## ACS Help

The following pages provide help for using the basic functions of the ACS, as well as descriptions of all implemented plugins.

#### **ACS Basic Functions**

The figure below shows the main panel of the AsTeRICS Configuration Suite after the application has started. The screen is divided in four main areas, the menu area (1), the *Deployment* area (2) (where the drawing will be done), the *GUI Designer* area (3), switchable with the *Deployment* area and the *Properties* area (4), where the settings of the components can be adapted.



after Startup

Additionally, the main menu will open, if the AsTeRICS-Button is pressed in the menu bar - see figure below. All functions of the main menu are also reachable via buttons in the tabs, except the *About* and the *Print* dialog. This dialog shows general information about the ACS and the AsTeRICS project.



Menu Bar with Open Main Menu

#### **Create and Edit a Model**

To create a new model, add one or several components. To do so, select the tab Components, and from the sub-menu select a group of components, depending on what is supposed to be added. The three component groups are *Sensors*, *Processors* and *Actuators*. Within these sections, the components are devided in subsections, making it easier finding one plugin out of the hugh amount of available plugins. Additionally, there is a fifth section *Saved Groups* where saved groups can be added to the drawing.



Now, this added component can be manipulated in the drawing area. This can be done with the functions in the Edit-tab or directly by using the mouse or the keyboard.

#### **Model Manipulation via Mouse**

After a component has been added to the drawing field, it can be moved by using the drag and drop functionality. Several elements (components, channels, eventchannels) can be selected by drawing a selection rectangle (just press the left mouse button and move the mouse) or by pressing the *Ctrl*-Key and clicking on each element. All the selected elements can be also moved using drag and drop. All selected components are marked with a blue rectange in the background, the component, which has the keyboard focus (and displayed in the property editor), is surrounded with a blue border. Channels can be drawn by pressing the mouse button over an output port and dragging the channel to an input port. Connecting eventchannels is similar to the channels.



Selection of Components

#### **Model Manipulation via Keyboard**

All elements within the drawing area can be set on focus, using the *Tab*key or the arrow-keys. To select an element, the *Space*-key has to be used, to select several elements, use *Ctrl*- and *Space*-Keys at the same time. By pressing the *Alt-Key, the ribbon menu shortcuts will appear at the menus. The App*-Key opens the context menu. More information about the usage of the keyboard within the ACS can be found in the section *Component Context Menu*.

#### Selection

It is possible to select several components, channels and event channels at the same time. The selection can be made with the mouse (by dragging a selection rectangle or by the usage of the *Ctrl*-key, like in many standard programs) or with the keyboard using the *Space* key. The selected components are marked with a blue frame. The figure below shows a model with several selected components.



Model with three selected components

## The Edit Tab

The edit tab is used for manipulating properties of the components and their interconnections. The list below provides a detailed description of the available operations found on the Edit Tab as shown in the figure.



Description of the Model Properties Group:

- *Edit Model ID* edits the unique model ID. This ID is generated automatically and is used by the ARE to store model based informations belonging to the editing model.
- Show Model Description shows the model description dialog. Within this dialog, the model description will be shown and can be edited. The description is divided in the parts of Short Description, Model Requirements and Detailed Description

Description of the *Edit Components* Group:

- *Move Component* enables the move mode of a component. Now the component can be moved around the drawing board using the arrow keys. Using the enter-key or tabbing to another component ends the move mode.
- Component Properties shows the properties of the component.

Description of the *Edit* Group:

- *New Channel* indicates that a channel is about to be drawn. To start drawing a channel, click on an output port or use the context menu.
- *New Eventchannel* indicates that an event channel is about to be drawn. To start drawing an event channel, click on an event trigger port or use the context menu.
- Cut cuts out the selected elements and store the cutted elements in

the clipboard.

- *Copy* copies the selected elements to the clipboard.
- Paste copies the elements from the clipboard to the drawing field.
- *Delete Selection* deletes the selected elements. This can also be done with the delete-key.
- Undo the last editing action like move a component, add/delete a component or also the channel and event channel operations. Setting properties and events (things done in the property window) are excluded form undo.
- *Redo* the last editing action that has been undone with *Undo*. Setting properties and events (things done in the property window) are excluded form redo.
- *Group* the selected components (incl. channels and event channels) to one group. The selected components must not contain a group. All connected ports and event ports will be available as external connection points of the group. The figure above shows a very simple example, the grouping of two components to one group.



Grouping of two components

- *Ungroup* the selected group, showing the single components of the group.
- Save Group saves a selected group, so the group can be reused in this or in other models. The saved groups are available in

#### Components / Saved Groups

After at least two components have been added to the deployment, they can be connected to each other. A connection always has to start at the output port (right hand side of a component) connecting to an input port (left side of the component). One output port can be connected to several input ports, but an input port can only receive data from one output port. Additionally the data types of the ports must match in order to be able to connect them, see the section on Channels.

#### **Component Context Menu**

All editing functions, which require the usage of the mouse, can also be done with the keyboard, using the context menu of the component. The context menu appears using the right mouse key, the space key or the application key.

out	
Add Channel	▶ out
Connect Channel	
Drop Channel	
Add Event Channel	
Connect Event Channel	
Drop Event Channel	
Move	
Delete	
Properties	
Show Component Status	
Remove Conflict Marker	

Component with the Opened Component Context Menu

#### Channels

Channels are the main way to transmit data from one component to another. A channel always transmits information from the output port to the input port. The data type of the channel is always equal the data type of the output port. The components of the AsTeRICS platform process one or several of the following data types, represented by the ports of the components:

- Boolean: can be true or false
- Byte: numbers from -128 to 127
- Char: one character
- Integer: numbers from approx. -2 billion to +2 billion
- Double: huge amount of positive and negative floating point numbers
- String: a string of characters (up to whole sentences)

The ports can be connected to ports with the same data type or following these connection rules:

- byte to integer
- byte to double
- char to integer
- char to double
- integer to double
- double to integer

#### **Events**

The AsTeRICS platform knows two concepts of connecting two components to each other. The first one is channels, where data is transported from one component to another. The second one is the events-concept. Events are single or continuous happenings, which should trigger an action at the receiver. After an event channel has been established between a trigger and a listener, the events have to be set in the events tab (which appears in the property area - by default on the right side of the ACS). In this event tab, there is a table with two columns: the left column lists the event listeners, the right column the event triggers. So, with the selection box on the right side (second column), the triggering event for the listener will be set. One component can send and receive events from several other components. The following figure shows the setting of events.



with Active Events Tab

#### **Setting the Properties**

Each component has 0 to several properties, by means of which the behaviour of the component can be adapted or functions can be enabled. These properties are not limited to the component itself. Also the ports (input and output, but not the event ports) can have properties. The usage of the properties and their effects can be found in the documentation of the components. The following figure shows the property editor editing the characteristics of the component "generator". By default, the property area is on the right hand side of the ACS.

	⇒ ×	Properties (Ctrl-P)		• џ
		Component: generator	.1	
		Search		
		<ul> <li>Internal Properties</li> </ul>		
generator.1	E	Component Descriptic	Signal Generator	
		Component Name	generator.1	
out 🛑		easyConfig		
		ComponentType	sensor	*
		<ul> <li>Properties</li> </ul>	1	
		send_interval	20	<b>a</b>
		waveform	sine	-
		frequency	2	
		amplitude	100	
		phaseshift	0.5	
		offset	0	

Property Tab showing a Component's Properties

#### **Dynamic Properties**

Beside the standard properties, components can also have dynamic properties. This means, properties in the property editor can become values from the component on the ARE. This feature might be useful in several cases, as example if a plugin is hardware dependent (selecting a soundcard or a midi player), or depending on files on the target system, like several sound files.

If a plugin is implementing a dynamic property, the values will be requested from the ARE, as soon as the ACS is synchronised with the ARE. Then, the received values can be selected in a combobox within the property editor (as shown in the figure below).



Plugin with Dynamic Property

## Tooltips

Tooltips are used to provide context information when editing the model. They can be found when hovering over an item of interest. The following items provide tooltips:

- Deployment (drawing) area, when hovering
  - a component
  - an input port of a component
  - an output port of a component
  - the event listener item of a component
  - the event trigger item of a component
  - an event channel line
- Property editor, when hovering
  - a property key
- Input port editor, when hovering
  - $\circ$  a property key
- Output port editor, when hovering
  - a property key
- Event channel editor, when hovering
  - $\circ\;$  the column heading of event trigger or event listener
  - an entry of an event trigger or event listener
- Components menu, when hovering
  - a menu item
- Components search, when hovering
  - a search result item

#### **Open and Save Models**

In the system tab, models can be saved on the local file system (*Save Model*, *Save Model* as), or loaded from the local file system (*Open Model*). *New Model* cleans up the drawing field, preparing everything for a new model.



## **Control the ARE**

In the system-tab, the group ARE handles the functionalities for connecting to and communicating with the ARE. The connection to the ARE is handled by the AsTeRICS Application Programming Interface (ASAPI).



Tab, ARE and ARE Storage Group

 Connect to ARE connects the ACS with the ARE. The Connect to ARE dialog appears, asking for the connection data. The host name (IP-address of the host) can be found in the ARE configuration, the default port should be 9090. Beside this connection dialog, also auto connection can be used, see the section Options / General Settings.

Host Name:	localhost	Port Number:	9090
			Ok Cancel

The ARE Connection Dialog

When the connection has been established, two special cases can occur:

- An active model (deployment) has been detected on the ARE. The user will be asked to download this model or to proceed without downloading it.
- An active model (deployment) has been detected and is running on the ARE. The user will be asked to download this model and switch the ACS to *Running* mode or to proceed without downloading it.
- Disconnect from ARE closes the connection to the ARE.
- *Upload Model* transmits the model in the drawing from the ACS to the ARE. The model on the ARE will be overwritten. Uploading the

model to the ARE does not start the model on the ARE.

- *Download Model* transmits the active model from the ARE to the ACS. The model on the drawing area will be overwritten.
- *Download Bundles* transmits the bundel description (describing the components) form the connected ARE to the ACS. These bundel descrptions will be available as components in the components-tab. Then bundles can be managed with the *Bundle Manager*.

The group ARE Storage deals with the storage on the ARE. The storage is an area within the ARE where models can be stored and also activated using the ARE interface.

- *Store Model on ARE* transmits the model in the drawing from the ACS to the ARE storage. A dialog appears to set the filename.
- Load Model from Storage transmits a model from the ARE storage to the ACS. The model on the ACS drawing area will be overwritten. A dialog appears to select the filename of the model.
- Activate a Stored Model: A dialog appears to select the filename of a model in the storage. This model will be set active in the ARE and also will be started. Furthermore, the model on the ACS drawing area will be overwritten with the selected model and the ACS switches to run-mode.
- *Delete a Stored Model* deletes a model from the ARE storage using a file dialog.
- Set as Autorun sets the model as autorun model. This model will be started automatically when the ARE starts.

Starting and stopping a model can be done with the buttons in the group Model.



- *Start Model* starts the model on the ARE and switches the ACS into run-mode. This means that now no components, channels and event channels can be added, edited or deleted. The drawing area is greyed out.
- Pause Model pauses the model on the ARE.
- Stop Model stops the model on the ARE and ends the run-mode.

## **GUI Designer**

The GUI Designer allows to arrange the GUI Elements of components with a graphical user interface. These GUI Elements will then be shown on the ARE GUI. All GUI elements will be drawn and deleted automatically, if a component with a GUI is added or deleted. Within the GUI Designer, the size and the position of the GUI Elements can be changed.



GUI Designer

#### Miscellaneous

Within the tab *Miscellaneous* (Misc.), status reports and logging files can be requested from the connected ARE, options can be set, the *Component Collection Manager* can be called and external tools can be launched. The three supported external tools are the *Plugin Creation Wizard*, the *Plugin Activation Wizard* and the *Language Translation File Creator*.



# Status Reporting and Error Logging

Within the tab *Miscellaneous* (Misc.), status information and error logging can be requested from the ARE (if an active connection to the ARE is available).

	222223	0222223				
		Q.	:	9	9 <b>1</b> >	76
Get ARE Sł Status	how Logfile from ARE	Component Collection Manag	Options ger	Plugin Creation Wizard	Plugin Activation Wizard	Create Language Translation File
	Mise	cellaneous			External Tools	

• *Get ARE Status* opens a window, showing status messages from the ARE and from the components within the ARE. These messages can be copied to the clipboard or saved to a file:

Status	Component	Message
ok	ARE	
deployed	ARE	
ok	ARE	
ok	ARE	
deployed	ARE	
running	ARE	
ok	ARE	

The ARE Status Window

• Show Logfile from ARE shows the logging file from the ARE, containing status and error messages:

	form the ARE:			
06.05.2011 09:0	1:12 eu.asterics.component.a	actuator.corecimdisplay.CoreCi	mDisplayInstance handleO	kPress
INFO: Ok butto	n press			
06.05.2011 09:0	1:12 eu.asterics.component.a	actuator.corecimdisplay.Model	MenuItem action	
INFO: Loading	model: bartest1.acs			
06.05.2011 09:1	.8:06 eu.asterics.mw.are.Asap	iSupport runModel		
INFO: eu.asterio	cs.mw.are.AsapiSupport.runM	lodel: model running		
06.05.2011 09:1	.8:10 eu.asterics.mw.are.Asap	iSupport stopModel		
06.05.2011 09:1 INFO: eu.asterio	.8:10 eu.asterics.mw.are.Asap :s.mw.are.AsapiSupport.stop1	iSupport stopModel Model: model stopped		
06.05.2011 09:1 INFO: eu.asterio	8:10 eu.asterics.mw.are.Asap cs.mw.are.AsapiSupport.stop1	iSupport stopModel Model: model stopped		
06.05.2011 09:1 INFO: eu.asterio	8:10 eu.asterics.mw.are.Asap s.mw.are.AsapiSupport.stop1	iSupport stopModel Model: model stopped		
06.05.2011 09:1 INFO: eu.asterio	8:10 eu.asterics.mw.are.Asap :s.mw.are.AsapiSupport.stop1 	iSupport stopModel Model: model stopped		Þ

The ARE Logging Window

Beside this general information, the status of each component can be requested using the *Show Component Status* in the context menu of each component. The following figure shows a component with a red background - this means an error occurred in this component. The status window shows this error and with the button *Remove Error Marker* the red background can be removed.

X   @   AsTeRICS Configuration Suite		-							- 0	23
Connect Disconnect Upload Download to ARE Model ARE	Edit Misc.	Load Model Actin from Storage Stored ARE Storage	vate a Delete a i Model Stored Model	Start Model Model Model Model	New Open Model Model	Save Model Mo	Save odel as			
Deployment (Ctrl-D)	ARE Status	ntur marcana far can	energet advisout 1		- • ×	⇒×	Properties (Ctrl-P) Component: adcinput Search Internal Properties Component Descripti	c ADC CIM sensor f	or inputs	• ‡
	Status	Component adcinput.1	Message Could not get port of	ontroller for CIM ID: 401			easyConfig ComponentType Properties input_active_in1	sensor		¥
							input_active_in2 input_active_in3 input_active_in4 input_active_in5			
							input_active_ino input_active_in7 input_active_in8 periodic_update			۲
	Remove Erro	or Marker	Close	Copy to Clipboard	Save to File	J				
۲ <u>۱۱۱</u>						+	Properties (Ctrl-P)	♂ Output Ports		
ARE Status: Synchronised						100%		Ctrl +/-		

A Component in Error State with the opened Status Window

#### **Component Collection Manager**

The Component Collection Manager (see Figure below) is a small tool in which downloaded component collections (the description of the available plugins within the AsTeRICS Runtime Environment) can be saved and administered. Within the component collection manager, the following functionalities are provided:

- Use Default sets the default ACS component collection as active component collection
- Set as Autostart sets the active component collection as autostart component collection, which will be loaded at ACS startup.
- Save Component Collection saves the active component collection (e.g. a downloaded component collection from the ARE) into the ACS folder.
- The Saved Component Collections list shows all saved component collections. A component collection can be selected and set active.

nent Collection
nent Collection
ction as Autostart
ve Close

Component Collection Manager

## Print

The ACS also includes very basic print functionality, located in the main menu. The model will be printed on one A4 page, properties or set events will not be printed.

#### Options

Within the *miscellaneous* tab, the options dialog can be opened. This dialog is divided in three parts: *General Settings*, *Dialogs* and *Colours*.

#### **General Settings**

Se	t the Editor-Options for the AsTeRI	CS Configuratio	on Suite 🛛 🗃	
	General Settings Dialogs Cole	ours		
	Reset window arrangement:	RESET		
	Language:	English	<ul> <li>(setting will take effect after restart of ACS)</li> </ul>	
	Connection Data:	Oetect AR	E automatically	
		O Host:	localhost	
		Port:	9090	
		Connection 1	Timeout (ms)	
			20000	
	ARE Status Update	📝 Enable au	utomatic ARE status update	
		Update frequ	uency (ms)	
		Q	1000	
	Automatic backup files	🔽 Enable au	utomatic creation of backup files	
			Ok Cancel	
_				Ontic

Dialog, General Settings

- *Reset Window Arrangement* resets all layout settings to default values.
- Language gives the possibility to select the ACS language between English, German, Spanish and Polish. The properties of the components will not be affected by this, as they are dependent on the component description. After changing the language, a restart of the ACS is required for the changes to take effect.
- Connection Data is responsible for the connection of the ACS with the ARE. The Host contains the IP-address of the ARE, the Port its IP-port. Default port is 9090. If Detect ARE automatically is selected, the host information will be ignored and the ACS tries to find the ARE in the network. If more then one ARE will be detected in the network, an ARE selection dialog will appear. Connection Timeout

sets time network timeout time (in milliseconds). If the ARE is not reacting after this time, the connection to the ARE will be closed.

- ARE Status Update enables or disables an automatic update of the ARE status. This status update works while the ACS is in run-mode. The Update Frequency sets the time between two status updates (in milliseconds).
- Automatic Backup Files creates a backup file during each Save File process. The backup file gets the ending .backup.

#### **Dialogs Settings**

Set the Editor-Options for the AsTeRICS Configuration Suite	
General Settings Dialogs Colours	
Show naming-dialog on component-insert	
Show host/port-dialog on ARE-connection	
Show confirmation message after connecting an Event Channel	
Show confirmation message on successful ARE-connection	
Ask before overriding a model on the ARE?	
Ask before overriding local model?	
Show confirmation message to override model when connecting to ARE	
Show confirmation message to override and run model when connecting to ARE	
Ok Cancel	
	Optic

Dialog, Dialogs Settings

Within this options tab, dialogs can be activated or deactivated. In the ACS, several dialogs have the option *Show this dialog every time*. If a dialog has been deactivated there, it can be reactivated in the options dialog.

## **Colours Settings**

Set the Editor-Options for the AsTeRICS Co	nfiguration Suite		
General Settings Dialogs Colours			
Component Header	Choose Colour		
Group Header	Choose Colour		
Component Body	Choose Colour		
Input Ports	Choose Colour		
Output Ports	Choose Colour		
Input Eventport	Choose Colour		
Output Eventport	Choose Colour		
		Ok Cancel	
			Optic

Dialog, Colours Settings

Within the *Colours* tab, the colours of the different parts of a component can be changed. The colour chooser not only allows changing the colour, also the transparency can be changed.

#### Modes

The ACS can enter several different modes, depending on the status of the connected ARE. The connection status is displayed at the bottom left corner of the ACS. Possible ACS modes are:

- Disconnected
- <u>Connected</u>
- <u>Synchronized</u>
- <u>Running</u>
- Pause

#### Disconnected

This is the standard mode after the ACS has been started. The drawing area is enabled, new components can be added and channels between the components can be established. Models cannot be uploaded or downloaded and also the ARE storage is not accessible, due to the fact that there is no ARE connected.

#### Connected

This mode is reached after the ACS has been connected to the ARE. The drawing area is enabled, new components can be added and channels between the components can be established. Models can be uploaded or downloaded, also the ARE storage is accessible. The status and the logging file can be requested from the ARE.

## Synchronized

After a model has been uploaded or downloaded, the ARE is synchronized with the ACS. The model can now be started on the ARE, using the *Start Model* button. Adding or removing components and editing channels or event channels will cause a switch back to the mode *connected*. Changing properties of the components will not change the mode, as these changes are transmitted to the ARE in the background. Also the status and the logging file can be requested from the ARE.

#### Running

After the *Start Model* button has been pressed, the ACS is in the *running* mode. Within this mode, the drawing area is disabled (indicated by a grey background) and the buttons in the components tab and the edit tab are disabled, so elements can only be selected or moved. The following figure shows a screenshot of the ACS in running mode.



ACS in the Running Modus
# Pause

This mode is similar to the running mode, with the difference, that the model on the ARE is not running but in a pause state.

# **Keyboard Control**

The AsTeRICS Configuration Suite is fully accessible by keyboard. The following sections describe the usage in the different parts of the ACS.

#### Menus

The whole ACS is fully accessible by keyboard. The menu items of the ACS menus can be acessed pressing the *Alt*-key, followed by the keys, being displayed. Screenshot 1 shows the menu after pressing *Alt*, Screenshot 2 shows the System menu after pressing *Alt* followed by pressing *Y*.



Screenshot: The menu after pressing Alt followed by pressing Y

## The Drawing Area

The drawing area can be reached, pressing *Ctrl+D*, the focus will be on the first element. The focus between the elements can be changed using the *Tab* key or the Arrow keys. Elements can be selected using the *Space* Key, multiple elements can be selected, pressing *Ctrl+Space*. Pressing the *App* Key (also called Context Menu key) opens the context menu. Screenshot 3 shows a selected component with the opend context menu.



Screenshot: Component with opend context menu

#### The Context Menu

- *Add Channel* creates a new channel. An output port must be selected to initiate a new channel.
- *Connect Channel* sets the end point of the new channel. Select an empty input port to finish the channel
- Drop Channel deletes an initiated but unfinished channel
- Add Event Channel creates a new event channel. An event trigger

must be selected to initiate a new event channel.

- *Connect Event Channel* sets the end point of the new event channel. Select an event listener to finish the event channel. Afterwards, the events must be set in the Event Editor
- Drop Event Channel deletes an initiated but unfinished event channel
- *Move Selection* moves the selected elements on the drawing area by using the arrow keys. The move operation can be stopped by pressing *Enter*.
- *Delete Selection* deletes the selected components, channels and event channels.
- *Properties* sets the focus to the Property Editor
- Show Component Status displays a component status window.
- *Remove Conflict Marker* removes the marker, indicating a version conflict between the component collection of the ARE and the ACS.

## The GUI Editor

Accessing the GUI Editor is quite similar to the Drawing area. Using the Ctrl+G key combination sets the focus on the first element. The App Key (also called Context Menu key) opens the context menu, offering the options of:

- *Resize* chnages the width and hight of the GUI element by using the arrow keys.
- *Stop Resize* stops the resize operation. This function will automatically be called if the focus is set to any other element (e.g by pressing the *Tab* or the *App* key).
- *Move* moves the GUI element on the GUI area by using the arrow keys.
- *Stop Move* stops the move operation. This function will automatically be called if the focus is set to any other element (e.g by pressing the *Tab* or the *App* key).

#### **Property and Port Editor**

*Ctrl+G* sets the focus to the property editor tab. Using the *Tab* key sets the focus to the property editor, using the arrow keys allows to navigate to the other tabs (input and output ports, event triggers and listeners). Within the property editor, navigation will be handled using the *Tab* key to select and edit the different properties. Within the input and output ports, the list of ports can be navigated using the arrow keys (so shown in screenshot 4).

noseX		
Output Port: nose)	<	
Search		
<ul> <li>Internal Propertie</li> </ul>	es	
Port Label	noseX	
Port Alias		
PortDataType	integer	-
Description	Output port of nose X coordinate	
noseY		
Output Port: nose)	/	
Barch Search		
<ul> <li>Internal Propertie</li> </ul>	es	
Port Label	noseY	
Port Alias		
PortDataType	integer	-
Description	Output port of nose Y coordinate	
chinX		
chinY		

Screenshot: Port Editor with opend and closed output ports

### **Event Editor**

With the Event Editor (reachable with the shortcut Ctrl+E), events within an event channel will be set. Once again, the navigation will be handled using the *Tab* key. When a combobox is in focus, the value can be changed using the up an down arrow keys. Furthermore, the combobox can be opend using the *Enter* key. Screenshot 5 shows the event editor with an opend combo box.

T' 4	DI 16 DI 19 IT 1	
Timer.1	PlatformDigitalIn.1	
start	in1Low	•
start		
stop		
reset	in1High	
	in3High	
	in1Low	
	in2Low	
	in3Low	

Screenshot: Event Editor with opend combobox

# **Keyboard Shortcuts**

#### Keyboard Shortcuts

Кеу	Command	
F1	Open Help window	
F5	Start the model	
F6	Pause the model	
F7	Stop the model	
Ctrl + A	Select all elements in the drawing area	
Ctrl + C	Copy all selected elements to the clipboard	
Ctrl + X	Cut out all selected elements and copy them to the clipboard	
Ctrl + V	Paste the elements from the clipcoard to the drawing area	
Ctrl + Z	Undo last operation	
Ctrl + Y	Redo last operation (after Undo)	
Ctrl + S	Save the drawing	
Del	Delete the selected elements	
Ctrl + +	Zoom in	
Ctrl + -	Zoom out	
Ctrl + D	Set focus on the drawing area	
Ctrl + G	Set focus on the GUI area	
Ctrl + P	Set focus on the Property editor	
Ctrl + E	Set focus on the Event editor	
Alt	Display keyboard shortcuts of the menu elements	
Tab	Navigate to the next element (in several areas)	
Arrow Keys	Move Focus in several areas, also move in scroll areas	
Alt + F4	Close the ACS	

# Plugins

In the following sections, all the plugins available in the AsTeRICS framework will be presented. The plugins can be divided in 3 groups: sensor modules, processing modules and actuator modules.

- Sensors include all software modules which make physical or simulated data available to other AsTeRICS plugins. Examples include a digital switch interface, an analogue sip/puff sensor, a webcam or a signal generator.
- *Processors* include mathematical transformations, feature detectors, threshold level monitors etc. which are necessary to scale or combine signals or detect interesting events in the data streams acquired from the sensors.
- Actuators enable assistive functionalities like mouse- or keyboard replacement, visual or acoustic feedback or direct interaction with the environment, e.g. infrared remote control, home automation or physical manipulation.

The plugins represent the functional blocks of AsTeRICS. They can be combined to tailored Assistive Technology setups via the graphical AsTeRICS Configuration Suite software application (ACS), where plugins can be selected, connected via channels, and parameterized. Thus, the ACS can be considered as a user friendly environment to arrange the plugins. The result of this configuration process is an .xml file containing the deployment model for the AsTeRICS Runtime Environment (ARE).

# Sensors

Sensors are defined as components which generate data to be processed within the processing chain set up in the AsTeRICS Runtime Environment. Sensors can either generate data in the component itself (e.g. a signal generator) or they can read data from an external sensor (such as an ADC or a digital GPIO port). Sensors only contain output ports as no input data from within the ARE is needed.

# Acceleration

# **Component Type: Sensor (Subcategory: Inertial Measurement)**

The Acceleration input component is a sensor which is used in combination with the Acceleration CIM created in the course of the AsTeRICS project. The component provides three output ports which provide acceleration values for x, y and z axis. Via the component properties, the sampling rate and the sensitivity (acceleration range) can be set.



## Requirements

This software component requires an Acceleration CIM (CIM ID: 0x0501) connected to the platform.



#### **Output Port Description**

- **xAcc [double], yAcc [double], zAcc [double]:** These three output ports provide actual acceleration values of the X-, Y- and Z axis. The values are updated as a new data packet comes in (the sampling rate is set via the property update\_frequency.
- total [double]: This output port provides the total acceleration value, calculated by the square root of (x\*x + y\*y + z\*z).

### **Event Listener Description**

- **start:** An incoming event on this port starts the Acceleration CIM. (If the property autostart is set, the CIM is started automatically when the model is started).
- **stop:** An incoming event on this port stops the Acceleration CIM.
- calibrate: An incoming event on this port samples the current values of X, Y and Z axis and stores them as new "offset values". These offset values are subtracted from the incoming x, y and z values. Please note that this is not a real calibration of the sensor, but can set a baseline for the output values to zero, for example when the sensor is in resting pose.

### **Properties**

- **autoStart [boolean]:** This property specifies if the Acceleration CIM shall be started together with the model (if value is true) or if the Acceleration CIM receives a sepertate start condition via the dedicated event listener port (if value is false).
- **discreteSteps [integer]:** If this property has a positive value greater than zero, the effective range of values for the X,Y and Z axis (-8192 to 8192) is mapped to a smaller number of values given by the property. For example: If he value is 10, the whole range of acceleration is mapped to values from -5 to 5. This can be helpful if a small number of different values is desired, e.g. for controlling the mouse cursor speed.
- **updateFrequency [integer]:** This property defines the bandwidth of the acceleration sensor and respectively the update rate of acceleration values on the output ports. The update rate is twice as fast as the selected bandwidth. The bandwidth can be selected as 10Hz, 20Hz, 40Hz, 75Hz, 150Hz or 300Hz. This means that if a bandwidth of 40Hz is selected, the x, y and z values are updated 80 times per second and frequencies up to 40 Hz can be measured with the acceleration sensor. Values of 20 Hz to 75 Hz are recommended for normal operation where body movements are measured (the highest possible frequency 300Hz can cause performance troubles in the current platform implementation because the update rate of 600 times per second is too fast).
- accelerationRange [integer]: This property defines the sensitivity of the acceleration sensor (the measureable range of acceleration, defined in g). The range can be selected as 1g, 1.5g, 2g, 3g, 4g, 8g or 16 g via the combo box selection in the ACS. The selected acceleration range is mapped to values of -8192 to 8192 on the output ports. A value of 3 g is recommended for normal operation where body movements are measured.

# Analog In

# **Component Type: Sensor (Subcategory: Generic Control Input)**

The AnalogIn component is a sensor which is used in combination with the ADC CIM. The component provides two output ports which correlate to the inputs IN1 and IN2 of the CIM. The component can be set up to sample the inputs periodically or upon an incoming event. Both inputs can be used for voltage and resistance measurement.



## Requirements

This software component requires an ADC CIM (CIM ID: 0x0901) connected to a USB port.

#### **Output Port Description**

• **in1 to in2 [integer]:** Each of these output port corresponds to one input of the ADC CIM. The measured values are forwarded directly from the CIM thus corresponding to the description given in the CIM specification. Depending on the type of connected sensor the values either correspond to a voltage or a resistive value.

#### **Event Listener Description**

• **adcSampleTrigger:** An incoming event on this port will cause the ADC CIM to sample data on its inputs. This event will only take effect if the periodic\_update property is set to 0.

### **Properties**

- **periodicUpdate [integer]:** This property defines the time between data sampling requests to the ADC CIM (in milliseconds). The software component will send requests to the CIM in the given intervals. Due to current limitations of the ADC CIM it is recommended to use time intervals not lower than 50 milliseconds (20 samples per second).
- activateInput1 to activateInput2 [Boolean]: These properties correspond the CIM's ADC inputs IN1 to IN8, if the property is true, the corresponding output port of the software component will send the sampled data.
- **uniqueld:** unique number of the CIM if more than one CIMs of the same type are used. The module flashes a LED for identification when the ID is selected. **Supports value suggestions from ARE** (dynamic property)

# **Autostart Event**

#### **Component Type: Sensor (Subcategory: Simulation)**

The AutostartEvent component sends an event after model start, with a delay defined by the Delay property.

Deployment (Ctrl-D) GUI Designer (Ctrl-G) 🖛	Properties (Ctrl-P)		<b>▼</b> Ą
A	Component: Autostar	tEvent.1	
-	Search		
	<ul> <li>Internal Properties</li> </ul>		
	Component Descripti	<ul> <li>Sends the event after model start, with de</li> </ul>	fin€
	Component Name	AutostartEvent.1	
AutostartEvent.1	ComponentType	sensor	-
	<ul> <li>Properties</li> </ul>		
	Delay	1000	\$
, ,			
• • • • • • • • • • • • • • • • • • •	Properties (Ctrl-P)	🗲 Event Trigger	

AutostartEvent plugin

# **Event Trigger Description**

• **output** The output event.

## **Properties**

• Delay [integer]: The event delay[ms].

# **Button Grid**

# **Component Type: Sensor (Subcategory: Graphical User Interface)**

The Button Grid component is a simple GUI on-screen keyboard. It sends events after buttons have been pressed.



## **Event Trigger Description**

• **button1...button20:** These events are fired as the corresponding buttons are pressed.

### **Properties**

- caption [string]: The text shown on the component caption.
- horizontalOrientation [boolean]: If selected, the keyboard will be placed horizontally, otherwise the keyboard will be placed vertically.
- textColor [integer, combobox selection]: The color of the caption text ("default" keeps the standard setting)
- **backgroundColor [integer, combobox selection]:** The color of the button background ("default" keeps the standard setting)
- **borderColor [integer, combobox selection]:** The color of the button frame ("default" keeps the standard setting)
- borderThickness [integer]: The size of the button frame
- **selectionFrameColor [integer, combobox selection]:** The color of the selection frame (active button, "default" keeps the standard setting)
- **selectionFrameThickness [integer]:** The size of the selection frame
- **displayGUI [boolean]:** if selected, the GUI of this component will be displayed if not, the GUI will be hidden and disabled.
- **buttonCaption1...buttonCaption20 [string]:** The text shown on the buttons:1...20. If the text is empty, the button will not be displayed at all.
- **toolTip1...toolTip20 [string]:** The text shown as tooltip when the mouse hovers over a corresponding button. If the text is empty, no tooltip will be displayed.

# CellBoard

# **Component Type: Sensor (Subcategory: Graphical User Interface)**

The CellBoard plugin is a GUI plugin which can display a panel with multiple cells. It can be used as a simple on-screen keyboard. Each cell can contain text and a picture, and it can provide acoustic feedback on selection. Cells can emit command strings and sounds or switch to other cellboards (which can be stored as .xml files). The CellBoard plugin provides basic scanning options (to select a cell via a stepwise process). It also provides a small editor for designing the cells and saving the whole cellboard grid as an .xml file.

Deployment (Ctrl-D) = ×	Properties (Ctrl-P)	· · · · ·
	Component: CellBoard	i.1
	Search	
	<ul> <li>Internal Properties</li> </ul>	^
	Component Class	sensor 💌
	Component Name	CellBoard.1
	Component Type	asterics.CellBoard
	Component Description	Adjustable matrix of button cells with scann
CellBoard.1	<ul> <li>Properties</li> </ul>	
	caption	Cell Board
actCell	rows	3
column actCellCaptior	columns	3
cellNumber actCellText	textColor	black
	backgroundColor	white
	scanColor	red 💌
	scanMode	none
selected centre	scanCycles	3
scankow	hoverTime	1000
scanColumn	hoverIndicator	fixed frame
	hoverFrameThickness	4
	enableEdit	X
	enableClickSelection	X
	commandSeparator	
	keyboardFile	
	ignoreKeyboardFilePr	c 🗌
	displayGUI	X
~	cellText1	~
< >	Prop 🖙 Inpu	🖻 Outp 📝 Even 🔰 Even

CellBoard plugin

### Using the cell editor

The cell editor can be opened by right-clicking a cell. (This option can be enabled/disabled by using the property "enableEdit"). The cell editor allows to define the content of a cell (cell caption, action string, cell image and sounds for cell-selection or acoustic preview during cell scanning). "Switch to Cellboard" allows to specify a cellboard grid which will be loaded when this cell is selected. When "back" is entered here, the previously loded cellboard will be restored. Thus, complex communication/selection grids can be layered. The whole current cellboard is saved as an .xml file when "Save Cellboard" is clicked. Existing .xml files can be selected to be loaded on startup via the "keyboardFile"-property of the cellboard plugin in the ACS property editor.

Cell Board-			
first	second	thrid	
forth	fifth	•	Cell Editor ×
seventh	eighth	fifth @IRTRANS: snd	Cell Action String asterics_tv,on;@SPEECH: TV on Path to Cell Image
			Path to Soundfile
			Path to PreviewSound
		Switch to Cel	board (use "back" to go to previous grid)
		data/cellBoardKeyb	Cellboard filename: oards/dummy.xml Save Cellboard Cancel

CellBoard editor

#### **Input Port Description**

- **row [integer]:** This port can be used in "directed scanning" mode to set the row of the highlighted cell.
- **column [integer]:** This port can be used in "directed scanning" mode to set the column of the highlighted cell.
- **cellNumber [integer]:** This port can be used in "directed scanning" mode to set the number of the highlighted cell.
- **xmlFile [string]:** Name of a keyboard file (default location in data/cellBoardKeyboards) which can be loaded or saved. A load operating can be triggered via the dedicated event listener port.

### **Output Port Description**

- actCell [integer]: Sends the number of the currently scanned cell.
- actCellCaption [string]: Sends the cell caption of the currently scanned cell.
- actCellText [string]: Sends the action string of the currently scanned cell
- selectedCell [integer]: Sends the number of the selected cell.
- **selectedCellCaption [string]:** Sends the cell caption of the selected cell.
- **selectedCellText [string]:** Sends the action string of the currently active cell
- **scanRow [integer]:** Sends the number of the currently scanned row (in row/column scanning mode)
- **scanColumn [integer]:** Sends the number of the currently scanned column (in row/column scanning mode)

### **Event Listener Description**

- **scanMove:** Moves the scanning highlight frame in the row-column and column-row scanning modes.
- **scanSelect:** Selects the highlighted cell in the row-column, column-row and directed scanning modes.
- **moveUp:** Moves the scanning highlight up in the directed scanning mode (wrap around is possible).
- **moveRight:** Moves the scanning highlight to the right in the directed scanning mode (wrap around is possible).
- **moveLeft:** Moves the scanning highlight to the left in the directed scanning mode (wrap around is possible).
- **moveDown:** Moves the scanning highlight down in the directed scanning mode (wrap around is possible).
- **load:** loads a keyboard from the given xml-filename (property "keyboardFile").

## **Event Trigger Description**

- **cellClicked:** This event is triggered when one of the cells is clicked.
- **scanCancel:** This event is triggered when the row/column scanning process is cancelled because the maximum number of scan cycles has been reached.
- **cell1...cell36:** This event is triggered when the given cell is selected.

### **Properties**

- **caption [string]:** The component caption.
- rows [integer]: The number of the cell rows (1-36, rows x cols <= 100).</li>
- **columns [integer]:** The number of the cell columns (1-36, rows x cols <= 100).
- textColor [integer]: The color of the cell text.
- **backgroundColor [integer]:** The color of the cell background.
- scanColor [integer]: The color of the scanning highlighting.
- scanMode [integer]: This property defines the scanning mode:
  - "none": The board is not scanned. Users can select the cell by clicking on it.
  - "row-column": In this mode, rows of the cells are scanned. When the user selects the row, cells in the row are scanned. The scanning frame can be moved using the scanMove event port. The row and the cell itself can be selected using the scanSelect event port.
  - "column-row": In this mode, columns of the cells are scanned. When the user selects the column, cells in the column are scanned. The scanning frame can be moved using the scanMove event port. The column and the cell itself can be selected using the scanSelect event port.
  - "directed": In this mode only one cell is highlighted. The user can move the scan highlighting via the moveUp, moveRight, moveLeft or moveDown event ports or the row, cellNumber and column input ports. The cell can be selected using the scanSelect event port.
  - *"hover selection":* In this mode the user can select the cell by hovering the mouse pointer above the cell.
- **scanCycles [integer]:** This parameter is used in "row/column scanning" mode. After the number of scan cycles is reached, the scanning switches back to the highest level.
- **hoverTime [integer]:** This parameter is used in "hover selection" mode. It defines the time of hovering needed to select the cell (in milliseconds).
- hoverTimeIndicator [integer]: This parameter adjusts the way a

hover selection prgress is indicated to the user:

- "fixed frame": No graphical indication the frame of the currently active cell is shown but does not change.
- "growing frame": The frame of the active cell is growing until the value of property "hoverSelectionThickness" is reached - then the cell is selected.
- "backgorund color": The background color of the cell changes slowly from the selected "backgroundColor" to the selected "scanColor" - then the cell is selected.
- **hoverFrameThickness [integer]:** Maximum thickness of the cell frame (only valid for the hover selection mode with indicator "growing frame").
- **enableEdit [boolean]:** If selected, the cells can be right-clicked to display the cell editor, which also allows to save the xml-keyboard file.
- **enableClickSelection [boolean]:**If selected, all cells can be selected via a left click, regardless of the scanning process
- **commandSeparator [string]:** If a character (or characters) are specified here, they will be used to tokenize the cell text which is sent to the "selectedCellText" output port when teh cell is selected. This can be used to send multiple strings (seperately) to this output port. In the example screenshot above, a semicolon (";") is used to send two different action string to the output port when the cell is selected. A "StringFilter" plugin can then be used to route these commands to different plugins.
- **keyboardFile [string]:** Name of a cellboard .xml file (the sugeested files are located in folder data/cellBoardKeyboards or in folder "models"). This file will be loaded at startup of the model. If no file is specified, only the cell contents defined via the ACS editor will be available.**Supports value suggestions for existing xml-files from the ARE (dynamic property)**
- **ignoreKeyboardFileProperties [boolean]:** If selected, the general cellboard parameters (rows/columns/colors/scan mode etc.) will be kept as defined via the ACS editor when a cellboard .xml file is loaded. If not selected, these parameters will be loaded from the .xml file, thereby overriding the values defined in the ACS editor.
- **dispayGUI [boolean]:** If selected, the component will be displayed in the ARE GUI. Otherwise the componentent will be hidden !

The following properties will be overwritten if cellboard .xml files are loaded:

- **cellText1...cellText36 [string]:** The text displayed on the cell.
- **cellImage1...cellImage36 [string]:** The path of the image displayed on the cell.
- actionText1...actionText36 [string]: The text sent through the selectedCellText output port, when the cell is selected.
# **Digital In**

# **Component Type: Sensor (Subcategory: Generic Control Input)**

The DigitalIn component provides an interface to read the digital inputs of the GPIO CIM. On state changes of the connected signals (transitions to high or low level), the component generates corresponding triggerevents.

DigitalIn	Screenshot: DigitalIn plugin	
DigitalIn		
nlugin	nlugin	DigitalIn

### Requirements

This component requires the DigitalIn (GPO) CIM (CIM Id: 0x0701) connected to an USB port.

#### **Event Trigger Description**

- **in1High to in6High:** Each of these event ports is linked to one input port, if the device connection to this input port delivers a signal which changes to high level, an event will be raised on the corresponding port.
- **in1Low to in6Low:** Each of these event ports is linked to one input port, if the device connection to this input port delivers a signal which changes to low level, an event will be raised on the corresponding port.

#### **Properties**

- activateEventIn1 to activateEventIn6 [Boolean]: These properties declare for each port whether or not a signal transition on the actual input port should result in an event being triggered in the ARE. If a property is set true for one input, it will raise events on signal transitions, if it is set to false it will not.
- **periodicUpdate:** Period in milliseconds for update messages about state of device inputs. If the property is set to zero, the plugin will receive messages from the CIM on signal transitions, if the property is non-zero, the CIM will send status messages in the defined intervals without extra event messages on signal changes. Both modes will have the same effect on the software plugin, it will raise events on its trigger ports.
- **uniqueld:** unique number of the CIM if more than one CIMs of the same type are used. The module flashes a LED for identification when the ID is selected. **Supports value suggestions from ARE** (dynamic property)

### **EDF Reader**

#### **Component Type: Sensor (Subcategory: File System)**

This plugin reads an .edf file and sends the data to its output ports



ReadEDF

#### **Output port Description**

• **CH1 - CH8[double]:** these ports send the samples of the recorded signals.

#### **Properties**

• FileName[string]: specifies the name of the file, in which the data has been recorded Supports value suggestions from ARE (dynamic property)

## **Edit Box**

# **Component Type: Sensor (Subcategory: Graphical User Interface)**

This component creates a GUI edit field which can send the text to other components. The text will be sent to the output port when enter is pressed or when the edit box looses the input focus (when the cursor is removed from the edit field).

Edit Box			
Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Properties (Ctrl-P)		<b>→</b> û
	componentType: Edit	Box.1	
EditBox.1	Component Description	<ul> <li>Text edit box</li> </ul>	
	Component Name	EditBox.1	
output	ComponentType	sensor	-
	<ul> <li>Properties</li> </ul>		
	caption	Edit Box	
	default		
	textColor	black	•
	backgroundColor	white	•
	insertAction	do nothing	•
•	Properties (Ctrl-P)	Toutput Ports 🔰 Event Listener	EditBo
	plugin		

### **Output Port Description**

• **output [string]:** String output port.

### **Event Listener Description**

- **clear:** Removes the text from the component.
- **send:** Send the text value to the String output port.

#### **Properties**

- caption [string]: Caption of the component.
- **default [string]:** The default text, which is set at startup.
- textColor [integer]: Defines color of the text.
- **backgroundColor [integer]:** Defines background color.
- **insertAction [integer]:** Defines behaviour of the component after the text has been sent to the output port. The text in the component can be selected or removed.
- **sendDefaultValue [boolean]:** When this checkbox is checked the default String value is sent to the String output port when the model gets started.
- **displayGUI [boolean]:** If selected, the GUI of this component will be displayed if not, the GUI will be hidden and disabled.

## Enobio

# **Component Type: Sensor (Subcategory: Bioelectric Measurement)**

This component interfaces the Enobio sensor to the AsTeRICS system. It is in charge of driving the USB interface, commanding the proprietary protocol that Enobio uses, filtering the signal for removing the environmental noise and performing an automatic offset compensation for each channel in order to keep the electrophysiological signal correctly calibrated and avoid the effects that the skin contact may introduce in the signal. The component delivers the sampled signal in the Enobio electrodes through four output ports (one per channel). In addition, there is another output port which reports the calibration status of the channels and information regarding the sample loses due to environmental issues in the wireless link. The output ports (sampled data and status) deliver 250 values per second, which corresponds to the sample rate in the Enobio electrodes.



#### Requirements

This software component requires an Enobio receiver connected to the platform, the Enobio device switched on and the electrodes correctly placed on the user.



Enobio device

#### **Output Port Description**

- **Channel1 to Channel4 [integer]:** Each output corresponds to the sampled data from its corresponding Enobio channel. The integer represents the microvolts of the electro-physiological signal read by Enobio. The data might be pre-processed according to the value of the properties of the component.
- **Status [integer]:** This port provides information regarding both the calibration status of the four channels and the status of the wireless link. For every integer value that is available in the data output ports, another integer value is available in this port with the corresponding status information. The information is proprietary codified within a 16-bit integer. This includes information of calibration status of each channel and the status of the wireless link. This information would be kept away for the moment form the ARE programmers and provided upon request if necessary.

#### **Event Trigger Description**

- **externalSignalPosEdgeEvent:** This event is fired if the external signal toggles from low to high level.
- **externalSignalNegEdgeEvent:** This event is fired if the external signal toggles from high to low level.

#### **Properties**

- IsChannel1Activated to IsChannel4Activated [Boolean]: If this property is set to true, the corresponding channel is calibrated, thus the raw data from this channel will be meaningful.
- HighPassFilterInChannel1 to HighPassFilterInChannel4 [Boolean]: If this property is set to true, a high pass filter is applied to the data from the corresponding channel.
- LineNoiseFilter [Boolean]: If this property is set to true, a 50 Hz band pass filter is applied to the data before it is passed to the output port. This filter is useful when the environmental electrical noise is present in the signal.

## EOG

#### **Component Type: Sensor (Subcategory:Sensor Modules)**

This component provides the EOG signal (electro-oculogram) from the EOG sensor board, which has been developed in course of the Bachelor theses of Benedikt Rossboth at the UAS Technikum Wien. The EOG sensor board delivers information about eye movements and eye blinks which can be used in alternative user input setups.

For operation, three surface electrodes must be fixed to the users head (one electrode aside and above one eye, and one DRL electrode e.g. at the wrist of the user). In a proof-of-concept model, a musical instrument was implemented where a user can control midi tones via eye movements. For detailed information please refer to the file Documentation/DIYGuides/EOG\_Rossboth.pdf

GUI Designer (Ctrl-G) Deployment (Ctrl-D) = ×	Properties (Ctrl-P)	🗕 û	
	Component: EOG.1		
	Search		
	<ul> <li>Internal Properties</li> </ul>		
EOG.1	Component Name	EOG.1	
	Component Type	asterics.EOG	
horizontal	Component Descriptic	connects to EOG CIM and measures horizontal a	
vertical	Component Class	sensor 💌	
	<ul> <li>Properties</li> </ul>		
	updatePeriod	100	
< > >	🖀 Properties (Ctrl-P)	T Output Ports	t
E	OG sensor pluqi	n	L

#### Requirements

The EOG sensor has to be connected to a USB port. The design documentation of the electronic circuit can be found in the thesis pdf. The firmware for the EOG sensor's microcontroller is available in the folder CIMs/EOG\_CIM. The image below shows the breadboard setup of the circuit and a measurement of 6 eye blinks (above) and a left/right eye movement (below).



EOG sensor application

#### **Output Port Description**

- **horizontal [integer]:** the currently measured horizontal eye movement value
- **vertical [integer]:** the currently measured vertical eye movement value

#### **Properties**

• **updatePeriod [integer]:** the update period for values in milliseconds.

## **EShoe**

#### **Component Type: Sensor (Subcategory: Sensor Modules)**

This component provides an interface to the eShoe which is an insole for shoes with sensor inlays, capabale of measuring multiple degrees of freedom (pressure, acceleration and angular velocity) for gait analysis and rehabilitation purposes. See: <u>eShoe-info (german)</u>. The COM Port number for the device must be specified in the plugin properties.

GUI Designer (Ctrl-G)	⇒ ×	Properties (Ctrl-P)		₹ Ф
	^	Component: EShoe.1		
		Search		
EShoe.1		<ul> <li>Internal Properties</li> </ul>		_
		Component Class	sensor	*
InputString AccX		Component Name	EShoe.1	
AccY		Component Type	asterics.EShoe	
AccZ		Component Description	eShoe-Plugin to gather data from	the
Heel		<ul> <li>Properties</li> </ul>		
Manuk		ComPort	1	<b>÷</b>
MetaV		SamplingRate	50	•
Metal		SDMemory		
GyroX				
GyroY				
GyroZ				
Angle 🛑				
	``			
			UU 7 EVe	ES
		plugin		

#### Requirements

This software component requires an EShoe connected via Bluetooth to a COM Port.



Schematic view of the EShoe (left insole), light grey circles mark pressure sensors (big toe (BT), metatarsal head I (M1), metatarsal head V (M5), heel (HE)), the grey box marks the positon of the triaxial acceleration sensor (a) and triaxial gyroscope (ω).
V. David, H. Jagos, S. Litzenberger, and M. Reichel, "Instrumented insole for mobile and long distance motion pattern measurement," Procedia Eng., vol. 34, pp. 760-765, 2012.

#### **Output Port Description**

- **Channel1 to Channel3 [Short]:** Output of the acceleration in X, Y and Z direction. The values range for -1285,81632m/s<sup>2</sup> to 1285,77708m/s<sup>2</sup>.
- **Channel4 to Channel7 [Unsigned Short]:** Output of the pressure sensors of heel, metaV, metal and toe. The unit of the output data is gram.
- **Channel8 to Channel10 [Short]:** Output of the gyroscope in X, Y and Z direction. The values range for -2280,6528°/s to 2280,5832°/s.
- **Channel11 [Short]:** This is the angle between subsurface and insole of the users foot. The values range for -327,68° to 327,67°.

#### **Properties**

- **COMPort [String]:** The name of the COM port, for example COM2 or COM17.
- **SamplingRate [Integer]:** The sampling rate of the eShoe. Could be chosen between 50Hz and 200Hz.
- **SDMemory [Boolean]:** The storage on the SD card in the eShoe can be turned on if checked.

### **Event Generator**

The event generator plugin can be used to periodically send event triggers at a given time interval.

### Requirements

No special hardware or software required

### **Port Description**

No input or output ports available

### **Event Trigger Description**

• **event\_out\_1:** Events will be generated and sent to this port.

#### **Properties**

- **generation\_delay [integer]:** The time interval for generating events in milliseconds.
- event\_payload [string]: A string value which will be sent with the event as a parameter (currently not used by other plugins).

## Eyetracker

#### **Component Type: Sensor (Subcategory: Computer Vision)**

This component provides different computer vision tracking algorithms which can be selected via the "trackingMode" property. The available modes include "blob-tracking", "calibrated eye-tracking" and "calibrated eye-tracking with head pose estimation". The two eye-tracking modes deliver estimations of the x/y-positions where the user is looking on the computer screen which can be used for cursor control (gaze estimation).

#### **Plugin modes**

- **only blob tracking** In this mode, the plugin just outputs the x/y coordinates of a round surface detected in the live camera images. No calibration procedure is needed. The plugins starts immediately with the coordinate output at the ports "x" and "y". A possible application for this mode is tracking of a round marker which could be placed anywhere on the body.
- calibrated eye tracking In this mode, the plugin expects close-up images of an eye, recorded by a head-mounted setup. The suggested hardware configuration is the AsTeRICS headmounted SVM system (see image below). The eye-pupil location is determined and mapped to an estimated position at the computer screen where the person is looking at. For this mode, the plugins needs to be calibrated. First, a rectangular region of interest (ROI) has to be selected by clicking into the live camera image while holding the CTRL/STRG key. The ROI should cover the area where the eyeball is moving when the user is looking in different directions. Then, the calibration sequence can be started by sending an event to the "calibrate" event listener port. During calibration, the cursor is moved to several locations on the screen and the user is supposed to look at these positions without moving the head. The changing of the cursor position is indicated with an acoustic signal. Calibration succeeds if all calibration location can be recorded without excessively distorted values. If the calibration cannot be

accomplished successfully, an acoustic signal is emitted and the Eyetracker plugin does not start to put out x/y data. In this case the calibration procedure has to be repeated. The coordinate-output of the eye tracking is only working correctly if no head movements occur. Any head movement will compromise the correctness of the x/y estimation for the cursor position..

• calibrated eye tracking with head pose estimation This mode is an extension of the "calibrated eye tracking" mode. The calibration has to be performed as above. Additionally to the head-mounted camera, an external LED-frame mounted on the computer monitor is used to minimize the negative effects of head movements to the tracking accuracy. Please note that this mode needs the headmounted SVM device plus the external Led-Frame.



Eyetracker in operation (calibrated eyetracking mode)

#### Requirements

A camera has to be available in the operating system, to detect round blobs or the eye-pupil position. Depending on the mode (if head pose estimation is selected), a head mounted camera which films a close-up of the eye and an external LED-mount are needed.



head mounted SVM device

#### **Input Port Description**

- pt1x pt4x [integer]:
- **pt1y pt4y [integer]:** These 8 input ports can be connected to the corresponding output ports of the Sensorboard plugin. The Sensorboard delivers the location of 4 IR-led tracking points in the field-of-view of the IR-object-tracking camera of the headmounted SVM. This information can be used to compensate the head movement to increase gaze estimation stability. The 8 input ports have to be synchronized (turn the synchronized-property on for every input port). **These 8 input ports support synchronization**

#### **Output Port Description**

- **x [integer]:** The x-coordinate delivered by the tracking algorithm. The meaning of this value depends on the selected mode.
- **y** [integer]: The y-coordinate delivered by the tracking algorithm. The meaning of this value depends on the selected mode.

#### **Event Listener Description**

- **calibrate:** if this event is triggered, calibration procedure for the Eyetracker is started (this is not relevant for the mode "only blob tracking").
- offsetCorrection: this event is useful for eyetracking mode, when a drift of the cursor position has occured. When the event is triggered, the plugin stops the output of x/y coordinates for two seconds, where the user has time to fix the exact cursor position with the eyes. A new offset will be calculated to match the cursor position after the 2 seconds pause.
- **showCameraSettings** an incoming event displays the settings window for the camera device, where parameters like image brightness or contrast can be adjusted.
- togglePoseInfoWindow an incoming event displays the pose info window, where the current location of the 4 IR tracking points for head-pose compensation can be seen. For a normal head orientation these 4 points should be centered in the middle of the window (change the angle of the frontal camera to adjust the position of the yellow dots).
- **startEvaluation** an incoming event displays the accuracy evaluation window and starts evaluation. When the user follows the cross to 9 positions in the window, the x/y coordinates of the cursor are stored to a file in the ARE folder which can then be used to calculate and compare the gaze accuracy.
- **saveProfile** an incoming event saves the camera settings to a file of the given name (property cameraProfile). Use with caution a saved settings-profile does only work with the same camera which was used to save the settings.

#### **Event Trigger Description**

- **blinkDetected:** triggered if the blob detection is lost for a timespan bigger than minBlinkDuration and lower than maxBlinkDuration. This can be used for a single eye-blink detection.
- **longBlinkDetected:** triggered if the blob detection is lost for a timespan bigger than maxBlinkDuration. This can be used for a long eye-blink detection.
## **Properties**

- **cameraSelection [string, combobox selection]:** using this property, the utilized camera can be chosen. Possible values range from "first camera" to "fith camera". If only one camera is available in the system, "first camera" shall be chosen.
- **cameraResolution [string, combobox selection]:** This selection box provides several standard camera resolutions. Changing the resolution affects accuracy and performance (CPU load of the runtime system). Provided selections include "160x120", "320x240", "640x480", "800x600", "1024x768" and "1600x1200". If the selectied resolution cannot be delivered by the image acquisition device, the next matching resolution is chosen by the plugin.
- **cameraProfile [string]** a filename for the camera settings profile to be saved (property cameraProfile). Use with caution a saved settings-profile does only work with the same camera which was used to save the settings.
- **cameraDisplayUpdate [integer]:** This property allows to select the update rate for the camera display in milliseconds. If "0" milliseconds is chosen, no window for the live-video will be displayed. If "100" is chosen, the live image window will be updated 10 times a second. This property does not influence the frame rate of the camera nor the processing interval for new camera frames, only the display in the GUI is adjusted.
- tracking mode [string, combobox selection]: The selection of the plugin's mode of operation ("only blob tracking", "calibrated eye tracking", or "calibrated eye tracking with head pose estimation")
- **xMin [integer]:** the minimum value for the x-coordinate output
- **xMax [integer]:** the maximum value for the x-coordinate output. If "0" is selected, the plugins auto-detects the screen resolution and uses the X-Size of the computer screen.
- **yMin [integer]:** the minimum value for the y-coordinate output
- **yMax [integer]:** the maximum value for the y-coordinate output. If "0" is selected, the plugins auto-detects the screen resolution and uses the Y-Size of the computer screen.
- **calibrationStepsX [integer]:** the number of rows for generating calibration positions

- **calibrationStepsY [integer]:** the number of columns for generating calibration positions. For example: if 4 x-steps and 3 y-steps are chosen, the user has to look at 12 cursor positions during the calibration phase. More positions increase the gaze-tracking accuracy but result in a longer calibration phase.
- **averaging [integer]:** the length of the averaging window for smoothening the ouput values.
- screenSize [double]: the diameter of the computer screen (important if head pose correction is used)
- **minBlinkDuration [integer]:** the minimum time for a short blink to be detected (a short blink is detected if the blink time is bigger than minBlinkDuration and lower than maxBlinkDuration.
- **maxBlinkDuration [integer]:** the maximum time for a short blink to be detected (a long blink is detected if the blink time is bigger than maxBlinkDuration).

# Eyetribe

#### **Component Type: Sensor (Subcategory: Computer Vision)**

This component provides access to the raw gaze tracking data from the <u>EyeTribe</u> eye tracking device. The data includes the estimated gazepoint (x/y), the eye location (x/y), fixation time and eye close time. The plugin connects to a running EyeTribe server.



Eyetribe plugin

## Requirements

The Eyetribe tracker must be connected to an USB3.0 port and the Eyetribe server software must be running.



Eyetribe hardware

#### **Output Port Description**

- **gazex [integer]:** The x-coordinate of the estimated gazepoint on the computer screen (in pixels)
- gazey [integer]: The y-coordinate of the estimated gazepoint on the computer screen (in pixels)
- **posx [integer]:** The x-coordinate of the eye pupil(s) left / right / both pupils can be selected via property
- **posy [integer]:** The y-coordinate of the eye pupil(s) left / right / both pupils can be selected via property
- **fixationTime [integer]:** The time period for fixation of a particular spot on the screen
- **closeTime [integer]:** The time period for closing both eyes (or eye tracking signal lost)

### **Event Listener Description**

- **startCalibration:** if this event is triggered, calibration procedure for the Eyetracker is started.
- offsetCorrection: this event is useful when a drift of the cursor position has occured. When the event is triggered, the plugin stops the output of x/y coordinates for several seconds, where the user has time to fix the exact cursor position with the eyes. (see description below).
- **removeLastOffsetCorrection** when this event is triggered, the last offset correction point will be removed useful if the offset correction did not work as intended.

## **Event Trigger Description**

- **blink:** triggered if the eye detection is lost for a timespan bigger than minBlinkTime and lower than midBlinkTime. This can be used for a single eye-blink detection.
- **longBlink:** triggered if the blob detection is lost for a timespan bigger than midBlinkTime and lower than maxBlinkTime. This can be used for a long eye-blink detection.
- **fixation:** triggered if the user looks at a particular location on the screen for longer than defined in via the fixationTime property.
- **fixationEnd:** triggered if the user looks stops looking at a particular location (which triggered a fixation event).

## **Properties**

- **minBlinkTime [integer]:** the minimum time for a short blink to be detected (a short blink is detected if the blink time is bigger than minBlinkTime and lower than midBlinkTime.)
- **midBlinkTime [integer]:** the time for seperating short blinks from long blinks.
- **maxBlinkTime [integer]:** the maximum time for a short blink to be detected (a long blink is detected if the blink time is bigger than midBlinkTime and lower than maxBlinkTime..
- **fixationTime [integer]:** the minimum time for triggereing a fixation event when the user looks at a particular spot on the screen
- **pupilPositionMode [combobox]:** selects the mode for calculation of the eye position output port values. (left eye / right eye or an average of both eyes)
- offsetCorrectionRadius [integer]: defines the distance to an offset correction point where this offset correction points starts to influece the eye coordinates (see description below)
- offsetCorrectionMode [combobox]: selects the mode for the offset correction measurement. (manual or automatic mode, see above description)

#### **Calibration and Offset Correction**

- A sucessful gaze estimation needs prior calibration. This should be done using the application/GUI provided by the EyeTribe software. However, a calibration can also be initiated using the provided startCalibration EventListener port (see below). When starting the calibration process using this event, the mouse cursor is positioned to 9 screen locations, starting at the left upper corner and iterating via middle and right position through 3 rows until the right bottom corner is reached. During calibration, follow the cursor with the eyes (reducing head movements to a minimum). For each calibration position, an acoustic signal indicates the time when the samples are taken. When the calibration process is finished, the plugin continues sending the measurement data from it's output ports.
- The offset correction event listener port allows setting so called

"offset correction points" at desired screen locations. This is possible during the normal operation of the eye tracking. This is useful when certain locations on the screen cannot be reached because of a calibration problem but a new calibraiton is either not desired or not successful / precise enough. The goal of the offset correction is to reduce the error between the real gaze point and the estimated (weak) gaze point where usually the cursor is positioned, by adding a small offset value. After starting the offset correction (which is indicated by an acoustic signal) look at the intended spot on the screen. After 1 second, another acoustic signal indicates that the coordinates have been saved. Now look at the mouse cursor (the weak gaze point which shall be corrected). The next step differs according to the selected offset correction mode (which can be chosen via the **offsetCorrectionMode** property:

- manual offset correction: try to follow the cursor with the eyes. It should move slowly towards the original gaze point, correcting the error. This manual calibration phase takes about 4 seconds.
- automatic offset correction: look at the cursor. After 1 second, the offest is measured.

After the offset correction has been performed, the given offset value is stored into an internal list. As soon as the estimated gaze point enters an area around the offset correction point (the area size is defined via the property offsetCorrectionRadius), an appropriate fraction of the offset value is added to the estimated gaze point coordinates. Here, linear approximation is used, so that looking at the original (weak) gaze estimation point will add the full offest value, resulting the corrected gaze estimation point.

# **Facetracker CLM**

#### **Component Type: Sensor (Subcategory: Computer Vision)**

The FacetrackerCLM component is one of the vision based plugin. At this stage of development it offers the same functionalities for mouse emulation as the other sensor plugin named <u>facetrackerLK</u>.

The underlying mechanism is based on the tracking of facial features detected in a first initialization stage by fitting a deformable face model on the image as soon as the region of interest is identified thanks to the OpenCV implementation of the Viola-Jones classifier.

The plugin outputs at each frame the relative offsets of a series of measures based on the tracked points with respect to the previous frame. These offsets may be integrated or used directly as inputs for the mouse emulator actuators. Usually it is a good idea to use the relative displacement of the centre of the face (PosX and PosY) to guide the mouse movements and reserve the other measurements or events to implement other optional functionalities (the head roll, pitch and yaw angles and relative scale of the face).

The FacetrackerCLM also introduces the detection of facial gestures that can be then exploited in the ACS models to directly perform actions or trigger events. To this purpose in this version there are two distinct event related outputs:

• Detection of the raising of both eyebrows.

For a closer explaination of the EyeTrackerCLM functions and the training algorithm for eye states please refer to the document FacetrackerCLM.pdf in the folder /documentation.

	X         System         Compensation         Set         Mic.         Set	Start Model Ruser Model Stop Model Stop Model Mage Stare Model of
	Aff the set of the set	Math         Louil           Pata         * 0           Pata         * 0 </td
Provide State Synthematical State Carteria	# Pro_ / Sve / Sve	

FacetrackerCLM plugin

FacetrackerCLM plugin

# Requirements

A camera has to be available in the operating system (preferably a consumer USB camera).

# **Input Port Description**

There are no input ports for this plugin.

#### **Output Port Description**

- **Roll[double]:** The output port Roll outputs the relative change of the roll angle of the head pose (degrees).
- **Pitch[double]:** The output port Pitch outputs the relative change of the pitch angle of the head pose (degrees).
- Yaw[double]: The output port Yaw outputs the relative change of the yaw angle of the head pose (degrees).
- **PosX[double]:** The output port PosX outputs the relative displacement of the x coordinate (image coordinates) of the tracked point (approximatively located around the nose).
- **PosY[double]:** The output port PosX outputs the relative displacement of the y coordinate (image coordinates) of the tracked point (approximatively located around the nose).
- **Scale[double]:** The output port Scale outputs the relative change in scale of the apparent size of the fitted face model in the current image.
- **EyeLeft[int]:** The output port EyeLeft outputs 0 if the left eye is opened, 1 if closed in the current image.
- **EyeRight[int]:** The output port EyeRight outputs 0 if the right eye is opened, 1 if closed in the current image.

### **Event Listener Description**

- **reset:** Forces reinit of the fitting of deformable model for the face in order to reset the tracking points.
- showCameraSettings: Displays the camera settings dialog on screen
- setReferencePose: sets the reference pose

## **Event Trigger Description**

• **EyebrowsRaised:** this event gets raised everytime the plugin detects a specific configuration of the facial landmarks corresponding to a *surprise* expression.

#### **Properties**

- cameraSelection [string, combobox selection]: this property determines the index of the input camera. Possible values range from "first camera" to "fifth camera". If only one camera is available in the system, "first camera" is the correct choice.
- cameraResolution [string, combobox selection]: this selection box provides several standard camera resolutions. Changing the resolution affects accuracy and performance (CPU load of the runtime system). Provided selections include "160x120", "320x240", "640x480" and "800x600". If the selected resolution cannot be delivered by the image acquisition device, the closest matching resolution is chosen automatically by the plugin.
- **cameraDisplayUpdate [integer]:** this property allows to select the update rate for the camera display in milliseconds. If "0" milliseconds is chosen, no window for the live-video will be displayed. If "100" is chosen, the live image window will be updated 10 times a second. Please note that this property does not influence the frame rate of the camera nor the processing interval for new camera frames, only the display in the GUI is adjusted.
- **modelName [string]:** this property informs the plugin about which is the trained model to load. The file is searched in the the FacetrackerCLM/EyeStateModels folder inside the plugin "data" folder. Specify the name without .yml extension.

# EyeX

#### **Component Type: Sensor (Subcategory: Computer Vision)**

This component provides access to the raw gaze tracking data from the <u>Tobii EyeX</u> eye tracking device. The data includes the estimated gazepoint (x/y), the eye location (x/y), fixation time and eye close time. The plugin connects to the deveice via the Tobii EyeX SDK.



EyeX plugin

Tobii

## Requirements

The Tobii EyeX tracker must be connected to an USB3.0 port and the Tobii EyeX server software must be running.



Tobii EyeX hardware

#### **Output Port Description**

- gazeX [integer]: The x-coordinate of the estimated gazepoint on the computer screen (in pixels)
- gazeY [integer]: The y-coordinate of the estimated gazepoint on the computer screen (in pixels)
- **posX [integer]:** The x-coordinate of the eye pupil(s) left / right / both pupils can be selected via property
- **posY [integer]:** The y-coordinate of the eye pupil(s) left / right / both pupils can be selected via property
- **fixationTime [integer]:** The time period for fixation of a particular spot on the screen
- **closeTime [integer]:** The time period for closing both eyes (or eye tracking signal lost)

### **Event Listener Description**

- offsetCorrection: when offset correction mode "CorrectionSpots" is selecte, an new correction spot can be added via this event. When the event is triggered, the plugin stops the output of x/y coordinates for several seconds, where the user has time to fix the exact cursor position with the eyes, thereby defining the x/y correction offset.
- **removeLastOffsetCorrection** when this event is triggered, the last offset correction spot will be removed useful if the offset correction did not work as intended.
- **stopOffsetCorrection** when this event is triggered, any offset correction mode is stopped.
- **createAndCalibrateGuestProfile** creates a new guest profiles and initiates calibration.
- **calibrateCurrentProfile** initiates calibration of the currently active profile.
- **switchToOffsetCorrectionSpots** selects uses multiple correction spot which can be added via envet "offsetCorrection".
- **switchToPermanentOffsetCorrection** permanently adds the input port values of xOffset and yOffset to the gazepoint.
- **switchToCombinedOffsetCorrection** uses the xOffset / yOffset input port values to fine-tune the gazePoint (this switches off gaze-updates until a certain gaze offset is detected.)
- activate activates the eye tracker.
- **deactivate** deactivates the eye tracker.

## **Event Trigger Description**

- **blink:** triggered if the eye detection is lost for a timespan bigger than minBlinkTime and lower than midBlinkTime. This can be used for a single eye-blink detection.
- **longBlink:** triggered if the blob detection is lost for a timespan bigger than midBlinkTime and lower than maxBlinkTime. This can be used for a long eye-blink detection.
- **fixation:** triggered if the user looks at a particular location on the screen for longer than defined in via the fixationTime property.
- **fixationEnd:** triggered if the user looks stops looking at a particular location (which triggered a fixation event).

## **Properties**

- **enabled [boolean]:** selects if the eyetracker is enabled on startup of the model.
- **averaging [integer]:** selects the size of an internal averager for the gaze point x/y values.
- **minBlinkTime [integer]:** the minimum time for a short blink to be detected (a short blink is detected if the blink time is bigger than minBlinkTime and lower than midBlinkTime.)
- **midBlinkTime [integer]:** the time for seperating short blinks from long blinks.
- **maxBlinkTime [integer]:** the maximum time for a short blink to be detected (a long blink is detected if the blink time is bigger than midBlinkTime and lower than maxBlinkTime..
- **fixationTime [integer]:** the minimum time for triggereing a fixation event when the user looks at a particular spot on the screen
- **pupilPositionMode [combobox]:** selects the mode for calculation of the eye position output port values. (left eye / right eye or an average of both eyes)
- offsetCorrectionRadius [integer]: defines the distance to an offset correction point where this offset correction points starts to influece the eye coordinates (see description below)
- offsetCorrectionMode [combobox]: selects the mode for the offset correction. (correction spots, permanent offset correction or combined correction)

#### **Calibration and Offset Correction**

- A sucessful gaze estimation needs prior calibration. This should be done using the application/GUI provided by the Tobii EyeX software, but can also be accomplished by the provided event listeners.
- The offset correction modes allow application of drift correction during normal operation of the eye tracking. This is useful when certain locations on the screen cannot be reached because of a calibration problem but a new calibration is either not desired or not successful / precise enough. The goal of the offset correction is to reduce the error between the real gaze point and the estimated

(weak) gaze point where usually the cursor is positioned, by adding small offset values.

There are three offset correction modes provided, which can be chosen via the **offsetCorrectionMode** property or the respective events:

• offset correction spots

when a new spot is set via the provided event listener, eye tracking is stopped for several seconds. Look at the cursor. After 1 second, the offest is measured. When the tracking starts again, the measured offset is applied in the area of the selected offsetCorrectionRadius. Here, linear approximation is used, so that looking at the original (weak) gaze estimation point will add the full offest value, resulting the corrected gaze estimation point. Multiple spots can be added at different screen loactions.

 permanent offset correction: the correction values for x and y which are provided via the input ports xOffset and yOffset are permanenty added to the gazepoint location. A joystick, mouthmouse or similar input device can be used to provide these values.

 combined offset correction: if the correction values for x and y which provided via the input ports xOffset and yOffset are not zero, the eyetracking stops and the correction values are applied to the last gazepoint. Eye-tracking starts again when the measured gazepoint differs from the corrected gazepoint by more than the selected offsetCorrectionRadius.

# **Facetracker LK**

#### **Component Type: Sensor (Subcategory: Computer Vision)**

This component provides a face tracking computer vision algorithm which puts out estimated movement of a users' nose and chin in x and y coordinates. The underlying mechanism builds upon the OpenCV library (in particular a trained cascade of haar-like features and an optical flow algorithm). The x- and y- coordinates can be used in camera-mouse configurations or to enable selection or control tasks. The values are only provided if a face can be tracked by the algorithm and are updated with the achievable frame rate. Note that the x- and y- values represent relative movement in pixels and have to be accumulated (e.g. via the integrate component) to generate e.g. absolute mouse positions.





FacetrackerLK plugin

## Requirements

A camera has to be available (this can be any webcam or a camera which is available as image acquisition device via the operating system). The picture below shows the Logitech Webcam 9000 Pro.



Logitech Webcam 9000 Pro

#### **Output Port Description**

- **noseX [integer]:** This value specifies the relative change in the x coordinate of the user's nose with respect to the previous image frame.
- **noseY [integer]:** This value specifies the relative change in the y coordinate of the user's nose with respect to the previous image frame.
- **chinX [integer]:** This value specifies the relative change in the x coordinate of the user's chin with respect to the previous image frame.
- **chinY [integer]:** This value specifies the relative change in the y coordinate of the user's chin with respect to the previous image frame.

#### **Event Listener Description**

- **init:** if this event is triggered, the face recognition procedure is initiated. This can be useful if the correct face position has been lost due to drifting of the LK algorithm.
- **showCameraSettings** an incoming event displays the settings window for the camera device, where parameters like image brightness or contrast can be adjusted.
- **saveProfile:** if this event is triggered, the camera profile is saved.

### **Properties**

- **cameraSelection [string, combobox selection]:** using this property, the utilized camera can be chosen. Possible values range from "first camera" to "fith camera". If only one camera is available in the system, "first camera" shall be chosen.
- **cameraResolution [string, combobox selection]:** This selection box provides several standard camera resolutions. Changing the resolution affects accuracy and performance (CPU load of the runtime system). Provided selections include "160x120", "320x240", "640x480", "800x600", "1024x768" and "1600x1200". If the selectied resolution cannot be delivered by the image acquisition device, the next matching resolution is chosen by the plugin.
- **cameraDisplayUpdate [integer]:** This property allows to select the update rate for the camera display in milliseconds. If "0" milliseconds is chosen, no window for the live-video will be displayed. If "100" is chosen, the live image window will be updated 10 times a second. Please note that this property does not influence the frame rate of the camera nor the processing interval for new camera frames, only the display in the GUI is adjusted.
- **cameraProfile [string]:** Filename of the camera profile.

# **FS20 Receiver**

#### **Component Type: Sensor (Subcategory: Home Control)**

The FS20Receiver receives commands of the home automation system FS20 for ELV Electronics. Depending on the received commands, events will be fired.

Deployment (Ctrl-D) GUI Designer (Ctrl-G) 🗢	Properties (Ctrl-P)	<b>~</b> џ	L
A	Component: FS20Recei	iver.1	
	Search		
	<ul> <li>Internal Properties</li> </ul>		
FS20Receiver.1	Component Descriptio	Receiver Plugin for the FS20 Home aut	
	Component Type	asterics.FS20Receiver	
fs20command	Component Name	FS20Receiver.1	
	Component Class	sensor 💌	
	<ul> <li>Properties</li> </ul>		
	Housecode	[11111111 🚖	
	Sendaddress	[1111 😤	
•		-	
	Troperties (Ctrl-P)	TOutput Ports 📝 Event Trigger	FS2

Receiver Plugin

## Requirements

This component requires the FS20 PCE (see <u>http://www.elv.de/FS20-PC-Empfauml;nger-FS20-PCE/x.aspx/cid\_74/detail\_10/detail2\_31219</u>) connected to an USB port.



FS20 PCE Receiver

#### **Output Port Description**

• **fs20command [string]:** The received FS20 command will be send out of the plugin. The data has the following format: housecode\_sendaddress\_command, e.g. 1111111\_3343\_17

## **Event Trigger Description**

Each received command triggers an event, being mapped to this command. The following table will describe this events:

Event	Command
Off	0
Level1	1
Level2	2
Level3	3
Level4	4
Level5	5
Level6	6
Level7	7
Level8	8
Level9	9
Level10	10
Level11	11
Level12	12
Level13	13
Level14	14
Level15	15
Level16	16
OnOldLevel	17
Toggle	18
Dim Up	19
Dim Down	20
Dim Up and Down	21
Program internal timer	22
Off for timer then old brightness level	24

Command Mapping

On for timer then off	25
On old brightness level for timer then off	26
On for timer then old brightness level	30
On for old level then previous state	31

## **Properties**

- **housecode [integer]** The housecode, the system should react on. The housecode has 8 digits, each from 1 to 4. This property influences the event triggers, but not the *fs20command* output port.
- **sendaddress [integer]** The sendaddress, the system should react on. The sendaddress has 4 digits, each from 1 to 4. This property influences the event triggers, but not the *fs20command* output port.

# **HeadPositionHC**

#### **Component Type: Sensor (Subcategory:Computer Vision)**

This plug-in provides a selection of up to 12 choices via the head. A choice can be selected through looking to the side. The angle of the head is calculated with the position of the ear and the mouth. The detection of the features is realised with HaarCascade.

Cor	nponent: HeadPosit	ionHC.1	
	Al Search		
dPositionHC.	Z* Bearen		
	nternal Properties	Used Devision U.C.1	
	mponent Name	HeadPositionHC.1	110
Number	mponent Type	asterics.HeadPositionHC	
60	mponent Descriptio	Picking between choi	ces w
Co	mponent Class	sensor	*
	Properties		1000
	oicesEachSide	4	-
An	gle1LeftSide	160	-
An	gle2LeftSide	120	-
An	gle3LeftSide	100	÷
An	gle4LeftSide	65	\$
An	gle5LeftSide	-100000	\$
An	gle6LeftSide	100000	-
An	gle1RightSide	20	\$
An	gle2RightSide	60	\$
An	gle3RightSide	80	\$
An	gle4RightSide	110	\$
An	gle5RightSide	100000	\$
An	gle6RightSide	-100000	\$
Pa	PathForHaarCascade C:/HaarCascade/		
Ca	meralD	0	\$
Co	unterResettingROI	10	-

HeadPositionHC sensor plug-in

the
### Requirements

A webcam or an other camera has to be connected to the computer.

### **Output Port Description**

• **CellNumber:** the number of the current selected Choice

## **Event Triggerer Description**

• **Select:** will be triggered when the position is held for an amount of time

#### **Properties**

- **ChoiceEachSide [integer]:** sets the choices, which are visible and selectable for each side. (1-6 Choices each side are possible)
- **Angle1LeftSide [integer]:** sets the upper value of the angle for the first choice on the left side. Every value lower than this value and higher than the value of Angle2LeftSide selects the first choice of the left side.
- **Angle2LeftSide [integer]:** sets the upper value of the angle for the second choice on the left side. Every value lower than this value and higher than the value of Angle3LeftSide selects the second choice of the left side.
- **Angle3LeftSide [integer]:** sets the upper value of the angle for the third choice on the left side. Every value lower than this value and higher than the value of Angle4LeftSide selects the third choice of the left side.
- **Angle4LeftSide [integer]:** sets the upper value of the angle for the fourth choice on the left side. Every value lower than this value and higher than the value of Angle5LeftSide selects the fourth choice of the left side.
- **Angle5LeftSide [integer]:** sets the upper value of the angle for the fifth choice on the left side. Every value lower than this value and higher than the value of Angle6LeftSide selects the fifth choice of the left side.
- **Angle6LeftSide [integer]:** sets the upper value of the angle for the sixth choice on the left side. Every value lower than this value selects the sixth choice of the left side.
- **Angle1RightSide [integer]:** sets the lower value of the angle for the first choice on the right side. Every value higher than this value and lower than the value of Angle2RightSide selects the first choice of the right side.
- Angle2RightSide [integer]: sets the lower value of the angle for the second choice on the right side. Every value higher than this value and lower than the value of Angle3RightSide selects the first choice of the right side.
- Angle3RightSide [integer]: sets the lower value of the angle for the third choice on the right side. Every value higher than this value and

lower than the value of Angle4RightSide selects the first choice of the right side.

- **Angle4RightSide [integer]:** sets the lower value of the angle for the fourth choice on the right side. Every value higher than this value and lower than the value of Angle5RightSide selects the first choice of the right side.
- **Angle5RightSide [integer]:** sets the lower value of the angle for the fifth choice on the right side. Every value higher than this value and lower than the value of Angle6RightSide selects the first choice of the right side.
- **Angle6RightSide [integer]:** sets the lower value of the angle for the sixth choice on the right side. Every value higher than this value selects the first choice of the right side.
- **PathForHaarCascade [string]:** Filepath to the folder, where the HaarCascade-files are stored.
- CameralD [integer]: the ID of the Camera.
- **CounterResettingROI [integer]:** the number of frames to wait before resetting the ROI (finding the face) when no feature is found.
- **CounterToSendSelectEvent [integer]:** the number of frames, which the person has to hold the position until the select event is triggered.

## HoverPanel

# **Component Type: Sensor (Subcategory: Graphical User Interface)**

This component creates a panel window with selectable opacity which can be placed anywhere on the screen/desktop. No window decoration is used. The plugin provides x/y input port for coordinates and the panel reacts if these coordinates are located within the panel. Via a selectable hover time, events can be triggered if the coordinates stay within the panel for a given time. Note that the hoverPanel represents a seperate window and is not a child window of the ARE window.

Deployment (Ctrl-D)	Properties (Ctrl-P)	<b>~</b> ↓
/	Component: HoverPa	nel.1
	Search	
	<ul> <li>Internal Properties</li> </ul>	
	Component Class	sensor 💌
	Component Name	HoverPanel.1
HoverPanel.1	Component Type	asterics.HoverPanel
	Component Descripti	c The HoverPanel allows interaction
↓ ×	<ul> <li>Properties</li> </ul>	
Фу	dataSource	mouseCursor 🔹
	caption	hover
	fontSize	14
	textColor	lightgray 💌
	backgroundColor	black 💌
	activationColor	cyan 💌
	dwellTime	200
	idleTime	50
	opacity	50
	stayActive	
×		
	🔄 📑 Prop 📑 Inpu	🔰 Even   🔰 Even

HoverPanel plugin



2 HoverPanels on screen

### **Input Port Description**

- x [integer]: x coordinate
  y [integer]: y coordinate

### **Event Listener Description**

- **activate:** activates the hover panel.
- **deactivate:** deactivates the hover panel.

#### **Event Trigger Description**

- **selected:** triggered if coordinates are in range of the panel position for the hover time period.
- enter: triggered if coordinates enter the range/location of the panel.
- exit: triggered if coordinates exit the range/location of the panel.

#### **Properties**

- dataSource [integer, combobox selection]: If "mouse coursor" is selected, the coordinates are grabbed from the current position of the mouse cursor. If "coordinateInputs" is selected, the input ports (x/y) are used.
- **caption [string]:** The caption of the hover panel.
- fontSize [integer]: The size of the caption text.
- textColor [integer]: Defines color of the caption text.
- backgroundColor [integer]: Defines background color.
- activationColor [integer]: Defines color for the hover selection process.
- dwellTime [integer]: The hover time for a selection.
- **idleTime [integer]:** The time how long the panel will stay inactive after a selection.
- **opacity [integer]:** the opacity of the hver panel (0-100%).
- **stayActive [boolean]:** If selected, the hover panel will remain in active state after a selection (deactivation via event).

# **Joystick Capture**

# **Component Type: Sensor (Subcategory: Standard Input Devices)**

The JoystickCapture component provides data from the first detected Joystick controller which is available on the local system. Six Joystick analog positions (axis), the Point-Of-View angle and up to 20 Buttons are available as output ports and event triggers.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Event Trigger		-
	*	Triggers	Description	ſ
	=	pressedButton1	button 1 (Triangle) has been pressed	
		releasedButton1	button 1 (Triangle) has been released	
JoystickCapture.		pressedButton2	button 2 (Circle) has been pressed	
		releasedButton2	button 2 (Circle) has been released	_
×		pressedButton3	button 3 (Cross) has been pressed	
y 🛑		releasedButton3	button 3 (Cross) has been released	
z		pressedButton4	button 4 (Square) has been pressed	
r		releasedButton4	button 4 (Square) has been released	
		pressedButton5	button 5 (L1) has been pressed	
		releasedButton5	button 5 (L1) has been released	
		pressedButton6	button 6 (R1) has been pressed	
		releasedButton6	button 6 (R1) has been released	
		pressedButton7	button 7 (L2) has been pressed	
		releasedButton7	button 7 (L2) has been released	
	-	4		Þ.
<u>, m.</u>	•	Properties (Ctrl-P)	Toutput Ports 💉 Event Trigger	

JoystickCapture plugin

### Requirements

A joystick has to be connected and available to the operating system.



PC compatible GamePad

#### **Output Port Description**

- x [integer]: The X-Position of the Joystick.
- **y** [integer]: The Y-Position of the Joystick.
- z [integer]: The Z-Position of the Joystick.
- r [integer]: The R-Position of the Joystick.
- **u [integer]:** The U-Position of the Joystick.
- v [integer]: The V-Position of the Joystick.
- pov [integer]: The POV-angle of the Joystick.

#### **Event Trigger Description**

- pressedButton1 pressedButton20: This event is fired if the corresponding button of the Joystick has been pressed.
- releasedButton1 releasedButton20: This event is fired if the corresponding button of the Joystick has been released.

#### **Properties**

• **updatePeriod [integer]:** This property value sets the update time for refreshing the joystick information (in milliseconds).

## **Key Capture**

# **Component Type: Sensor (Subcategory: Standard Input Devices)**

This component provides access to keystrokes of a single key on the keyboard. The keystroke capture does not depend on a particular window or text field to have the input focus. For every keypress and release of the specified key an event gets fired. Possible applications include triggering functions by keyboard input, interfacing to speech recognition software or remapping keys to other keycodes.

<b>=</b>	Properties (Ctrl-P)		<b>•</b> q		
	Component: KeyCaptu	re.1			
	Search				
	<ul> <li>Internal Properties</li> </ul>				
	Component Name	KeyCapture.1			
	Component Type	asterics.KeyCapture			
	Component Descriptio	captures key presses and releases for a single keyboard key			
	Component Class	sensor	*		
	<ul> <li>Properties</li> </ul>		-		
	keyCode	0	۲		
KeyCapture.1 =	block				
keyCode					

KeybCapture plugin

#### Requirements

A keyboard which generates keystrokes or a software component which injects keystrokes into the operating system message queue.

### Mapping of keyCodes to actual keys

Кеу	KeyCode
A	30
В	48
С	46
D	32
E	18
F	33
G	34
Н	35
	23
J	36
K	37
L	38
M	50
N	49
0	24
Р	25
Q	16
R	19
S	31
Т	20
U	22
V	47
W	17
Х	45
Y	21
Z	44
0	11

	1	2	
	2	3	
	3	4	
	4	5	
	5	6	
	6	7	
	7	8	
	8	9	
	9	10	
	F1	59	
	F2	60	
	F3	61	
	F4	62	
	F5	63	
	F6	64	
	F7	65	
	F8	66	
	F9	67	
	F10	68	
	F11	87	
	F12	88	
	F13	91	
	F14	92	
	F15	93	
	F16	99	
	F17	100	
	F18	101	
	F19	102	
	F20	103	
	F21	104	
	F22	105	
I	r	· · · · · · · · · · · · · · · · · · ·	I

F23	106
F24	107
ESC	1
-	12
=/+	13
Backspace	14
Tab	15
Caps Lock	58
Cursor Up	57416
Cursor Left	57419
Cursor Right	57421
Cursor Down	57424
NumPad 0	82
NumPad 1	79
NumPad 2	80
NumPad 3	81
NumPad 4	75
NumPad 5	76
NumPad 6	77
NumPad 7	71
NumPad 8	72
NumPad 9	73
Ctrl left	29
Ctrl right	3613
Alt left	56
Alt right	3640
Context Menu	3677
Windows Key	3675 or 3676
SPACE	57
Print Screen	3639
Scroll Lock	70

Pause	3653
Enter	28
Page Up	3657
Page Down	3665
,	51
	52

The full list of keycodes is defined here: https://github.com/kwhat/jnativehook/blob/master/src/java/org/jnativehook/

#### **Input Port Description**

• **keyCode [integer]:** The keycode of the key which should be captured. For a list of the mapping between keycodes and keys see the graphics above.

#### **Event Trigger Description**

- **keyPressed:** This event port fires an event if the key with the specified keyCode was pressed.
- **keyReleased:** This event port fires an event if the key with the specified keyCode was released.

#### **Event Listener Description**

- **enable:** The keystate is captured and events get fired
- **disable:** The plugin does not fire any event.
- **block:** After this event, the keystroke of the key with the specified keyCode is not forwarded to the operating system.
- **unblock:** After this event, the keystroke of the key with the specified keyCode is forwarded to the operating system.

#### **Properties**

- **block [boolean]:** If this property is set to true, no key press activities will be routed to the operating system
- **keyCode[integer]:**The keycode of the key which should be captured. For a list of the mapping between keycodes and keys see the graphics above.

## **Keyboard Capture**

# **Component Type: Sensor (Subcategory: Standard Input Devices)**

This component provides access to keystrokes input via a standard keyboard. The keystroke capture does not depend on a particular window or text field to have the input focus. The keycodes of the pressed keys and accumulated words (multiple keys separated by ) are provided at the output ports of this component. Possible applications include triggering functions by keyboard input, interfacing to speech recognition software or remapping keys to other keycodes.



KeyboardCapture plugin

#### Requirements

A keyboard which generates keystrokes or a software component which injects keystrokes into the operating system message queue.

#### **Output Port Description**

- **keyCode [integer]:** This port sends the virtual keycode of the last pressed key as an integer value.
- **words [string]:** This ports accumulates keystrokes and sends them as a string as soon as a blank separator (space key) appears in the key input stream.

#### **Event Trigger Description**

• **recognizedCommand1 - recognizedCommand10:** These event ports fire an event if one of seven command strings has been detected in the current input stream of keys.

#### **Properties**

- **blockEvents [boolean]:** If this property is set to true, no key press activities will be routed to the operating system they will not be processed by other applications and disappear. If the property value is set to false, keystrokes will be passed back to the operating system and processed as usual.
- **command1 [string] command10 [string]:** Seven string properties to specify command strings. The component looks for these command strings in the input stream of keystrokes. If a command string matches, the corresponding event trigger port is raised. This can be useful for example to define voice commands which should trigger certain actions in other ARE plugins.

# Kinect

# **Component Type: Sensor (Subcategory: Standard Input Devices)**

The Kinect component interfaces to the Microsoft Kinect camera over the OpenNI library. It provides the skeleton Data of exactly one person in front of the kinect camera.

For the installation of the OpenNI framework and the drivers see for example <u>here</u>.

	Properties (Ctrl-P)		₹ д
*	Component: kinect.1		
	Search		
cinect.1	<ul> <li>Internal Properties</li> </ul>		
	ComponentType	sensor	*
HeadX	Component Descriptio	My Plugin description	
HeadY	Component Name	kinect.1	
HeadZ	<ul> <li>Properties</li> </ul>		
leftHandX 📻	visualize	×	
leftHandY	centerZeroPoint		
rightFootX			
leftFootY			
leftFootZ			

#### Requirements

- A Microsoft Kinect must be connected to the system.
- The installed drivers must be compatible to the framework. e.g. The SensorKinect drivers work perfectly.
- The OpenNI Framework must be installed.
- The Nite Middleware must be installed

#### **Output Port Description**

The range of the coordinates is 0 - 640 for the x coordinates and 0 - 480 for the y coordinates if centerZeroPoint is set false and -320 - 320 for the x coordinate and -240 - 240 for the y coordinate when set to true.

- HeadX [double]: The X position of the Head
- HeadY [double]: The Y position of the Head
- HeadZ [double]: The Z position of the Head
- leftHandX [double]: The X position of the left Hand
- leftHandY [double]: The Y position of the left Hand
- leftHandZ [double]: The Z position of the left Hand
- rightHandX [double]: The X position of the right Hand
- rightHandY [double]: The Y position of the right Hand
- rightHandZ [double]: The Z position of the right Hand
- **rightFootX [double]:** The X position of the right Foot
- **rightFootY [double]:** The Y position of the right Foot
- rightFootZ [double]: The Z position of the right Foot
- leftFootX [double]: The X position of the left Foot
- leftFootY [double]: The Y position of the left Foot
- leftFootZ [double]: The Z position of the left Foot

#### **Properties**

- **visualize [boolean]:** When set to true a windows is displayed which shows the tracked skeleton. While the window is simply grey, no skeleton has been found yet.
- centerZeroPoint [boolean]: Specifies wether the 0,0 point is at the top left corner (centerZeroPoint=false) or in the middle of the frame (centerZeroPoint=true)

# KinectJ4K

# **Component Type: Sensor (Subcategory: Standard Input Devices)**

The Kinect component interfaces to the Microsoft Kinect v1 camera over the <u>J4K library</u>. It provides the skeleton data of exactly one person in front of the kinect v1 camera.

For the installation the SDK from Microsoft is needed. The plugin was tested with the SDK-Version 1.5 (see <u>here</u>).


KinectJ4K plugin

## Requirements

- A Microsoft Kinect v1 must be connected to the system.
- The drivers from Microsoft (usually installed with the SDK) are required.

### **Output Port Description**

You can find more information about the Coordinate Spaces here.

- FootLeftX [double]: The X position of the left Foot
- FootLeftY [double]: The Y position of the left Foot
- FootLeftZ [double]: The Z position of the left Foot
- FootRightX [double]: The X position of the right Foot
- FootRightY [double]: The Y position of the right Foot
- FootRightZ [double]: The Z position of the right Foot
- AnkleLeftX [double]: The X position of the left Ankle
- AnkleLeftY [double]: The Y position of the left Ankle
   AnkleLeftZ [double]: The Z position of the left Ankle
- AnkleRightX [double]: The X position of the right Ankle
- AnkleRightY [double]: The Y position of the right Ankle
- AnkleRightZ [double]: The Z position of the right Ankle
- KneeLeftX [double]: The X position of the left Knee
- KneeLeftY [double]: The Y position of the left Knee
- KneeLeftZ [double]: The Z position of the left Knee
- KneeRightX [double]: The X position of the right Knee
- KneeRightY [double]: The Y position of the right Knee
- KneeRightZ [double]: The Z position of the right Knee
- HipLeftX [double]: The X position of the left side of the Hip
- **HipLeftY** [double]: The Y position of the left side of the Hip
- HipLeftZ [double]: The Z position of the left side of the Hip
- HipCenterX [double]: The X position of the center of the Hip
- HipCenterY [double]: The Y position of the center of the Hip
- HipCenterZ [double]: The Z position of the center of the Hip
- **HipRightX [double]:** The X position of the right side of the Hip
- HipRightY [double]: The Y position of the right side of the Hip
- HipRightZ [double]: The Z position of the right side of the Hip
- SpineX [double]: The X position of the Spine
- SpineY [double]: The Y position of the Spine
- SpineZ [double]: The Z position of the Spine
- ShoulderLeftX [double]: The X position of the left Shoulder
- ShoulderLeftY [double]: The Y position of the left Shoulder
- ShoulderLeftZ [double]: The Z position of the left Shoulder
- **ShoulderCenterX [double]:** The X position of the area between

the Shoulders

- **ShoulderCenterY [double]:** The Y position of the area between the Shoulders
- **ShoulderCenterZ [double]:** The Z position of the area between the Shoulders
- **ShoulderRightX [double]:** The X position of the right Shoulder
- **ShoulderRightY [double]:** The Y position of the right Shoulder
- ShoulderRightZ [double]: The Z position of the right Shoulder
- ElbowLeftX [double]: The X position of the left Elbow
- ElbowLeftY [double]: The Y position of the left Elbow
- **ElbowLeftZ [double]:** The Z position of the left Elbow
- ElbowRightX [double]: The X position of the right Elbow
- ElbowRightY [double]: The Y position of the right Elbow
- ElbowRightZ [double]: The Z position of the right Elbow
- WristLeftX [double]: The X position of the left Wrist
- WristLeftY [double]: The Y position of the left Wrist
- WristLeftZ [double]: The Z position of the left Wrist
- WristRightX [double]: The X position of the right Wrist
- WristRightY [double]: The Y position of the right Wrist
- WristRightZ [double]: The Z position of the right Wrist
- HandLeftX [double]: The X position of the left Hand
- HandLeftY [double]: The Y position of the left Hand
- HandLeftZ [double]: The Z position of the left Hand
- HandRightX [double]: The X position of the right Hand
- HandRightY [double]: The Y position of the right Hand
- HandRightZ [double]: The Z position of the right Hand
- HeadX [double]: The X position of the Head
- HeadY [double]: The Y position of the Head
- HeadZ [double]: The Z position of the Head

## Legacy Analog In

# **Component Type: Sensor (Subcategory: Generic Control Input)**

The LegacyAnalogIn component is a sensor which is used in combination with the legacy ADC CIM (CIM Id: 0x0401). The component provides eight output ports which correlate to the inputs IN1 to IN8 of the CIM. The component can be set up to sample the inputs periodically or upon an incoming event.



## Requirements

This software component requires an ADC-DAC CIM (CIM ID: 0x0401) connected to a USB port.

SV USB       SV USB       EXT_PWR       EXP         OUT1       OUT2       OUT3       OUT4       GPADC       GPDAC       GPOUT       GPIN	ADC,	/DAC				1	0	)	
OUT1 OUT2 OUT3 OUT4 GPADC GPDAC GPOUT GPIN	5V USB	SV USB	EXT_PW	R EXP		/	Ast	Feric	S
	OUT1	OUT2	OUT3	OUT4	GPADC	GPDAC	GPOUT	GPIN	
IN1         IN2         IN3         IN4         IN5         IN6         IN7         IN8	IN1	IN2	IN3	IN4	IN5	IN6	IN7	IN8	•

Legacy ADC/DAC CIM

### **Output Port Description**

• **in1 to in8 [integer]:** Each of these output port corresponds to one input of the ADC CIM. The measured values are forwarded directly from the CIM thus corresponding to the description given in the CIM specification.

## **Event Listener Description**

• **adcSampleTrigger:** An incoming event on this port will cause the ADC CIM to sample data on its inputs. This event will only take effect if the periodic\_update property is set to 0.

## **Properties**

- **periodicUpdate [integer]:** This property defines the time between data sampling requests to the ADC CIM (in milliseconds). The software component will send requests to the CIM in the given intervals. Due to current limitations of the ADC CIM it is recommended to use time intervals not lower than 50 milliseconds (20 samples per second).
- activateInput1 to activateInput8 [Boolean]: These properties correspond the CIM's ADC inputs IN1 to IN8, if the property is true, the corresponding output port of the software component will send the sampled data.

## Legacy Digital In

# **Component Type: Sensor (Subcategory: Generic Control Input)**

The LegacyDigitalIn component provides an interface to read the digital inputs of the legacy GPIO CIM (CIM Id: 0x0201). On state changes of the connected signals (transitions to high or low level), the component generates corresponding trigger-events.



LegacyDigitalIn plugin

## Requirements

This component requires the legacy GPIO CIM (CIM Id: 0x0201) connected to an USB port.



GPIO CIM

## **Event Trigger Description**

- **in1High to in8High:** Each of these event ports is linked to one input port, if the device connection to this input port delivers a signal which changes to high level, an event will be raised on the corresponding port.
- **in1Low to in8Low:** Each of these event ports is linked to one input port, if the device connection to this input port delivers a signal which changes to low level, an event will be raised on the corresponding port.

## **Properties**

- activateEventIn1 to activateEventIn8 [Boolean]: These properties declare for each port whether or not a signal transition on the actual input port should result in an event being triggered in the ARE. If a property is set true for one input, it will raise events on signal transitions, if it is set to false it will not.
- **periodicUpdate:** Period in milliseconds for update messages about state of device inputs. If the property is set to zero, the plugin will receive messages from the CIM on signal transitions, if the property is non-zero, the CIM will send status messages in the defined intervals without extra event messages on signal changes. Both modes will have the same effect on the software plugin, it will raise events on its trigger ports.

## LineReader

#### **Component Type: Sensor (Subcategory: File System)**

This component reads lines from a text file and sends them to an output port. The line reading can be triggered by an incoming event.



### **Input port Description**

• **skipLines (integer):** When an integer number is received at this port, the component tries to skip this number of lines. The next line will be read from the resulting position.

## **Output port Description**

• **actLine (string):** A line of text which has been read from the file (as line termination character, the "enter"-key (\n) is used.

## **Event Listener Description**

- **readNextLine:** When this event is triggered, the component tries to read one line of text from the file and sends it to the output port.
- **resetToFirstLine:** When this event is triggered, the internal file pointer is reset of the beginning of the file.

## **Event Trigger Description**

• **endOfFile:** This event is fired when no line can be read from the file (file end or other error occurred)

## **Properties**

• **fileName [string]:** The full path and file name of the text file to read. The path can be given as absolute path or relative to the ARE executable's directory

## Lipmouse

#### **Component Type: Sensor (Subcategory:Sensor Modules)**

This component provides signals from the Lipmouse module, which allows computer control via a mouthpiece. The user can create input signals with 3 degrees of freedom by interacting with the mouthpiece: left/right, up/down and sip/puff. The horizontal and vertical movements are measured via force sensors inside the lipmouse module. The sip/puff actions are measured via a pressure sensor. The lipmouse can be adjusted to very low force that it can be used by persons with severly reduced motor capabilites (e.g. late stage musculuar dystrophy or quadraplegia up to C1/C2 lesions).

A	Component: Lipmouse.	1			
	Search				
	<ul> <li>Internal Properties</li> </ul>				
Lipmouse.1	Component Name Lipmouse.1				
	Component Type	asterics.Lipmouse			
×	Component Descriptio	Lipmouse CIM			
	Component Class	sensor			
	<ul> <li>Properties</li> </ul>				
pressure	uniqueID	not used			
	periodicADCUpdate	50			
	sipThreshold	505			
	sipTime	700			
	puffThreshold	520			
	puffTime	700			

Lipmouse sensor plugin

## Requirements

The Lipmouse module must be connected to a USB port. Firmware for the lipmouse sensor can be found in the folder CIMs/Lipmouse\_CIM. The free "teensy loader" tool can be used to update the firmware of the lipmouse module. An alternative stand-alone firmware and configuration GUI is provided in the folder CIMs/StandAlone\_Modules\_FLipWare. Detailed design documentation will be provided in the future.



Lipmouse sensor application

## **Output Port Description**

- X [integer]: the force applied to the Lipmouse mouthpiece in xdirection
- Y [integer]: the force applied to the Lipmouse mouthpiece in ydirection
- **pressure [integer]:** the pressure value applied to the sip/puff sensor

## **Event Listener Description**

- **calibration:** if this event is received, the x/y force value is set to 0 (removing any drift of the sensor values)
- setLed1: if this event is received, Led1 will be turned on
- clearLed1: if this event is received, Led1 will be turned off
- setLed2: if this event is received, Led2 will be turned on
- clearLed2: if this event is received, Led2 will be turned off
- setLed3: if this event is received, Led3 will be turned on
- clearLed3: if this event is received, Led3 will be turned off

## **Event Triggerer Description**

- **sip:** will be triggered when pressure decreases under the selected sip treshold and increses back before the selected sipTime has passed
- **longSip:** will be triggered when pressure decreases under the selected sip treshold and increases back after the selected sipTime has passed
- **sipStart:** will be triggered at the moment when pressure decreases under the selected sip treshold
- **sipEnd:** will be triggered at the moment when pressure increases above the selected sip treshold
- **puff:** will be triggered when pressure increases above the selected puff treshold and decreases back before the selected puffTime has passed
- **longPuff:** will be triggered when pressure increases above the selected puff treshold and decreases back after the selected puffTime has passed
- **puffStart:** will be triggered at the moment when pressure increases above the selected puff treshold
- **puffEnd:** will be triggered at the moment when pressure decreases under the selected puff treshold
- button1Pressed: will be triggered when button1 of the lipmouse is pressed
- **button1Released:** will be triggered when button1 of the lipmouse is released
- button2Pressed: will be triggered when button2 of the lipmouse is pressed
- button2Released: will be triggered when button2 of the lipmouse is released
- button3Pressed: will be triggered when button3 of the lipmouse is pressed
- button3Released: will be triggered when button3 of the lipmouse is released

## **Properties**

- **uniqueID [integer]:** a unique identifier, useful if more then one lipmouse modules are used (dynamic property).
- **periodicADCUpdate [integer]:** the update rate for force and pressure values in milliseconds.
- **sipThreshold [integer]:** threshold value for sip actions.
- **sipTime [integer]:** threshold time for sip/longSip events.
- puffThreshold [integer]: threshold value for puff actions.
- **puffTime [integer]:** threshold time for puff/longPuff events.

## LipmouseIR

#### **Component Type: Sensor (Subcategory:Sensor Modules)**

This component provides signals from the Lipmouse module, which allows computer control via a mouthpiece. The user can create input signals with 3 degrees of freedom by interacting with the mouthpiece: left/right, up/down and sip/puff. The horizontal and vertical movements are measured via force sensors inside the lipmouse module. The sip/puff actions are measured via a pressure sensor. The lipmouse can be adjusted to very low force that it can be used by persons with severly reduced motor capabilites (e.g. late stage musculuar dystrophy or quadraplegia up to C1/C2 lesions). Additionally the Lipmouse has an infrared interface which enables it to control electronic devices. Through a receiver it is possible to record commands from existing remote controls and store the codes to a database. When needed, the IR signal is reconstructed and transmitted to the device to be controlled.

-	Properties (Currer)					
	Component: Lipmouse					
	Search					
LipmouseIR.1	<ul> <li>Internal Properties</li> </ul>	<ul> <li>Internal Properties</li> </ul>				
DeviceType	Component Name	LipmouseIR.1				
DeviceName X	Component Type	asterics.LipmouseIR				
	Component Descriptio	LipmouseIR CIM				
Pressure	Component Class	sensor				
	<ul> <li>Properties</li> </ul>					
	uniqueID	not used				
	periodicADCUpdate	50	-			
	sipThreshold	505	<b>a</b>			
	sipTime	700	<b>a</b>			
	puffThreshold	520	<b>a</b>			
	puffTime	700	-			
	IRCodeFilePath	./data/sensor.LipmouseIR/IRCodes.csv				

LipmouseIR sensor plugin

## Requirements

The LipmouseIR module must be connected to a USB port. Firmware for the lipmouseIR sensor can be found in the folder CIMs/Lipmouse\_CIM. The free "teensy loader" tool can be used to update the firmware of the lipmouseIR module. Detailed design documentation will be provided in the future.

## **Input Port Description**

- DeviceType [string]: Type of the device to be controlled via IR (e.g.: TV)
- **DeviceName [string]:** Name of the device to be controlled via IR (e.g.: Sony, Samsung, etc.)
- **DeviceFunction [string]:** Name of function of the device (e.g.: On, Off, VolumeUp, etc.)

## **Output Port Description**

- X [integer]: the force applied to the Lipmouse mouthpiece in xdirection
- Y [integer]: the force applied to the Lipmouse mouthpiece in ydirection
- **pressure [integer]:** the pressure value applied to the sip/puff sensor

## **Event Listener Description**

- SendIRCode: Send an IR Code to the Lipmouse
- RecordIRCode: Record an IR Code through the Lipmouse
- **calibration:** if this event is received, the x/y force value is set to 0 (removing any drift of the sensor values)
- setLed1: if this event is received, Led1 will be turned on
- clearLed1: if this event is received, Led1 will be turned off
- **setLed2:** if this event is received, Led2 will be turned on
- clearLed2: if this event is received, Led2 will be turned off
- setLed3: if this event is received, Led3 will be turned on
- clearLed3: if this event is received, Led3 will be turned off

## **Event Triggerer Description**

- **StartRecord:** will be triggered when Lipmouse is recording
- **StopRecord:** will be triggered when Lipmouse has finished recording
- **sip:** will be triggered when pressure decreases under the selected sip treshold and increses back before the selected sipTime has passed
- **longSip:** will be triggered when pressure decreases under the selected sip treshold and increases back after the selected sipTime has passed
- **sipStart:** will be triggered at the moment when pressure decreases under the selected sip treshold
- **sipEnd:** will be triggered at the moment when pressure increases above the selected sip treshold
- **puff:** will be triggered when pressure increases above the selected puff treshold and decreases back before the selected puffTime has passed
- **longPuff:** will be triggered when pressure increases above the selected puff treshold and decreases back after the selected puffTime has passed
- **puffStart:** will be triggered at the moment when pressure increases above the selected puff treshold
- **puffEnd:** will be triggered at the moment when pressure decreases under the selected puff treshold
- button1Pressed: will be triggered when button1 of the lipmouse is pressed
- **button1Released:** will be triggered when button1 of the lipmouse is released
- button2Pressed: will be triggered when button2 of the lipmouse is pressed
- **button2Released:** will be triggered when button2 of the lipmouse is released
- button3Pressed: will be triggered when button3 of the lipmouse is pressed
- button3Released: will be triggered when button3 of the lipmouse is released

## **Properties**

- **uniqueID [integer]:** a unique identifier, useful if more then one lipmouse modules are used (dynamic property).
- **periodicADCUpdate [integer]:** the update rate for force and pressure values in milliseconds.
- **sipThreshold [integer]:** threshold value for sip actions.
- **sipTime [integer]:** threshold time for sip/longSip events.
- puffThreshold [integer]: threshold value for puff actions.
- **puffTime [integer]:** threshold time for puff/longPuff events.
- **IRCodeFilePath [string]:** Filepath to the file, where the IR Codes are stored.

### **IR Code Database**

The database which contains the IR codes as well as the information about the type and name of the device and the specific function is a comma separated value file. The first value is the type, the second one is the name and the third one is the function. The following 512 values are the IR code. This database is automatically generated and maintained if new IR codes are recorded with the Lipmouse with IR functions.

## MicGPI

# **Component Type: Sensor (Subcategory: Generic Control Input)**

This component reads the input from the microphone or line-in of a computer's sound device. The averaged amplitude / sound pressure level and the dominant frequency can be calculated and used for control purposes. It is highly recommended to use a headset microphone to avoid unwanted detections from ambient noise. The sampling rate can be set (300-44100 Hz), the other device properties are mono (1 channel) and 8 bit resoltion.

It is possible to attach a momentary switch via standard 3.5mm jack plug to the mic/line input of the soundcard (or a cheap USB soundcard) and use this component for detection of switch presses.

**Disclaimer:** Attaching a switch to mic/line input does not work with all sound cards. Although we are not aware of a permanent damage to a sound card, you do this on your own risk !



## Requirements

A internal or external sound device with mic or line input is required. If the component is utilized as switch input, a momentary switch must be attached to the line-in or microphone input jack. No additional input circuit is required. Any sort of filtering of the signal like background noise cancelling must be disabled.
## **Input Port Description**

- **thresholdLow [int]:** This input port sets the low threshold below a signal must move that a inLow event gets triggered.
- **thresholdHigh [int]:** This input port sets the high threshold above a signal must move that a inHigh event gets triggered.

## **Output Port Description**

- **pressure [double]:** This port outputs the sound amplitude or pressure level of the mic input signal (the values depend on the selected calculation mode).
- **frequency [integer]:** In case frequency calculation is enabled, this port outputs the dominant frequency of the spectrum (in Hz).

## **Event Trigger Description**

- **inLow:** This event is fired if the calculated level goes below the low threshold (can be used to detect switch press/release or sound pressure level).
- **inHigh:** This event is fired if the calculated level goes above the high threshold (can be used to detect switch press/release or sound pressure level).

## **Properties**

- **samplingRate [int]:** Defines the sampling rate in Hz (allowed values are in the rang of 300 44100 Hz)
- **sampleSize [int]:** Defines the size of the sample buffer. Must be a power of 2. The values 32/64/128/256/512/1024/2048 can be selected from a combobox. This buffer size determines the calculation-interval on the output ports.
- mode [int, combobox selection]: Defines the way how the output signal (which is also compared to the threshold values) is calculated. The calculation is applied on a block of samples of the selected size. Available options are: average sample values, average absolute sample values, min value, max value and max absolute value
- **thresholdLow [int]:** This property sets the low threshold below a signal must move that a inLow event gets triggered.
- **thresholdHigh [int]:** This property sets the high threshold above a signal must move that a inHigh event gets triggered.
- **noiseLevel [double]:** This property defines a minimum level fo a valid sound signal. All noise which is below this level will be attenauted to avoid unwanted detections.
- **calculateFrequency [boolean]:** If true, an FFT will be performed to calculate the frequency spectrum and output the dominant frequency to the associated port.
- **printSpectrum [boolean]:** If this property is set to true and frequency calculation is enabled, the FFT spectrum will be printed to the console (only reasonable in debug mode).
- **captureDevice:** [string]: This property defines the sound card from which the audio samples should be analysed. (dynamic property, values are suggested when ARE is in synced state)

## **Mouse Capture**

# **Component Type: Sensor (Subcategory: Standard Input Devices)**

This component provides access to mouse input activities of connected a standard mouse, like mouse x/y movement or button press/release activities. Mouse wheel and third mouse button are supported.



MouseCapture plugin

## Requirements

A mouse or mouse-equivalent must be connected to the computer/personal platform.

## **Output Port Description**

- **mouseX [integer]:** This port provides current absolute mouse X-position or relative mouse X-movement as integer value (depending on the mode of operation).
- **mouseY [integer]:** This port provides current absolute mouse Y-position or relative mouse Y-movement as integer value (depending on the mode of operation).

## **Event Listener Description**

- **blockEvents:** After this incoming event no mouse activities will be routed to the operating system.
- **forwardEvents:** After this incoming event all mouse activities will also be forwarded to the operating system.
- **toggleBlock:** An incoming event toggles the current blockEvents. If the mouse activities were forwarded they will now be kept and vica versa.

## **Event Trigger Description**

- **leftButtonPressed:** This port fires an event as the left mouse button is pressed.
- **leftButtonReleased:** This port fires an event as the left mouse button is released.
- **rightButtonPressed:** This port fires an event as the right mouse button is pressed.
- **rightButtonReleased:** This port fires an event as the right mouse button is released.
- **middleButtonPressed:** This port fires an event as the middle mouse button is pressed.
- **middleButtonReleased:** This port fires an event as the middle mouse button is released.
- **wheelUp:** This port fires an event as the mouse wheel is turned one step away from the user.
- **wheelDown:** This port fires an event as the mouse wheel is turned one step towards the user.

## **Properties**

blockEvents [boolean]: This property defines the mode of operation of the mousehook component. If this property is set to true, no mouse activities will be routed to the operating system - they will not be processed by other applications, the mouse cursor will not move and no clicks will be actually performed by the operation system. This can be useful if the mouse activity shall be transferred e.g. from the AsTeRICS personal platform to another system (via the HID actuator) or the mouse should be trapped to control a particular GUI or menu structure. In this case, the mouse component outputs only relative mouse movements in X and Y axis at the corresponding output ports. If the property value is set to false, mouse activities will be passed back to the operating system and will be processed as usual. In this case, the mousehook component provides the absolute mouse positions at the component?s output ports.

# **OpenBCI**

# **Component Type: Sensor (Subcategory: Bioelectric Measurement)**

This component provides an interface to bioelectric amplifiers which are compatible to the openBCI packet fromat. For more information please refer to the <u>OpenBCI</u> website. The COM Port number where the device is connected must be specified in the plugin properties.

GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)	<b>~</b> ↓			
		Component: openBCI.1				
		Search				
		<ul> <li>Internal Properties</li> </ul>				
openBCI.1		Component Class	sensor			
		Component Name	openBCI.1			
Channel1		Component Type	asterics.openBCI			
Channel2		Component Descriptic	This plugin receives an openBCI da			
Channel3		<ul> <li>Properties</li> </ul>				
Channeld		ComPort	COM4			
Channel		BaudRate	115200			
Channels						
Channelo						
Channel7 (						
Channel8 🧰						
AccX 🗲						
AccY 🗖						
AccZ 🧰						
	-					
	2	- DOL - Lunia				

OpenBCI plugin

## Requirements

This software component requires an Open BCI compatible device connected to a COM Port, which sends packet protocol for channel data.

## **Output Port Description**

- **Channel1 to Channel8 [integer]:** Each output corresponds to the sampled data from its corresponding channel. A calibration procedure to obtain the factor for calculation of microvolts must be performed with the amplifier device.
- AccX, AccY, AccZ [integer]: The current values of the 3 axis of the acceleration sensor.

## **Properties**

- **COMPort [String]:** The name of the COM port, for example COM2 or COM17.
- **Baudrate [Integer]:** The baud rate for the transmission. Standard is 115200.

## **OpenVibe**

# **Component Type: Sensor (Subcategory: Bioelectric Measurement)**

The OpenVibe plugin allows obtaining data from a connection to the OpenVibe BCI software which is currently one of the most prominent BCI frameworks. OpenVibe supports a wide range of biosignal and EEG acquisition devices and provides sophisticated signal processor training and classification algorithms. The graphical design concept is similar to the ACS, and plugins can be committed by the open source community. To send data from OpenVibe to the ARE plugin, a dedicated OpenVibe Plugin called "AsTeRICS connection" has been created. This OpenVibe plugin sends up to 16 channels of signal data and up to 61 different stimulation events to the ARE plugin via a UDP connection, by using the Open Sound Control (OSC) protocol. The following figure shows this communication flow.



flow from OpenVibe to AsTeRICS

## Requirements

The OpenVibe framework has to be installed and the AsTeRICS connection plugin must be available (this means that OpenVibe has to be built from sources and the AsTeRICS connection plugin is included in the build process). Furthermore, the correct UDP port must be set in the properties of both plugins, and the ARE plugin has to be running when OpenVibe is started (because the ARE plugin acts as server and opens a listening port).



OpenVibe plugin

## **Output Port Description**

• CH1 – CH16 [double]: these are the output ports for the OpenVibe signals. Up to 16 signals can be sent

## **Event Listener Description**

- OVTK\_StimulationId\_Label\_00 0C
- OVTK\_StimulationId\_Letter\_0 Z
- OVTK\_StimulationId\_Label\_Target
- OVTK\_StimulationId\_Label\_NonTarget These are the Event triggers which can be linked to the corresponding OpenVibe Stimulation IDs. In total, 56 different stimulations can be processed

## **Properties**

• **Port [integer]:** the UPD port which is opened by the AsTeRICS plugin to wait for the OpenVibe connection

## **Osc Server**

#### **Component Type: Sensor (Subcategory: Communication)**

The OscServer component enables the ARE to receive messages using the OpenSoundControl (OSC) protocol. The OscServer can receive various OSC data messages which can be divided in the individual data segments and forwarded to the output ports. The properties are used for the segmentation of the individual information segments of a whole OSC message. This plugin utilizes the NetUtil java library (http://www.sciss.de/netutil/) for the OSC implementation, it is (C)opyrighted 2004-2011 by Hanns Holger Rutz and released under the GNU Lesser General Public License.



OscServer plugin

## Requirements

- Any OSC client software which sends data to the server e.g. ARE OscOutClient Plugin, various OSC Apps for Android and various PC software.
- Check your firewall configuration and network settings to ensure that OSC messages are not blocked.
- Exact knowledge about the structure of the OSC message, to determine the OSC message structure refere to the OSC client documentation or utilze a neworksniffer e.g. Wireshark (www.wireshark.org)

## **Functional Principle**

Symbolic OSC Message	/OSC/address/pattern	 fff	 Arugment0,	Argument1,	Argument2,
Coresponding Plugin		CH[1-8] = float			
Properites	AddressCH[n]	CH[9-12]= string	ArgNrCH[0]	ArgNrCH[1]	ArgNrCH[2]

Symbolic OSC Message

Each time the OscServer Plugin receives a OSC message it decompose it according to the plugin properties (AddressCH[n], ArgNrCH[n]) and forward it to the output ports. The OscServer is able to receive OSC messages with arguments of the type float and string. The OSC datatype float is typcased to the ARE type double.

## **Output Port Description**

- **out 1-8 [double]:** Forward data form the OSC message.
- out 9-12 [string]: Forward strings from the OSC message

## **Properties**

- **Port [integer]:** This value specifies the OscServer port.
- AddressCH[n] [string]: This value specifies the OSC input address e.g. "/path/to/sensor/accxyz".
- **ArgNrCH[n] [integer]:** The Argument Number defines which single argument of an entire message is picked out and forwarded to the output. The first argument of an OSC message begins with the index value zero, e.g. the first argument has the index 0, the second the index 1 and so on.

## **Referred Plugins**

- OscOutClient
- OpenVibe OscGestureFollower

## **P2Parser**

# **Component Type: Sensor (Subcategory: Bioelectric Measurement)**

This component provides an interface to bioelectric amplifiers which are compatible to the openEEG P2 packet fromat (e.g. the ModualarEEG or the SMTEEG by Olimex). For more information please refer to the <u>OpenEEG</u> website. The COM Port number where the device is connected must be specified in the plugin properties.

GUI Designer (Ctrl-G) 🗧	Properties (Ctrl-P)	<b>→</b> Å		
<u>م</u>	Component: p2_parser.	1		
E	Search			
	<ul> <li>Internal Properties</li> </ul>			
	Component Class	sensor 💌		
p2_parser.1	Component Name	p2_parser.1		
	Component Type	asterics.p2_parser		
Channel1	Component Descriptio	My Plugin description		
Channel2	<ul> <li>Properties</li> </ul>			
Channel3	ComPort	COM12		
Channeld	BaudRate	57600		
Channels				
Channelo				
×				
	Properties (Ctrl-P)	T Output Ports		

OpenEEG P2 Parser plugin

## Requirements

This software component requires an OpenEEG compatible device connected to a COM Port, which sends the P2 packet protocol for channel data.

### **Output Port Description**

• **Channel1 to Channel6 [integer]:** Each output corresponds to the sampled data from its corresponding channel. The values range for -512 to 512. A calibration procedure to obtain the factor for calculation of microvolts must be performed with the amplifier device.

## **Properties**

- **COMPort [String]:** The name of the COM port, for example COM2 or COM17.
- **Baudrate [Integer]:** The baud rate for the transmission. Standard is 57600.

## **Platform Analog In**

# **Component Type: Sensor (Subcategory: Generic Control Input)**

The PlatformAnalogIn component is a sensor which is used in combination with the analog inputs of the personal platform. The component provides two output ports which correlate to the inputs IN1 and IN2 of the platform. The component can be set up to sample the inputs periodically or upon an incoming event. Both inputs can be used for voltage and resistance measurement.



## Requirements

This component requires the Core CIM (CIM Id: 0x0602) of the AsTeRICS Personal Platform.



The AsteRICS Personal Platform

### **Output Port Description**

• **in1 to in2 [integer]:** Each of these output port corresponds to one input of the ADC CIM. The measured values are forwarded directly from the CIM thus corresponding to the description given in the CIM specification. Depending on the type of connected sensor the values either correspond to a voltage or a resistive value.

## **Event Listener Description**

• **adcSampleTrigger:** An incoming event on this port will cause the ADC CIM to sample data on its inputs. This event will only take effect if the periodic\_update property is set to 0.

## **Properties**

- **periodicUpdate [integer]:** This property defines the time between data sampling requests to the ADC CIM (in milliseconds). The software component will send requests to the CIM in the given intervals. Due to current limitations of the ADC CIM it is recommended to use time intervals not lower than 50 milliseconds (20 samples per second).
- activateInput1 to activateInput2 [Boolean]: These properties correspond the CIM's ADC inputs IN1 to IN8, if the property is true, the corresponding output port of the software component will send the sampled data.

# **Platform Digital In**

#### **Component Type: Sensor (Subcategory: Personal Platform)**

The PlatformDigitalIn component provides an interface to read the digital inputs of the AsTeRICS Personal Platform. On state changes of the connected signals (transitions to high or low level), the component generates corresponding trigger events.



PlatformDigitalIn plugin

## Requirements

This component requires the Core CIM (CIM Id: 0x0602) of the AsTeRICS Personal Platform.



The AsteRICS Personal Platform
### **Event Trigger Description**

- **in1High to in3High:** Each of these event ports is linked to one input port, if the device connection to this input port delivers a signal which changes to high level, an event will be raised on the corresponding port.
- **in1Low to in3Low:** Each of these event ports is linked to one input port, if the device connection to this input port delivers a signal which changes to low level, an event will be raised on the corresponding port.

• activateEventIn1 to activateEventIn3 [Boolean]: These properties declare for each port whether or not a signal transition on the actual input port should result in an event being triggered in the ARE. If a property is set true for one input, it will raise events on signal transitions, if it is set to false it will not.

## Proximity

#### **Component Type: Sensor (Subcategory:Sensor Modules)**

This component provides the distance sensor signal from the Proximity sensor board, which has been developed in course of the Bachelor Theses of Franziska Horak at the UAS Technikum Wien. The Proximity sensor board can be head mounted and offers an alternative input method via minimal movements of facial features - e.g. chin movements. In several proof-of-concept models, on-screen keyboard control and mouse control could be shown. For detailed information please refer to the file Documentation/DIYGuides/ProximitySensor\_Horak.pdf



Proximity sensor plugin

### Requirements

The Proximity sensor has to be connected to a USB port. Design files and firmware for the proximity sensor can be found in the folder CIMs/Proximity\_CIM.



Proximity sensor application

### **Input Port Description**

• **input [integer]:** this input port allows setting the threshold value for the creation of events

### **Output Port Description**

• **distance [integer]:** the currently measured distance from sensor to subject

### **Event Trigger Description**

- LowToHigh: this event is triggered when the distance increases above the threshold value
- **HighToLow:** this event is triggered when the distance decreases below the threshold value

- threshold [integer]: the threshold value for creating events.
- sendingMode [combobox selection]: if "Continuous data" is selected, the distance values are sent to the output port, if "Events: below->above", "Events: above->below" or "Events: both" are selected, the respective event triggers will be generated.

### **Razor IMU**

# **Component Type: Sensor (Subcategory: Inertial Measurement)**

The RazorIMU plugin provides the serial output of the 9DOF Razor Inertial Measurement Unit at three output ports: pitch, yaw and roll. These three values represent the orientation in terms of rotation along the three axes of the coordinate system.

Deployment (Ctrl-D) GUI Designer (Ctrl-G) =	Properties (Ctrl-P)		<b>▼</b> ₽
	componentType: Razo	IMU.1	
	Search		
E	<ul> <li>Internal Properties</li> </ul>		
	Component Descriptio	9 Degree-of-Freedom Razor Ine	rtial Measurer
	Component Name	RazorIMU.1	
RazorIMU.1	ComponentType	sensor	
	<ul> <li>Properties</li> </ul>		
pitch 🛑	comPort	COM7	
yaw 🛑	baudRate	57600	<b>÷</b>
roll	operationMode	PitchYawRoll	
<	Troperties (Ctrl-P)	Output Ports	
Razo	orIMU plugin		

### Requirements

This plugin requires a 9DOF Razor IMU module connected to the AsTeRICS platform via a UART/USB converter cable (e.g. an FTDI cable) which creates a COM port. The Razor IMU module (and also the converter cable) is available at SparkFun electronics. It has to be updated with the Sparkfun 9DOF Razor IMU AHRS firmware. (The COM port must be determined by looking in the device manager window and cannot be automatically detected like with dedicated AsTeRICS CIMs.) The required baud rate is 57600.



RazorIMU module

### **Output Port Description**

- **pitch [double]:** The value for the current pitch.
- yaw [double]: The value for the current yaw.
- roll [double]: The value for the current roll.

- **comPort [String]:** The name of the COM port the IMU is connected to.
- **baudRate [integer]:** The baud rate the IMU is transferring its data at.
- **operationMode [String]:** Designates the operation mode (currently only "PitchYawRoll" is available).

## **CSV Reader**

#### **Component Type: Sensor (Subcategory: File System)**

This plugin reads a whole .csv file and sends the seperate lines as strings to the output port.

⇒ ×	Properties (Ctrl-P)		<b>▼</b> ₽
	Component: ReadCSV.1 Search		
ReadCSV.1	<ul> <li>Internal Properties</li> </ul>		
	Component Class	sensor	-
Output	Component Name	ReadCSV.1	
	Component Type	asterics.ReadCSV	
	Component Descriptio	Open and read a CSV file	
	<ul> <li>Properties</li> </ul>		
	FilePath	.\\data\\csv\\	

ReadCSV

### **Event Listener Description**

• **read:** Opens a file-chooser menu to select a csv-file which should be opened and read. The file is read immideately after selection (attention: this blocks the ARE model execution until the file contents are read completely !)

#### **Output port Description**

• **Output[string]:** The port sends the individual lines of the file as strings. An example for the string is: "data1;data2;data3". To seperate the string into individual strin values, the StringSplitter can be used.

• **FilePath[string]:** Defines the default Path were the file is located. (TBD: add property for default FileName and line-by-line read option)

### **RFID** Reader

#### **Component Type: Sensor (Subcategory: Others)**

The RFIDReader component provides an interface to the ID-Innovations RFID reader modules. These modules are available for example from Sparkfun electronics. The recognized ID-Tags are transferred from the module to a COM port, where the RFIDReader components reads the ID and puts it to the output port as an ASCII string. The RFIDReader can be useful to switch to a different a model (e.g. for another users or use cases) or to change parameters of a running model.

Deployment (Ctrl-D) GUI Designer (Ctrl-G) 🗧	Properties (Ctrl-P)	<b>~</b> ↓
×	componentType: RFIE	DReader.1
_	Search	
	<ul> <li>Internal Properties</li> </ul>	
	Component Descripti	<ul> <li>Interfaces to the ID Innovations RFID reader</li> </ul>
RFIDReader.1	Component Name	RFIDReader.1
	ComponentType	sensor
taoID	<ul> <li>Properties</li> </ul>	
	comPort	COM8
	baudRate	9600 😤
	Branartiar (Ctrl.D)	" Output Poste
DEID	Doodor plugip	

RFIDReader plugin

### Requirements

This software component requires an ID-Innovations RFID reader to be connected to a COM port via the Sparkfun USB breakout board or a UART/USB bridge or converter cable. Dedicated drivers have to be installed (e.g. the FTDI VCP drivers for the Sparkfun breakout board). All needed components are contained in the Sparkfun RFID Starter Kit:



RFID Starter Kit

### **Output Port Description**

• **tagID [string]:** A recognized TagID is put out on this port as a sequence of 12 hexadecimal numbers in an ASCII string.

- **comPort [string]:** The COM port where the RFID reader module is connected to (e.g "COM5")
- **baudRate [integer]:** The baudrate for communication with the RFID reader module, should be 9600 for the ID Innovations modules

### Sensorboard

#### **Component Type: Sensor (Subcategory:Sensor Modules)**

This component provides the sensor signals from the Sensorboard CIM module, which has been developed in course of the Master Thesis of Yat-Sin Yeung at the UAS Technikum Wien. The Sensorboard is part of the head-mounted Smart Vision Module setup but can also be used as a stand-alone unit for movement analysis. The Sensorboard contains a 3-axis accelerometer, a 3-axis gyroscope, a 3-axis compass module, one analogue pressure sensor (a sip/puff sensor) and a connection to an optical IR-object tracking sensor which can track the position of up to 4 infrared LEDs in the field of view of the sensor. Furthermore, the Sensorboard contains a USB hub so that a USB camera can be connected.

The sensorboard is necessary for the headpose-compensated eye gaze tracking applications of the SVM. It can be used also for other applications like remote IR-led tracking or as inertial measurement unit. For a detailled description of the Sensorboard and its application for eye-tracking please refer to the files

Documentation/DIYGuides/SVM\_Eyetracking\_Yeung.pdf and Documentation/DIYGuides/SmartVisionModule.pdf

(Ctrl-D) GUI Designer (Ctrl-G)	Output Ports		<b>▼</b> ‡
-	/ accX		
	Output Port: accX		
	Search Search		
	E A Internal Properties		0
	Port Label	accX	
	Port Alias	ucov.	
	PortDataType	integer	5
	Description	acceleration X	-
	4		
compassZ pt1X	Internal Properties		
	Component Name	SensorBoard.1	
	Component Type	asterics.SensorBoard	
	Component Descript	ic Interfaces the sensor board CIM which provides a	icce
	Component Class	sensor	*
	<ul> <li>Properties</li> </ul>		-
	refreshInterval	50	Ð
	·		

Sensorboard plugin

### Requirements

The Sensorboard has to be connected to an USB port. Design files and firmware of the Sensorboard can be found in the folder CIMs/Sensorboard.



The Sensorboard PCB

### **Output Port Description**

- **accX/Y/Z [integer]:** the three axis output of the acceleration sensor
- gyroX/Y/Z [integer]: the three axis output of the gyroscope sensor
- compassX/Y/Z [integer]: the three axis output of the compass sensor
- pt1x pt4x [integer]:
- **pt1y pt4y [integer]:** The x/y corrdinates of the IR-LED tracking camera (0-1022, 1023 if no LED detected)
- pressure [integer]: the output of the pressure (sip/puff) sensor

• **refreshInterval [integer]:** the refresh interval for sensor values in milliseconds (should not be less than 20).

# **Signal Generator**

#### **Component Type: Sensor (Subcategory: Simulation)**

The SignalGenerator component can generate several output waveforms for component tests or other purposes like timing or event generation. Available waveforms are sine, sawtooth, rectangle and random signal data.



SignalGenerator plugin

### **Output Port Description**

• **out [double]:** The generated waveform is provided at this port.

- **sendInterval [integer]:** This value specifies the output rate in milliseconds. Please note that the output rate has to be fast enough to assemble the selected output frequency. For example if a frequency of 2 Hz is set, the send\_interval should not be greater than 125 milliseconds according to the sample theorem.
- waveForm [integer]: The waveform types random, sine, sawtooth and rectangle can be selected.
- frequency [double]: The frequency of the output signal in Hertz.
- **amplitude [double]:** The amplitude of the output signal.
- **phaseShift [double]:** The output signal is phase-shifted by this value (in milliseconds).
- offset [double]: Amplitude-offset of the output signal (this value is added to each generated waveform value).

# **Signal Shaper**

#### **Component Type: Sensor (Subcategory: Simulation)**

The SignalShaper component can be used to generate signal composed from basic linear functions.

Deployment (Ctrl-D) GUI Designer (Ctrl-G) 🗧	Properties (Ctrl-P)	▼ .
	Component: SignalSha	per.1
	Search	
=	<ul> <li>Internal Properties</li> </ul>	A
	Component Description	This component allow to shape signal from
	Component Name	SignalShaper.1
SignalShaper.1	ComponentType	sensor 💌
	<ul> <li>Properties</li> </ul>	
output	interval	20 😫
	numberOfLines	1
	behaviourAfterFinish	do nothing 📃
	beginValue1	0
	endValue1	0
	time1	0
	beginValue2	0
	endValue2	0
-	time2	0 *
* 🔲 🔹 👘	Properties (Ctrl-P)	TOutput Ports 💉 Event Listener
Sigr	nalShaper plugin	
endValue1 / b	beginValue2	endValue2
	ł	beginValue3 / endValue3



Example composed signal created from three linear signals

### **Output Port Description**

• **output [double]:** The output port for the signal.

### **Event Listener Description**

• **start:** Start the signal generation. If this event is received during signal generation, the signal generation is restarted.

- interval [integer]: The sampling rate in milliseconds.
- **numberOfLines [integer]:** The number of linear signals used to shape the output signal.
- **behaviourAfterFinish [integer]:** Defines the component behaviour after all signal lines have been sent:
  - **do nothing:** stop any action.
  - **repeat:** Send the signal lines from beginning.
  - **send the last value:** Continue to send the last value of the signal.
- **beginValue1...beginValue5 [double]:** The begin value of the linear signals.
- endValue1...endValue5 [double]: The end value of the linear signals.
- time1...time5 [integer]: The duration of the linear signals in milliseconds.

# Slider

# **Component Type: Sensor (Subcategory: Graphical User Interface)**

The Slider component generates a slider with adjustable range of values and size on the ARE desktop. This slider can be used to change important parameters of the model during runtime. Furthermore, an incoming signal can be adjusted by the slider component, using a gain factor property.

	0		
0 -800 -600 -400 -200	0 200	400 600 800	100
Deployment (Ctrl-D) GUI Designer (Ctrl-G) =	Properties (Ctrl-P)	-	<b>ņ</b>
	componentType: Slider	.1	
	Search		
	<ul> <li>Internal Properties</li> </ul>		
	Component Descriptio	a slider GUI element which provides the sli	c
	Component Name	Slider.1	-
Slider.1 >	ComponentType	sensor	
	<ul> <li>Properties</li> </ul>		
value 🛑	min	0	3
	max	100	3
	default	50	3
	caption	my slider	
	majorTickSpacing	20	3
	minorTickSpacing	5	3
	- Constant - Constant	horizontal	
	alignment	nonzontal	

plugin

Slider

#### **Input Port Description**

- **setValue [integer]:** Sets the slider position to the incoming value. Note that this value is not propagated to the output port (to avoid loops).
- **in [double]:** input port for an incoming signal which can be amplified by the slider component

#### **Output Port Description**

- **value [integer]:** This port provides the currently selected slider value (position). Only integer values are possible.
- **out [double]:** The amplified (or attenuated) input signal (out = in \* gain \* slider value)

- **min [integer]:** The minimum value of the slider range
- max [integer]: The maximum value of the slider range
- **default [integer]:** The defualt position of the slider at model startup (this value is not automatically sent to the port at model startup.
- gain [double]: The amplification value for an (optional) incoming signal (out = in \* gain \* slider value)
- caption [string]: A label for the slider
- **majorTickspacing [integer]:** Coarse sections for the slider value captions
- **minorTickspacing [integer]:** Fine sections for the slider value captions
- alignment [integer, combobox selection]: Slider orientation in the GUI, can be horizontal or vertical
- fontSize [integer]: Font size of the caption
- **storeValue [boolean]:** if the storeValue property is enabled, the current value of the slider position is stored and restored when the model is started next time. Note that this overrides the defaultValue property.
- **displayGUI [boolean]:** if selected, the GUI of this component will be displayed if not, the GUI will be hidden and disabled.

## **Space Navigtor 3D Mouse**

# **Component Type: Sensor (Subcategory: Standard Input Devices)**

This component interfaces the 3D connexion 3D Mouse device.


## Requirements

The 3D Mouse device connected to the platform



SpaceNavigator 3DMouse

#### **Output Port Description**

- mouseX [integer]: Data of axis X.
- mouseY [integer]: Data of axis Y.
- mouseZ [integer]: Data of axis Z.
- mouseRx [integer]: Data of rotation of axis X.
- mouseRy [integer]: Data of rotation of axis Y.
- mouseRz [integer]: Data of rotation of axis Z.
- **buttons [integer]:** Data of selected buttons combination.

## **Properties**

• **interval [integer]:** The interval of capturing 3D mouse state (ms).

# **Textfield Reader**

# **Component Type: Sensor (Subcategory: Graphical User Interface)**

Similar to the Keyboard Hook component, this component provides access to keystrokes coming from a standard keyboard or injected via software. The difference to the Keyboard Hook is that the Text Field Reader opens a GUI element with a text input filed and processes key input only from this text field. The keycodes of the pressed keys and accumulated words (multiple keys separated by ) are provided at the output ports of this component. Possible applications include triggering functions by keyboard input or remapping keys to other keycodes.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	<b>₽</b> P	roperties (Ctrl-P)		<b>•</b> ņ
Deployment (Ctrl-D) GUI Designer (Ctrl-G)	componentType: Text     Search     Internal Properties     Component Descriptic     Component Name     ComponentType     Properties     command1     command2     command3	roperties (Ctrl-P) componentType: Textfi a Internal Properties Component Descriptic Component Name ComponentType Properties command1 command2 command4	ieldReader.1 Captures key input from a te TextfieldReader.1 sensor	
		command5	five	
	-	command6	six	•
<		☞ Pr ☞ O 🥖 Ev		

TextfieldReader plugin

### Requirements

A keyboard which generates keystrokes or a software component which injects keystrokes into the operating system?s message queue.

#### **Output Port Description**

- **keys [integer]:** This port sends the virtual keycode of the last pressed key as an integer value.
- **words [string]:** This ports accumulates keystrokes and sends them as a string as soon as a blank separator (space key) appears in the key input stream.

### **Event Trigger Description**

• **recognizedCommand1** - **recognizedCommand7**: These event ports fire an event if one of seven command strings has been detected in the current input stream of keys.

#### **Properties**

- **command1 [string] command7 [string]:** Seven string properties to specify command strings. The component looks for these command strings in the input stream of keystrokes. If a command string matches, the corresponding event trigger port is raised. This can be useful for example to define voice commands which should trigger certain actions in other ARE plugins.
- **displayGUI [boolean]:** If selected, the GUI of this component will be displayed if not, the GUI will be hidden and disabled.

# Timer

#### **Component Type: Sensor (Subcategory: Simulation)**

The timer component can measure time in milliseconds, provide current time on an output port and trigger events when a specified time period has passed. It can be used to influence other features (e.g. to provide acceleration) or to enable dwell selection, for example clicking when a certain time has passed.



plugin

## **Input Port Description**

• **period [integer]:** The number of milliseconds the timer period property is set to.

### **Output Port Description**

• **time [integer]:** The number of milliseconds which have passed since the timer has been started.

#### **Event Listener Description**

- **start:** An incoming event starts the timer.
- **stop:** An incoming event stops the timer (pause, the current time value is not reset to 0).
- **reset:** An incoming event resets the time value to 0 (but does not stop the timer).

## **Event Trigger Description**

• **periodFinished:** This event is triggered when the given time has passed.

#### **Properties**

- **mode [integer]:** This property selects one of several possible modes of operation of the timer:
  - *"one shot":* The timer runs once from 0 to the specified time, and then stops.
  - *"repeat n times":* The timer completes the time period several times (the number is selected with the repeat counter property).
  - *"endless loop":* the timer completes the time period until it is stopped by an event at the stop listener port.
  - "once and continue time output": The time period is completed once, the timer is not stopped but continues to send the last time value (this is useful to generate increasing values with persisting maximum value, e.g. for utilization as accelerated speed value).
  - "measure time between start and stop": The timer sends the time passed from start to stop events to the output port (when stop was received).
- **repeatCounter [integer]:** The number of periods to finish for the "repeat n times " mode.
- **timePeriod (ms) [integer]:** The time period covered by this counter in milliseconds.
- **resolution (ms) [integer]:** The update rate of the timer in milliseconds. This value defines how often the current time value is updated and sent to the output port. It thereby defines the accuracy for the timer.
- waitPeriod (ms) [integer]: This value specifies how long the timer is bypassed before it actually starts (after receiving a start event).
- **autostart [boolean]:** This property defines if the timer will be started automatically together with the model (true) or if it will be started by an event (false).

# Tobi TiC

# **Component Type: Sensor (Subcategory: Bioelectric Measurement)**

The Tobi TiC plugin allows obtaining data from a connection to the <u>Tobi</u> <u>Framework</u> (Tools for Brain Computer Interaction). Tobi defined standard for BCI interfaces, e.g. beteween classification and actuator modues. Tobi supports a wide range of biosignal and EEG acquisition devices. The TiC interface is dedicated to transfer classifier results to actuators. This plugin implements a simple TiC reader via a TCP socket. Up to 5 TOBI class labels can be defined which will be used to extract values from incoming TiC-compatible xml messages. These class labels and the sever socket which is openend by the plugin can be defined viy plugin properties.



TiC plugin

### Requirements

TiC-compatible messages must be sent to the server port which is opened by the plugin.



Internal structure of a TOBI iC message

#### Example iC message:

```
<tobic version="0.1.1.0" frame="100">

<classifier name="errp" description="ErrP classifier" vtype="prob" ...

ltype="custom">

<class label="detected">0.953813</class>

</classifier>

<classifier name="smr" description="SMR classifier" vtype="prob" ...

ltype="biosig">

<class label="0x769">0.520190</class>

<class label="0x770">0.479810</class>

</classifier>

</classifier>
```

example TOBI iC message

### **Output Port Description**

• **value1 – value5 [double]:**these are the output ports for the Tobi class values. Up to 5 signals can be sent

#### **Properties**

- **tcpPort [integer]:** the TCP port which is opened by the AsTeRICS plugin to wait for the Tobi connection
- **classLabel1 classLabel5** [string]: the classLabels to extract values from the Tobi TiC messages. Values will be directed to the corresponding output ports of the plugin.

# **TuioReactivision**

#### **Component Type: Sensor (Subcategory: Sensor Modules)**

This component integrates reacTIVision into AsTeRICS

oReactivision.	Component: TuioReact	ivision.1	
	Search		
MarkerID 🛑	<ul> <li>Internal Properties</li> </ul>		
	Component Class	sensor 💌	
ipos 🛑	Component Name	TuioReactivision.1	
Ypos 🛑	Component Type	asterics.TuioReactivisio	
	Component Descriptio	integrates Tuio into as	
ed 🛑	<ul> <li>Properties</li> </ul>		
	TextOutput		
	MarkerAllocation		
Accel	reactivisionPath	C:\	
	TextFile	C:\	

TuioReactivision plug-in

## Requirements

This component requires that the reacTIVision.exe is running --> Download: http://reactivision.sourceforge.net/. It can either be started manually, with the ApplicationLauncher plug-in or via a reference in the tuioReactivision plug-in's property "reactivisionPath"

#### **Output Port Description**

- Marker ID [Integer]: The fiducial marker's ID
- Session ID [Float]]: Temporary object ID
- **xPos [Float]:** The marker's position on the x-axis
- yPos [Float]: The marker's position on the y-axis
- angel [Float]: The marker's angle
- motion speed [Float]: Movement vector which shows the marker's motion speed
- rotation speed [Float]: Rotation vector which shows the marker's rotation speed
- motionAccel [Float]: The marker's motion acceleration
- rotationAccel [Float]: The marker's rotation acceleration
- **Text [String]:** Sends the text assigned to the marker which is recognised on the display

#### **Properties**

- **Text Output [Boolean]:** If it is true, a string can be assigned to a marker. Therefor, a text file is necessary
- Marker Allocation [Boolean]: If it is true, an event port can be assigned to a marker. Therefor, a text file is necessary. Only Text Output or Marker Allocation can be true
- reactivisionPath [Boolean]: Contains the path to the reacTIVision.exe
- **Text File [String]:** Contains the path of the text file in which the marker is either assigned to a text or an event port (e.g. "1-3" marker three is assigned to event trigger one )



Examplary text file when a marker should be assigned to a string (Marker number is on the left side)

	listelnt - Editor	- 🗆 🗙
Datei Bearbeiten Format Ansicht ?		
1-0		^
2-1		
3-2		
4-3		
5-4		
6-5		

Examplary text file ehen a marker should be assigned to an integer (Marker number is on the right side)

## **Event Trigger**

• Event Trigger Port 1-6 can only be used when marker allocation is true.

# WiiMote

# **Component Type: Sensor (Subcategory: Standard Input Devices)**

The WiiMote component interfaces to the Nintendo WiiMote controller via Bluetooth and the WiiYourself! library. It provides various sensor values including the Wiimote buttons and the tracked IR-points of the Wiimote IR front camera and the Joystick position of the Nunchuk extension.

Contains WiiYourself! wiimote code by gl.tter - see http://gl.tter.org

For Wiimote connection and pairing, please see e.g. here.



### Requirements

A Wiimote controller has to be available and paired with the Bluetooth radio module of the system.



WiiMote and Nunchuk

#### **Output Port Description**

- **pitch [integer]:** The calculated pitch orientation of the Wiimote controller (in degrees)
- **roll [integer]:** The calculated roll orientation of the Wiimote controller (in degrees)
- point1X [integer]: The X position of the first visible IR point (0-1023)
- **point1Y [integer]:** The Y position of the first visible IR point (0-768)
- point2X [integer]: The X position of the second visible IR point (0-1023)
- **point2Y [integer]:** The Y position of the second visible IR point (0-768)
- **nunX [integer]:** The X-position of the analog joystick on the Nunchuk extension (if connected)
- **nunY [integer]:** The Y-position of the analog joystick on the Nunchuk extension (if connected)
- **battery [integer]:** The battery level (0-100%)

#### **Event Trigger Description**

- pressedUp: Up direction button pressed
- releasedUp: Up direction button released
- pressedDown: Down direction button pressed
- releasedDown: Down direction button released
- pressedLeft: Left direction button pressed
- releasedLeft: Left direction button released
- pressedRight: Right direction button pressed
- releasedRight: Right direction button released
- pressedA: A direction button pressed
- releasedA: A direction button released
- pressedB: B direction button pressed
- releasedB: B direction button released
- pressed1: 1 direction button pressed
- released1: 1 direction button released
- pressed2: 2 direction button pressed
- released2: 2 direction button released
- pressedPlus: Plus direction button pressed
- releasedPlus: Plus direction button released
- pressedMinus: Minus direction button pressed
- releasedMinus: Minus direction button released
- pressedHome: Home direction button pressed
- releasedHome: Home direction button released
- **pressedNunchuckC:** NunchuckC direction button pressed
- releasedNunchuckC: NunchuckC direction button released
- pressedNunchuckZ: NunchuckZ direction button pressed
- releasedNunchuckZ: NunchuckZ direction button released

### **Properties**

• **updatePeriod [integer]:** This property defines how often the WiiMote is queried for new data (in milliseconds)

# **XFacetrackerLK**

#### **Component Type: Sensor (Subcategory: Computer Vision)**

#### **OS: Windows, Linux, Mac OSX**

This component provides a cross-platform face tracking computer vision algorithm which puts out estimated movement of a users' nose and chin in x and y coordinates. The underlying mechanism builds upon the JavaCV (OpenCV) library (in particular a trained cascade of haar-like features in combination with a Lukas Kanade optical flow algorithm is used to track a face and its movement). The x- and y- coordinates can be used in camera-mouse configurations or to enable selection or control tasks. The values are only provided if a face can be tracked by the algorithm. Note that the x- and y- values represent relative movement in pixels and have to be accumulated (e.g. via the integrate component) to generate e.g. absolute mouse positions.

### Requirements

A camera has to be available (this can be any webcam or a camera which is available as image acquisition device via the operating system). The picture below shows the Logitech Webcam 9000 Pro. Also the camera device of a Kinect sensor or PS3Eye should work. Furthermore, a video stream of an IP camera can be used for tracking.



Logitech Webcam 9000 Pro

#### **Output Port Description**

- **noseX [integer]:** This value specifies the relative change in the x coordinate of the user's nose with respect to the previous image frame.
- **noseY [integer]:** This value specifies the relative change in the y coordinate of the user's nose with respect to the previous image frame.
- **chinX [integer]:** This value specifies the relative change in the x coordinate of the user's chin with respect to the previous image frame.
- **chinY [integer]:** This value specifies the relative change in the y coordinate of the user's chin with respect to the previous image frame.

#### **Event Listener Description**

- **init:** if this event is triggered, the face recognition procedure is initiated. This can be useful if the correct face position has been lost due to drifting of the LK algorithm.
- **showCameraSettings** an incoming event displays the settings window for the camera device (only on Windows), where parameters like image brightness or contrast can be adjusted.

#### **Properties**

- frameGrabber[string, combobox selection]:Name of FrameGrabber to use (Default, VideoInput, OpenCV, FFmpeg, OpenKinect, PS3Eye, IPCamera).
- frameGrabberFormat[string]:Format for grabber, e.g. FFmpeg: 'dshow' (default), 'vfwcap', 'gdigrab'.
- deviceList [string, combobox selection]:List of available devices, if supported by grabber
- **cameraSelection [string]:** The camera device to be used use camdIdx e.g. '0' (VideoInput, OpenCV, OpenKinect, PS3Eye), or camName e.g. 'video=Integrated Camera' or 'desktop' (FFmpeg), stream-url for IPCamera.
- **cameraResolution [string, combobox selection]:** This selection box provides several standard camera resolutions. Changing the resolution affects accuracy and performance (CPU load of the runtime system). Provided selections include "160x120", "320x240", "640x480", "800x600", "1024x768" and "1600x1200".
- **titleVideoFrameWindow [string]:**The title of the window showing the video frame.

### **Example Configuraitons**

#### Default

Let the plugin probe available framegrabbers and choose the first available one. The following grabber are tried in the following order:

- 1. OpenCV
- 2. VideoInput
- 3. FFmpeg

Configuration

- frameGrabber:Default
- cameraSelection:0

#### OpenCV

- frameGrabber:OpenCV
- cameraSelection:0

#### VideoInput

- frameGrabber:VideoInput
- cameraSelection:0

#### FFmpeg

- frameGrabber:FFmpeg
- frameGrabberFormat (or vfwcap):dshow
- cameraSelection (device name as of device manager):video=Integrated Camera

#### **MJPEG stream of SmartPhone camera**

Install the following Android app on your smartphone and start streaming the front camera: <u>Camera Stream - WiFi IP Webcam (Web Host LLC)</u>

Enter the http url displayed at your smartphone and add /video to the url

e.g.

http://192.168.1.100:8080/video

Set the cameraSelection property of the XFacetrackerLK plugin to this url.

Configuration

- frameGrabber:IPCamera
- cameraSelection:http://192.168.1.100:8080/video

# **Processors**

Processors are components which modify data provided by other plugins. Thus processors contain input ports as well as output ports and are placed in the middle of the ARE processing chain. Types of processors are either data modifiers (such as amplifiers and averagers) or flow control elements (such as conditional path selectors).
## AdjustmentCurve

# **Component Type: Processor (Subcategory: Signal Shaping)**

The AdjustmentCurve component allows transformation of an incoming signal to an outgoing signal. The signal mapping can be freely arranged in a drawing window (GUI) during runtime of the model. The resulting mapping can be saved as a curve file. The GUI is optional - an existing curve can be loaded to perform the signal mapping without the GUI.

GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)	<b>▼</b> Ū	
		Component: Adjustme	ntCurve.1	
		Search		
		<ul> <li>Internal Properties</li> </ul>		
		Component Class	processor 💌	
		Component Name	AdjustmentCurve.1	
		Component Type	asterics.AdjustmentCurve	
AdjustmentCury		Component Descriptio	AdjustmentCurve	
el		<ul> <li>Properties</li> </ul>		
() in		filename	curve1	
out CurveName		displayGui	×	
		inMin	0	
		inMax	1000	
		outMin	0	
		outMax	1000	
		mode	autoupdate min and max	
		fontSize	14	
	-	caption	signal mapper	
٠ 🔲	Þ	Properties (Ctrl-P)	Event Listener 3 Output Ports 3 Input Ports	
	Adj	ustmentCurve pl	lugin	





#### **Input Port Description**

- **in [double]:** This port receives the input values which will be mapped to output values.
- **CurveName [string]:** When this port receives a string, the plugin tries to load a curve file of this name from the plugin's data subdirectory (ARE/data/processor.adjustmentcurve).

• **out [double]:** This port provides the resulting output value.

#### **Event Listener Description**

- **displayGui:** An incoming event makes the GUI visible.
- hideGui: An incoming event makes the GUI invisible.
- **loadCurve:** An incoming event loads a curve of the current filename (as given in the plugin property or received from the input port "curveName").
- **saveCurve:** An incoming event saves the current mapping curve under the given filename. This event has teh same function like the "save"-button which is available in the GUI window of the plugin.

#### **Properties**

- **filename [string]:** The filename of the curve file. If the curve file exists in the plugin's subdirectory (ARE/data/processor.adjustmentcurve), this curve is loaded. If the file does not exist, a new curve can be drawn in the GUI and saved under this name into the plugin's data subfolder (ARE/data/processor.adjustmentcurve).
- **display GUI [boolean]:** If checked, the GUI of the adjustment curve plugin will be shown and the user can modify and save the curve in real time by dragging, creating or deleting curve points.
- **intMin [double]:** Sets the minimum value of the input range.
- **outMax [double]:** Sets the maximum value of the input range.
- **outMin [double]:** Sets the minimum value of the output range.
- outMax [double]: Sets the maximum value of the output range.
- mode [combobox]: "autoupdate min and max" modifies the input range if incoming values exceed the current minimum or maximum, "clip to min and max" which keeps the values as set by the min/max properties.
- fontSize [integer]: The size of the font for dispaying text in the GUI.
- caption [string]: The caption of the AdjustmentCurve GUI.

## Amplifier

#### **Component Type: Processor (Subcategory: Basic Math)**

This component takes an input and multiplies it with a gain factor set in the properties. It can be used to amplify or reduce signal values or to invert them.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	₹	Properties (Ctrl-P)	▼ џ
		componentType: Ample       Image: Search       Internal Properties	ifier.1
Amplifier.1		Component Descriptio	Multiplies input with a proporti
		Component Name	Amplifier.1
sigOut		ComponentType	processor 💌
		<ul> <li>Properties</li> </ul>	
		factor	1
	v		
	+	Properties (Ctrl-P)	Tinput Ports 2 Output Ports
ARE Status: Connected 100%		Ctrl +	-/-

Amplifier plugin

#### **Input Port Description**

• **sigIn [double]:** The input value to be amplified.

• **sigOut [double]:** The amplified output value.

#### **Properties**

• **factor [double]:** The gain factor that inputs are multiplied with.

## Arduino

# Component Type: Processors (Subcategory: Microcontroller Interface)

The Arduino component provides an interface to the Arduino Uno microcontroller and makes available analog inputs and digital inputs and outputs.



plugin

#### Requirements

An Arduino UNO microcontroller board has to be connected to an USB port. The Arduino CIM firmware must have been downloaded into the Arduino to communicate via the CIM protocol with the Arduino component.



Arduino UNO board

#### **Input Port Description**

- **pwm3 [integer]:** PWM channel 3, output value (range: 0-255). The output signal depends on the mode (PWM or servo PWM)
- pwm5 [integer]: PWM channel 5, output value (range: 0-255). The output signal depends on the mode (PWM or servo PWM)
   Warning:Due to hardware limitations, either PWM or IO mode is available for pin 5 AND 6. It is not possible to use these pins individually!
- pwm6 [integer]: PWM channel 6, output value (range: 0-255). The output signal depends on the mode (PWM or servo PWM)
   Warning:Due to hardware limitations, either PWM or IO mode is available for pin 5 AND 6. It is not possible to use these pins individually!

• **A0 - A5 [integer]:** The readings of the 6 analog channels of the Arduino (0-1023)

#### **Event Listener Description**

- **setPin2 setPin13:** An incoming event will set the corresponding digital output pin on the Arduino to high level (5V)
- **clearPin2 clearPin13**: An incoming event will set the corresponding digital output pin on the Arduino to low level (0V)

#### **Event Trigger Description**

- **pin2ChangedToLow pin13ChangedToLow**: This event is triggered if the corresponding input pin on the Arduino switches from high to low level (connected to 0V)
- **pin2ChangedToHigh pin13ChangedToHigh:** This event is triggered if the corresponding input pin on the Arduino switches from low to high level (connected to 5V)

#### **Properties**

- **uniqueID [integer]:** If more than one Arduino CIM is used, this property allows the identification of the module. (dynamic property)
- **periodicADCUpdate [integer]:** This property defines how often the ADC values are measured and put out on the ports A0-A5 (0=disable ADC)
- **pin2Mode pin13Mode [integer]:** This property defines the mode of the Arduino Pins. The value can be selected via a ComboBox. Possible selections are:
  - not used
  - Input without pullup resistor
  - Input with pullup resistor
  - output, default low
  - output, default high
  - PWM servo (PWM for servo driving, 1-2ms pulse), available only on pin 3,5 and 6
  - 500Hz PWM (normal 0-100% PWM), available only on pin 3,5 and 6

## **Audio Selector**

## **Component Type: Processor (Subcategory: Audio and Voice)**

This plug-in manages the audio tracks present in the data/music folder and different external request working as an interface with the wavefileplayer plug-in



AudioSelector plugin

#### Requirements

To work along with wavefileplayer plug-in.

• TrackName [string]: of the Track to be played. Supports value suggestions from ARE (dynamic property)

#### **Event Listener Description**

- **StartStop:** Togle between play stop state request.
- NextTrack: Play next track request.
- VolumeUp: Put the volume up request.
- VolumeDown: Put the volume down request.

#### **Event Trigger Description**

- **Play:** Play Track Request.
- Pause: Stop Track Requests.
- VolumeUp: Volume Up request.
- VolumeDown: Volume Down Request.

### Averager

#### **Component Type: Processor (Subcategory: Basic Math)**

The averager component takes an input stream and forwards the average of certain amount of buffered last inputs. The size of the buffer can be set via a property. The component can be used to eliminate spikes in an input data stream. Furthermore the unit can also be used as an accumulator which accumulates all inputs within a certain time interval.



Averager plugin

#### **Input Port Description**

• **input [double]:** This port reads the inputs to be averaged.

• **output [double]:** This port provides the current average of the buffered inputs.

#### **Properties**

- **mode [integer]:** Denotes the operating mode of the unit, three modes are available:
  - *average:* unit performs averaging calculations and emits double values average with integer.
  - *output:* unit performs calculations and rounds result to integer.
  - *accumulate:* unit accumulates inputs for certain amount of time.
- **bufferSize [integer]:** Specifies the size of the buffer in the averaging modes or the amount of milliseconds to accumulate in the accumulator mode.

# Basic Tremor Reduction algorithms

# **Component Type: Processor (Subcategory: Singal Shaping)**

This plugin contains three algorithms for user hand tremor reduction: Arithmetic Mean, Outlier Reduction and Exponential Smoothing. The Arithmetic Mean algorithm calculates the cursor position as an average of the past n cursor positions. N is defined by the bufferSize parameter. The Outlier Reduction algorithm keeps the maximum distance between two followed cursor positions. The maximum distance is defined by the maxDistance parameter. If the distance between two cursor positions is greater than maxDistance, it is reduced to the value of maxDistance. The Exponential Smoothing algorithm implements the Exponential smoothing technique. The factor parameter defines the Exponential Smoothing and the degree parameters define the degree of the equation.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Properties (Ctrl-P) Component: BasicTRale	gorithms.1	• #	
	<ul> <li>Internal Properties</li> </ul>			
	Component Descriptio	Basic tremor reduction algorithms		
BasicTRalgorith	Component Name	BasicTRalgorithms.1		
ms.l	ComponentType	processor	-	
inputX	<ul> <li>Properties</li> </ul>			
inputY	algorithm	Exponential Smoothing	-	
outputY bufferSize	eventsType	Absolute positions	-	
maxDistance	bufferSize	20	<b>a</b>	
factor	maxDistance	10		
	factor	0.5		
	degree	1	-	
*				
• •	TProperties (Ctrl-P)	Input Ports 🔄 Output Ports	Ba	

Tremor Reduction algorithms plugin

#### **Input Port Description**

- **inputX [integer]:** Input mouse X position.
- inputY [integer]: Input mouse Y position.
- **bufferSize [integer]:** The new buffer size value for the Arithmetic Mean algorithm.
- **maxDistance [double]:** The new maximum distance value for the Outlier Reduction algorithm.
- **factor [double]:** The new factor value for Exponential Smoothing algorithm.

- outputX [integer]: Output mouse X position.
  outputY [integer]: Output mouse Y position.

#### **Properties**

- **algorithm [integer]:** Defines the algorithm used for the tremor reduction.
- eventsType [integer]: Defines if the mouse coordinates are absolute or relative.
- **bufferSize [integer]:** The buffer size value for the Arithmetic Mean algorithm.
- **maxDistance [double]:** The maximum distance value for the Outlier Reduction algorithm.
- **factor [double]:** The factor value for Exponential Smoothing algorithm.
- **degree [integer]:** The degree of the equation for Exponential Smoothing algorithm.

## Benchmark

#### **Component Type: Processor (Subcategory: Others)**

This component may be used to perform benchmark of data throughput at a particular location of the model / design. It counts port activity of data and event ports per given time.



Benchmark plugin

#### **Input Port Description**

• **in [double]:** Input port for numeric values. Incoming activity increases the data counter.

- dataCount [integer]: The current value of the data counter.
- eventCount [integer]: The current value of the event counter.

#### **Event Listener Description**

- eventIncrease: Incoming events increase the event counter.
- **resetCounter:** An incoming event resets data counter and event counter to 0.

#### **Properties**

• **time [integer]:** The time period in milliseconds for measuring data activity and events. After the time has passed, the current values of data counter and event counter are provided at the output port, and the counters are reset to zero.

## **Blink Detection**

# **Component Type: Processor (Subcategory: DSP and Feature Detection)**

This component detects the shape that a blink produces in an electrooculogram signal. The plugin analyses the input samples and recognises both a single blink and a double blink. Here a single blink is defined by an action whereby both eyes are simultaneously and voluntary closed and open. A double blink refers to the repletion of this action twice in an consecutive way. When one of these conditions is found the corresponding event is fired. In addition, a true Boolean will be output to the corresponding output port.

	* 	Triggers BlinkDetected	Description	
	E.	BlinkDetected		
		Diminoreteeteo	Creating an event with the blink detection	
		DoubleblinkDetected	Creating an event with the double blink detection	
BlinkDetection 1	*	Properties (Ctrl-P)	<sup>™</sup> Input Ports ∮ Event Trigger	R
		- riopentes (Ctri-P)	- Inputroits / Event ingger	- 8
# Requirements

The input signal shall correspond to a 250-Hz sampled electro-oculogram signal, i.e., an output port of the <u>Enobio</u> component when the corresponding electrode is placed on the user's forehead.

• **input [integer]:** Input values that correspond to a 250-Hz sampled electro-oculogram signal.

# **Event Trigger Description**

- **BlinkDetected:** This event port fires an event if a blink is detected in the input sequence of integers.
- **DoubleblinkDetected:** This event port fires an event if a double blink is detected in the input sequence of integers.

# **Blink Detector**

# **Component Type: Processor (Subcategory: DSP and Feature Detection)**

This component detects the shape that a blink produces in an electrooculogram signal. The plugin analyses the input samples and recognises both a single blink and a double blink. Here a single blink is defined by an action whereby both eyes are simultaneously and voluntary closed and open. A double blink refers to the repletion of this action twice in an consecutive way. When one of these conditions is found the corresponding event is fired. Note that the plugin will only detect simple or double blinks with a duration smaller than BlinkLength miliseconds. Similarly, it will only detect double blinks that are separated by less than DoubleBlinkSeparation miliseconds. In addition, the "strength" of the blinks is defined by the maxThreshold and minThreshodld properties:



signal double blink

EEG



Double blink derivative

In order to propperly configure these 4 properties, they should be previously obtained by running the <u>BlinkDetectorTrainer</u> plugin for each different subject. In order to correctly detect simple and double blinks, the input signal is internally decimated by a factor of 11 and derivated. For debugging purposes, the decimated samples and the derivated samples are output to the corresponding output ports.

X   🛄 👩 🤗   AsTeRICS Confi	iguration Suite			×
Y System Componer	nts Edit Misc.			
Sensors Processors Actuators	E M Specials Saved Groups			
Components				
Deployment (Ctrl-D) =	Properties (Ctrl-P)		<b>•</b> p	4
	Component: BlinkDetector.1			Q
	Search Search			tput
	<ul> <li>Internal Properties</li> </ul>		-	Po
E	Component Name	BlinkDetector.1		rt.
	Component Type	asterics.BlinkDetector		
	Component Descriptio	Blink detector. It detects both a simple and a double blinks		
	Component Class	processor [	•	
	<ul> <li>Properties</li> </ul>		_	
	SampleRate	250	÷	
BlinkDetector.1	maxThreshold	10000	=	
	BlinkLength	300	1	
Filtered Samp	DoubleBlinkSeparatior	500	1	
Differential 🚛	minThreshold	-10000 8	Ŧ	
<	🐨 Properties (Ctrl-P)	🖥 Input Ports 🔰 Event Trigger	_	20
ARE Status: Disconnected	100%	Ctrl +/-		

Blink Detector plugin

# Requirements

The input signal shall correspond to a 250-Hz sampled electro-oculogram signal, i.e., an output port of the <u>Enobio</u> component when the corresponding electrode is placed on the user's forehead.

• **input [integer]:** Input values that correspond to a 250-Hz sampled electro-oculogram signal.

### **Output Port Description**

- **Filtered Sample [integer]:** For each input sample, this output port delivers the decimated sample with a decimation factor of 11.
- **Differential [integer]:** For each input sample, this output port delivers the derivated sample (after the decimation).

# **Event Trigger Description**

- **BlinkDetected:** This event port fires an event if a blink is detected in the input sequence of integers.
- **DoubleblinkDetected:** This event port fires an event if a double blink is detected in the input sequence of integers.

## **Properties**

- **sampleRate [integer]:** Sample rate of the input signal in Hertz.
- **maxThreshold [integer]:** Positive threshold for a peak in the derivated signal to be considered as a potential blink (see "Double blink derivative" figure).
- **minThreshold [integer]:** Negative threshold for a peak in the derivated signal to be considered as a potential blink (see "Double blink derivative" figure).
- BlinkLength [integer]: Duration of one blink in miliseconds (see "Double blink derivative" figure).
- **DoubleBlinkSeparation [integer]:** Separation (in miliseconds) between two blinks that correspond to a double blink (see "Double blink derivative" figure).

# **Blink Detector Trainer**

# **Component Type: Processor (Subcategory: DSP and Feature Detection)**

This component calculates the maxThreshold, minThreshold, BlinkLength and DoubleBlinkSeparation customized properties of the <u>Blink Detector</u> plugin for each specific subject. For a description of the meaning of these properties, please see <u>Blink Detector</u>. The training of the system consists on asking the subject to follow a protocol. This protocol consists on a series of 5 simple blinks and 5 double blinks. Note that the subject can perform only **one** simple (or double) blink each time the protocol indicates so through its Protocol port. When the protocol finishes, the results show up through the Results output port.



Detector plugin

# Requirements

The input signal shall correspond to a 250-Hz sampled electro-oculogram signal, i.e., an output port of the <u>Enobio</u> component when the corresponding electrode is placed on the user's forehead.

• **input [integer]:** Input values that correspond to a 250-Hz sampled electro-oculogram signal.

### **Output Port Description**

- **Protocol [string]:** Actions to be performed by the user. Note that the user must perform just one blink (or double blink) each time the corresponding message is delivered through this port.
- **Results [string]:** Final parameters calculated for the specific subject. They will delivered when the protocol has finished.

## **Event Listener Description**

- **StartProtocol [integer]:** Starts the training protocol. The actions to be performed by the subject will be delivered through the Protocol port.
- **StopProtocol:** Stops the training protocol.

## **Properties**

- **sampleRate [integer]:** Sample rate of the input signal in Hertz.
- **language [list]:** Language of the messages thrown through the Protocol port while the protocol is running. The user can chose English or Spanish.

# Comparator

#### **Component Type: Processor (Subcategory: Basic Math)**

This component compares the numerical values of two input ports and provides output depending on the result of the comparison.

GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		<b>•</b> 4	
		Component: Comparate	or.1		
		Parch Search			
		<ul> <li>Internal Properties</li> </ul>			
Comparator.1		Component Class	processor	-	
		Component Name	Comparator.1		
		Component Type	asterics.Comparator		
		Component Descriptic	Comparator for two inputs		
		<ul> <li>Properties</li> </ul>			
		condition	a greater b	•	
		outputMode	output a if condition met		
		eventMode	create event only if condition changes		
		threshold	1		
		threshold2	10		
	-				
-	+	Properties (Ctrl-P)	Event Trigger 🖉 Output Ports 🏾 🗇 Input	Ports	
		Comparator plu	Igin		

- inA [double]: Input port for signal a. This input port supports synchronization
- inB [double]: Input port for signal b. This input port supports synchronization

# **Output Port description**

• **out [double]:** Output port of the comparator (value according to condition and operational mode).

# **Event Trigger Description**

- **conditionTrue:** This event is triggered when the comparator condition switches from false to true.
- **comparatorFalse:** This event is triggered when the comparator condition switches from true to false.

# **Properties**

- **condition [integer]:** Defines the condition to be met. Available conditions are "a greater b", "a equals b", "a lower b", "a greater threshold", "a equals threshold", "a lower threshold" and "a between threshold and threshold2".
- **outputMode [integer]:** Defines the mode of operation, respectively which values are put to the output port of the plugin. Following modes are available: "output min", "output max" and "output a if condition met".
- **eventMode [integer]:** Defines the mode of event generation (if events are created on every comparison of input values or only if the output condition changes).
- **threshold [double]:** Defines the threshold value for the condition modes "a greater than threshold", "a equals threshold" and "a lower than threshold".
- **threshold2 [double]:** Defines the threshold2 value for the condition mode "a between threshold and threshold2".

# **Compute Bandpower**

# **Component Type: Processor (Subcategory: DSP and Feature Detection)**

This component computes the power that an input signal has in a specific frequency band. The plugin stores as many values as the DataLen property indicates before providing a new value in the output port. This solution is based on the FFT so only the bins corresponding to the specified band are considered. This approach removes the contribution of the out-band frequencies to the final value. This approach improves the power output of the filter plugin which uses the filtered signal for computing the output value so the frequencies out of the pass band contributes to the final value since the filter is implemented as FIR filter with a finite number of coefficients so the frequency response will not never be perfect.



Compute Bandpower plugin

# Requirements

The values in the input port shall correspond to a time series.

• **input [double]:** Input port for the values of time series which power in band is computed.

#### **Output Port description**

• **output [double]:** Output of the value that corresponds to the power of the signal present in the last DataLen samples. If the input signal is expressed in volts, then the output is expressed in squared volts.

# **Properties**

- **DataLen [integer]:** Defines the length of the time series over which the band power computation is performed. Only power of two values are allowed for this property.
- **SampleRate [integer]:** Defines the sample rate of the input time series. It is defined in samples per second.
- **StartBandFrequency [integer]:** Defines the beginning of the band to be analysed. It is defined in Hertz.
- EndBandFrequency [integer]: Defines the end of the band to be analysed. It is defined in Hertz.

# **Constant Dispatcher**

#### **Component Type: Processor (Subcategory: Basic Math)**

This component sends double values from the chosen slot.



ConstantDispatcher plugin

• **slotDispatch [integer]:** Sends the value from the slot defined by number.

# **Output Port Description**

• **output [double]:** The port for the output value.

## **Event Listener Description**

- **dispatchSlot1...dispatchSlot20** : Sends the double value from the slot: 1...20.
- **dispatchNextSlot:** Sends double value from the next slot.
- dispatchPreviousSlot: Sends double value from the previous slot.
- **dispatchSlotSeries:** Sends slots values in sequence from slot 1 to slot defined by the Number property with the delay defined by the Delay property.

# **Properties**

- number [integer]: Number of used slots.
- **delay [integer]:** Delay in ms used in sequence slot dispatch.
- slot1...slot20 [double]: The slot for the value: 1...20.
- **autosendSlot [integer]:** Number of slot which is automatically sent at start (0=disable).

# **DataType conversion**

# **Component Type: Processor (Subcategory: Data Converters)**

#### This is a component for testing implicit data type conversion.

The component sends the given property value converted to the data type of the used output port. A value received at any of the input ports is printed out at the console.

- inByte [byte]: Input data of type byte.
- inChar [char]: Input data of type char.
- inInteger [integer]: Input data of type integer.
- **inDouble [double]:** Input data of type double.
- inString [string]: Input data of type string.
- inBoolean [boolean]: Input data of type boolean.

### **Output Port Description**

- **outByte [byte]:** Output data of type byte.
- outChar [char]: Output data of type char.
- outInteger [integer]: Output data of type integer.
- **outDouble [double]:** Output data of type double.
- **outString [string]:** Output data of type string.
- outBoolean [boolean]: Output data of type boolean.

# Deadzone

# **Component Type: Processor (Subcategory: Signal Shaping)**

The purpose of this component is to define active and passive areas (zones) for one- or two dimensional sensor values. This function could also be called "resting zone" or "centerzone". It can be useful for example when a mouse pointer should not move before a certain value of a sensor (displacement of an analogue sensor, strength of activity) is reached.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	➡ Properties (Ctrl-P)		<b>▲</b> [
Deadzone.1	Component: Deadzor	Component: Deadzone.1	
	Search	Search	
	Internal Properties	Internal Properties	
	Component Descripti	Defines active/passive Zone for x/y values	
	Component Type	asterics.Deadzone	
	Component Name	Deadzone.1	
inY autY	Component Class	processor	-
	<ul> <li>Properties</li> </ul>	<ul> <li>Properties</li> </ul>	
	xCenter	0	
	yCenter	0	
	radius	10	
	mode	only inner values	-
	•		
	Deadzone p	lugin	
- inX [double]: Input of x values. This input port supports synchronization
- inY [double]: Input of y values. This input port supports synchronization
- **radius [double]:** The radius of the active/passive zone around the centre point.

- outX [double]: Output of modified x values.
  outY [double]: Output of modified y values.

#### **Event Listener Description**

• **setCenter:** An incoming event stores the current x- and y- input values to represent the centre (It defines them as offset values). This can be useful for sensor calibration because it defines the "baseline" or resting position.

#### **Event Trigger Description**

- **enterZone:** This event is triggered when the x or x- and y- values enter the specified radius around the centre.
- **exitZone:** This event is triggered when the x or x- and y- values leave the specified radius around the centre.

- **xCenter [double]:** This value defines the x- position of the centre (the middle of the active/passive zones).
- **yCenter [double]:** This value defines the y- position of the centre (the middle of the active/passive zones).
- radius [double]: The radius of the active/passive zone around the centre point.
- **mode [integer]:** Selects the mode of operation of the centerzone component, following modes are available:
  - "only inner values": x- and y- values are passed to the output ports only if the distance to the centre is lower than the given radius.
  - *"only outer values":* x- and y- values are passed to the output ports only if the distance to the centre is greater than the given radius.
  - "deadzone": x- and y- values are passed to the output ports only if the distance to the center is lower than the given radius, and additionally a correction of the values is performed so that they start with 0 when leaving the inner zone. This is useful for defining a "deadzone" for sensor values, where an inactive area shall be provided and no sudden acceleration is desired when leaving this inactive area.

### **Decimation**

#### Component Type: Processor (Subcategory: Basic Math)

The decimation operation performed by this component consists in an anti-aliasing low band pass filter plus a down-sampling. The component outputs a computed sample after receiving a certain number of input samples according to the down-sampling-ratio property value. So the resultant signal is like the original signal, but sampled to a lower ratio determined by the mentioned property.



Decimation plugin

• **input [double]:** Input port for the signal to be decimated.

• **output [double]:** Output port of the decimated signal.

• **DownSamplingRatio [integer]:** Defines the ratio between the number of samples in the input and output ports.

## Delay

#### **Component Type: Processor (Subcategory: Basic Math)**

This plugin passes received double values after a defined delay.



Delay plugin

• in (double): Input port for the incoming signal.

• out (double): Output port for the delayed signal.

• **delay [integer]:** Delay between input and output signal (in milliseconds).

### Derivative

#### **Component Type: Processor (Subcategory: Basic Math)**

This component computes the first derivative operation of the input signal by using an approximation by a numerical differentiation method using five points. Due to the fact that the component takes into consideration the previous four samples plus the current one, it will output the derivative value corresponding to the centre sample of the five ones. It means two samples earlier from the one that is received in the input port.



• input [double]: Input port for the signal to be derived.

• **output [double]:** Output port of the first derivative of the input signal. The output values present a delay of two samples because of the reason given in the general description.

• **SampleFrequency [integer]:** Defines the frequency which the input signal is sampled, so the distance between two consecutive samples, which is used by the component, is defined.

## Differentiate

#### **Component Type: Processor (Subcategory: Basic Math)**

The differentiate component outputs the difference of the previous to the current input value on the output port. This functionality can be considered as a simplified implementation of the derivative component.



• in [double]: Input port for signal.

• **out [double]:** Output of difference value.

• **resetValue [double]:** The value that is used in the first difference calculation after start.

## Dissimilarity

# **Component Type: Processor (Subcategory: DSP and Feature Detection)**

This component computes the Euclidean distance between two input vectors. The component stores the input values from both input ports until DataLen values have been received, then the computation is performed and sent to the output port.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		<b>→</b> ‡
	* H	componentType: Dissin	nilarity.1	
		Search		
		<ul> <li>Internal Properties</li> </ul>		
		Component Descriptic	Dissimilarity processor	
Dissimilarity.1		Component Name	Dissimilarity.1	
input1		ComponentType	processor	*
output		<ul> <li>Properties</li> </ul>		
		DataLen	100	<b></b>
	F	Properties (Ctrl-P)	Input Ports 2 Output Ports	

Dissimilarity plugin

- input1 [double]: Input port for the first signal.
  input2 [double]: Input port for the second signal.

• **output [double]:** Output port for the dissimilarity computation. A value is provided every time DataLen samples arrive to the input ports.

• **DataLen [integer]:** Defines the length of the signals over which the dissimilarity is computed.

## **Double To String**

# **Component Type: Processor (Subcategory: Data Converters)**

This component converts the double values at the input port to string values at the output.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		<b>▼</b> ₽	
	•	componentType: Doub	bleToString.1		
			Search		
		<ul> <li>Internal Properties</li> </ul>			
DaubleTeString		Component Descriptio	Converts double to string		
		Component Name	DoubleToString.1		
1		ComponentType	processor	*	
	Ŧ	Properties (Ctrl-P)	Input Ports 3 Output Ports		

DoubleToString plugin

• input [double]: Input port for the double values to be converted.

• **output [string]:** Output port for the converted string values.

## **ECMAScriptInterpreter**

#### **Component Type: Processor (Subcategory: Scripting)**

This component is a general purpose processor that can relays the input and incoming events to a script compatible to the ECMA script specification (e.g. JavaScript). The script is specified by the property scriptname. If the property is left empty, the component will load the file "script.js" from local storage. If this file does not exist, the component will generate the file in local storage and fill it with a default "pass-through" script.

There are certain constraints for the script:

- the script has to contain an object named scriptclass.
- the object has to implement a method dataInput(input\_index, input\_data)
- the object has to implement a method eventInput(event\_index)

The script is provided with the following external variables:

- output: an array of size 8 representing 8 IRuntimeOutputPorts
- eventout: an array of size 8 representing 8 IRuntimeEventTriggererPorts
- property: an array of size 8 holding strings with the property inputs from the components property fields

The sendData method of the output variables has to be called with a string. If necessary this needs to be converted into a Java string, this can be done like this:

```
str = new java.lang.String(in_data);
output[in_nb].sendData(str.getBytes());
```

For more information please see a demo script in the plugin source code!

	Dearch			
ECMAScriptInter	<ul> <li>Editor Properties</li> </ul>			
preter.1	EnableGrid	×		
inputPort1	ShowGrid	X		
outputPort1	GridSteps	small 💌		
outputPort2	ScreenRes	SixteenNine		
outputPort3	<ul> <li>ARE Properties</li> </ul>			
outputPort4	Decoration	X		
outputPort5	Fullscreen			
	AlwaysOnTop			
outputPort7	ToSystemTray			
inputPort8	ShowControlPanel	X		

ECMAScriptInterpreter plugin

• inputPort1 - inputPort8 [string]: input ports for script parameters

• outputPort1 - outputPort8 [string]: output ports for script results

### **Event Listener Description**

• **elpPort1** - **elpPort8**: 8 event listener ports which can be used by the script code.

### **Event Trigger Description**

• **etpPort1 - elpPort8:** 8 event trigger ports which can be used by the script code.

- **scriptname [string]:** a valid filename of an ECMA-compatible script (e.g. Javascript) which shall be interpreted
- value1 value8 [string]: 8 properties which can be used by the script.

## **Event Block**

# Component Type: Processor (Subcategory: Event and String Processing)

This plugin, depending on its state, can pass or block events from the input port.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		<b>▼</b> ₽
	A III	Component: EventBlock	k.1	
		Internal Properties		
EventBlock1		Component Descriptic	Pass or block event	
		Component Name	EventBlock.1	
		ComponentType	processor	-
		<ul> <li>Properties</li> </ul>		
		block		
		blockAfterEvent		
	-			
		Troperties (Ctrl-P)	🐔 Event Listener 📝 Event Trigger	

EventBlock plugin
### **Event Listener Description**

- input: Input port for the events.
- **pass:** Pass the events.
- **block:** Block the events.
- **change:** Change the state of component: pass/block to the opposite.

### **Event Trigger Description**

• **output :** Output port for events.

- **block [boolean]:** If is set to true, the component will block the events after start.
- **blockAfterEvent [boolean]:** If is set to true, the component will block the events after passing one event.

### **Event Cascade**

## **Component Type: Processor (Subcategory: Event and String Processing)**

The EventCascade component can be used to generate a sequence (or loops) of events with selectable delay times. The sequence can be started and stopped via incoming events.



EventCascade plugin

### **Event Listener Description**

- **nextEvent:** Triggers the next event in the event cascade (with wraparound)
- **previousEvent:** Triggers the previouse event in the event cascade (with wrap-around)
- **startCascade:** Starts the event sequence
- **startCascade:** Stops the event sequence
- reset: Sets the next event number to zero (first in the sequence)

### **Event Trigger Description**

• trigger1 - trigger15: The available event trigger outputs

- **activeTriggers [integer]:** How many event triggers are active (defines the wrap-around)
- **loops [integer]:** How many loops will be performed before the event sequence stops (0=endless loop)
- **autoStart [boolean]:** Defines if the event cascade will be automatically started at model startup
- delayBeforeTrigger1 delayBeforeTrigger15[integer]: Delay time before the corresponding trigger event is created

### **Event Counter**

## Component Type: Processor (Subcategory: Event and String Processing)

This component counts events. It can increase and decrease a counts via incoming events.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)	<b>-</b> ↓
		Component: EventCour	nter.1
		Search	
		<ul> <li>Internal Properties</li> </ul>	
		Component Name	EventCounter.1
		Component Type	asterics.EventCounter
		Component Descriptio	Two-direction event counter
EventCounter.1		Component Class	processor
		<ul> <li>Properties</li> </ul>	
output 🛑		mode	no limit
		minValue	0 😫
		maxValue	1000 🚖
		initialValue	0 🗧
		wrapAround	
		sendInitialValue	
+		🕈 Properties (Ctrl-P) 🦪	Event Listener Cutput Ports

EventCounter plugin

### **Input Port Description**

• **setValue [integer]:** Sets the event counter to the incoming value. Note that this value is not propagated to the output port (to avoid loops).

### **Output Port Description**

• **output [integer]:** Sends the number of events.

### **Event Listener Description**

- increase: Increases the number of events.
- **decrease:** Decreases the number of events.
- resetToZero: Sets the event counter to zero.
- resetToInitial: Sets the event counter to the initial value (property).
- sendNow: sendes the current value of the counter to the output port

- mode [integer]: Defines counting mode:
  - *no limit:* The component counts events without any limitation.
  - *limit maximum:* In this mode, the maximum value of the counter is limited by the maxValue property.
  - *limit minimum:* In this mode, the minimum value of the counter is limited by the minValue property.
  - *limit minimum and maximum:* In this mode, both maximum and minimum values of the counter are limited by the maxValue amd minValue properties.
- minValue [integer]: Defines the minimum value of the counter.
- maxValue [integer]: Defines the maximum value of the counter.
- initialValue [integer]: Defines the initial value of the counter.
- wrapAround [boolean]: if selected and the appropriate mode is set, exceeding the maximum value will wrap to the minimum value and vice versa.
- **sendInitialValue [boolean]:** if selected, the initial value is sent at the startup.
- **autoSend [boolean]:** if selected, the changes of the event counter are sent immediately to the output port

## **Event Delay**

# Component Type: Processor (Subcategory: Event and String Processing)

This plugin passes received events after a defined delay.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		<b>•</b> ‡		
	*	Component: EventDela	ay.1			
	=	Search				
		<ul> <li>Internal Properties</li> </ul>				
EventDelay.1		Component Description	Delalys events			
		Component Name	EventDelay.1			
		ComponentType	processor	-		
		<ul> <li>Properties</li> </ul>				
		delay	100	۲		
		Description (Chill D)	/ French Linkson / French Tringer	_		
		Properties (Ctrl-P)	Event Listener Zevent Trigger			
EventDelay plugin						

### **Event Listener Description**

• **input:** Input port for the events.

### **Event Trigger Description**

• **output :** Output port for events.

• **delay [integer]:** Delay between input and output event (in milliseconds).

## **Event Dispatcher**

# **Component Type: Sensor (Subcategory: Event and String Processing)**

This component translates incoming strings to events. Ten string slots for commands and ten associated Event Trigger Ports are available. This component can be useful to generate an event from a string command (which is generated by OSKA or another string-sending component).



EventDispatcher plugin

### **Input Port Description**

• **cmd [string]:** The incoming command string.

### **Event Trigger Description**

• recognizedCommand1 - recognizedCommand10: the events triggered by the recognized commands

• **command1 - command10 [string]:** Ten string slots for commands. If an incoming string matches one of these property values, the associated event trigger is raised.

## **Event Flip Flop**

This component stores the state, driven by an event. When the event-in event is received and the internal status is 1, event-out1 is fired and the internal status set to 2. When the event-in event is received and the internal status is 2, event-out2 is fired and the internal status set to 1.

# **Component Type: Processor (Subcategory: Event and Signal Processing)**



Flip Flop plugin

### **Event Listener Ports**

- **event-in:** Event input to change the state of the flip-flop and fire an out-event.
- **selectOut1:** selects state 1 (event-out1 trigger port will send the next incoming event)
- **selectOut2:** selects state 2 (event-out2 trigger port will send the next incoming event)

### **Event Trigger Description**

- event-out1: Event fired, if event-in received and stored state is 1.
- event-out2: Event fired, if event-in received and stored state is 2.

No Properties.

## **EventRouter**

This component allows routing up to 6 incoming events to one of 8 corresponding output (trigger) event ports.

# **Component Type: Processor (Subcategory: Event and Signal Processing)**

GUI Designer (Ctrl-G) Deployment (Ctrl-D)	Ŧ	Event Listener	<b>→</b> ‡
	*	Listeners	Description
		eventIn	the events to be routed
		select1	select route 1
EventRouter.1		select2	select route 2
		select3	select route 3
		select4	select route 4
		select5	select route 5
		select6	select route 6
		select7	select route 7
		select8	select route 8

Router plugin

### **Event Listener Description**

- eventIn1-6: The incoming events to be routed.
- **select1-select8:** selects one of the 8 corresponding event routes (trigger output ports) events entering the eventIn ports
- selectNext: selects the next event route
- selectPrevious: selects the next event route

### **Event Trigger Description**

- eventOut1-eventOut8: 8 event trigger ports where the events entering the listener port eventIn can be routed to.
- eventOut2\_1-eventOut2\_8: 8 event trigger ports where the events entering the listener port eventIn2 can be routed to.
- eventOut3\_1-eventOut3\_8: 8 event trigger ports where the events entering the listener port eventIn3 can be routed to.
- eventOut4\_1-eventOut4\_8: 8 event trigger ports where the events entering the listener port eventIn4 can be routed to.
- eventOut5\_1-eventOut5\_8: 8 event trigger ports where the events entering the listener port eventIn5 can be routed to.
- eventOut6\_1-eventOut6\_8: 8 event trigger ports where the events entering the listener port eventIn6 can be routed to.

activeRoutes (integer): number of active routes (relevant for the selectNext and selectPrevious functions and the wrapAround feature)
wrapAround (boolean): if selected selectNext srats again at the first route when the last active route was reached, and selectPrevious continues at the last active route when triggered at the first route.

## **EventStateMachine**

This component allows definition of a sequence of events (up to 9 different incoming events are supported). If the sequence appears at the event listener ports in the given order, an event is fired by the EventStateMachine plugin. Optionally, a timing can be specified for a valid event sequence.

# Component Type: Processor (Subcategory: Event and Signal Processing)



State Machine plugin

### **EventStateMachine Description**

- in1-in9: Incoming events
- **reset:** Resets the current state detection to the first element of the sequence

### **Event Trigger Description**

- **finalStateReached:** fired if the last element of the sequence is reached.
- **stateError:** fired if an incoming event does not fit the sequence.
- **stateChanged:** fired if a new state of the sequence is reached.

- **stateSequence [string]:** A sequence of digits separated by commas, representing the order of incoming events, for example: "1,5,2,3,1".
- stateTiming [string]: An (optional) sequence of timings for the above events, specified in min/max milliseconds for a state transition. One timing information consists of a numeric value for the maximum time or two numeric values separated by a forward slash (/) for minimum and maximum time. For example the timing "1000/2000,500/1000,x,3000" defines that the transition from first to second state my take a minimum of 1000 milliseconds and a maximum of 3000 milliseconds; The transition to the next state may trake 500-1000 milliseconds, the next transition has no timing contraints, the next transition may take up to 3000 milliseconds. If this property is left empty, no timing constraints are active for the event state transitions.
- **resetOnIncorrectEvent [boolean]:** if checked, an incoming event which does not fit the current element of the sequence will reset the state detection to the first element of the sequence.

## FABI

# **Component Type: Processor (Subcategory: Microcontroller Interface)**

This component provides an interface for a FABI controller and enables the programming process. It is possible to configure the buttons which are connected to the FABI controller and store the set to the built in EEPROM.

~	Component: FABI.1	
FABI.1	Search	]
	Component Name	FABI.1
ButtonMode	Component Type	asterics.FABI
SlotSaveName	Component Descriptio	Plugin for connection to F
SlotLoadNam	Component Class	processor
MoveMouseX	<ul> <li>Properties</li> </ul>	
MoveMouseY	Stepsize	3
Text	COMPort	1
key		

### Requirements

The Plugin requires a FABI2.0 compatible Version as well as the right COM Port. The Baudrate is predefined at 9600 Baud.

#### **Input Port Description**

- ButtonMode [integer]: Number of the button
- SlotSaveName [string]: Name of the slot to save the set of modes
- SlotLoadName [string]: Name of the slot to load the set of modes
- MoveMouseX [integer]: Number of pixels to move the mouse in X direction (right)
- **MoveMouseY [integer]:** Number of pixels to move the mouse in Y direction (down)
- Text [string]: Text for the KeyWrite command
- key [string]: Text for the KeyPress command (e.g. AT KP KEY UP presses the "Cursor-Up" key, AT KP KEY CTRL KEY ALT KEY DELETE presses all three keys) supported key identifiers for key press command (AT KP): KEY A KEY B KEY C KEY D KEY E KEY F KEY G KEY H KEY I KEY J KEY K KEY L KEY M KEY N KEY O KEY P KEY OKEY RKEY SKEY TKEY UKEY VKEY WKEY X KEY Y KEY Z KEY 1 KEY 2 KEY 3 KEY 4 KEY 5 KEY 6 KEY 7 KEY 8 KEY 9 KEY 0 KEY F1 KEY F2 KEY F3 KEY F4 KEY F5 KEY F6 KEY F7 KEY F8 KEY F9 KEY F10 KEY F11 KEY F12 KEY RIGHT KEY LEFT KEY DOWN KEY UP KEY ENTER KEY ESC KEY BACKSPACE KEY TAB KEY HOME KEY PAGE UP KEY PAGE DOWN KEY DELETE KEY INSERT KEY END KEY NUM LOCK KEY SCROLL LOCK KEY SPACE KEY CAPS LOCK KEY PAUSE KEY SHIFT KEY CTRL KEY ALT KEY RIGHT ALT KEY GUI KEY RIGHT GUI

### **Output Port Description**

- List [string]: Lists the slots stored in the EEPROM of the FABI controller
- **ID [string]:** Shows the ID of the FABI version
### **Event Listener Description**

- **ID** : identification string will be returned (e.g. "FABI Version 2.0") Corresponding FABI command is : "AT ID"
- ButtonMode : button mode setting for a button (e.g. AT BM 2 -> next command defines the new function for button 2) Corresponding FABI command is : "AT BM num"
- ClickLeft : click left mouse button Corresponding FABI command is : "AT CL"
- ClickRight : click right mouse button Corresponding FABI command is : "AT CR"
- ClickDoubleLeft : click double with left mouse button Corresponding FABI command is : "AT CD"
- ClickMiddle : click middle mouse button Corresponding FABI command is : "AT CM"
- **PressLeft :** press/hold the left mouse button Corresponding FABI command is : "AT PL"
- **PressRight :** press/hold the right mouse button Corresponding FABI command is : "AT PR"
- **PressMiddle :** press/hold the middle mouse button Corresponding FABI command is : "AT PM"
- **ReleaseLeft :** release the left mouse button Corresponding FABI command is : "AT RL"
- **ReleaseRight :** release the right mouse button Corresponding FABI command is : "AT RR"
- ReleaseMiddle : release the middle mouse button Corresponding FABI command is : "AT RM"
- WheelUp : move mouse wheel up Corresponding FABI command is : "AT WU"
- WheelDown : move mouse wheel down Corresponding FABI command is : "AT WD"
- MoveMouseX : move mouse in x direction (e.g. AT X 4 moves 4 pixels to the right) Corresponding FABI command is : "AT MX num"
- MoveMouseY : move mouse in y direction (e.g. AT Y -10 moves 10 pixels up)
   Corresponding FABL command is : "AT MX num"

Corresponding FABI command is : "AT MY num"

- **KeyWrite :** keyboard write text (e.g. AT KW Hello! writes "Hello!") Corresponding FABI command is : "AT KW text"
- **KeyPress** : key press: press/hold all keys identified in text (e.g. AT KP KEY\_UP presses the "Cursor-Up" key, AT KP KEY\_CTRL KEY\_ALT KEY\_DELETE presses all three keys) The possible KeyPress commands are described in detail at the input port "key"! Corresponding FABI command is : "AT KP text"
- KeyRelease : key release: releases all keys identified in text Corresponding FABI command is : "AT KR text"
- KeyReleaseAll : release all: releases all currently pressed keys and buttons
  - Corresponding FABI command is : "AT RA"
- Save : save settings and current button modes to next free eeprom slot under given name (e.g. AT SAVE mouse1) Corresponding FABI command is : "AT SAVE text"
- Load : load button modes from eeprom slot (e.g. AT LOAD mouse1 -> loads profile named "mouse1")
   Corresponding FABI command is : "AT LOAD text"
- List : list all saved mode names Corresponding FABI command is : "AT LIST"
- Next : next mode will be loaded (wrap around after last slot) Corresponding FABI command is : "AT NEXT"
- Clear : clear EEPROM content (delete all stored slots) Corresponding FABI command is : "AT CLEAR"
- Idle : idle command (no operation) Corresponding FABI command is : "AT IDLE"

- **Stepsize [integer]:** set mouse wheel stepsize (e.g. AT WS 3 sets the wheel stepsize to 3 rows) The stepsize is set when the plugin is started
- **COMPort [integer]:** COM Port of FABI.

### FabiCronusMax

## **Component Type: Processor (Subcategory: Microcontroller Interface)**

This component provides an interface for the FABI controller and the CronsuMax USB stick. It enables the programming process of the FABI as well as the configuration for all supported gaming consoles. It is possible to configure the buttons which are connected to the FABI controller and store the set to the built-in EEPROM. Via the plugin the CronusMax stick controls the selected gaming device through the external buttons connected to the FABI. The set of buttons can be defined in a configuration file and modified for each game and console.



FabiCronusMax plugin



### Requirements

The Plugin requires a FABI2.0 compatible Version as well as the right COM Port. The Baudrate for the FABI controller is predefined at 9600 Baud. Additionally a CronusMax device is necessary as well as the software GTuner (GTuner download)

- InConsole [string]: Input of the selected Console (e.g. PS3)
- InGame [string]: Input of the selected Game (e.g. Need For Speed)
- InMode [Integer]: Input of the selected Mode (e.g. 1)

#### **Output Port Description**

- **OutConsole [string]:** Output of the selected Console (e.g. PS3)
- **OutGame [string]:** Output of the selected Game (e.g. Need For Speed)
- **OutMode [Integer]:** Output of the selected Mode (e.g. 1)
- OutModel [string]: Output for the next model to be started
- **OutButtons [string]:** Output for the buttons in current mode, separated through a comma ','

### **Event Listener Description**

- **ModeSwitcher:** Switch between the modes.
- GameSwitcher: Switch between the Games.
- **ConsoleSwitcher:** Switch between the consoles.

### **Event Trigger Description**

- **Busy:** Triggered if Fabi is busy.
- Ready: Triggered if Fabi is ready.
- loadModel: Triggered to load new model.

- ComPort [integer]: COM Port of FABI.
  ModeFilePath [string]: Path to the file with the configuration.

### **Configuration file**

The configuration file has to be a \*.CSV file and the data has to be separated with ','. The following two lines show the structure of a configuration file:

XBOXONE, BattleField, Mode, KEY\_A, KEY\_B, Mode, KEY\_1, KEY\_2 XBOX360, Formel1, Mode, KEY\_A, KEY\_B, KEY\_C, KEY\_D, KEY\_E

The first field defines the console and the second one defines the game. "Mode" signals that the following fields are the keycodes which are connected to the buttons in the right order. There can be up to 10 modes per game and up to 6 buttons per mode.

## Filter

# **Component Type: Processor (Subcategory: DSP and Feature Detection)**

This plugin implements a FIR Filter

loyment (Ctrl-D) GUI Designer (Ctrl-G)	Propertie	es (Ctrl-P)		• 1
	Comp	onent: Filter.1		
	A Company	↓ Search		
	Inte	rnal Properties	N	
	Comp	oonent Name	Filter.1	
	Comp	oonent Type	asterics.Filter	
	Comp	oonent Descriptio	Filter plugin	
Filter.1	Comp	oonent Class	processor	*
( Innut	🔺 Pro	perties		
Output	Order	·	500	•
SignalPower	Cutof	fFreq1	0	
	Cutof	fFreq2	0	
	Samp	lingRate	500	-
	Туре		Low pass	-
	Signa	IPowerUpdateRa	[1	-
	Signa	IPowerBufferSize	125	

Filter plugin

• Input [double]: The signal to be filtered.

### **Output Port Description**

- Output [double]: The filtered signal.
  SignalPower [double]: Signal Power on the band pass.

- **Order [integer]:** Order of the filter. It is recommended to use orders around the sampling rate.
- **CutoffFreq1 [double]:** Cutoff frequecy for low and high pass filter types. In case of band pass filters it is the low cutoff frequency of the band.
- **CutoffFreq2 [double]:** In case of band pass filters it is the high cutoff frequency of the band.
- **SamplingRate [integer]:** Sampling rate of the input signal.
- **Type [integer]:** Type of filter (low, high or band pass filter).
- **SignalPowerUpdateRate [integer]:** Update ratio for the SignalPower output port. The SignalPower output port provides a new value every time the Output port has provided N values, where N is the value of this property.
- **SignalPowerBufferSize** [integer]: Length of the buffer that keeps the filtered signal that is used to compute the power of the signal.

### **FS20 Command Interpreter**

#### **Component Type: Processor (Subcategory: Home Control)**

The FS20CommandInterpreter receives strings containing commands of the home automation system FS20 for ELV Electronics. Depending on the received commands, events will be fired.

Deployment (Ctri-D)     GUI Designer (Ctri-G)       FS20CommandI	Component: FS20Comr	mandInterpreter.1
	Internal Properties	
nterpreter.1	Component Descriptio	Plugin to interprete FS20 commandos
command	Component Type	asterics.FS20CommandInterpreter
	Component Name	FS20CommandInterpreter.1
	Component Class	processor
	<ul> <li>Properties</li> </ul>	
	Housecode	11111111 💽
	Sendaddress	1111 🕃
· · · · · · · · · · · · · · · · · · ·		
) •	Troperties (Ctrl-P)	🤊 Input Ports 📝 Event Trigger

Command Interpreter Plugin

• **command [string]:** The command string containing the FS20 Command. The data must have the following format: housecode\_sendaddress\_command, e.g. 1111111\_3343\_17

### **Event Trigger Description**

Each received command triggers an event, being mapped to this command. The following table will describe this events:

Event	Command
Off	0
Level1	1
Level2	2
Level3	3
Level4	4
Level5	5
Level6	6
Level7	7
Level8	8
Level9	9
Level10	10
Level11	11
Level12	12
Level13	13
Level14	14
Level15	15
Level16	16
OnOldLevel	17
Toggle	18
Dim Up	19
Dim Down	20
Dim Up and Down	21
Program internal timer	22
Off for timer then old brightness level	24

Command Mapping

On for timer then off	25
On old brightness level for timer then off	26
On for timer then old brightness level	30
On for old level then previous state	31

- **housecode [integer]** The housecode, the plugin should react on. The housecode has 8 digits, each from 1 to 4.
- **sendaddress [integer]** The sendaddress, the plugin should react on. The sendaddress has 4 digits, each from 1 to 4.

### HRVAnalysis

## **Component Type: Processor (Subcategory: DSP and feature extraction)**

This component calculates various Heart Rate Variability (HRV) parameters from an incoming signal of raw ECG-data. For a detailed description of the HRV parameters and a guide how to use optical and electrical heart rate sensors see the work of Andreas Schreiber (in /documentation/DIYGuides/HRVAnalysis\_Schreiber.pdf and /documentation/OpticalPulseSensor\_Schreiber.pdf)

*	Component: HRVAnaly	sis.1	
	Search		
HRVAnalysis.1	<ul> <li>Internal Properties</li> </ul>		
	Component Name	HRVAnalysis.1	
	Component Type	asterics.HRVAnalysis	
	Component Descriptic	HRV - Analysis	
	Component Class	processor	-
thasso of	<ul> <li>Properties</li> </ul>		
	samplerate	100	
	OutlierRange	1.5	

HRVAnalysis plugin

• **HRVInput [double]:** Input port for the raw signal

### **Output Port description**

- **runtime [double]:** the current time, since the first sample, in seconds
- **pulserate [double]:** the current calculated pulserate
- **SDNN [double]:** the standard deviation of all RR-intervals
- **rMSSD [double]:** the square-root of the average sum of the quadratic differences between neighboring RR-intervals
- **SDSD [double]:** the current standard deviation of successive differences between neighbouring RR-intervals
- **pNN50 [double]:** the numer of successive RR-intervales that differ by more than 50ms (expressed as percentage of all RR-intervals)
- **pNN20 [double]:** the numer of successive RR-intervales that differ by more than 20ms (expressed as percentage of all RR-intervals)
- DD [double]: the deviation of 2 succesive RR-intervals

### **Event Listener Description**

• **start:** An incoming event starts the HRV analysis

- **samplerate [double]:** specifies the sample rate of the incoming signal.
- **outlierRange [double]:** Defines factor of the mean R-amplitudes which is used as a threshold to detect spikes / signal artefacts.

## IIRFilter

## **Component Type: Processor (Subcategory: DSP and Feature Detection)**

This plugin provides adjustable Infinite Impulse Response Filters, based on the Java DSP Library: <u>http://www.source-code.biz/dsp/java</u>

GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		, ų
	-	Component: IIRFilter.1		
		Search		1
		<ul> <li>Internal Properties</li> </ul>		-
		Component Name	IIRFilter.1	
		Component Type	asterics.IIRFilter	
IIRFilter.1		Component Descriptic	This plugin provides variou	us
		Component Class	processor	•
out		<ul> <li>Properties</li> </ul>		
magnitude 🛑		passType	bandpass	•
		characteristicType	butterworth	•
		order	10	8
		samplingFrequency	100	
		fc1	10	
		fc2	20	
		ripple	-1	
	*			

IIRFilter plugin

• in [double]: The signal to be filtered

- out [double]: The filtered signal
- **magnitude [double]:** the magnitude of the filtered signal in the selected passband (only calculated if the passType = bandpass !)

- **passType [integer] (combobox selection):** can be lowpass, highpass, bandpass or bandstop
- **characteristicType [integer] (combobox selection):** can be butterworth, bessel or chebyshev
- order [integer]: the filter order (values from 1 to 12 recommended)
- **samplingFrequency [integer]:** the sampling rate of the input signal
- **fc1 [double]:** corner frequency (lower corner frequency for bandpass filter)
- **fc2 [double]:** higher corner frequency (ignored in case of highpass or lowpass types)
- **ripple [double]:** the passband ripple supression, must be a negative value (only for chebyshev types, ignored for other types)

## Int To String

# **Component Type: Processor (Subcategory: Data Converters)**

This component converts the incoming integer values to the string values at the output port.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P) 🗸 🗸
IntToString.1	* III	componentType: IntToString.1         Search         Internal Properties         Component Descriptic         Component Name         IntToString.1         ComponentType         processor         Properties         hexadecimalOutput
< >	•	Troperties (Ctrl-P)

IntToString plugin

• input [integer]: Input port for the integer values to be converted.

### **Output Port Description**

• **output [string]:** Output port for the converted string values.

• **hexadecimalOutput [boolean]:** If this property is set, the integer is converted into a hexadecimal string.

### Integrate

#### **Component Type: Processor (Subcategory: Basic Math)**

The integrate component performs successive addition of incoming signal values. This is useful for transforming relative movement information coming from a sensor into absolute position values.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	
	componentType: Integrate.1       Image: Search       ▲ Internal Properties
	Component Descriptic Integration with upper and low
Integrate.1	Component Name Integrate.1
	ComponentType processor 💌
out	<ul> <li>Properties</li> </ul>
	resetValue 0
	upperLimit 50000
	IowerLimit -50000
	wrapAround
	-
×	<b>∄</b> Pr <b>∄</b> In <b>∄</b> O <b>≠</b> Ev

Integrate plugin

• **in [double]:** The input port for signal values.
### **Output Port Description**

• **out [double]:** Output of the integrated values.

#### **Event Listener Description**

• **reset:** An incoming event at this port sets the current accumulator value to the rest value (specified in the rest property field).

#### **Properties**

- **resetValue [double]:** The initial value of the accumulator, which is set when starting the model or when an event comes in at the elp\_rest event listener port.
- **upperLimit [double]:** The maximum accumulator value (the integration sum will stay at this value and not get higher even if positive values come in at the input port).
- **IowerLimit [double]:** The minimum accumulator value (the integration sum will stay at this value and not get lower even if negative values come in at the input port) Upper and lower limit are useful e.g. to set bounds for mouse movement etc.
- wrapAround [boolean]: If this property is set to true, the accumulator value is set to the lower\_limit if it gets greater than the upper\_limit (overflow), and to the upper\_limit if it would get lower than the lower\_limit (underflow).

## **Math Evaluator**

#### **Component Type: Processor (Subcategory: Basic Math)**

The math evaluator is a component with four inputs. These inputs can be combined in a mathematical expression which is entered via a property of the component. The expression parser used is <u>JEPlite2</u> which supports arithmetic as well as numerous mathematical expressions.

The list of supported mathematical functions is sin(), cos(), tan(), asin(), ,acos(), atan(), sqrt(), log(), ln(), angle(), abs(), mod(), sum(), rand(), umin(), add().

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	➡ Properties (Ctrl-P)		<b>→</b> Ū
	Component: MathEvalu	iator.1	
MathEvaluator.1	₹ Search	€ AL Search	
	Internal Properties		
	Component Descriptio	Mathematical expression evaluator	
	Component Type	asterics.MathEvaluator	
	Component Name	MathEvaluator.1	
	Component Class	processor	*
() inC	<ul> <li>Properties</li> </ul>		
	expression	a+b	
	<b>•</b>		
		- L the	

MathEvaluator plugin

#### Requirements

The component depends on the JEPlite library which is included in the component's JAR file.

#### **Input Port Description**

• **inA to inD [double]:** The inputs which can be accessed in the mathematical expression via a to d. **These 4 input ports support synchronization** 

### **Output Port Description**

• **out [double]:** the result of the expression.

#### **Properties**

• **expression [string]:** Mathematical expression to be evaluated.

## MinMax

#### **Component Type: Processor (Subcategory: Basic Math)**

This component stores maximum and minimum of an incoming signal and provides these values at the output ports. Together with the signaltranslation component, the minmax component can be used to auto-scale input values to a desired signal range.



### **Input Port Description**

• **in [double]:** Input signal for min/max calculation.

#### **Output Port Description**

- **outMax [double]:** The current maximum value found in the signal.
- **outMin [double]:** The current minimum value found in the signal.

#### **Event Listener Description**

• **reset:** An incoming event sets the current minimum and maximum to the values defined in the associated property fields defaultMin and defaultMax.

#### **Properties**

- **defaultMin [double]:** This is the default minimum value which is set when the model is started or if an event comes in at the rest event listener port.
- **defaultMax [double]:** This is the default maximum value which is set when the model is started or if an event comes in at the reset event listener port.

## **MotionAnalysis**

# **Component Type: Processor (Subcategory: DSP and Feature Detection)**

The MotionAnalysis Plugin procides a visual feedback for one or more channels. With this Plugin it is able to save a movement and compare it to later movements. To test this plugin MotionAnalysis\_Example.acs can be used.

#### Save

To save an exercise the save button has to be pressed. A new file with the name of the filename properity is created. To start the start button has to be pressed. The save funktion can be paused when the stop button is pressed. To restart the start button has to be pressed. To stop the save funktion the stopsave button has to be pressed.

#### Load

To load a file the load button has to be pressed. Then the explorer is opened an a file can be choosen. If a file is selected it is atomatically loaded in the graph. To start an exercise the start button has to be pressed. The exercise stops automatically when it is finished and a result is sent to the result output port.



roperties (Ctrl-P)			
Component: MotionAn	alysis.1		
E AL			
<ul> <li>Internal Properties</li> </ul>			
Component Name	MotionAnalysis.1		
Component Type	asterics.MotionAnalysis		
Component Descriptic	Display to visualize and compare		
Component Class	processor 💌		
<ul> <li>Properties</li> </ul>			
displayBuffer	3		
drawingMode	crop to min and max 🔹		
displayMode	redraw on incoming sample 💌		
drawingInterval	100 🗧		
min	-100		
max	100		
gridColor	black 💌		
loadchannelColor	blue 💌		
drawchannelColor	red 💌		
backgroundColor	white 💌		
fontSize	14		
caption	Exercise		
filename	Test		
filepath	.\\data\\actuator.motionanalysis\		
diviation	20		
limitation	80		

MotionAnalysis plugin



Screenshot: MotionAnalysis plugin

#### **Input Port Description**

- **channel1 [double]:** The input signal for channel one.
- **channel2** [double]: The input signal for channel two.
- **channel2** [double]: The input signal for channel three.
- **channel2 [double]:** The input signal for channel four.

#### **Output Port Description**

- **result [string]:** Sends the match between the loaded movement and the aktual movement.
- percent [double]: Sends the actual position in percent of time.

#### **Properties**

- **displayBuffer [integer]:** This property value specifies how often the oscilloscope window is drawn. For example if the display buffer size is 0, the oscilloscope traces are redrawn at every incoming value. If the display buffer size is set to 10, 10 values are stored in a buffer and drawn at once as the tenth value is received. This significantly reduces the computational resources spent for drawing the oscilloscope, which is useful especially at high update rates.
- **drawingMode [integer]:** Declares whether the y axis is adapting to mininum and maximum values automatically or to stay in preset bounds. This only affects the drawchannel not the loadchannel or the save option.
- **displayMode [integer]:** Affects the time when oscilloscope is redrawn. Can be set to the values "redraw on incoming samples" or "redraw periodically".
- **drawInterval [integer]:** Redraw interval in milliseconds (if periodic drawing is used).
- min [integer]: Preset minimum value for y axis of oscilloscope.
- max [integer]: Preset maximum value for y axis of oscilloscope.
- gridColor [integer]: The colour of the value-grid.
- **loadchannelColor [integer]:** The colour of the signal trace for the loaded value.
- **drawchannelColor [integer]:** The colour of the signal trace for the actual value.
- **backgroundColor [integer]:** The colour of the background of the oscilloscope window.
- fontSize [integer]: The size of the oscilloscope's caption.
- caption [string]: The caption to be displayed on the oscilloscope.
- **filename [string]:** The name of the saved file. There is added a time and date to not overwrite a file.
- filepath [string]: The path in wich the files are saved.
- **diviation [integer]:** The allowed diviation of the loaded value and the actual value in one point of time.
- **limitation [integer]:** The limitation of how much percent of match must be reached to raise an event.

#### EventListener

- Start: Starts a movement.
- **Stop:** Stops a movement.
- Save: Starts to save a movement.
- **Stopsave:** Stops to save a movement.
- Load: Loads a movement.

### EventTrigger

• Inrange: Raises a event when the result is higher than the limitation.

## **MultiSource**

# **Component Type: Processor (Subcategory: Signal Pathways)**

This plugin passes signals from up to four input ports to one output port.



MultiSource plugin

### **Input Port Description**

• input1...input4 [double]: Input ports 1 to 4

#### **Output Port Description**

• **output [double]:** The output port where all input signals are routed.

## **MultiSourceString**

# Component Type: Processor (Subcategory: Signal Pathways)

This plugin passes string inputs from up to four input ports to one output port.



MultiSourceString plugin

### **Input Port Description**

• input1...input4 [string]: Input ports 1 to 4.

### **Output Port Description**

• output [string]: Output port.

### **Neural Network Loader**

# **Component Type: Processor (Subcategory: DSP and Feature Detection)**

This plugin uses the <u>Encog framework</u> (version: 3.0.1). The plugin can load a neural network configuration stored in the Encog EG file. The neural network's output is calculated for the input data.



NeuralNetworkLoader plugin

### **Input Port Description**

• input1...input32 [double]: The Neural network inputs.

### **Output Port Description**

• **output1...output32 [double]:** The Neural network outputs.

#### **Properties**

• filePath [string]: The EG file path.

How to prepare example EG file for the Neural Network Loader plugin.

## **Create Encog EG file**

This document describe how to create sample <u>Encog framework</u> EG file that store neural network. The example shown how to create the basic neural network that performs XOR operation.

- Download the encog-workbench-3.0.1-release.zip package from the <u>Encog page</u>, unzip it and run the Encog Workbench using command: "java -jar encog-workbench-3.0.1-executable.jar". Create new folder for the project.
- 2. To create Neural network, select "File > New file...". Select: the "Machine Learning Method (\*.eg)" and enter name of the EG file. Then, select the "Feedforward Neural Network" and click OK. On the next window, enter 2 into the "Input Neuron Count" text box. Click Add and enter 4 neurons for the hidden layer. enter 1 into the "Output Neuron Count" text box. Change the Activation Function Hidden and Activation Function Output to the Activation Sigmoid. Then, click OK.



The structure of the neural network

3. Next step is to create the Training File. Select "File > New file...". Select: "Training File (\*.egb)" and enter name of the EGB file. Click OK. On the next window, enter 4 into the the "Training Set Elements" text box, enter 2 into the "Input Field Count" text box, enter 1 into the "Output Field Count" text box. Click OK. Click on the file and fill the table for the XOR operation.

Input 1	Input 2	Ideal 1	Significance		
0	0	0	1		
0	1	1	1		
1	0	1	1		
1	1	0	1		
The training set					

4. When the training set is ready, the neural network can be trained. Click on the neural network file and choose the "Train" option. In the Training Set select the training file, in the Neural Network select the neural network file and click OK. Select the Propagation - Resilient (RPROP) training method and click OK. Enter 0.01 into the Maximum Error Percent(0-100) box, choose the RPROP type: "RPROP+ (classic)" and click OK. Click the Start button to Train the network. When the Max Error is reached click Close.

The neural network file can be loaded by the <u>Neural Network Loader</u> plugin. To load the neural network, enter the EG file path into the filePath plugin property. For this example, the input1 input port and the input2 input port will send the input data for the neural network and the output1 output port will give the XOR operation result.

### **One Event Many Actions**

# Component Type: Processor (Subcategory: Event and String Processing)

This component allows the user to control up to 10 actions using just one or two input events. In the first step, the event actions are scanned so that the user can choose the action, in the next step the selected action event is triggered. The detailed functionality depends upon the selected mode.



OneEventManyActions plugin

#### **Event Listener Description**

- **input:** The control event.
- **inputOff:** Control event used in mode 2 for selecting the action by scanning.

#### **Event Trigger Description**

- **action1...action10:** The event triggers for the actions selected by user.
- **actionShown11...actionShown110:** These events are used in the scanning process to highlight the currently selected action via a GUI element (e.g. the CellBoard).

#### **Properties**

- actionsNumber [integer]: Number of action used.
- mode [integer]: The component's working mode:
  - **mode 1:** The input event starts the scanning, the inputOff event selects the action.
  - **mode 2:** The first input event starts the scanning, the second input event selects the action.
  - **mode 3:** The first input event starts the scanning and highlights the first action, the next input events highlight the next actions. If there is no input event for the selected delay time, the currently highlighted action is selected.
- **delay [integer]:** Delay used for the scanning process (in milliseconds).
# openHAB

# **Component Type: Processors (Subcategory: Home Control)**

The openHAB component interfaces to an openHAB instance, which is used to configure and use a home control environment. Usually, openHAB is stand-alone. Therefore it has its own GUI, many many possible interfacing solutions (called bindings) and a configuration tool (based on Eclipse). More information is available here: <u>openHAB wiki</u>.

This component uses the provided REST API of openHAB to read and write the state of different nodes (called items) within the openHAB system.

0
0
0
$\bigcirc$
6 °C
open

openHAB demo with different options (light, heating, temperature, ...)

# Requirements

A functional openHAB installation, which is accessible via the web interface (the plugin connects via HTTP REST API). You can run either HTTP or HTTPS, in order to fulfill any security requirements. In addition, it is also possible to provide HTTP basic authentication with username/password. Please note, that any saved password in the AsTeRICS model is stored in the model file in PLAINTEXT!

To run openHAB without password authentication, start openHAB with this command:

bash ./start\_debug.sh

To start with password authentication, use following command: bash ./start\_debug.sh -Djava.security.auth.login.config=./etc/login.conf The user configuration is handled via this file:

*openHAB\_runtime/configurations/users.cfg* Please note, that the first line is necessary, so do not remove it!

Further information on configuration and usage of openHAB is available on the openHAB GitHub page (<u>openHAB wiki</u>).

## **Input Port Description**

- **item1in [string]:** New state for item1 (the corresponding name is set in the property item1in). For example: set the property item1in to Light\_GF\_Bed\_Ceiling and send "ON" to the input port to switch on the light which is connected to this item.
- **item2in [string]:** New state for item2 (the corresponding name is set in the property item2in). Example: see input port item1in
- **item3in [string]:** New state for item3 (the corresponding name is set in the property item3in). Example: see input port item1in
- **item4in [string]:** New state for item4 (the corresponding name is set in the property item4in). Example: see input port item1in
- **item5in [string]:** New state for item5 (the corresponding name is set in the property item5in). Example: see input port item1in
- **item6in [string]:** New state for item6 (the corresponding name is set in the property item6in). Example: see input port item1in
- actionString [string]: NOT IMPLEMENTED YET: more flexible input, where a random item (referenced by the name) can be set.

### **Output Port Description**

• **item1 [string] - item6 [string]:** The current state of the items1 to items6, corresponding to the item names of properties item1out to item6out

# **Event Trigger Description**

• **item1change - item6change:** This event is triggered if the corresponding item (set by the properties item1event to item6event) changes its state. The initial value is NOT raising an event.

# **Properties**

- **updaterate [integer]:** Time in milliseconds, which will ellapse between each status update. Default: 1s (1000ms)
- **hostname [string]:** Hostname to connect to. It is possible to use a hostname, an IP adress or a FQDN
- **port [string]:** Port of the openHAB installation. Defaults: 8080 for HTTP, 8443 for HTTPS. Please take care of any blocking firewall.
- **protocol [string]:** Protocol to connect to openHAB. Either http or https may be used (recommended: https).
- **lazyCertificates [boolean]:** If this property is set, any SSL related certificate check will be removed for the given hostname. This affects the hole ARE.
- **username [string]:** This property is used, if the HTTP basic authentication of openHAB is used. Provide the username here.
- **password [string]:** This property is used, if the HTTP basic authentication of openHAB is used. Provide the password here.
- item1in [string]: Item name, which is used for the input port 1 (set an openHAB item)
- item2in [string]: Item name, which is used for the input port 2 (set an openHAB item)
- item3in [string]: Item name, which is used for the input port 3 (set an openHAB item)
- **item4in [string]:** Item name, which is used for the input port 4 (set an openHAB item)
- **item5in [string]:** Item name, which is used for the input port 5 (set an openHAB item)
- **item6in [string]:** Item name, which is used for the input port 6 (set an openHAB item)
- **item1out [string]:** Item name, which is used for the output port 1 (fetch an openHAB item with the given updaterate)
- **item2out [string]:** Item name, which is used for the output port 2 (fetch an openHAB item with the given updaterate)
- **item3out [string]:** Item name, which is used for the output port 3 (fetch an openHAB item with the given updaterate)
- **item4out [string]:** Item name, which is used for the output port 4 (fetch an openHAB item with the given updaterate)

- **item5out [string]:** Item name, which is used for the output port 5 (fetch an openHAB item with the given updaterate)
- **item6out [string]:** Item name, which is used for the output port 6 (fetch an openHAB item with the given updaterate)
- item1event [string]: Item name, which is used to raise an event if the state is changed (item1change)
- item2event [string]: Item name, which is used to raise an event if the state is changed (item2change)
- item3event [string]: Item name, which is used to raise an event if the state is changed (item3change)
- **item4event [string]:** Item name, which is used to raise an event if the state is changed (item4change)
- **item5event [string]:** Item name, which is used to raise an event if the state is changed (item5change)
- **item6event [string]:** Item name, which is used to raise an event if the state is changed (item6change)

# **Additional hints**

- The model will stop with an error message, if one of the item names in the properties is not found.
- There is no feedback for checking a successful state change. E.g.: if your write to a read-only item (temperature sensor), nothing will happen
- Use the *lazyCertificates* property with care, it will disable a major part of the SSL handshaking for the whole Java session. It should be limited to the given hostname only, but without warranty.
- The username/password combination from the properties is saved in PLAINTEXT in the model file, so handle it with care.

# **Osc Gesture Follower**

# **Component Type: Processor (Subcategory: DSP and Feature Detection)**

The OscGestureFollower component copules the ARE with the external gesture recognition software GestureFollower. The program is copyrighted by IRCAM. GestureFollower is stored in the ARE subfolder tools/GestureFollower. Gesture data can be stored and loaded from files in this subdirectory, these files have the extension ".mubu". The communication between GestureFollower and the ARE is based on the OpenSoundControl (OSC) protocol. This plugin utilizes the NetUtil java library (http://www.sciss.de/netutil/) for the OSC implementation, it is (C)opyrighted 2004-2011 by Hanns Holger Rutz and released under the GNU Lesser General Public License.



OscGestureFollower howto

# Requirements

- The plugin requires the gfOSC\_v1.exe in subfolder tools/GestureFollower, which implements the actual gesture follower algorithms.
- Check your firewall configuration and network settings to ensure that OSC messages are not blocked.

# **Functional Principle**

Input data is received through CH1 to CH4 e.g. from sensors like the acceleration measurement unit. Not all inputs must be connected, but the synchronized checkboxes have to be checked correct. The events must be connected like illustrated in the picture. First the gestures must be teached in. To teach in the first gesture, send an event into the 'learn1' eventListener. After finishing the first gesture, send an event to 'learn2' to teach in the second gesture, and so on. After all gestures are teached in, send the 'stoplearn' event. To clear all gestures send the 'clear' event. To start the gesture recognition process, send the 'follow' event. To stop the gesture following process, send the 'stop' event. The 'load' and 'save' events can be used to load or store the learned gesture data into the given filename.

# **Input Port Description**

• CH1 - CH4 [double]: The input port which receive data values. These 4 input ports support synchronization

#### **Output Port Description**

• **likeliest [double]:** While the gesturefollower is in 'follwing mode' the most likeliest gesture is indicated on the likeliest output port. Before it can sample the input data and recognize a gesture, some data must be teached in.

## **Properties**

- **InPort [integer]:** This value specifies the Port where OscMessages form the gesture follower are received.
- **OutPort [integer]:** This value specifies the Port where OscMessages are send to.
- filename [string]: Filename for the gesture data (load or save, .mubu file stored in the ARE subfoler tools/GestureFollower/). Supports value suggestions from ARE (dynamic property).

### **Event Listener Ports**

- stop: this event stops the gesture following
- stoplearn: this event stops the gesture learning process
- learn1 learn5: these events select gestures 1 5 for learning
- learn1 learn5: these events select gestures 1 5 for learning
- clear: this event clears learned gestures
- follow: this event starts the gesture recognition phase
- load: this event loads gesture data from file
- **save:** this event saves gesture data to file

# **Referred Plugins**

- OscOutClient
- OpenVibe OscServer

# OskaExternalScanning1D

#### **Component Type: Processor (Subcategory: OSKA)**

This component interacts with the On Screen Keyboard Application (OSKA) and forwards key selections and command (action-) strings to other ARE plugins. The scanning function of OSKA can be controlled via this plugin, allowing 1-dimensional input values to be used for direct scanning position selection. This is done in a two-step fashion where first the columns are highlighted and then keys of a column are highlighted. Selection of columns and cells is performed upon incoming events. The event input ports allow switching from row- to column scanning, moving the cell highlighter and selecting a cell.



OskaExternalScanning1D plugin

# Requirements

This component requires Sensory Software's OSKA keyboard. OSKA is automatically started by the plugin if it is located in the expected path ("../OSKA/Start Keyboard.exe").

#### **Input Port Description**

• **posKey [double]:** This port takes values for the selection of keys (respectively columns). Depending on the value of the property "inputRange", the incoming values are expected to be in a range from 0 to 1 (where 0 represents the first selection and 1 the last) or they identify the absolute key/column position as integer values. The input moves either the highlighted column or the highlighted key in the selected column to the corresponding position. A selection of the currently highlighted item can be triggered by incoming events.

#### **Output Port Description**

- **action [string]:** This output sends the action strings (which have been added to a key in the OSKA editor) when this key is pressed.
- **keycodes [string]:** If a key is selected which contains the "@KDB: ..." action string, the key codes are extracted from the action string and sent via this port, e.g. to the remoteKeyboard plugin's corresponding input port for keyboard input emulation.

## **Event Listener Description**

- **switch:** Incoming events will trigger a state change in the scanning state machine. On the first press highlighting will switch from column highlighting to cell highlighting. On the second press, the currently highlighted OSKA cell will be selected.
- **highlightNext:** Incoming events highlight the next column (or cell in a column).
- **highlightPrev:** Incoming events highlight the previous column (or cell in a column).

# **Event Trigger Description**

• eventOut1-eventOut10: These events can be triggered by selecting a cell which contains an @OSKA,event .. action string, (for example @OSKA,event 3).

# **Properties**

- **port [integer]:** This property defines the TCP port that the component listens on for connections of the OSKA.
- **title [string]:** This property defines the caption to be displayed in OSKA?s title bar.
- **oskaPath [string]:** The absolute path to the OSKA player as well as the program name is needed in order for the component to start OSKA by itself.
- **keyboardPath [string]:** If this property does not hold an empty string, the component will ask OSKA to load the keyboard referred to by this property. When the ACS is synchronized with the ARE (connected and model deployed) available keyboards can be selected from a drop-down listbox. (dynamic property)
- **highlightStyle [integer]:** This property selects the style of highlighting used in OSKA, the value range is from 0 to 2.
- **settingsFile [string]:** if this property is not empty OSKA will be started with this settings file as a command line parameter. When the ACS is synchronized with the ARE (connected and model deployed) available settings files can be selected from a drop-down listbox **Supports value suggestions from ARE (dynamic property)**
- **windowDecorated [boolean]:** if true, Oska will display a decorated window frame, otherwise only a lightweight frame.
- eventScanningEnabled [boolean]: if true, incoming events can move the highlighter.
- valueScanningEnabled [boolean]: if true, incoming values can move the highlighter.
- **inputRange [boolean] (combobox):** the range of the incoming input port values: either float values between 0 and 1, or integer values or integer values from 0 to the maximum element (number of cells in a respective row or column of the grid).
- **resizeEnabled [boolean]:** if true, the Oska will be resized to the dimensions specified in the GUI designer (slower).

# **Oska External Scanning2D**

#### Component Type: Processor (Subcategory: OSKA)

This component interacts with the On Screen Keyboard Application (OSKA) and forwards key selections and command (action-) strings to other ARE plugins. The scanning function of OSKA can be controlled via this plugin, allowing 1-dimensional input values to be used for direct scanning position selection. Scanning is operated using two float inputs which take inputs in a range defined by the according property (between 0.0 and 1.0. or integer values). The two inputs control the position of the highlighted cell in a grid with values being mapped to a corresponding position in the grid.



OskaExternalScanning2D plugin

# Requirements

This component requires Sensory Software's OSKA keyboard. OSKA is automatically started by the plugin if it is located in the expected path ("../OSKA/Start Keyboard.exe").

#### **Input Port Description**

- **posColumn [double]:** This port takes values for the selection of the X-position (column selection). Depending on the value of the property "inputRange", the incoming values are expected to be in a range from 0 to 1 (where 0 represents the first column and 1 the last) or they identify the absolute column position as integer values. **This input port supports synchronization**
- **posRow [double]:** This port takes values for the selection of the Yposition (row selection). Depending on the value of the property "inputRange", the incoming values are expected to be in a range from 0 to 1 (where 0 represents the first row and 1 the last) or they identifiy the absolute row position as integer values. **This input port supports synchronization**

#### **Output Port Description**

- **action [string]:** This output sends the action strings (which have been added to a key in the OSKA editor) when this key is pressed.
- **keycodes [string]:** If a key is selected which contains the "@KDB: ..." action string, the key codes are extracted from the action string and sent via this port, e.g. to the remoteKeyboard plugin's corresponding input port for keyboard input emulation.

# **Event Listener Description**

- **press:** Incoming events will trigger a press action on the selected key in OSKA.
- **highlightNextX:** Incoming events will highlight the next cell in the selected row.
- **highlightPrevX:** Incoming events will highlight the previous cell in the selected row.
- **highlightNextY:** Incoming events will highlight the next cell in the selected column.
- **highlightPrevY:** Incoming events will highlight the previous cell in the selected column.

# **Event Trigger Description**

• eventOut1-eventOut10: These events can be triggered by selecting a cell which contains an @OSKA,event .. action string, (for example @OSKA,event 3).

# **Properties**

- **port [integer]:** This property defines the TCP port that the component listens on for connections of the OSKA.
- **title [string]:** This property defines the caption to be displayed in OSKA?s title bar.
- **oskaPath [string]:** The absolute path to the OSKA player as well as the program name is needed in order for the component to start OSKA by itself.
- **keyboardPath [string]:** If this property does not hold an empty string, the component will ask OSKA to load the keyboard referred to by this property. When the ACS is synchronized with the ARE (connected and model deployed) available keyboards can be selected from a drop-down listbox. (dynamic property)
- **highlightStyle [integer]:** This property selects the style of highlighting used in OSKA, the value range is from 0 to 2.
- highlightResetPosition [integer]: This property selects the behaviour of the highlighter after a key has been pressed (do nothing / got to top left / got to center / got to bottom right).
- **settingsFile [string]:** if this property is not empty OSKA will be started with this settings file as a command line parameter. When the ACS is synchronized with the ARE (connected and model deployed) available settings files can be selected from a drop-down listbox. (dynamic property)
- **windowDecorated [boolean]:** if true Oska will display a decorated window frame, otherwise only a lightweight frame.
- eventScanningEnabled [boolean]: if true, incoming events can move the highlighter.
- valueScanningEnabled [boolean]: if true, incoming values can move the highlighter.
- **inputRange [boolean] (combobox):** the range of the incoming input port values: either float values between 0 and 1, or integer values or integer values from 0 to the maximum element (number of cells in a respective row or column of the grid).
- **resizeEnabled [boolean]:** if true, the Oska will be resized to the dimensions specified in the GUI designer (slower).

# **Oska Internal Scanning**

#### **Component Type: Processor (Subcategory: OSKA)**

This component interacts with the On Screen Keyboard Application (OSKA) and forwards key selections and command (action-) strings to other ARE plugins. OSKA is set to use its internal scanning methods and the component exposes the two button input events that Oska can work with.



OskaInternalScanning plugin

# Requirements

This component requires Sensory Software's OSKA keyboard. OSKA is automatically started by the plugin if it is located in the expected path ("../OSKA/Start Keyboard.exe").

#### **Output Port Description**

- **action [string]:** This output sends the action string which is attached to a specific key on the keyboard to connected components.
- **keycodes [string]:** This output sends the key codes which are attached to a key via the @KDB command.

## **Event Listener Description**

- **increaseScanspeed:** Incoming events will increase the internal scanning speed of OSKA.
- **decreaseScanspeed:** Incoming events will decrease the internal scanning speed of OSKA.
- **pressSwitch1:** Incoming events start the automatic scanning or switch to the next selection (to speed up the scanning).
- **pressSwitch2:** Incoming events switch from column- to row scanning (or in the next step select the cell). If the scanning is stopped, it will be started.

# **Event Trigger Description**

• eventOut1-eventOut10: These events can be triggered by selecting a cell which contains an @OSKA,event .. action string, (for example @OSKA,event 3).
- **port [integer]:** This property defines the TCP port that the component listens on for connections of the OSKA.
- **title [string]:** This property defines the caption to be displayed in OSKA?s title bar.
- **oskaPath [string]:** The absolute path to the OSKA player as well as the program name is needed in order for the component to start OSKA by itself.
- **keyboardPath [string]:** If this property does not hold an empty string, the component will ask OSKA to load the keyboard referred to by this property. When the ACS is synchronized with the ARE (connected and model deployed) available keyboards can be selected from a drop-down listbox. (dynamic property)
- **scanSpeed [integer]:** This property relates to the internal row column scanning method of OSKA and sets the speed of scanning, the value range is between 1 and 10.
- **highlightStyle [integer]:** This property selects the style of highlighting used in OSKA, the value range is from 0 to 2.
- **settingsFile [string]:** if this property is not empty OSKA will be started with this settings file as a command line parameter. When the ACS is synchronized with the ARE (connected and model deployed) available settings files can be selected from a drop-down listbox. (dynamic property)
- **windowDecorated [boolean]:** if true, Oska will display a decorated window frame, otherwise only a lightweight frame.
- **resizeEnabled [boolean]:** if true, the Oska will be resized to the dimensions specified in the GUI designer (slower).

## **Path Multiplexer**

# Component Type: Processor (Subcategory: Signal Pathways)

The Path Multiplexer component forwards signal from the selected input port to the output port.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Event Listener	<b>~</b> ↓
	*	Listeners	Description
PathMultiplexer.		passPort1	passes data from the port 1
		passPort2	passes data from the port 2
		passPort3	passes data from the port 3
		passPort4	passes data from the port 4
		passNextPort	pass from the next port
		passPreviousPort	passes from the previous po
input4			
	-		
< []]	•		. 🍠 Ev

PathMultiplexer plugin

#### **Input Port Description**

• **input1 to input4 [double]:** The input ports for signals to be multiplexed 1..4.

### **Output Port Description**

• **output [double]:** The output port.

#### **Event Listener Description**

- **passPort1 to passPort4:** The event send to this port sets the forwarding signal from the appropriate input port:1...4 to the output port.
- **passNextPort:** The event send to this port sets the forwarding signal from the next input port. If the current used is the port defined by the number property, the signal will be forward from the input port 1.
- **passPreviousPort:** The event send to this port sets the forwarding signal from the previous input port. If the current used is the port 1, the signal will be forward from the input port defined by the number property.

• **number [integer]:** The maximum port number in use (can be 1 to 4).

## Pathselector

# Component Type: Processor (Subcategory: Signal Pathways)

The pathselector component allows routing of an incoming numerical signal between up to 4 output ports. The desired output port can be directly selected by a dedicated event listener port, or the signal can be switched to the next or previous output port. This component can be used to utilize one particular signal for different purposes, e.g. controlling different actuators within a single model by sequentially switching between them.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	₹	Event Listener	<b>~</b> ↓					
		Listeners	Description					
		select1	selects out-port 1					
PathSelector.1		select2	selects out-port 2					
		select3	selects out-port 3					
		select4	selects out-port 4					
		selectNext	selects next out-port					
out2		selectPrevious	selects previous out-port					
out3								
	-							
<		ℬ Pr ℬ In ℬ O	💈 Ev					
Dath Calestan alvair								

PathSelector plugin

### **Input Port Description**

• in [double]: The incoming signal to be routed.

#### **Output Port Description**

• **out1 to out4:** Four output ports where the incoming signal can be routed to.

#### **Event Listener Description**

- **select1 to select4:** An incoming event at these ports directly activates the associated output path (e.g. as an event comes in at select 3, the input signal will be routed to out3.
- **selectNext:** The next output port is selected for signal output. The maximum number of active ports is set via the active ports property. If the current number of active ports is already the maximum one, the select next event will wrap around the active port number and one port will be selected.
- **selectPrevious:** The previous output port is selected for signal output. If the current port is out1, the select previous event will switch to the port with the maximum number given by the active ports property.

• **activePorts [integer]:** The maximum port number in use (can be 2 to 4).

### **PeakDetector**

#### **Component Type: Processor (Subcategory: Basic Math)**

The Peakdetector component can be used to detect peaks (top values of the signal) and valleys (bottom values) in a signal. Additionally it can calculate the time between two peaks or two valleys or between a peak and a valley. Optionally, top and bottom values can be compared with an average of the most recent top / bottom values - this allows to detect only peaks which are for example greater than 150% of the last 5 averaged peak values.



PeakDetector plugin

### **Input Port Description**

• in [double]: The incoming signal.

#### **Output Port Description**

- **top:** The peak value of the signal.
- **bottom:** The bottom value of the signal.
- **time:** The time in ms or beats per minute (BPM) (depending on the property timeMode) between two peaks (Mode=detect tops), two valleys (Mode=detect bottoms) or a valley and a peak (Mode=detect both) depending on the selected mode in the properties.

#### **Event Trigger Description**

- **topDetected:** The event gets fired if a new top value was detected in the input signal.
- **bottomDetected:** The event gets fired if a new bottom value was detected in the input signal.

- **mode:** Determines which time frame will be used for the time output signal (see description of the output port time).
- **comparePeaks:**the number of top / bottom values which are taken into account for averaging (0 = disable)
- validTopPercentage: the percentage of the recent averaged top values which constitutes a valid top value (0 = all top values are valid)
- validBottomPercentage:the percentage of the recent averaged bottom values which constitutes a valid bottom value (0 = all bottom values are valid)
- **timeMode:** Determines the unit for the measured time frame between top/bottom values. Options are beats per minute (BPM) or milliseconds.

## **Protocol SSVEP Train**

# **Component Type: Processor (Subcategory: DSP and Feauture extraction)**

This component is in charge of managing and configuring the SSVEP training recording protocol. It sends out the corresponding SSVEP stimulation frequencies.



ProtocolSSVEPTrain plugin

#### Requirements

This plugins must be connected to the FlickeringLightStimulator plugin (which triggers the external SSVEP stimulation panels) or to the SSVEPStiumlator plugin.

#### **Output Port Description**

- **StimFrequency [integer]:** Current stimulation frequency in Hz under test.
- Freq2 [integer]: Background stimulation frequency in Hz for panel 2.
- Freq3 [integer]: Background Stimulation frequency in Hz for panel 3.
- Freq4 [integer]: Background Stimulation frequency in Hz for panel 4.

#### **Event Listener Description**

- **StartProt:** Start protocol request.
- **StopProt:** Stop protocol request.
- **Continue:** Next Stimulation trial request
- **Repeat:** Repeat last stimulation trial request.
- **Stop:** Abort current stimulation trial request.

#### **Event Trigger Description**

- **StartTrial:** Event sent when a stimulation trial starts.
- **StopTrial:** Event sent when a stimulation trial ends.
- **StartStim:** Event sent when a stimulation period starts.
- **StopStim:** Event sent when a stimulation period ends.
- **ReadyStim:** EEvent sent 1 second befor the stimulation periods starts.
- **UpdatePanelConfig:** Event sent requesting a stimulation panels reconfiguration.

- NumRepetitions [integer]: Number of stimulation periods per trial.
- **RepStimDuration [integer]:** Stimulation period duration in seconds.
- **RepNonStimDuration [integer]:** Non-Stimulation period duration in seconds.
- freqStim1 [integer]: Stimulation frequency number 1 in Hz.
- freqStim2 [integer]: Stimulation frequency number 2 in Hz.
- freqStim3 [integer]: Stimulation frequency number 3 in Hz.
- freqStim4 [integer]: Stimulation frequency number 4 in Hz.
- freqStim5 [integer]: Stimulation frequency number 5 in Hz.
- freqStim6 [integer]: Stimulation frequency number 6 in Hz.
- freqStim7 [integer]: Stimulation frequency number 7 in Hz.
- freqStim8 [integer]: Stimulation frequency number 8 in Hz.
- freqStim9 [integer]: Stimulation frequency number 9 in Hz.
- freqStim10 [integer]: Stimulation frequency number 10 in Hz.
- RandomizeFreq [boolean]: Randomize stimulation frequencies order.

## Quantizer

#### **Component Type: Processor (Subcategory: Basic Math)**

This plugin performs a quantization of the input signal. The value of the output signal is the input value rounded to the nearest multiple of the quantizationStep property value.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)	• 1
		componentType: Quantizer.1	
		Component Descriptic Performs the quantization of the input signal	
		Component Name Quantizer.1	
Quantizer.1		ComponentType processor	*
input		<ul> <li>Properties</li> </ul>	
output		quantizationStep 1	
	•		
·		Toperties (Ctrl-P)	
ARE Status: Disconnected		100% Ctrl +/-	

Quantizer plugin

### **Input Port Description**

• **input [double]:** Input port for the values to be quantized.

### **Output Port Description**

• **output [double]:** Output port for the quantized values.

• quantizationStep [double]: The quantization step.

## **Regular Expression**

# **Component Type: Processor (Subcategory: Event and String Processing)**

This component processes strings with regular expressions. It can work in two modes: match strings with the pattern or replace string parts which match the pattern with another string.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		<b>→</b> ‡	
		Component: RegularExpression.1			
		Search			
		<ul> <li>Internal Properties</li> </ul>			
RegularExpressio n.l input output		Component Descriptic Process the string with regular expressions.		expressions.	
		Component Name	RegularExpression.1		
		ComponentType	processor		
		<ul> <li>Properties</li> </ul>			
		pattern			
		replace			
		replaceString			
	-				
* <u> </u>	•	Properties (Ctrl-P)	Tinput Ports 📑 Output Ports	Event Trigger	

RegularExpression plugin

### **Input Port Description**

• **input [string]:** Input port for strings.

#### **Output Port Description**

• **output [string]:** The port for strings which match the pattern or which were changed.

#### **Event Trigger Description**

- match : This event is sent if the string matches the pattern.
- **notMatch :** This event is sent if the string doesn't match the pattern.
- **replace** : This event is sent if parts of the string where replaced by the replaceString.
- **notReplace** : This event is sent if no replacement occurred.

- pattern [string]: Regular expression pattern.
- **replace [boolean]:** If the property is set to true, the component will search parts of the string which match the pattern and replace these parts with the replaceString, otherwise the component will match the whole string with the pattern.
- **replaceString [string]:** The string which replaces expressions which matching the pattern.

### **Relative Move Sampler**

#### **Component Type: Processor (Subcategory: Basic Math)**

This plugin was developed for models where a constant rate of value updates is needed. The plugin sums incoming relative position packages for three coordinates. The sums are sent to the output ports according to sampling rate. When the sums are sent, the accumulation values are reset to zero.



Relative Move Sampler

#### **Input Port Description**

- **inputX [integer]:** Input X position.
- inputY [integer]: Input Y position.
  inputZ [integer]: Input Z position.

#### **Output Port Description**

- **outputX [integer]:** Output X position.
- **outputY** [integer]: Output Y position.
- outputZ [integer]: Output Z position.

• **samplingRate [integer]:** Defines the rate of the sampling (in Hz).

## Sample and Hold

#### **Component Type: Processor (Subcategory: Basic Math)**

This component provides a sample-and-hold function for up to 4 input signals. This can be useful for storing a particular sensor values (e.g. for sensor calibration purpose).



SampleAndHold plugin
## **Input Port Description**

• **in1 - in4 [double]:** four input ports for double values to be captured on demand.

## **Output Port Description**

• out1 - out4 [double]: last captured values.

## **Event Listener Description**

• **sampleNow:** When an event comes in, the signal values of the input ports are captured and sent to the output ports.

## Sampler

#### **Component Type: Processor (Subcategory: Basic Math)**

This plugin generates a constant rate of sampling for the input port signal.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		•
	* H	componentType: Samp	ler.1	
		Component Descriptio	Sets the constant rate of the sampling	
Sampler.1		Component Name	Sampler.1	
		ComponentType	processor	*
input		<ul> <li>Properties</li> </ul>		
ouput		samplingRate	10	
		responseTime	0	
		sendNullSamples		
	Ŧ			
•		Properties (Ctrl-P)	Input Ports	
		Sampler plugir	ו	

## **Input Port Description**

• **input [double]:** Input port for the signal.

## **Output Port Description**

• **output [double]:** Output port for signal with constant rate of the sampling.

## **Properties**

- **samplingRate [double]:** The rate of sampling, which defines the data generation frequency at the output port.
- **responseTime [integer]:** Response time in milliseconds. If the time from arrival of the last input data exceeds the response time, the plugin stops sending the output data. If the responseTime is set to 0, it is not used.
- **sendNullSamples [boolean]:** If this property is set to true and there is no input data or the response time is exceeded the plugin sends samples equal to zero.

## **Signal Translation**

# **Component Type: Processor (Subcategory: Signal Shaping)**

The signal translation component is used to translate an input value which resides in a certain value range to a given output range. Interpolation of the position in the output range is done linearly. The component provides two inputs which allow other components to set the minimum and maximum value of the input range.



SignalTranslation plugin

#### **Input Port Description**

- **in [double]:** This port receives the input values which will be translated to the new range.
- **setMax [double]:** This port sets the value of the maximum property in the component.
- **setMin [double]:** This port sets the value of the minimum property in the component.

## **Output Port Description**

• **out [double]:** This port sends the translated values corresponding to the output range.

## **Properties**

- **inMin [double]:** Sets the minimum value of the input range, input values below this value will be clipped to the minimum.
- **inMax [double]:** Sets the maximum value of the input range, input values above this value will be clipped to the maximum.
- **outMin [double]:** Sets the minimum value of the output range.
- **outMax [double]:** Sets the maximum value of the output range.

## **Speech Processor**

# **Component Type: Processor (Subcategory: Audio and Voice)**

The SpeechProcessor component provides methods to use a speech recognition engine and a speech synthesizer, provided via the Microsoft Speech Platform Server version 11 (see http://www.microsoft.com/enus/download/details.aspx?id=27225). The Microsoft Speech Platform provides enables recognition of spoken words and generation of synthesized speech (text-to-speech, TTS). Engines and language packs for 26 languages can be downloaded for free (see above link). The language (culture) of speech recognition and synthesis can be adjusted as a plugin property (currently English, German, Spanish and Polish are supported by the plugin, if the language packs are installed.) The supported voice commands can be set by the plugin properties. Recognized commands trigger events and are put to an output port as stings. Moreover, the component can receive strings which are spoken via the selected speech synthesizer. The component features special commands for activation, deactivation and speaking all supported commands



SpeechProcessor plugin

## Requirements

• This component requires Microsoft's Speech Platform version 11 Runtime and the desired language packs to be installed on the platform running the plugin.

#### **Input Port Description**

• **speak [string]:** This port receives strings containing sentences or words that should be spoken via a speech synthesizer which fits the selected language (text-to-speech).

#### **Output Port Description**

• **command [string]:** This output sends strings which have been recognized by the speech recognition engine (matching one of the commands defined by property values).

## **Event Listener Description:**

• **help:** and incoming event starts the help mode where all supported voice commands will be spoken.

## **Event Triggerer Description:**

- **activated:** triggered when the recognition is activated via the defined activation command.
- **deActivated:** triggered when the recognition is deactivated via the defined command.
- recognizedCommand1 recognizedCommand10: If an incoming word matches a command defined in the plugin properties, the corresponding event will fire.

## **Properties**

- **language [integer] (combobox selection):** A combobox which defines which speech pack is to be used (currently: English, German, Spanish, Polish) these language packs need to be installed !
- **recognitionConfidence [double]:** This value defines the needed confidence for a valid recognition. The value can range from 0 to 1. A higher value results in a more precise pronunciation needed to detect a command, a lower level could lead to more false-positive recognition results.
- **speechLoopDelay [integer]:** This value defines the minimal time between two speech commands. It is used to avoid recognition-loops
- activationCommand [string]: a speech command to start the recognition of the other commands.
- **deactivationCommand [string]:** a speech command to stop the recognition of the other commands.
- helpCommand [string]: a speech command to speak out all supported voice commands.
- mode [integer] (combobox selection): This property selects one out of three operating modes for the speech recognition:
  - "always active": the command recognition is always running
  - "voice-triggered activation and deactivation": the command recognition is started by the recognition of the activation command and stopped by the deactivation command (these commands are defined in the according properties).
  - "automatic deactivation after command recognition": after a recognized command, the speech recognition will be bypassed until another activation command has been recognized.
  - "speech recognition disabled (TTS only)": In this mode, the speech recognition engine is not used whichs saved CPU power in text-to-speech-only applications.
- **command1 to command10**: The command strings which are checked by the speech recognition engines (these build the recognition grammar).

## **SSVEP Detect**

This component is in charge of evaluating the SSVEP response (up to 4 different frequencies) among the frequencies defined by the user. It also calculates the config file based on previously recorded training files that will be used to evaluate the detection, and is also in charge of updating its parameters according to the config file

# **Component Type: Processor (Subcategory: DSP and Feauture extraction)**

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		•
	Component: SSVEPDetect.1			
		Search		
	=	<ul> <li>Internal Properties</li> </ul>		
		Component Name SSVEPDetect.1		
SSVEPDetect.1		Component Type	asterics.SSVEPDetect	
		Component Descriptio	This plugin is in charge of ev	va
FreqP1		Component Class	processor	*
FreqP2		<ul> <li>Properties</li> </ul>		
FreqP3		SF1GO1	1	
FreqP4		SF1GOz	1	_
NumberOfPar		SF1GO2	1	_
		SF2GO1	1	
		SF2GOz	1	
		SF2GO2	1	
		SF3GO1	1	
		SF3GOz	1	_
	*	SF3GO2	1	
	+	🕈 Prop 🏾 🕈 Input	🐔 Even 🏓 Even	

## Requirements

SSVEPTrainFunction.exe and Matlab 2008B runtime engine

#### **Input Port Description**

- **O1 [double]:** Input port for the EEG channel O1. **This input port** supports synchronization
- Oz [double]: Input port for the EEG channel Oz. This input port supports synchronization
- **O2 [double]:** Input port for the EEG channel O2. **This input port supports synchronization**
- UserName [string]: Name Identifying current the user.
- NumberOfPanels [string]: Number of stimulation panels.
- FreqP1 [string]: Stimulation frequency of panel 1.
- FreqP2 [string]: Stimulation frequency of panel 2.
- FreqP3 [string]: Stimulation frequency of panel 3.
- FreqP4 [string]: Stimulation frequency of panel 4.

#### **Output Port Description**

- FreqP1 [integer]: Stimulation frequency of panel 1.
- FreqP2 [integer]: Stimulation frequency of panel 2.
- FreqP3 [integer]: Stimulation frequency of panel 3.
- FreqP4 [integer]: Stimulation frequency of panel 4.

### **Event Listener Description**

- StartStim: Event Informing that the stimulation period has started.
- StopStim: Event Informing that the stimulation period has finished.
- **CalculateConfigFile:** Event requesting the calculation of the configuration file
- **UpdateFromConfigFile:** Event requested an update of the properties according to the configuration file.
- **UpdatePanelsConfig:** Event reporting the stimulation panels plugin to update the stimulation frequencies.

## **Event Trigger Description**

- **UpdatePanelsConfig:** Event reporting the stimulation panels plugin to update the stimulation frequencies.
- **NonStimFreqD:** Event reporting that none stimulation frequency could be detected.
- **StimFreq1D:** Event reporting that stimulation frequency number 1 was detected.
- **StimFreq2D:** Event reporting that stimulation frequency number 2 was detected.
- **StimFreq3D:** Event reporting that stimulation frequency number 3 was detected.
- **StimFreq4D:** Event reporting that stimulation frequency number 4 was detected.

## **Properties**

- **SF1GO1 [double]:** Spatial filter coefficient for stimulation frequency 1 and electrode O1.
- **SF1GOz [double]:** Spatial filter coefficient for stimulation frequency 1 and electrode Oz.
- **SF1GO2 [double]:** Spatial filter coefficient for stimulation frequency 1 and electrode O2.
- **SF2GO1 [double]:** Spatial filter coefficient for stimulation frequency 2 and electrode O1.
- **SF2GOz [double]:** Spatial filter coefficient for stimulation frequency 2 and electrode Oz.
- **SF2GO2 [double]:** Spatial filter coefficient for stimulation frequency 2 and electrode O2.
- **SF3GO1 [double]:** Spatial filter coefficient for stimulation frequency 3 and electrode O1.
- **SF3GOz [double]:** Spatial filter coefficient for stimulation frequency 3 and electrode Oz.
- **SF3GO2 [double]:** Spatial filter coefficient for stimulation frequency 3 and electrode O2.
- **SF4GO1 [double]:** Spatial filter coefficient for stimulation frequency 4 and electrode O1.
- **SF4GOz [double]:** Spatial filter coefficient for stimulation frequency 4 and electrode Oz.
- **SF4GO2 [double]:** Spatial filter coefficient for stimulation frequency 4 and electrode O2.
- **StimFreq1 [integer]:** Stimulation frequency number 2 in Hz. If its value is 0 it will not be evaluated in the SSVEP detection..
- StimFreq2 [integer]: Stimulation frequency number 2 in Hz. If its value is 0 it will not be evaluated in the SSVEP detection.
- StimFreq3 [integer]: Stimulation frequency number 3 in Hz. If its value is 0 it will not be evaluated in the SSVEP detection.
- **StimFreq4 [integer]:** Stimulation frequency number 4 in Hz. If its value is 0 it will not be evaluated in the SSVEP detection.
- **BestHarm1 [integer]:** Best harmonic to detect stimulation frequency 1.
- BestHarm2 [integer]: Best harmonic to detect stimulation frequency

2.

- **BestHarm3 [integer]:** Best harmonic to detect stimulation frequency 3.
- **BestHarm4 [integer]:** Best harmonic to detect stimulation frequency 4.

## **String Append**

# **Component Type: Processor (Subcategory: Event and String Processing)**

Appends an incoming string or ASCII character to a stringbuffer, the updated stringbuffer is sent to an output port. Incoming events are available to send the final string to a dedicated output port and to clear the stringbuffer (or set it to a default value respectively).

GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)	.▼ Д		
		Component: StringAppend.1			
	=	Search			
		Internal Properties			
StringAppend.1		Component Class	processor		
inStr actResult		Component Name	StringAppend.1		
		Component Type	asterics.StringAppend		
		Component Descriptic	This plugin allows assembly of		
		<ul> <li>Properties</li> </ul>			
		autoSendAtEnter			
		defaultValue			
	•				
	۲.	🕈 Pr 📝 Ev 📽 O	. 🎯 In		

StringAppend plugin

## **Input Port Description**

- inStr [string]: String input port.
  inChar [integer]: ASCII code input port.

#### **Output Port Description**

- actResult [string]: String output port for current stringbuffer content.
- **finalResult [string]:** String output port for final stringbuffer content (triggered by event or enter).

#### **Event Listener Ports**

- **sendNow:** sends the current content of the stringbuffer to the "finalResult" output port.
- **sendNowAndClear:** sends the current content of the stringbuffer to the "finalResult" output port and sets the stringbuffer to the default value.
- deleteCharacter: deletes the last character of the stringbuffer.
- **clear:** sets the stringbuffer to the default value.

## **Properties**

- **autoSendAtEnter [boolean]:** If true, the stringbuffer is sent and cleared when the ASCII value for Enter/Return is received by the inChar input port.
- **defaultValue [string]:** An optional string text which is used a initial value for the stringbuffer.

## StringDelay

# **Component Type: Processor (Subcategory: Event and String Processing)**

This plugin passes received string values after a defined delay.

GUI Designer (Ctrl-G) = ×	Properties (Ctrl-P)		<b>▼</b> ‡		
	Component: StringDela	iy.1			
StringDelay.1	Search Search				
	<ul> <li>Internal Properties</li> </ul>				
	Component Class	processor	-		
	Component Name	StringDelay.1			
	Component Type	asterics.StringDelay			
	Component Descriptic	delays the incoming string by a given time			
	<ul> <li>Properties</li> </ul>				
	Delay	1000	\$		
	StringDelay plug	in			

## **Input Port Description**

• in (string): Input port for the incoming string.

## **Output Port Description**

• **out (string):** Output port for the delayed string.

## **Properties**

• **delay [integer]:** Delay between input and output of the string (in milliseconds).

## **String Dispatcher**

# **Component Type: Processor (Subcategory: Event and String Processing)**

This component has twenty slots for text strings. These strings can be sent to the output port via incoming events or by directly addressing a slot number using the input port.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Properties (Ctrl-P)		🗕 t
	componentType: String	gDispatcher.1	
=	Search		
	<ul> <li>Internal Properties</li> </ul>		*
	Component Descriptio	Send test from chosen slot	
StringDispatcher	Component Name	StringDispatcher.1	
1	ComponentType	processor	-
slotDispatch	<ul> <li>Properties</li> </ul>		-
output	delay	300	<b>a</b>
	slot1		
	slot2		
	slot3		
	slot4		
	slot5		
	slot6		
	slot7		
-	slot8		*
+	Prop The Input	TOutp 💉 Even	

StringDispatcher plugin
• **slotDispatch [integer]:** Sends the string from the slot defined by number.

• **output [string]:** String output port.

#### **Event Listener Description**

- **dispatchSlot1...dispatchSlot20:** These events cause text from the slot: 1..20 to be sent.
- **dispatchNextSlot:** This event causes text from the next (not empty) slot to be sent.
- **dispatchCurrentSlot:** This event causes text from the current slot to be sent.
- **dispatchPreviousSlot:** This event causes text from the previous (not empty) slot to be sent.
- **resetToFirstSlot:** This event resets the internal slot counter (for next/previous) to the first slot.
- **dispatchSlotSeries:** This event causes text in sequence from all not empty slots to be sent, with a delay defined by the delay property.

#### **Properties**

- **delay [integer]:** The interval (in milliseconds) which will be used for sending strings sequentially from all slots.
- slot1...slot20 [string]: 20 slots which contains the text to be sent

# **String Expander**

# **Component Type: Processor (Subcategory: Event and String Processing)**

Adds the preString and postString strings to the incoming string and sends the new string to the output port.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		<b>-</b> 4
	-	Component: StringExpa	ander.1	
		€ 2↓ Search		
		<ul> <li>Internal Properties</li> </ul>		
		Component Descriptio	Adds the preString and postString strings to the incoming	g st
		Component Type	asterics.StringExpander	
StringExpander.1		Component Name	StringExpander.1	
() input		Component Class	processor	•
Deployment (Ctrl-D) GUI Designer (Ctrl-G)		<ul> <li>Properties</li> </ul>		
		preString		
		postString		
		trim		
	-			
· _	Þ.	Troperties (Ctrl-P)	Tinput Ports	

StringExpander plugin

- **input [string]:** String input port.
- **preString [string]:** String which will be placed before the input string (as leading string).
- **postString [string]:** String which will be placed after the input string (as trailing string).

• **output [string]:** String output port.

#### **Properties**

- preString [string]: default leading String.
- postString [string]: default trailing String.
- trim [boolean]: if selected, all leading and trailing white-space characters will be removed from the input string.

# **String Filter**

# Component Type: Processor (Subcategory: Event and String Processing)

Applies a given filter text to the incoming string. It can be selected if only strings that contain the filter text are passed to the output port, and/or if the filter text shall be removed from the incoming string. Please note that the filter text is applied case sensitive.

Deployment (Ctrl-D) =	×	Properties (Ctrl-P)	<b>▼</b> 1
		Component: StringFilte	er.1
		Search	
		<ul> <li>Internal Properties</li> </ul>	
StringFilter.1		Component Class	processor
		Component Name	StringFilter.1
Ceployment (Ctrl-D)		Component Type	asterics.StringFilter
		Component Descriptic	removes a substring from a string and/or pass
		<ul> <li>Properties</li> </ul>	
StringFilter.1 in out		filterText	dummy
		passOnlyIfContains	X
		cropFilterText	X
< >>		Properties (Ctrl-P)	TINPUT Ports Dutput Ports
		StringFilter plug	nin

• in [string]: String input port.

• **out [string]:** String output port for processed/filtered string.

#### **Properties**

- filterText [string]: The filter text.
- **passOnlylfContains [boolean]:** If selected, only strings containing the filter text will be passed.
- **cropFilterText [boolean]:** If selected, the filter will be removed before the input string is passed to the output port.

# **String Path Multiplexer**

# Component Type: Processor (Subcategory: Signal Pathways)

The String Path Multiplexer component forwards strings from the selected input port to the output port.

Deployment (Ctrl-D) GUI Designer (Ctrl-G) 🗢	Properties (Ctrl-P)	~ џ
	Component: StringPat	hMultiplexer.1
	Search	
=	Internal Properties	
	Component Description	Forwards strings from the selected input to 1
	Component Name	StringPathMultiplexer.1
StringPathMultip	ComponentType	processor 💌
lexer.1	<ul> <li>Properties</li> </ul>	
inputl	number	2
input2		
input3		
input4		
·		
•	🕈 Prop 🏾 🕈 Inpu	Toutp 🔰 Even

StringPathMultiplexer plugin

• **input1 to input4 [string]:** The input ports for strings to be multiplexed.

• **output [string]:** The string output, which sends data of the selected input port.

#### **Event Listener Description**

- **passPort1 to passPort4:** The string output, which sends data of the selected input port.
- **passNextPort:** selets the next input port. If the currently used port is the maximum port (defined by the number property), input port 1 will be selected.
- **passPreviousPort:** selects the previous input port. If the currently used port is port 1, the maximum port (defined by the number property) will be selected.

### **Properties**

• **number [integer]:** The maximum port number (can be 1 to 4).

# **String Path Selector**

# Component Type: Processor (Subcategory: Signal Pathways)

The StringPathSelector component allows routing of incoming strings between up to 4 output ports. The desired output port can be directly selected by a dedicated event listener port, or the strings can be switched to the next or previous output port.

Deployment (Ctrl-D) GUI Designer (Ctrl-G) 🗧	Properties (Ctrl-P)	▼ џ
A	Component: StringPath	Selector.1
	Search	
=	<ul> <li>Internal Properties</li> </ul>	
	Component Descriptio	StringPathselector routes signal to one of fo
	Component Name	StringPathSelector.1
StringPathSelect or.1	ComponentType	processor 💌
	<ul> <li>Properties</li> </ul>	
outl	activePorts	2
out2 out2 out4		
* >	🕈 Prop 🏾 🕈 Inpu 🔅	🕈 Outp 💉 Even

StringPathSelector plugin

• **in [string]:** The incoming string port to be routed.

• **out1 to out4:** Four output ports where the incoming strings can be routed to.

#### **Event Listener Description**

- **select1 to select4:** An incoming event at these ports directly activates the associated output path (e.g. as an event comes in at select 3, the input string port will be routed to out3.
- **selectNext:** The next output port is selected for string output. The maximum number of active ports is set via the activePorts property. If the current number is already the maximum one, the select next event will wrap around the active port number and port 1 will be selected.
- **selectPrevious:** The previous output port is selected for string output. If the current port is out1, the maximum port number (given by the activePorts property) will be selected.

#### **Properties**

• **activePorts [integer]:** The maximum port number in use (can be 1 to 4).

# **String Splitter**

# **Component Type: Processor (Subcategory: Event and String Processing)**

Splits a string with seperators in up to 16 outputports.

	~	Component: StringSplit	tter.1						
		Search							
	StringSplitter.1	<ul> <li>Internal Properties</li> </ul>							
C	Disput	Component Class	processor						
	out1	Component Name	StringSplitter.1						
	out2	Component Type	asterics.StringSplitter						
	out3	Component Descriptio	Split the string in up to 16 strings						
	out4	<ul> <li>Properties</li> </ul>							
	out5	Seperator	;						
	out6								
	out7								
	out8								
	out9								
	out10								
	out11								
	out12								
	out13								
	out14								
	out15								
	out16								
-									

StringSplitter plugin

• **input [string]:** String input port. The string which has to be splitted.

• **output 1 to 16[string]:** String output ports. Sending the seperated Data.

## **Properties**

• Seperator [string]: Defines the seperator.

# **String To Double**

# **Component Type: Processor (Subcategory: Data Converters)**

This component converts the incoming string values into the integer values at the output port.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		<b>▼</b> ₽
Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Р + Ш	Properties (Ctrl-P) componentType: String Properties Internal Properties Component Descriptic Component Name ComponentType	gToDouble.1 Converts string to values StringToDouble.1 processor	
·	•	Properties (Ctrl-P)     ☐	Input Ports 3 Output Ports	

StringToDouble plugin

• **input [string]:** Input port for the string values to be converted.

• **output [double]:** Output port for the converted double values.

# **String To Int**

# **Component Type: Processor (Subcategory: Data Converters)**

This component converts the incoming string values into the integer values at the output port.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P) 🗸 🗸
	* H	componentType: StringToInt.1       Image: StringToInt.1       Image: StringToInt.1       Image: StringToInt.1       Image: StringToInt.1
		Component Descriptic Converts string to integer.
		Component Name StringToInt.1
StringToInt.1		ComponentType processor 💌
		Properties
output		hexadecimalInput
	Ŧ	
· •		The Properties (Ctrl-P) The Ports Output Ports
S	Strir	ngToInt plugin

• **input [string]:** Input port for the string values to be converted.

• **output [integer]:** Output port for the converted integer values.

#### **Properties**

• **hexadecimalInput [boolean]:** If this property is set, the input string data are regarded as the hexadecimal data string.

## **Text Sender**

# **Component Type: Processor (Subcategory: Event and String Processing)**

This component sends the string data through the output port when it receives the trigger event.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)	-	џ
	*	componentType: TextS	ender.1	
	-	Search		
		<ul> <li>Internal Properties</li> </ul>		
TextSender.1		Component Descriptio	Send text when gets event	
		Component Name	TextSender.1	
output 🛑		ComponentType	processor	-
		<ul> <li>Properties</li> </ul>		
		text	This is a sample text.	
	Ŧ			
· ·		Troperties (Ctrl-P)	Output Ports 💉 Event Listener	

TextSender plugin

• **setText [string]:** Updates the text with the incoming string.

• **output [string]:** String output port.
### **Event Listener Description**

• **sendText:** This event causes text to be sent.

### **Properties**

• text [string]: The default text to be sent.

# Threshold

#### Component Type: Processor (Subcategory: Basic Math)

This component takes an input and compares it to a given threshold and generates an according output value dependent on the operation mode. The component allows three operation modes: binary, deadzone and cut off which are described in the property section. The component allows operating with a hysteresis by setting the thresholds for transition from below to above and vice versa differently. Transitions over the thresholds are always evaluated arithmetically thus a below threshold to above transition always happens when the new value is mathematically greater than the threshold while the last input was less than threshold. Respectively the transition from above to below happens when the last value was greater than the threshold value and the new value is less than it. This statement also holds for negative values in the threshold. The component provides the possibility to raise events on threshold transitions either on below to above, above to below or on both.



Threshold plugin

### **Input Port Description**

• in [double]: the input to be evaluated.

### **Output Port Description**

• **out[double]:** the value that the threshold passes on for the given input.

### **Event Port Description**

- **eventPosEdge:** This event port will trigger an event when the value crosses the thresholdHigh property from below.
- **eventNegEdge:** This event port will trigger an event when the value crosses the thresholdLow property from above.

### **Properties**

- **thresholdHigh [double]:** The threshold that is checked for transitions from below the threshold to above.
- **thresholdLow [double]:** The threshold that is checked for transitions from above the threshold to below. Setting it to the same value as threshold\_high will remove the hysteresis from the threshold system.
- **outputHigh [double]:** The value to be sent to the output if the input is above the threshold in certain operation modes.
- **outputLow [double]:** The value to be sent to the output if the input is below the threshold in certain operation modes (see below).
- **operationMode [integer]:** There are three operation modes for the threshold component:
  - binary: the output will only generate two different values, the value of output\_high if the input is in the range above the threshold and the value of output\_low if the input is below the threshold.
  - cutoff: the input value will be passed through to the output as long as the value is below the threshold. If the input passes the threshold the output will take on the value set in output\_high.v
  - deadzone: the input value will be passed through to the output as long as the value is above the threshold. If the input falls below the threshold the output will take on the value set in output\_low.
- eventCondition [integer]: This property declares on which types of transitions an event will be raised:
  - *Below->above:* only transitions from below to above threshold raise events.
  - *Above->below:* only transitions from above to below threshold raise events.
  - *Both:* both types of transitions raise events.

### **Universal Remote Control**

# Component Type: Processor (Subcategory:Microcontroller Interfaces)

The Universal Remote Control (RC) enables the possibility to control all electronic devices in a household which are controlled remotely through infrared (IR) commands. THe necessary IR codes can be recorded with the RC itself and stored on a database on the computer. By choosing the manufacturer, name and function the IR code is deposed in this database and can be selected in order to send the code to the universal RC and therefore control electronic devices. The universal RC can also be used as a handheld gadget due to its built in battery and user interface. THe IR code database can be stored on an SD card.

UniversalRemote Control.1			Earch ✓ Internal Properties				
Devi	ceType	Ξ	Component Name	UniversalRemoteControl.1 asterics.UniversalRemoteControl			
Devi	ceName		Component Type				
Devi	coEurotio		Component Descriptic record ans send IR Codes through Universal Re				
U Devi	terunctio		Component Class	processor			
			<ul> <li>Properties</li> </ul>				
			IRCodeFilePath	./data/processor.UniversalRemoteControl/IRCodes.csv			

Universal Remote Control sensor plugin

the

### Requirements

For the use of the universal RC with a computer, it has to be be connected to a USB port. The firmware of the universal RC can be found in the AsTeRICS folder under /CIMs/UniversalInfraredRemoteControl/. A database will be automatically generated if there is not found one. For the usage of the universal RC as a handheld gadget, the SD card has to be formatted as a FAT16 volume and contain a database as well as the configuration file.

### **Input Port Description**

- DeviceType [string]: Type of the device to be controlled via IR (e.g.: TV)
- **DeviceName [string]:** Name of the device to be controlled via IR (e.g.: Sony, Samsung, etc.)
- **DeviceFunction [string]:** Name of function of the device (e.g.: On, Off, VolumeUp, etc.)

### **Event Listener Description**

- SendIRCode: Send an IR Code to the Universal Remote Control
- RecordIRCode: Record an IR Code through the Universal Remote Control

### **Event Triggerer Description**

- **StartRecord:** will be triggered when Universal Remote Control is recording
- **StopRecord:** will be triggered when Universal Remote Control has finished recording

### **Properties**

• **IRCodeFilePath [string]:** Filepath to the file, where the IR Codes are stored.

### **IR Code Database**

The database which contains the IR codes as well as the information about the type and name of the device and the specific function is a comma separated value file. The first value is the type, the second one is the name and the third one is the function. The following 512 values are the IR code. This database is automatically generated and maintained if new IR codes are recorded with the Universal Remote Control with IR functions.

### **Configuration File**

The settings of the universal RC can be stored in the config.csv file. This is a comma separated value file and contains configurations such as the speed and sensibility of the rotary encoder and external buttons as well as the name of the file that contains the IR codes. The content of the default config.csv file is:

JoystickSpeed,120 JoystickSensibility,3 File,IRCODES.CSV Sort,2

### WebSocket

#### **Component Type: Processor (Subcategory: Web)**

#### This is just a demo plugin and is not yet fully functional.

The websocket component takes an input stream and forwards the data to a websocket (<u>http://localhost:8082/ws/astericsData</u>). A demo webpage that connects to the websocket and visualizes the data can be accessed at <u>http://localhost:8082/</u>.

The websocket plugin can only be used if the the ARE was started with the following command:

#### start\_debug.bat --webservice

### **Input Port Description**

- **InA [double]:** This port reads the input to be forwarded.
- InB [double]: Not yet supported
- InC [double]: Not yet supported
- InD [double]: Not yet supported
- InE [double]: Not yet supported
- InF [double]: Not yet supported

### **Output Port Description**

- OutA [double]: Not yet supported
- OutB [double]: Not yet supported
- OutC [double]: Not yet supported
- OutD [double]: Not yet supported
- OutE [double]: Not yet supported
- OutF [double]: Not yet supported

### **Properties**

- host [string]: Not yet supported
  port [integer]: Not yet supported

# Yaak

#### **Component Type: Processor**

Yaak is a flexible on-screen-keyboard for the Android operating system. More information can be found in the pdf documentation in the Android/Yaak folder within the release package.

		Properties (Ctri-P)		•	
		Component: Yaak.1			
		Search			
		<ul> <li>Internal Properties</li> </ul>			
Yaak.1		Component Class	processor	-	
		Component Name	Yaak.1		
action 🛑		Component Type	asterics.Yaak		
		Component Descriptio	My Plugin description		
		<ul> <li>Properties</li> </ul>			
	=	hostname	localhost		
		port	44000	8	

### Requirements

• The Android based mobile device running Yaak must be connected to the AsTeRICS system over a TCP/IP connection and the used port must not be blocked by a firewall

### **Output Port Description**

• **action [string]:** gives the action string whenever a button of the keyboard gets triggered. The string for each button can be stored in the xml layout for the keyboards.

### **Properties**

- **hostname [string]:** The hostname of the mobile device (IP or hostname is valid)
- **port [integer]:** The port Yaak listens for incoming messages. This can be set in the xml file of a keyboard.

### Actuators

Actuators are components which take input data and control external processes with it. These processes can either control hardware peripherals (such as DACs or digital output ports) or make data visible (like in an oscilloscope). Actuators only have input ports since they are data sinks in the ARE.

# **Analog Out**

# **Component Type: Actuator (Subcategory: Generic Control Output)**

This plugin communicates with the DAC CIM and operates the analog outputs of the module. The plugin provides four input ports which correspond to the four DAC outputs of the CIM.



AnalogOut plugin

### Requirements

This software component requires an DAC CIM (CIM ID: 0x0401) connected to an USB port.

	ADC/DAC				$\wedge$				
:	O SV USB	SV USB	EXT_PWF			/	Asī	eRIC	S
	OUT1	OUT2	OUT3	OUT4	GPADC	GPDAC	GPOUT	GPIN	.0
	IN1	IN2	IN3	IN4	IN5	ING	IN7	IN8	•

ADC/DAC CIM

### **Input port Description**

- **out1 to out4 [integer]:** these input ports correspond to the DAC output of the same number on the CIM. The input is an integer and has a valid range between 0 and 240. The values represent the output voltage in 100mv steps, e.g. a value of 10 represents 1.0V, 143 represents 14.3V.
- **uniqueld:** unique number of the CIM if more than one CIMs of the same type are used. The module flashes a LED for identification when the ID is selected.

# **Application Launcher**

#### **Component Type: Actuator (Subcategory: File System)**

The ApplicationLauncher component can be used to run an external executable application. The application name is given to the plugin via an input port. A default application can be started via an incoming event. Togehter with the Keyboard- or RemoteKeyboard components, the ApplicationLauncher plugin can perform complex automation tasks, for example open Skype, choose a contact and make a call.



ApplicationLauncher plugin

### **Input Port Description**

• **filename [integer]:** The filename of the application to be started (including path).

### **Event Listener Description**

- **launchNow:** An incoming event on this port will start the (default or lastest received) application
- **closeNow:** An incoming event on this port will close the current application

### **Properties**

- **defaultApplication [string]:** Full path and filename of the default application
- **arguments [string]:** the commandline arguments for the application
- **workingDirectory [string]:** the working directory for the application (. is used for home directory of the application)
- closeCmd [string]:Optional close cmd, e.g. if started cmd has forked processes (e.g. OSKA) use: taskkill.exe /IM "OSKA Keyboard.exe" /T
- **autoLaunch [boolean]:** Defines if the default application is automatically launched at startup
- **autoClose [boolean]:** Defines if the current application is closed when the model is stopped
- **onlyByEvent [boolean]:** If this property is set to true, incoming application files names will not be started immediately (only the launchNow event will start the application)

# Preparation of the connection with Android Phone



Android Server Application

This document describes how to prepare connection between the AndroidPhoneControl component and the AsTeRICSPhoneServer application:

- 1. The connection between the AndroidPhoneControl component and the AsTeRICSPhoneServer application is made via TCP/IP connection. One of these components should work as a server, second as the client. The server should have the public IP or it should be in the same network where the client is.
- 2. In the model, select the connection mode in the connetionType property of the AndroidPhoneControl component. If the component should work as a server, put into the port property, the port which server will be use for its service. If the component should work as a client, put into the IP property IP of the server and into the port

property the server port.

- 3. Run the Android Server application on the Android phone. Select the connection mode from the Connection type property. If the application should work as a server put into the port number property the port which server will be use for its service. If the application should work as a client put into the Server IP property IP of the server and into the Port Number property the server port.
- 4. If the AndroidPhoneControl component is set to work as the server and Android Server application is set to work as client. First run the model, then, enable Android application using Enable server property. If Android Server application is set to work as a server and the AndroidPhoneControl component is set to work as a client first enable the Android application then run the model.

# AndroidPhoneControl

#### **Component Type: Actuator (Subcategory: Phone Interface)**

This component controls a mobile phone with Android operating system through the TCP/IP connection. Currently this component is able to perform such action as: call a remote phone, drop a phone call, send and receive SMS message.

Deployment (Ctrl-D) GUI Designer (Ctrl-G) 🗧	Properties (Ctrl-P)	<b>▼</b> ↓
AndroidPhoneC	Component: AndroidPl	Control Android phone with AsTeRICS applicatic
ontrol.1	ComponentType	actuator
RemotePhone SMSContent ReceivedSMS	Properties     connectionType	server 💌
Command ErrorNumber	port	21111
	defaultPhoneID	
< >	ar Prop ar Input	TOutp 🔰 Event 🔰 Event

AndroidPhoneControl plugin

### Requirements

Android phone running AsTeRICSPhoneServer application.

### **Input Port Description**

- **phoneID [string]:** The phone number used for outgoing SMS and make phone calls.
- **SMSContent [string]:** The SMS content which will be used for the send SMS action.
- **command [string]:** String command that can be sent to this component from other plugins to trigger phone actions. Currently supported commands are:
  - @PHONE: SMS:Phone\_ID, "Message\_content"
  - @PHONE: SMS
  - @PHONE: CALL: Phone\_ID
  - @PHONE: CALL
  - @PHONE: ACCEPT
  - @PHONE: DROP
  - @PHONE: SET\_ID: Phone\_ID
  - @PHONE: SET\_SMS: "Message\_content"
#### **Output Port Description**

- **remotePhoneID [string]:** This is a phone number of the caller or SMS sender.
- receivedSMS [string]: This is the content of the incoming SMS.
- errorNumber [integer]: The number of the error.

## **Event Listener Description**

- **sendSMS:** Sends the SMS message.
- makePhoneCall: Makes the phone call.
- acceptPhoneCall: Accepts the incoming phone call.
- dropPhoneCall: Drops the phone call.

## **Event Trigger Description**

- **idleState:** Phone is in the idle state.
- **ringState:** Phone is in the ring state.
- **connectedState:** Phone is connected with the remote phone.
- **newSMS:** There is a new SMS.
- error: An error occurred.

- **connectionType [integer]:** Defines connection type for the plugin: client or server.
- **IP [string]:** IP of the remote server used in the client mode.
- **port [integer]:** TCP/IP port of the service.
- **defaultPhoneID [string]:** Default phone number for outgoing SMS and phone calls.

Preparation of the connection with Android Phone.

# AREWindow

# **Component Type: Actuator (Subcategory: Graphical User Interface)**

This component allows moving the ARE window to desired locations on the screen and setting its state and modification options. Several default locations can be selected via incoming events (top, left, bottom, right or center of the screen). X- and Y- offset values can be defined - thus it becomes possible to set the ARE window e.g to a second screen (which is currently not supported in the ACS GUI designer).

GUI Designer (Ctrl-G) = X	Properties (Ctrl-P)		<b>•</b> Į
^	Component: AREWinde	ow.1	
	Search		
	<ul> <li>Internal Properties</li> </ul>		
AREWindow.1	Component Class	actuator	-
xPos	Component Name	AREWindow.1	
	Component Type	asterics.AREWindow	
yPos	Component Descriptio	provides basic ARE Window manipulation	
	<ul> <li>Properties</li> </ul>		
	xPos	0	-
	yPos	0	\$
	autoSetPosition		
	allowWindowModifica	X	
~			

AREWindow plugin

## **Input Port Description**

- **xPos [integer]:** The x offest value for positioning the ARE window
- **yPos** [integer]: The y offest value for positioning the ARE window

## **Event Listener Description**

- **moveToTop:** moves the ARE window to the top of the screen (y offset will be applied). The x position will not be changed.
- **moveToBottom:** moves the ARE window to the bottom of the screen (y offset will be applied). The x position will not be changed.
- **moveToLeft:** moves the ARE window to the left side of the screen (x offset will be applied). The y position will not be changed.
- **moveToRight:** moves the ARE window to the right side of the screen (x offset will be applied). The y position will not be changed.
- **moveToCenter:** moves the ARE window to the center of the screen (x and y offsets will be applied).
- minimize: minimizes the ARE window to the taskbar
- **restore:** restores the ARE window from the taskbar
- bringToFront: places the ARE window on top of other windows

- **xPos [integer]:** default value for the x offset
- **yPos** [integer]: default value for the y offset
- **autoSetPosition [boolean]:** if selected, the ARE window position will be modified at startup of the model according to the xPos and yPos properties. Furthermore, incoming values at the xPos or yPos ports will automatically position the ARE window.
- **allowWindowModification [boolean]:** If selected, the user can change the ARE window decoration and control panel by double- or right-clicking into the ARE window. These functions will be disabled when the property is not selected.

# **Bar Display**

# **Component Type: Actuator (Subcategory: Graphical User Interface)**

The Bar display generates a coloured bar graph to visualise a current signal value in the ARE environment (of course the LC - display or a computer monitor have to be connected to the platform). The Bar display features auto-scale of value range, display of a threshold value and selectable update rate and foreground / background colours.



BarDisplay plugin

## Requirements

Computer Monitor or LC-Display available for graphics output.

## **Input Port Description**

• **input [double]:** The input port for the signal to be displayed.

- **displayBuffer [integer]:** This number specifies how often an update of the bar graph is performed. For example if the display buffer is set to 10, ten values are accumulated and the average value is displayed after the tenth incoming value.
- **min [double]:** The default minimum of the signal range at model start (this value is automatically updated as lower values come in).
- **max [double]:** The default maximum of the signal range at model start (this value is automatically updated as higher values come in).
- **threshold [double]:** This value will be displayed with a marker in the bar graph (if enabled).
- **displayThreshold [boolean]:** This property enables (true) or disables (false) the threshold marker in the bar graph.
- **integerDisplay [boolean]:** This property selects if double values are rounded to integral values before being displayed in the bar graph.
- **mode [integer]:** Via this property the way how values which exceed the current min/max range of the bar graph component are handled: "clip to min and max" crops incoming values to the min/max range, "autoupdate min and max" scales the bar graph window and updates the min/max values to cover the incoming value.
- gridColor [integer]: The colour of the bar graph grid and descriptions.
- barColor [integer]: The colour of the bar display.
- **backgroundColour [integer]:** The colour of the window background.
- fontSize [integer]: The font size of the display's caption.
- **caption [string]:** The text of the display's caption.
- **displayGUI [boolean]:** If selected, the GUI of this component will be displayed if not, the GUI will be hidden and disabled.

# **Digital Out**

# **Component Type: Actuator (Subcategory: Generic Control Output)**

The DigitalOut plugin operates the output ports of the GPIO CIM. The output ports 1-2 are relais outputs where loads can be connected via a galvanic isolation barrier. The output ports 3-5 are open-collector outputs, where a pull-up resistor can be activated or deactivated using the plugin's properties. The plugin provides event listener ports which serve the activation or deactivation of an output channel, and a command port which accepts string parameters to set, clear and toggle particular output channels.



## Requirements

This component requires the GPIO CIM (CIM Id: 0x0801) to be connected to an USB port.

### **Input port Description**

- action [string]: The plugin reacts to incoming action strings starting with "@GPIO:" and a command. Valid commands are "set", "clear", "toggle" and "press". The command has to be followed by a comma and the port number, for example: "@GPIO:set,1" or "@GPIO:toggle,2". The "press"-command toggles the given output port two times with a delay of 500 milliseconds. The following examples illustrate the available action strings:
  - "@DIGITALOUT:set,1": Pin 1 of the GPIO CIM will be set
  - "@DIGITALOUT:clear,2":Pin 2 of the GPIO CIM will be cleared
  - "@DIGITALOUT:toggle,1": Pin 1 of the GPIO CIM will be changed
  - "@DIGITALOUT:press,4": Pin 4 of the GPIO CIM will be cleared and after 500ms it will be set again

## **Event Listener Description**

- **setOutput1 to setOutput5:** an incoming event on these ports will cause the corresponding output port on the CIM to go to the high level.
- **clearOutput1 to clearOutput5:** an incoming event on these ports will cause the corresponding output port on the CIM to go to the low level.
- **toggleOutput1 to toggleOutput5:**Toggles the state of the output port.
- **pressOutput1 to pressOutput5:**Presses the output port: Clears the state and after 500ms sets the state to high.

- pullupStateOut3 to pullupStateOut5 [boolean]: These properties specify if the internal pullup resistor shall be activated on the respective open collector output channels.
- **uniqueld:** unique number of the CIM if more than one CIMs of the same type are used. The module flashes a LED for identification when the ID is selected. **Supports value suggestions from ARE** (dynamic property)

## **Dot Meter**

# **Component Type: Actuator (Subcategory: Graphical User Interface)**

The Dot Meter generates a graphical representation of a 2-dimensional signal (for example x/y-coordinates) using a colored dot.



DotMeter example

## Requirements

Computer Monitor or LC-Display available for graphics output.

### **Input Port Description**

- x [double]: The x-input port for the signal to be displayed. This input port supports synchronization
- y [double]: The y-input port for the signal to be displayed. This input port supports synchronization

## **Event Listener Description:**

- **dotOn:** fill the dot with color.
- **dotOff:** show only outline of the dot (can be used to create a led indicator).

- **xMin [double]:** The default x-minimum of the signal range at model start (this value is automatically updated as lower values come in).
- **xMax [double]:** The default x-maximum of the signal range at model start (this value is automatically updated as higher values come in).
- **yMin [double]:** The default y-minimum of the signal range at model start (this value is automatically updated as lower values come in).
- **yMax [double]:** The default y-maximum of the signal range at model start (this value is automatically updated as higher values come in).
- **mode [integer]:** Via this property the way how values which exceed the current min/max range of the bar graph component are handled: "clip to min and max" crops incoming values to the min/max range, "autoupdate min and max" scales the bar graph window and updates the min/max values to cover the incoming value.
- dotSize [integer]: The radius of the dot.
- **centerLine [boolean]:** defines if a line to the window center is shown or not.
- **displayDot [boolean]:** defines if the dot is filled or not.
- **displayCaptions [boolean]:** defines if the value / captions are shown or not.
- gridColor [integer]: The colour of the graph grid and descriptions.
- **dotColor [integer]:** The colour of the dot.
- **backgroundColour [integer]:** The colour of the window background.
- fontSize [integer]: The font size of the display's caption.
- **caption [string]:** The text of the display's caption.
- **displayGUI [boolean]:** if selected, the GUI of this component will be displayed if not, the GUI will be hidden and disabled.

## EasyHomeControl

#### **Component Type: Actuator (Subcategory: Home Control)**

With the EasyHomeControl, commands for the EasyHome house automation system can be sent over the PCS device sold by ELV electronics. See <u>the EasyHome homepage</u> for details.

## Requirements

The EasyHome sender must be attached to the system!



Picture of the EasyHome sender



Picture of the EasyHome switch



Picture of the EasyHome dimmer

## **Troubleshooting problems under Win8/Win8.1**

Under Windows 8 and Windows 8.1, there can be problems concering the operation of the EasyHome device: If the red control led stays stays dark eaven if a command is send and the EasyHome sender cannot be used, the reason could be that the power management settings on Win8/8.1 switch the device off per default - this can be changed by applying the following steps:

- Open the Device Manager (usually can be selected by right-clicking at the bottom left corner of your windows desktop)
- Open the device group "Human Interface Devices"
- Locate the device which pops up when you insert or remove the EasyHome module to/from an USB port It should be a "HID compliant supplier defined device". You can distinguish different devices by right-clicking a device and looking into rider "details" and "properites" - there you can find the VID and PID values, e.g. under property "last known parent"
- Note the VID and PID values. For example if you see a line like "USB\VID\_1B1F&PID;\_C00F\EEE0000473" the values are VID:1B1F and PID:C00F
- Press the Windows-Key and "R" to open the command window. Type "regedit" and press enter
- Select the entry "HKEY\_LOCAL\_MACHINE" > "SYSTEM" > "CurrentControlSet" > "Enum" > "USB".
- In this list, select the entry for the EasyHome device (given by VID und PID you noted)
- Open the Subfolder "Device Paramteters" and double click the property "EnhancedPowerManagementEnabled" which appears in the right window section
- Change the property value from 1 to 0 and commit by clicking "OK"
- Restart you PC now hopefully the EasyHome works as intended.

## **Event Listener Description**

See table below for a list of all commands that can be triggered

Command

Mapping

CommandIDOffDevice1-150OnDevice1-151

To dim the light you have to sent an on signal when the light is already turned on.

- **nameDevice [string]:** Set the device name.
- **numberDevice [integer]:** Set the number of the device. The number must be the same if the shwitches shoud akt synchronized

# **EDF** Writer

#### **Component Type: Actuator (Subcategory: File System)**

This plugin writes the incoming data into an .edf file (http://www.edfplus.info/specs/edf.html)

The data gets live recorded, so it can be viewed simulaneously in the EDF-Browser ( http://www.teuniz.net/edfbrowser/ )

⇒ ×	Properties (Ctrl-P)		
	Component: WriteEDF.	2	
	Search		
	Internal Properties		
	Component Class	actuator	
	Component Name	WriteEDF.2	
	Component Type	asterics.WriteEDF	
	Component Descriptio	This plugin records the incoming data into an EDF fil	
	Properties		
	patientID	undefined	
	FileName	record	
	reSampling		
	SamplingRateCH1	100	
	SamplingRateCH2	100	
	SamplingRateCH3	100	
	SamplingRateCH4	100	
	SamplingRateCH5	100	
	SamplingRateCH6	100	
	SamplingRateCH7	100	
	SamplingRateCH8	100	
WriteEDF.2	PhysicalMinimumCH1	-32768	
	PhysicalMinimumCH2	-32768	
	PhysicalMinimumCH3	-32768	
CH2	PhysicalMinimumCH4	-32768	
ССНЗ	Physical Minimum CH5	-32768	
Фсн4	PhysicalMinimumCH6	-32768	
СН5	PhysicalMinimumCH7	-32768	
СН6	PhysicalMinimumCH8	-32768	
СН7	PhysicalMaximumCH1	32767	
СН8	PhysicalMaximumCH2	32767	
	PhysicalMaximumCH3	32767	
	PhysicalMaximumCH4	32767	
	PhysicalMaximumCH5	32767	
	PhysicalMaximumCH6	32767	
	PhysicalMaximumCH7	32767	
	PhysicalMaximumCH8	32767	
	DigitalMinimumCH1	-32768	
	DigitalMinimumCH2	-32768	
	DigitalMinimumCH3	-32768	
	DigitalMinimumCH4	-32768	
	DigitalMinimumCH5	-32768	
	DigitalMinimumCH6	-32768	
	DigitalMinimumCH7	-32768	
	DigitalMinimumCH8	-32768	
	DigitalMaximumCH1	32767	
	DigitalMaximumCH2	32767	
	DigitalMaximumCH3	32767	
	DigitalMaximumCH4	32767	
	DigitalMaximumCH5	32767	
	DigitalMaximumCH6	32767	
	DigitalMaximumCH7	32767	
	DigitalMaximumCH8	32767	

WriteEDF

## **Input port Description**

• **CH1 - CH8[double]:** these ports get the data, which will be recorded.

- **PatientID[string]:** specifies the name of the subject. Maximum number of characters: 80
- FileName[string]: specifies the name of the file (without extension ), in which the data will be recorded
- **reSampling[boolean]:** If this box is checked, the incoming data will be resampled with the chosen frequency of the properties samplingRateCH1 CH8. This could be neccessary, if the incoming data frequency is not known
- **SamplingRateCH1 CH8[integer]:** specifies the frequency of the incoming data. Values smaller than 1 will deactivate the corresponding channel
- **PhysicalMinimum CH1 CH8 [integer]:** specifies the minimum values of the incoming data
- **PhysicalMaximum CH1 CH8 [integer]:** specifies the maximum values of the incoming data
- **DigitalMinimum CH1 CH8 [integer]:** specifies the minimum values of the stored data
- **DigitalMaximum CH1 CH8 [integer]:** specifies the maximum values of the stored data

# **EmulateFaultyPlugin**

#### **Component Type: Actuator (Subcategory: Test)**

The component emulates a faulty plugin - which is a plugin that throws unexpected exceptions during start/pause/stop methods, or a plugin that has very long lasting method calls or even method calls hanging forever. The plugin is used to test the stability of the ARE in error situations. It supports the configuration of the duration of a method call including an endless method call.

## Requirements

No special hardware or software required.

### **Input Port Description**

- **inA [double]:** Input data of type double.
- inB [double]: Input data of type double.
- inC [string]: Input data of type string.
- inA [integer]: Input data of type integer.

## **Event Listener Description**

- eventA: An incmonig event A.
- eventB: An incoming event B.
- eventC: An incoming event C.
#### **Properties**

- **startException [boolean]:** Throw an exception when the plugin start method is called.
- **pauseException [boolean]:** Throw an exception when the plugin pause method is called.
- **stopException [boolean]:** Throw an exception when the plugin stop method is called.
- **resumeException [boolean]:** Throw an exception when the plugin resume method is called.
- **startDuration [integer]:** The duration in ms of the method call. If -1 is specified the method hangs forever and produces a thread dead lock.
- **pauseDuration [integer]:** The duration in ms of the method call. If -1 is specified the method hangs forever and produces a thread dead lock.
- **resumeDuration [integer]:** The duration in ms of the method call. If -1 is specified the method hangs forever and produces a thread dead lock.
- **stopDuration [integer]:** The duration in ms of the method call. If -1 is specified the method hangs forever and produces a thread dead lock.
- **inADuration [integer]:** The duration in ms of the method call. If -1 is specified the method hangs forever and produces a thread dead lock.
- **inBDuration [integer]:** The duration in ms of the method call. If -1 is specified the method hangs forever and produces a thread dead lock.
- **inCDuration [integer]:** The duration in ms of the method call. If -1 is specified the method hangs forever and produces a thread dead lock.
- **inDDuration [integer]:** The duration in ms of the method call. If -1 is specified the method hangs forever and produces a thread dead lock.
- eventADuration [integer]: The duration in ms of the method call. If -1 is specified the method hangs forever and produces a thread dead lock.

- eventBDuration [integer]: The duration in ms of the method call. If -1 is specified the method hangs forever and produces a thread dead lock.
- eventCDuration [integer]: The duration in ms of the method call. If -1 is specified the method hangs forever and produces a thread dead lock.

## **Enobio Display**

# **Component Type: Actuator (Subcategory: Graphical User Interface)**

This component displays the electrophysiological signals recorded by the Enobio device, so a user can check that the signals are being received and are correctly calibrated. The 4 Enobio channels are plotted with three different colours. When the signal is plotted in red it means that the channel is not being calibrated due to the configuration of the Enobio component. If the signal is plotted in yellow then this signal is in process of calibration. If the signals are plotted in green it indicates that the signal is correctly calibrated.



#### Requirements

This software component requires an Enobio receiver connected to the platform, the Enobio device switched on and the electrodes correctly placed on the user. In addition, the component expects an Enobio component connected to its inputs.

#### **Input Port Description**

- **Channel1 to Channel4 [double]:** Input ports for the corresponding output ports of the Enobio component.
- **Status [double]:** Input port to be connected to the status output port of the Enobio component.

### EnOcean

#### **Component Type: actuator (Subcategory: Home Control)**

The EnOcean plugin enables the ARE to interface to EnOcean wireless sensors. This plugin utilizes the Priscilla java library for the EnOcean implementation , it is (C)opyrighted by UAS FH Technikum Wien and released under the GNU General Public License (FSF v2). The EnOcean plugin provides an interface to the EnOcean sensors over an USB stick (EnOcean USB300) or an IP gateway. Interfacing to EnOcean devices is possible either by sending different data to actuators or receiving different types of values. This allows control of lightning, heat and ventilation devices and many more via the AsTeRICS platform.

	Component: enocean	1		
		÷ <b>±</b>		
	Search	Search Search		
	<ul> <li>Internal Properties</li> </ul>			
	Component Class	actuator		
	Component Name	enocean.1		
	Component Type	asterics.enocean		
	Component Descripti	<ul> <li>Enables the ARE to communicate with an EnOcean wireless network</li> </ul>		
	<ul> <li>Properties</li> </ul>			
	localIP	127.0.0.1		
	gatewayIP	172.0.10.2		
	USB	×		
	id1	0		
	sendType1	binary		
.1	dataValue1	0		
and	id2	0		
outputl	sendType2	binary		
output2	dataValue2	0		
output3	id3	0		
	sendType3	binary		
	dataValue3	0		
output5	id4	0		
output6	sendType4	binary		
	dataValue4	0		
	id5	0		
	sendType5	binary		
	dataValue5	0		
	id6	0		
	sendТуреб	binary		
	dataValue6	0		
	IDSlider1	0		
	sendTypeSlider1	binary		

EnOcean plugin

### Requirements

- An EnOcean gateway (either USB or IP)Basic knowledge on EnOcean device IDs

#### **Functional Description**

By starting the model, the EnOcean plugin establishes a connection to the EnOcean gateway specified by the plugin properties (defined by gatewayIP and USB checkbox). The gatewayIP is either the IP adress of the gateway or the COM port name (e.g. COM1 or /dev/ttyUSB0). There are 3 different ways of sending data to EnOcean devices. The first one is done by the 6 input slider ports. Each of them has properties for the source device ID (range: 0-127) and a data type (binary, temperature, illumination, humidity, temperature set point and fan level). On every update of the input data, an EnOcean data frame is generated and sent. The second method is done over the event input ports. There are 3 properties to define. The device ID and the data type have equal possible values as with the slider input. The third property defines the value to be sent. The first method is the most flexible one, by sending a command string to the command input port. It expects a special formatted string and parses the dedicated keywords and format to an EnOcean command. The string can be assembled individually with other string formatting plugins. Receiving sensor data is done by combining event and output ports. There are 2 properties, one for the type of input data (the data type are the same as stated before) and one for the source device ID (this property is a hexadecimal string, containing the full 32bit ID, printed on the sensor). The event ports are triggered each time, an event with the defined type and device ID is received. If an event is raised, at the same time the corresponding data value is available at the output ports.

#### **Input Port Description**

- command [string]: This port receives string commands for the EnOcean devices. The commands have to be in the format "@ENOCEAN:device\_id#type#value". An example for a valid command is"@ENOCEAN:25#binary#true". As a valid command is received by the input port, the value and data type is broadcasted to the devices with the given device ID offset (range: 0-127)
- **slider[1-6] [double]:** Input port for a double value which is converted to an integer (or another type, depends on the datatype) and broadcasted to all EnOcean devices in range

#### **Properties**

- **localIP [string]:** The local IP address of the interface which can reach the IP gateway.
- gatewayIP [string]: The destination IP address of the EnOcean gateway or the device name of the USB gateway (emulates a serial interface, e.g. COM1 or /dev/ttyUSB0)
- **USB [boolean]:** If checked, then the connection will be established through a serial interface, otherwise through an IP gateway
- **id[1-6] [integer]:** Source ID for input event[n]. The range is from 0 to 127.
- **sendType[1-6] [string]:** Send type for input event[n]. Following data types are allowed global (for every datatype property, port and event):
  - "binary": e.g. used for switching light actuators on or off.
  - "temperature": e.g. used to transmit a temperature value.
  - "illumination": e.g. used to transmit illumination values.
  - "humidity": e.g. used to transmit humidity values.
  - "setpoint": e.g. used to send set point value (+10K or -5% humidity)
  - "fan": e.g. used to set or simulate a defined fan level
- **dataValue[1-6] [string]:** The transmitted value for the input event [n] with the given id[n] and type (sendType[n])
- **IDSlider[1-6] [string]:** Source ID for slider[n]. The range is from 0 to 127.
- **sendTypeSlider[1-6] [string]:** The data type for sending slider[n] data with IDSLider[n]. The possible data types are stated above
- **IDTrigger[1-6] [string]:** Source ID for event trigger[n]. This is a full id, as printed on the device as 6 character string(e.g. FFEFA01C)
- **TypeTrigger[1-6] [string]:** The data type for the listening event[n]. If an incoming frame from the give ID (IDTrigger[n]) contains this data type, an event on event port [n] is raised. The possible data types are stated above

#### **Event Listener**

• **send[1-6]:** Each time a event is triggered, the coresponding EnOcean frame which is specified in the plugin properties, is transmitted.

#### **Event Trigger**

• **event\_out\_[1-6]:** Each time a frame is received with the given parameters (from properties IDTrigger[n] and TypeTrigger[n]), this event is raised

### **Event Visualizer**

# **Component Type: Actuator (Subcategory: Graphical User Interface)**

This plugin provides a graphical feedback for events. It can be used to monitor event activities and is mainly targeted for testing of configurations during setup time. The plugin offers a GUI (simple window with text output) where event names are displayed.

Deployment (Ctrl-D) GUI Desi	gner (Ctrl-G) 🛛 🗢	Properties (Ctrl-P)		<b>~</b> û
	*	componentType: Ever	ntVisualizer.1	
	E	Search		
		<ul> <li>Internal Properties</li> </ul>		
		Component Description	Simple Event Visualizer	
EventVisualizer.1		Component Name	EventVisualizer.1	
		ComponentType	actuator	Y
•	•	Properties (Ctrl-P)	∮ Event Listener	
	Events           Tue Jan 24 15:41:00           Tue Jan 24 15:41:01           Tue Jan 24 15:41:02           Tue Jan 24 15:41:03           Tue Jan 24 15:41:04	CET 2012: event_out_1_ CET 2012: event_out_1_ CET 2012: event_out_1_ CET 2012: event_out_1_ CET 2012: event_out_1_ CET 2012: event_out_1_e	aventDisplay aventDisplay aventDisplay aventDisplay aventDisplay	

EventVisualizer plugin

### Requirements

No special hardware or software required.

### **Port Description**

This plugin does not provide input or output ports.

### **Event Trigger Description**

• event\_in\_1: incoming events will be displayed in the GUI.

#### **Properties**

• **displayGUI [boolean]:** if selected, the GUI of this component will be displayed - if not, the GUI will be hidden and disabled.

## **File Writer**

#### **Component Type: Actuator (Subcategory: File System)**

This component writes the input values that are received in the input port to a text file, so these values can be analysed and processed off-line.

Deployment (Ctrl-D) GUI Designer (Ctrl-G) 🗧	Properties (Ctrl-P)		<b>→</b> ‡
·	componentType: FileW	/riter.1	
	Search		
	<ul> <li>Internal Properties</li> </ul>		
	Component Descriptio	File Writer actuator	
FileWriter.1	Component Name	FileWriter.1	
	ComponentType	actuator	
mpor	<ul> <li>Properties</li> </ul>		
	FileName	filewriter	
-	Properties (Ctrl-P)	Planut Portr	<b>5</b> 1-
	a Flopenies (curre)	input Forts	FIIE
	Writer plugi	'n	

#### **Input Port Description**

• input [double]: Input port for the values to be written to the text file.

#### **Properties**

• FileName [string]: Defines the name for the file where the input values are written. A timestamp indicating year, month, day, hour, minute and second of when the file is created is appended to the file name in order to identify different recordings.

# **Flickering Light Stimulator**

# **Component Type: Actuator (Subcategory: Brain Computer Interface)**

This plug-in allows the user to interact with the <u>Flickering Light Stimulator</u> panels by configuring their properties (stimulation frequency, duty cycle, intensity and colour) and start/stop the light stimulation. The stimulation frequency of the panels can be modified while the stimulation is running.

Sensors Processors Actuators	ts Edit Misc. E M Specials Saved • Groups •			
Deployment (Ctrl-D) =	Properties (Ctrl-P)		• д	1
	Component: Flickering	lightStimulator.1		0
				utp
FlickeringLightSt	Search			ut P
Imulator.1	<ul> <li>Internal Properties</li> </ul>	-		orts
panel E	Component Name	FlickeringLightStimulator.1	_	1.5
panel2	Component Type	asterics.FlickeringLightStimulator		
panel3	Component Descriptio	Flickering light stimulator	-	
panel4	Component Class	actuator	•	
	Properties	(e.	-	
	frequencyPanel1	1	-3	
	frequencyPanel2	1	-3	
	frequencyPanel3	<u></u>	-3	
	frequencyPanel4	1	-3	
	dcPanel1	50	-9	
	dcPanel2	50	<u> </u>	
	dcPanel3	50	<u> </u>	
	dcPanel4	50		
	intPanel1	100		
	intPanel2	100		
	intPanel3	100		
	intPanel4	100		
	red	X		
	blue	X		
	green	X		
	comPort	COM7		
	N	3000	۲	
<	Troperties (Ctrl-P)	🕈 Input Ports 🔰 Event Listener 🔰 Event Trigger		
ARE Status: Disconnected	100%	Ctrl +/-		

Flickering Light Stimulator plugin

#### Requirements

This software component requires at least one of the four Flickering Light Stimulator panels to be connected to any available COM port. The COM port number to which the panels are connected must be configured in the properties before starting the plugin.



Flickering Panels setup

#### **Input Port Description**

• **panel1 to panel4 [integer]:** input ports for the stimulation frequency of each panel in Hertz. Their value ranges from 1 to 50 Hertz.

#### **Event Listener Description**

- **startStim:** starts the stimulation of the flickering light panels when receiving an event.
- **stopStim:** stops the stimulation of the flickering light panels when receiving an event.
- **updateConfiguration:** updates the configuration of the panels (stimulation frequency, duty cycle, intensity and colour) when receiving an event.
- **startStimPeriod:** starts a stimulation of N miliseconds.

#### **Event Trigger Description**

• **stimPeriodFinished:** an event is emitted through this port when the stimulation period of N miliseconds has finished.

#### **Properties**

- **frequencyPanel1 [integer]:** stimulation frequency of panel 1 in Hertz. This property ranges from 1 to 50 Hertz.
- **frequencyPanel2 [integer]:** stimulation frequency of panel 2 in Hertz. This property ranges from 1 to 50 Hertz.
- **frequencyPanel3 [integer]:** stimulation frequency of panel 3 in Hertz. This property ranges from 1 to 50 Hertz.
- **frequencyPanel4 [integer]:** stimulation frequency of panel 4 in Hertz. This property ranges from 1 to 50 Hertz.
- **dcPanel1 [integer]:** duty cycle of panel 1. This property ranges from 1 (short cycle) to 100 (large cycle).
- **dcPanel2 [integer]:** duty cycle of panel 2. This property ranges from 1 (short cycle) to 100 (large cycle).
- **dcPanel3 [integer]:** duty cycle of panel 3. This property ranges from 1 (short cycle) to 100 (large cycle).
- **dcPanel4 [integer]:** duty cycle of panel 4. This property ranges from 1 (short cycle) to 100 (large cycle).
- **intPanel1 [integer]:** light intensity of panel 1. This property ranges from 1 (low intensity) to 100 (high intensity).
- **intPanel2 [integer]:** light intensity of panel 2. This property ranges from 1 (low intensity) to 100 (high intensity).
- **intPanel3 [integer]:** light intensity of panel 3. This property ranges from 1 (low intensity) to 100 (high intensity).
- **intPanel4 [integer]:** light intensity of panel 4. This property ranges from 1 (low intensity) to 100 (high intensity).
- red [boolean]: presence of red colour in all the panels.
- **blue [boolean]:** presence of blue colour in all the panels.
- green [boolean]: presence of green colour in all the panels.
- **comPort [string]:** COM port number to which the panels are connected. The string must be expressed as "COMx", where x is the COM port number.
- **N [integer]:** duration in miliseconds of the stimulation started by the startStimPeriod event.

## **FS20 Sender**

#### **Component Type: Actuator (Subcategory: Home Control)**

With the FS20 Sender commands for the FS20 house automation system can be sent over the PCS device sold by ELV electronics. See <u>the ELV</u> <u>FS20 homepage</u> for details.

### Requirements

The PCS sender must be attached to the system!



of the PCS FS20 sender

Picture

#### **Troubleshooting problems under Win8/Win8.1**

Under Windows 8 and Windows 8.1, there can be problems concering the operation of the FS20 PCS device: If the red control led stays on just a few seconds after plugging in the device, and then goes dark and teh PCS sender cannot be used, the reason could be that the power management settings on Win8/8.1 switch the device off per default - this can be changed by applying the following steps:

- Open the Device Manager (usually can be selected by right-clicking at the bottom left corner of your windows desktop)
- Open the device group "Human Interface Devices"
- Locate the device which pops up when you insert or remove the FS20 PCS module to/from an USB port It should be a "HID compliant supplier defined device". You can distinguish different devices by right-clicking a device and looking into rider "details" and "properites" - there you can find the VID and PID values, e.g. under property "last known parent"
- Note the VID and PID values. For example if you see a line like "USB\VID\_1B1F&PID;\_C00F\EEE0000473" the values are VID:1B1F and PID:C00F
- Press the Windows-Key and "R" to open the command window. Type "regedit" and press enter
- Select the entry "HKEY\_LOCAL\_MACHINE" > "SYSTEM" > "CurrentControlSet" > "Enum" > "USB".
- In this list, select the entry for the FS20 PCS (given by VID und PID you noted)
- Open the Subfolder "Device Paramteters" and double click the property "EnhancedPowerManagementEnabled" which appears in the right window section
- Change the property value from 1 to 0 and commit by clicking "OK"
- Restart you PC now hopefully the FS20 PCS works as intended.

#### **Input Port Description**

- **houseCode [int]:** The houseCode to which the command should be sent. Overrides the houseCode set in the properties
- **address [int]:** The address of the target device. Overrides the houseCode set in the properties.
- action [string]: Action input to send commands from other components which output a variable string, for example OSKA. The string format is as follows: @FS20:houseCode,address,command; e.g.@FS20:1111111,1234,18 to send the toggle command to the device with housecode 11111111 and address 1234. The delimiters ',', '\_' and ' ' are allowed. For the indices of the commands see the table below.

#### **Event Listener Description**

See table below for a list of all commands that can be triggered Command Mapping

Command	ID
Off	0
Level1	1
Leve2	2
Leve3	3
Leve4	4
Leve5	5
Leve6	6
Leve7	7
Leve8	8
Leve9	9
Level10	10
Level11	11
Level12	12
Level13	13
Level14	14
Level15	15
Level16	16
OnOldLevel	17
Toggle	18
Dim Up	19
Dim Down	20
Dim Up and Down	21
Program internal timer	22
Off for timer then old brightness level	24
On for timer then off	25

On old brightness level for timer then off	26
On for timer then old brightness level	30
On for old level then previous state	31

#### **Properties**

- **houseCode [integer]:** The default housecode for the component if there is no on the input port.
- **address [integer]:** The default address for the component if there is no on the input port.

## **GSM Modem**

#### **Component Type: Actuator (Subcategory: Communication)**

This component can perform send and receive SMS action through the GSM modem.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		џ
		componentType: GSM	IModem.1	
1		<ul> <li>Internal Properties</li> </ul>		
		Component Description	My Plugin description	
GSMModem.1		Component Name	GSMModem.1	
		ComponentType	actuator	
phoneID remotePhonei		<ul> <li>Properties</li> </ul>		
SMSContent receivedSMS		serialPort		
arrorNumber		pin		
		smsCenterID		
		defaultPhoneID		
·		🕈 Prop 🏾 🕈 Input	TOutp 🗲 Event 🗲 Event	

GSMModem plugin

### Requirements

A GSM modem with SMS option connected to the platform.

#### **Input Port Description**

- **phoneID [string]:** Phone number which will be used for the send SMS action.
- **SMSContent [string]:** SMS content which will be used for the send SMS action.
### **Output Port Description**

- **remotePhoneID [string]:** This is a phone number of the SMS sender.
- receivedSMS [string]: This is the content of the incoming SMS.
- errorNumber [integer]: The number of the error.

### **Event Listener Description**

• **sendSMS:** Sends the SMS message.

### **Event Trigger Description**

- **newSMS:** There is a new message.
- error: An error occurred.

### **Properties**

- **serialPort [string]:** The modem COM port. If this parameter is empty, the component uses the port of the first modem found.
- **pin [string]:**The PIN code for the SIM card. If the PIN is not needed this property should be empty.
- **smsCenterID [string]:** SMS Center ID. If the Center ID is not needed this property should be empty.
- **defaultPhoneID [string]:** This is a default phone number, which will be used for the send SMS actions.

### **Image Box**

# **Component Type: Actuator (Subcategory: Graphical User Interface)**

The Image Box is a GUI component which displays images loaded from image files.



ImageBox plugin

### **Input Port Description**

• **input [string]:** The path of the image file, which will be displayed.

### **Event Listener Description**

• **clear:** Removes the image from the component.

### **Event Trigger Description**

• **clicked:** The event is triggered when the user clicks on the component.

### **Properties**

- caption [string]: Caption of the component.
- **default [string]:** The path of the image file, which is displayed after start.
- **backgroundColor [integer]:** Defines background color.
- **displayGUI [boolean]:** If selected, the GUI of this component will be displayed if not, the GUI will be hidden and disabled.

### **IR Trans**

#### **Component Type: Actuator (Subcategory: Home Control)**

This component sends transmission commands to an infraredtransmitter. The different commands can be selected by the different events. The commands must be programmed into the IR-transmitter, before this component can be used. Furthermore, commands can also be sent to this component's input port ("action").

Action String example: The action string starts with "@IRTRANS:" and contains the remote control name and the command name. For example, if the IR-Transmitter is programmed with a database called "LG-TV" and the desired command is called "TvOn", the action string to play the IR-code is "@IRTRANS: snd LG-TV,TvOn".

Using the event based option, the property "prestring" is set to "snd LG-TV", and at property "send1" is set to "TvOn".

		C				
	*	Component: Ir I rans.1				
		Search				
		Internal Properties				
		Component Class	actuator			
		Component Name	IrTrans.1			
		Component Type	asterics.IrTrans			
		Component Descriptio	Infrared Transmission Module, interfaces to the IrTran	ns device		
		<ul> <li>Properties</li> </ul>				
		hostname				
		port	21000			
		prestring				
		send1				
		send2				
		send3				
		send4				
		send5				
		send6				
<u></u>		send7				
		send8				
		send9				

IRTrans plugin

### Requirements

The IRTrans module (USB, LAN or WiFi version) is required. It can be purchased from <u>http://www.irtrans.de</u>.



IRTrans universal IR remote module

### **Input Port Description**

• **action[string]:** A string, which will be sent to the IRTrans module, must start with "@IRTRANS".

#### **Output Port Description**

• **output[string]:** The IRTrans can also receive IR-Commands. With the IRTrans Configuration tool string commands can be defined for every received IR-Command which will be sent to a custom UDP server. If the target is the computer where the ARE runs on, the given string will be received by this output port.

### **Event Listener Description**

- **sendprop1:** sends the command, stored in the property send1 to the IRTrans (including the prestring).
- ...
- **sendprop24:** sends the command, stored in the property send24 to the IRTrans (including the prestring).

### **Properties**

- **hostname[string]:** The hostname/IP-address of the IRTrans. Use "localhost" if you have the IRTrans USB Version (IRTrans server tool must be running), or use the IP address of your IRTrans LAN or WiFi module.
- **port[string]:** The port of the IRTrans (default is 21000).
- **prestring[string]:** A string, being added before the strings of properties send1 to send24. Typically snd or snd , should be used.
- Send1[string]: This string (including the prestring) will be sent, if the event EventProp1 will be triggered.
- ...
- **Send24[string]:** This string (including the prestring) will be sent, if the event EventProp24 will be triggered.

*Comment:* This component is exactly tailored to the IRTrans module and needs expert knowledge to be configured.

## Keyboard

# **Component Type: Actuator (Subcategory: Input Device Emulation)**

The Keyboard component generates local keyboard input on the computer that the ARE is running on (per software emulation). The component supports key press/release actions and sending key sequences and special keycodes. Multiple instances of the Keyboard component can be used to provide different key actions.



Keyboard plugin

### Requirements

No special hardware or software required.

### **Input Port Description**

• **keyCodes [string]:** An incoming string which consists of alphanumeric characters and special key codes. The keys are sequentially generated as local keystrokes as the string is received, and as the sendKeys and other input related events of the component are being triggered.

### **Event Listener Description**

- **sendKeys:** An incoming event at this port generates all keycodes of the keycode string (sequentially).
- **pressKey:** An incoming event at this port generates a press (hold and release) event on the next keycode of the keycode string. After the last character, the send position will be reset to the first character.
- **holdKey:** An incoming event at this port generates a hold event on the next key of the keycode string (the key is pressed but not released). The holdKey listener can be used together with the releaseKey feature to create long key presses of single keys (e.g. of the cursor keys) to allow game control etc.
- **releaseKey:** An incoming event at this port releases the current key of the keycode string.

### **Properties**

- **keyCodeString [string]:** A string containing keys and keycodes. Please note that this string will be replaced by an incoming string at the keycodes input port. The keyCodeString can contain alphanumeric characters and special characters. Special characters are written in parentheses, for example {SHIFT}, {CTRL}, {ALT}, {BACKSPACE}, {ENTER} etc. Modifier keys like {SHIFT} or {ALT} are combined as they appear consecutively in the keystring, and are generated with the next printable character. For example, the keystrings "{SHIFT}" or "{CTRL}{ALT}{DEL}" are sent as single key values to the target computer. For a description of the currently supported special key codes please refer to Appendix B of the user manual.
- **inputMethod [integer]:** Declares whether to use sending window messages or system-wide SendInput API function as the way how keyboard input is injected.
- waitTime [integer]: Defines a number of milliseconds for a pause which shall occur when {WAIT} appears in the keycode string.

## KNX

#### **Component Type: actuator (Subcategory: Home Control)**

The KNX plugin enables the ARE to interface a KNX installation. This plugin utilizes the calimero java library (http://calimero.sourceforge.net/) for the KNX implementation, it is (C)opyrighted 2006-2008 by W. Kastner and released under the GNU General Public License (FSF v2 or later). The KNX plugin provides an interface to the KNX home automation bus via a KNXnet/IP router. KNX actuators with specific group addresses can be switched on or off, or a specific value can be sent to the KNX actuators. This allows control of lightning, heat and ventilation, blend control or other utilization of other home automation facilities via the AsTeRICS platform.



### Requirements

- A KNX infrastructure
- A little knowledge on KNX

### **Functional Description**

By starting the model, the KNX plugin establishes a connection to the KNX gateway specified by the plugin properties. Currently, this plugin supports only IP connection (due to the limitation of the underlaying calimero library, USB interfaces will be supported with Java8). The plugin has three different possibilities to interface KNX datapoints:

- Transmitting: Input ports (slider): Intended for dimming actuators
- Transmitting: Event Listeners: Used to send a defined value (with a given type) to a KNX group address.
- Transmitting: Action string: The most flexible way to send something to KNX. The command structure is: @KNX: group\_address,datapoint\_type,value. It is possible to use either a space, a comma or a number sign as separation tokens. The datapoint type is described later.
- Receiving: Event Triggers: An event trigger can be used to listen to a defined KNX group address. If there is ANYTHING happening on this dedicated group address, an event will be raised.
- Receiving: Output ports: The received value of a KNX group address will be interpreted by the given data point type, parsed to a string and sent out to the output ports

### **KNX datapoint types**

KNX itself doesn't provide any possibility to get information on how to interpret received data. This is the point where you need the datapoint types. There are a few main types and some sub types. For example: *1.001* is the datapoint type for a simple switch actuator/sensor. The input values are varying on the different datapoint types. For the switch example, the valid values are *on* and *off*. The full list of all datatypes is provided by the ARE. If you want to receive a full list, you have to place the plugin in your ACS model, connect to the ARE and upload the model (even without any functionality). Afterwards, the full list of all currently supported datapoint types is available. Every list entry contains following information:

- Description
- [Minimum,Maximum value]
- (Datapoint ID)

The example of the light switch:

*Switch[off,on] (1.001)* The literal name for this datapoint type is "Switch" and its ID is 1.001. Because this is a boolean value, off and on are the only valid values. For float values, these given values are representing the minimum and the maximum value.

### **Input Port Description**

- actionString [string]: This port receives string commands for the KNX component. The commands have to be in the format "@KNX: group\_address,datapoint\_type,value". The delimiters ',', '#' and ' ' are allowed. An example for a valid command is"@KNX:1/1/1,1.001,on". As a valid command is received by the input port, the value is sent to the KNX group address. The group addresses comply to the setup of the KNX network which can be configured with the ETS software.
- **slider[1-6] [double]:** Input port for a double value which is converted to a given datapoint (Property: DPTSlider[1-6]) and sent to the given KNX group address (Property: groupAddressSlider[1-6]

### **Output Port Description**

• **data [1-6] [string]:** These output ports are sending received data from the group addresses (Property: groupAddressOutput[1-6]). The data interpretation is controlled by the DPT (datapoint type, property: DPTOutput[1-6])

### **Properties**

- **localIP [string]:** The local IP address of the interface which can reach the KNXnetIP.
- KNXNetIP [string]: The destination IP address of the KNX gateway.
- NAT [boolean]: Enable NAT
- groupAddress[1-6] [string]: (Event Listener) Specifies KNX group addresses for the [n] Event Listener e.g. "1/1/1".
- dataValue[1-6] [string]: (Event Listener) Date value, which should be sent (triggered by the input event). The possible values are determined by the used DPT property (see the chapter on datapoint types)
- **DPTEvent[1-6] [string/dropdown]:** (Event Listener) Specifies the DPT (datapoint type) of the dataValue[1-6] which is sent to the KNX group address, triggered by the input event.
- groupAddressSlider[1-6] [string]: (Input port) Specifies KNX group addresses for the Slider[n].
- **DPTSlider[1-6] [string/dropdown]:** (Input port) Determines the datapoint type corresponding to the sliders [1-6].
- groupAddressTrigger[1-6] [string]: (Event Trigger) Specifies KNX group addresses for the Event Trigger [n]. The event is triggered if ANY action is happening on the given group address. If you need additional data processing, please use the output ports and process the received data values.
- groupAddressOutput[1-6] [string]: (Output port) Specifies the KNX group adresses, which are sent to the output ports. If there is anything happening on the given group adresses, the corresponding data is sent to the output ports.
- **DPTOutput[1-6] [string/dropdown]:** (Output port) Determines the datapoint type for the output port. The received data interpretation is defined by the DPTOutput property.

### **Event Listener**

- **send[1-6]:** Each time an event is triggered, the corresponding KNX command which is specified in the plugin properties (groupAddress[1-6],dataValue[1-6],DPTEvent[1-6]), is issued.
- **read[1-6]:** Trigger a read command on the output port. The settings for the group address and the DPT are the same as for the output ports (groupAddressOutput[1-6], DPTOutput[1-6]).

### **Event Trigger**

• **event\_out\_[1-6]:** When there is any activity on the given KNX group address, set by the properties (groupAddressTrigger[1-6]) an event is raised by this plugin.

## **Legacy Digital Out**

# **Component Type: Actuator (Subcategory: Generic Control Output)**

The DigitalOut plugin operates the output ports of the legacy GPIO CIM (CIM Id: 0x0201). The output ports 1-4 are open-collector outputs, where a pull-up resistor can be activated or deactivated using the plugin's properties. The output ports 5-8 are relais outputs where loads can be connected via a galvanic isolation barrier. The plugin provides event listener ports which serve the activation or deactivation of an output channel, and a command port which accepts string parameters to set, clear and toggle particular output channels.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)	<b>–</b> 1	
	*	componentType: DigitalOut.1		
		Search		
	=	<ul> <li>Internal Properties</li> </ul>		
ButtonGrid.1 DigitalOut.1		Component Descriptic	Digital Output control of GPIO-CIN	
		Component Name	DigitalOut.1	
action		ComponentType	actuator 💌	
		<ul> <li>Properties</li> </ul>		
		pullupStateOut1	×	
		pullupStateOut2		
		pullupStateOut3		
		pullupStateOut4		
	+			
<		TProperties (Ctrl-P)	7 Input Ports 🔰 Event Listener	
				Lega

DigitalOut plugin

### Requirements

This component requires the GPIO CIM (CIM Id: 0x0201) to be connected to an USB port.



Legacy GPIO CIM

### **Input port Description**

- action [string]: The plugin reacts to incoming action strings starting with "@GPIO:" and a command. Valid commands are "set", "clear", "toggle" and "press". The command has to be followed by a comma and the port number, for example: "@GPIO:set,1" or "@GPIO:toggle,2". The "press"-command toggles the given output port two times with a delay of 500 milliseconds. The following examples illustrate the available action strings:
  - "@DIGITALOUT:set,1": Pin 1 of the GPIO CIM will be set
  - "@DIGITALOUT:clear,2":Pin 2 of the GPIO CIM will be cleared
  - "@DIGITALOUT:toggle,1": Pin 1 of the GPIO CIM will be changed
  - "@DIGITALOUT:press,4": Pin 4 of the GPIO CIM will be cleared and after 500ms it will be set again

### **Event Listener Description**

- **setOutput1 to setOutput8:** an incoming event on these ports will cause the corresponding output port on the CIM to go to the high level.
- **clearOutput1 to clearOutput8:** an incoming event on these ports will cause the corresponding output port on the CIM to go to the low level.

### **Properties**

• **pullupStateOut1 to pullupStateOut4 [boolean]:** These properties specify if the internal pullup resistor shall be activated on the respective open collector output channels.

## LineWriter

#### **Component Type: Actuator (Subcategory: File System)**

This component writes lines from an input port to a text file.



ButtonGrid plugin

### **Input port Description**

• **actLine (string):** an incoming string is stored into the text file.
- **fileName [string]:** The full path and file name of the text file to be written. The path can be given as absolute path or relative to the ARE executable's directory
- addTimeToFileName [boolean]: If selected, a timestamp will be added to the filename, to avoid overwriting files
- **titleCaption [string]:** if not empty, this string parameter will be used as first line in the file (e.g. to create table caption in a csv file)
- **timestamp [integer, combobox selection]:** The type of timestamp which will be added befor each line (currently either "no timestamp" or "milliseconds" can be chosen).
- **lineEndMark [integer, combobox selection]:** The type of line end marker which will be added to each line (currently either "systemDefault", "newline" or "carriage-return + newline" can be chosen).
- **append [boolean]:** if selected, the lines will be appended to an existing file.

# MediaPlayer

#### **Component Type: Actuator (Subcategory: File System)**

This component allows to play media files supported by the local system. The supported file type depends on the installed media CODEC, but generally all media formats supported by the VLC player should work. Please note that the VLC player (32 bit version) should be installed in the default location: C:/Program Files/VideoLan or C:/Program Files (x86)/VideoLan



MediaPlayer plugin

- **filename [string]:** Input port for the mediafile name. The file must be present in the filesystem of the ARE. Sending to this port will change the media file.
- **position [double]:** Position in the media file (0-100%) Sending to this port will set the position in the media file.
- rate [double]: The playback speed of the media file (0-500%) Sending to this port will change the playback speed.

## **Output Port Description**

• **pos [double]:** The current playback position in percent.

### **Event Listener Description**

- **play:** Triggering this event will start (or resume) the mediafile playback.
- **pause:** Triggering this event will pause the mediafile playback.
- **stop:** Triggering this event will stop the mediafile playback.
- **reset:** Triggering this event will reset the mediafile playback to position 0.

- **filename [string]:** Initial mediafile name. The file must be present in the filesystem the ARE. This is a dynamic property: a synchronized ARE can suggest available files which are located in the data/videos and data/music subfolders of the ARE.
- **autoplay [boolean]:** if selected, the file will play automatically.
- **displayGUI [boolean]:** if selected, the GUI of this component will be displayed if not, the GUI will be hidden and disabled.
- **pathToVLC [string]:** path to the VLC installation (32-bit version needed).

# **Midi Player**

#### **Component Type: Actuator (Subcategory: Audio and Voice)**

The MidiPlayer component can generate midi tone output on the default midi device of the system. This can be used for audio feedback (e.g. for a selection or click) or for creation of musical instruments. The MidiPlayer Plugin features an optional GUI where the currently played tone can be seen.

GUI Designer (Ctrl-G)     Froperties (Ctrl-P)     Component: MidiPlayer.1     MidiPlayer.1     MidiPlayer.1									
GUI Designer (Ctrl-G) <ul> <li>X</li> <li>Properties (Ctrl-P)</li> <li>Component: MidiPlayer.1</li> <li>Search</li> <li>Internal Properties</li> <li>Component Name</li> <li>MidiPlayer.1</li> </ul>									
GUI Designer (Ctrl-G)     Froperties (Ctrl-P)     Component: MidiPlayer.1     MidiPlayer.1     MidiPlayer.1     MidiPlayer.1									
GUI Designer (Ctrl-G) WidiPlayer.1 MidiPlayer.1 MidiPlayer.1 MidiPlayer.1 MidiPlayer.1 MidiPlayer.1 Component Name MidiPlayer.1 Component Name MidiPlayer.1									
GUI Designer (Ctrl-G)           MidiPlayer.1           MidiPlayer.1           MidiPlayer.1           MidiPlayer.1	midiPlayer GUI								
GUI Designer (Ctrl-G) <ul> <li>X</li> <li>Properties (Ctrl-P)</li> <li>Component: MidiPlayer.1</li> <li>Search</li> <li>Internal Properties</li> <li>Component Name</li> <li>MidiPlayer.1</li> <li>MidiPlayer.1</li> <li>MidiPlayer.1</li> <li>Component Name</li> <li>MidiPlayer.1</li> <li>Component Value</li> <li>Component Value</li></ul>	_								
MidiPlayer.1         MidiPlayer.1         MidiPlayer.1             Search         Internal Properties         Component Name       MidiPlayer.1	<b>-</b> p								
MidiPlayer.1     Search       MidiPlayer.1     Component Name									
MidiPlayer.1  A Internal Properties  Component Name MidiPlayer.1  Component Value									
MidiPlayer.1 Component Name MidiPlayer.1									
Component Type asterics.MidiPlayer									
Component Descriptic This plugin is used for MIDI	арр								
Component Class actuator	•								
instrument  Properties	<ul> <li>Properties</li> </ul>								
scale midiDevice Gervill									
channel 1	\$								
instrument Vibraphone									
triggerThreshold 50	•								
triggerMax 100	\$								
pitchMin 0	•								
pitchMax 1000	\$								
toneScale alltones.sc									
playOnlyChangingNot 🗙									
displayGUI 🔀									

MidiPlayer plugin

- **trigger [integer]:** The input strength (tone trigger).
- pitch [integer]: The tone height input.
- **instrument [string]:** accepts a string which contains a valid instrument name to select this instrument
- **scale [string]:** accepts a string which contains a tone scale name to select this tone scale

- midiDevice [combobox]: The Midi Output device. This combobox allows selection of an installed Midi device for tone output. Supports value suggestions from ARE (dynamic property)
- **channel [combobox]:** The midi channel (1-16, 10 usually is the drum channel).
- **instrument [combobox]:** The midi instrument or controller (the first 127 selections are instruments, then 127 controllers are available. if a controller is selected, the incoming pitch value is use as control change value). **Supports value suggestions from ARE (dynamic property)**
- **triggerThreshold [integer]:** The value of the trigger input threshold. Tones will only be generated if the trigger input is bigger than this value. If the trigger input is not used, all tone heights received at the pitch input port are played with full volume.
- **triggerMax [integer]:** The maximum value of the trigger input. This value influences the volume of the played tones: Tones will get louder as the trigger input value approaches the triggerMax value.
- **pitchMin [integer]:** The minimum value for pitch input. Defines the pitch input value for the lowest tone output.
- **pitchMax [integer]:** The maximum value for pitch input Defines the pitch input value for the highest tone output.
- toneScale [integer]: One of several tone scales can be selected. The tonscales are stored in the plugin's subfolder in the ARE. Supports value suggestions from ARE (dynamic property)
- playOnlyChangingNotes [boolean]: If selected, only different note values are triggered (else, same note values can be played multiple times)
- **displayGUI [boolean]:** If this property value is set to true, the GUI window for the MidiPlugin will be displayed. The GUI window shows the current tone height and the available feedback tone scale.

# **Model Switcher**

#### **Component Type: Actuator (Subcategory: File System)**

The ModelSwitcher component allows to switch from the running model to another model which will be deployed and started. This makes it possible to build menus for different use-cases or switch from one usecase to another.



ModelSwitcher plugin

• **modelName [string]:** The name of the model (including extension, for example "CameraMouse\_sensitive.acs". The switch is performed as soon as the model name is received. The model must exist in the ARE/models folder of the runtime environment.

### **Event Listener Description**

• **switchModel:** An incoming event on this port will switch to the default model.

• **model[string]:** A fixed model name can be give here. This model must exist in the ARE/models folder of the runtime environment. The model switch is performed when the switchModel event is received.

## Mouse

# **Component Type: Actuator (Subcategory: Input Device Emulation)**

The Mouse component allows mouse cursor positioning and clicking on the computer the ARE is running on (by software emulation). The mouse x-position, y-position, press/release actions of three mouse buttons and mouse-wheel movements can be controlled via desired input values and event triggers.



#### plugin

#### Please Note: Mouse emulation on Windows 7

The mouse emulation on Windows 7 does not always work as expected due to User Account Control (UAC) settings. Especially when you want to use the Windows 7 On-Screen-Keyboard dragging the keyboard does not work. To troubleshoot <u>turn off the User Account Control (UAC)</u> – change the level to "Never notify"

- mouseX [double]: The desired X-Position of the mouse. This input port supports synchronization
- mouseY [double]: The desired Y-Position of the mouse. This input port supports synchronization
- **action [string]:** Input port for a command string. This command string allows to modify the action of the next left mouse click it can be set to trigger other types of mouse clicks. A command string may be composed of several items that are delimited by ',' or ' '. Following command strings are accepted:
  - "@MOUSE:nextclick,right": next left click event will create a right mouse button click.
  - "@MOUSE:nextclick,double": next left click event will create a double click.
  - "@MOUSE:nextclick,middle": next left click event will create a middle button click
  - "@MOUSE:nextclick,drag": next left click event will hold the left mouse button.
  - "@MOUSE:nextclick,release": next left click event will release the left mouse button.
  - "@MOUSE:action,enable": enables all mouse actions.
  - "@MOUSE:action,disable": disables all mouse actions.
  - "@MOUSE:action,toggle: enables / disables all mouse actions.

#### **Event Listener Description**

- **leftClick:** An incoming event at this port creates a mouse button click. A left mouse button click will be generated, unless a valid "nextclick..." command has been received at the cmd input which changed the click type (see above).
- **middleClick:** A click with the middle mouse button is generated.
- **rightClick:** A click with the right mouse button is generated.
- **doubleClick:** A double click with the left mouse button is generated.
- **dragPress:** The left mouse button is pressed (but not released again).
- dragRelease: The left mouse button is released.
- wheelUp: The mouse wheel is turned one position from the user.
- wheelDown: The mouse wheel is turned one position to the user.
- activate: enables all mouse actions.
- deactivate: disables all mouse actions.
- toggle: enables / disables all mouse actions.
- **absolutePosition:**after this event is triggered the incoming values for mouseX and mouseY are interpreted as absolute movement information
- **relativePosition:**after this event is triggered the incoming values for mouseX and mouseY are interpreted as relative movement information
- **nextClickRight** next left click event will create a right mouse button click.
- nextClickDouble next left click event will create a double click.
- nextClickMiddlenext left click event will create a middle button click
- **nextClickDrag** next left click event will hold the left mouse button.
- **nextClickRelease** next left click event will release the left mouse button.

- **enableMouse [boolean]:** The value of this property specifies if all mouse actions are bypassed (false) or enabled (true).
- **absolutePosition [boolean]:** If this property value is set to false, incoming values at the mouseX and mouseY input ports are interpreted as relative movement information. The values are summed up (integrated) to calculate the absolute position. If the property value is set to true, the values of the input ports are treated as absolute x/y positions.
- **xMin [integer]:** The minimum value for the X-coordinate (the mouse will not move farther to the left).
- xMax [integer]: The maximum value for the X-coordinate (the mouse will not move farther to the right). If the xMax property is set to 0, the horizontal screen resoltion will be assumed as maximum x-position for the mouse cursor.
- **yMin [integer]:** The minimum value for the Y-coordinate (the mouse will not move farther up).
- yMax [integer]: The maximum value for the Y-coordinate (the mouse will not move farther down) If the yMax property is set to 0, the vertical screen resoltion will be assumed as maximum y-position for the mouse cursor.

## **Mousecursor Icon**

# **Component Type: Actuator (Subcategory: Input Device Emulation)**

The MousecursorIcon plugin can be used to modify the current default system mouse cursor icon. A number of icon file names can be given as properties and activated by corresponding event listener ports. Currently, only the default system cursor can be modified. The cursor is restored to the default arrow when the model is stopped.

Deployment (Ctrl-D)	₹	Properties (Ctrl-P)	•		
		Component: MouseCur	rsorIcon.1		
		Search Search			
	m	<ul> <li>Internal Properties</li> </ul>			
		Component Name	MouseCursorIcon.1		
MourseCursorico		Component Type	asterics.MouseCursorIcon		
n.l		Component Descriptic	sets different cursor icnos for the local mouse curso		
() iconName		Component Class	actuator		
Convanie		<ul> <li>Properties</li> </ul>			
		icon1name	arrow		
		icon2name	arrow		
		icon3name	arrow		
		icon4name	arrow		
		icon5name	arrow		
		iconбname	arrow		
		icon7name	arrow		
		icon8name	arrow		
		icon9name	arrow		
	-				
	•	Properties (Ctrl-P)	Tinput Ports 📝 Event Listener		

MousecursorIcon plugin

• **iconName [strig]:** This input port accepts a filename, the plugin tries to load a cursor file with this name from the local plugin working directory.

#### **Event Listener Description:**

• **setIcon1 - setIcon9:** an incoming events sets the given cursor as new system cursor.

• iconName1 - iconName9 [string]: 9 slots for cursor file names.

# **Net Connection**

#### **Component Type: Actuator (Subcategory: Communication)**

This component provides interface to pass the data through the network. It allows to pass data to the another NetConnection component or to the application which uses the NetConnection Native ASAPI library.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)	<b>~</b> ↓	
	-	Component: NetConne	ction.1	
NetConnection.1	E	Parch Search		
		<ul> <li>Internal Properties</li> </ul>		
		Component Name	NetConnection.1	
integerInputP integerOutput			Component Type	asterics.NetConnection
integer/InputP integerOutput			Component Descriptio	This component provides the interface for con
integerOutput		Component Class	actuator 💌	
integerOutput integerOutput doubleOutput doubleOutput doubleOutput doubleOutput doubleOutput		<ul> <li>Properties</li> </ul>		
		connectionType	server 💌	
			IP	localhost
		port	21112	
		multisession		
doubleInputP doubleOutput				
doubleInputP doubleOutput				
stringInputPo				
stringOutputPo				
stringInputPo				
stringInputPo				
stringInputPo				
sangoupar				
	*			
· ·		TProp TInput	🗲 Event 🍼 Event 🍱 Outp	

NetConnection plugin

- **integerInputPort1...integerInputPort5 [integer]:** The integer values which are passed to these ports are sent to the remote receiver.
- **doubleInputPort1...doubleInputPort5 [double]:** The double values which are passed to these ports are sent to the remote receiver.
- **stringInputPort1...stringInputPort5 [string]:** The text values which are passed to these ports are sent to the remote receiver.

#### **Output Port Description**

- **integerOutputPort1...integerOutputPort5 [integer]:** The output ports for the integer values received from the remote sender.
- **doubleOutputPort1...doubleOutputPort5 [double]:** The output ports for the double values received from the remote sender.
- **stringOutputPort1...stringOutputPort5 [string]:** The output ports for the text values received from the remote sender.

### **Event Listener Description**

• **inputEvent1...inputEvent10:** The events which are sent to the remote receiver.

## **Event Trigger Description**

• **outputEvent1...outputEvent10:** The events received from the remote sender.

- **connectionType [integer]:** Describes connection mode: client or server.
- **IP [string]:** The IP address of the remote server.
- **port [integer]:** Port used in IP/TCP connection.
- **multisession [boolean]:** If the plugin is set to work as the server and this property is set, the plugin can connect to more than one client.

# **Osc Out Client**

#### **Component Type: actuator (Subcategory: Communication)**

The OscOutClient plugin enables the ARE to broadcast messages using the OpenSoundControl (OSC) protocol. This plugin utilizes the NetUtil java library (http://www.sciss.de/netutil/) for the OSC implementation, it is (C)opyrighted 2004-2011 by Hanns Holger Rutz and released under the GNU Lesser General Public License.



OscOutClient plugin

#### Requirements

- Nothing, works sand-alone within ARE
- Check your firewall configuration and network settings to ensure that OSC messages are not blocked.

#### **Functional Description**

The OscOutClient collects data form the inputs CH1-4 and assamble it to one OSC message with 4 arguments. The address for the OSC channel is set by the property AddressCh1. Furthermore the plugin has a StringIN input port. If the plugin receives a string it broadcast one OSC message with one string argument. The address for the string OSC channel is set by the property AddressStringCh.

• **CH[1-4] [double]:** Input port for the data, eg. a oszilloscope or any other double value.

- **Port [integer]:** This value specifies the OscOutClient port.
- **PeerAddress [string]:** Specifies the IP where the OSC server is listening.
- AddressCH1 [string]: This value specifies the OSC data output address e.g. "/path/to/receiver/accxyz".
- AddressStringCh [string]: This value specifies the OSC string output address.

## **Referred Plugins**

- OscServer
- OpenVibe OscGestureFollower

# Oscilloscope

# **Component Type: Actuator (Subcategory: Graphical User Interface)**

The Oscilloscope provides graphical output of one or two signal values. It is a very basic implementation but useful to visualize sensor values, changes in values and value history. The trace colours and update speed can be configured via component parameters.



Oscilloscope plugin

• in [double]: The input signal for the oscilloscope.

- **displayBuffer [integer]:** This property value specifies how often the oscilloscope window is drawn. For example if the display buffer size is 0, the oscilloscope traces are redrawn at every incoming value. If the display buffer size is set to 10, 10 values are stored in a buffer and drawn at once as the tenth value is received. This significantly reduces the computational resources spent for drawing the oscilloscope, which is useful especially at high update rates.
- **drawingMode [integer]:** Declares whether the y axis is adapting to mininum and maximum values automatically or to stay in preset bounds.
- **displayMode [integer]:** Affects the time when oscilloscope is redrawn. Can be set to the values "redraw on incoming samples" or "redraw periodically".
- **drawInterval [integer]:** Redraw interval in milliseconds (if periodic drawing is used).
- min [integer]: Preset minimum value for y axis of oscilloscope.
- max [integer]: Preset maximum value for y axis of oscilloscope.
- gridColor [integer]: The colour of the value-grid.
- **channelColor [integer]:** The colour of the signal trace for the channel.
- **backgroundColor [integer]:** The colour of the background of the oscilloscope window.
- fontSize [integer]: The size of the oscilloscope's caption.
- caption [string]: The caption to be displayed on the oscilloscope.
- **displayGUI [boolean]:** If selected, the GUI of this component will be displayed if not, the GUI will be hidden and disabled.
### **Phone Control**

#### **Component Type: Actuator (Subcategory: Phone Interface)**

This component controls a mobile phone with Windows Mobile operating system (v. 5.0 and above) through a Bluetooth connection. Currently supported functions are: Calling a phone number and accepting an incoming call, sending and receiving SMS messages. These functions can either be triggered by incoming events or by sending string commands to the phone plugin.

Deployment (Ctrl-D) GUI Designer (Ctrl-G) 🗧	Properties (Ctrl-P)	<b>▼</b> [	<b>џ</b>	
A	componentType: PhoneControl.1			
	€ 2↓ Search			
	<ul> <li>Internal Properties</li> </ul>			
	Component Descriptic	Phone plugin		
PhoneControl	Component Name	PhoneControl.1		
	ComponentType	actuator 💌		
phoneID	<ul> <li>Properties</li> </ul>			
SMSContent	defaultPhoneID			
command	bluetoothPhoneName			
erroriumber	port	-1 😫	3	
· •	🕈 Prop 🏾 🕈 Input	Toutp 🦸 Event 🔰 Event		
ARE Status: Disconnected	00%	Ctrl +/-		
Phor	neControl plugin			

### Requirements

A Phone running Windows Mobile, a Bluetooth dongle or Bluetooth HW support, Microsoft Bluetooth stack active. AsTeRICS Phone server application running on the mobile phone.

- **phoneID [string]:** This is the phone number which will be used for actions like: send SMS, make phone call.
- **SMSContent [string]:** This is the SMS content which will be used for sending SMS action.
- **command [string]:** String command that can be sent to this component from other plugins to trigger phone actions. Currently supported commands are:
  - @PHONE: SMS:Phone\_ID, "Message\_content"
  - @PHONE: CALL: Phone\_ID
  - @PHONE: ACCEPT
  - @PHONE: DROP

#### **Output Port Description**

- **remotePhoneID [string]:** This is a phone number of the caller or SMS sender.
- receivedSMS [string]: This is the content of the incoming SMS.
- errorNumber [integer]: The number of the error.

### **Event Listener Description**

- sendSMS: Sends SMS.
- makePhoneCall: Makes the phone call.
- acceptPhoneCall: Accepts the incoming phone call.
- dropPhoneCall: Drops the phone call.
- **reconnect:** Reconnects the phone.

### **Event Trigger Description**

- **idleState:** Phone is in the idle state.
- **ringState:** Phone is in the ring state.
- **connectedState:** Phone is connected with the remote phone.
- **newSMS:** There is a new SMS.
- error: The error occurs.

### **Properties**

- **defaultPhoneID [string]:** This is a default phone number, which will be used for send SMS and make phone call actions.
- **bluetoothPhoneName [string]:** This is a Bluetooth name of the phone which the component will connect.
- **port [integer]:** This is a Bluetooth port number.

## **Platform Digital Out**

# **Component Type: Actuator (Subcategory: Personal Platform)**

The PlatformDigitalOut plugin operates the output ports of the AsTeRICS Personal Platform. The output ports 1-2 are open-collector outputs with a deactivated pull-up resistor.

Deployment (Ctrl-D)	GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)	🛥 û
		*	Component: Platform	DigitalOut.1
			Search	
			<ul> <li>Internal Properties</li> </ul>	
	PlatformDigitalO		Component Name	PlatformDigitalOut.1
		Component Type	asterics.PlatformDigitalOur	
		Component Description	Control for the digital out	
			Component Class	actuator
		<ul> <li>Properties</li> </ul>		
l			pullupOutput1	
			pullupOutput2	
		÷		
<		•	2 Pro 2 Inp 1	Eve
	51.11	D: 11 10		

PlatformDigitalOut plugin

### Requirements

This component requires the Core CIM (CIM Id: 0x0602) of the AsTeRICS Personal Platform.



The AsteRICS Personal Platform

- **command [string]:** The plugin reacts to incoming action strings starting with "@GPIO:" and a command. Valid commands are "set", "clear", "toggle" and "press". The command has to be followed by a comma and the port number, for example: "@GPIO:set,1" or "@GPIO:toggle,2". The following examples illustrate the available action strings:
  - "@DIGITALOUT:set,1":Output port 1 of the Personal Platform will be set (5 Volt)
  - "@DIGITALOUT:clear,2":Output port 2 of the Personal Platform will be cleared (0 Volt)
  - "@DIGITALOUT:toggle,1":Output port 1 of the Personal Platform will be changed

### **Event Listener Description**

- **setOutput1 to setOutput2:** an incoming event on these ports will cause the corresponding output port to go to the high level.
- clearOutput1 to clearOutput2: an incoming event on these ports will cause the corresponding output port to go to the low level.

## **Platform LCD**

# **Component Type: Actuator (Subcategory: Personal Platform)**

The PlatformLCD component handles interaction with the display and the input buttons of the AsTeRICS Personal Platform. The component allows other components to send messages to the display.



PlatformLCD plugin

### Requirements

This component requires the Core CIM (CIM Id: 0x0601) of the AsTeRICS Personal Platform.



The AsteRICS Personal Platform

• **input [string]:** The Input of the port will be displayed on the Core CIM's display

### Pong

#### **Component Type: Actuator (Subcategory: Others)**

The Pong component is an implementation of the classic "pong"-game, where two players control their paddles and try to hit a moving ball. The Pong component offers different input modalities (speed or event-based) so that the game can be played via a wide range (and combination of) sensors. Two users can play the game using different sensors. Several game options can be controlled by the plugin properties.



plugin



### Application

A special "bike-Pong" interface has been developed at UAS Technikum Wien, which allows playing the game via bicycle ergometers. For this purpose, the Arduino microcontroller (and corresponding plugin) are utilized to measure the user interactions and deliver the data to the Pong plugin. The energy created by the pong-players can be calculated in calories (this mode makes only sense when the bike-ergonometers are connected, see properties) Please note that the bike mode only works well with a screen resolution of 1920x1090 because of the utilized background graphics.



bike-ergomenter controlled pong game

- **playerOnePos [integer]:** This input port defines the position of player one's paddle (0 to 300)
- playerTwoPos [integer]: This input port defines the position of player two's paddle (0 to 300)
- playerOneSpeed [integer]: This input port defines the speed of player one's paddle (-10 to 10)
- playerTwoSpeed [integer]: This input port defines the speed of player two's paddle (-10 to 10)

### **Event Listener Description**

- **startGame:** An incoming event starts/restarts the game
- **playerOneToggleDirection:** An incoming event changes the direction of player one's paddle (only relevant for event-based paddle control mode).
- **playerTwoToggleDirection:** An incoming event changes the direction of player two's paddle (only relevant for event-based paddle control mode).
- **playerOneMovement:** An incoming event moves player one's paddle one step (only relevant for event-based paddle control mode). This input is also used for the calculation of the total amount of engery.
- **playerTwoMovement:** An incoming event moves player two's paddle one step (only relevant for event-based paddle control mode). This input is also used for the calculation of the total amount of engery.

### **Properties**

- **controlMode [combobox selection]:** selects the mode for controlling the paddle positions. possible selections are: absolute position (via input port), speed (via input port) or single events.
- **speedStep [double]:** defines the amount of movement caused by one event.
- goalsToWin [double]: number of goals to win a game (player lives).
- eventsToCaloryMultiplier [double]: factor to calculate energy (in calories) from incoming events (especially for the bike ergometer application) Setting this property value to 0 deactivates the energy calculation and the respective game report screens (default).
- goalScoreBase [integer]: game points for one goal.
- touchScoreBase [integer]: game points for one ball hit.
- resetWaitTime [integer]: time to wait before resetting game screen.
- maxSpeed [double]: the maximum speed of the ball.
- **minXSpeed [double]:** the minimum X speed of the ball (to avoid deadlocks of the gameplay).
- **reflectionYImpulse [double]:** speed impluse gained from a vertical reflection of the ball.
- **soundFilePaddleTouch [string]:** a wav file which is played when the ball touches a paddle.
- **soundFileBoundsTouch [string]:** a wav file which is played when the ball touches the vertical bounds.
- **soundFileGoal [string]:** a wav file which is played when a player missed a ball.
- **soundFileEndGame [string]:** a wav file which is played when the game is over.

### **Remote Joystick**

## **Component Type: Actuator (Subcategory: Input Device Emulation)**

The RemoteJoystick component interfaces the AsTeRICS Personal Platform to a second computer via the HID actuator CIM (USB dongle, plugged into the target computer). The HID actuator emulates a standard HID Joystick device on the target computer (no special driver software is needed). The Joystick controller is compatible to the Playstation3 SixAxis controller and can be used for PS3 game interfacing. The X/Y axis represent the left stick, the Z/R axis represent the right sick. The button numbers correspond to the PS3-controller buttons as labeled in the event port descriptions. The Joystick analog positions, the Point-Of-View angle and up to 13 Buttons can be emulated via desired input values and event triggers. Note that multiple instances of the Remote-components (RemoteJoystick, RemoteKeyboard and RemoteMouse) can be used concurrently with one HID actuator USB dongle, e.g. to provide different actions from up to three different input devices on the target computer.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Event Listener	<b>▼</b> ‡
	*	Listeners	Description
	=	pressButton1	presses button 1 (Square)
		releaseButton1	releases button 1 (Square)
RemoteJoystick.		pressButton2	presses button 2 (Cross)
		releaseButton2	releases button 2 (Cross)
JoystickA		pressButton3	presses button 3 (Circle)
iovstickZ		releaseButton3	releases button 3 (Circle)
joystickR		pressButton4	presses button 4 (Triangle)
joystickPov		releaseButton4	releases button 4 (Triangle)
		pressButton5	presses button 5 (L1)
		releaseButton5	releases button 5 (L1)
		pressButton6	presses button 6 (R1)
		releaseButton6	releases button 6 (R1)
		pressButton7	presses button 7 (L2)
		releaseButton7	releases button 7 (L2)
	+	4	III +
<	•	Troperties (Ctrl-P)	Input Ports 🦸 Event Listener

RemoteJoystick plugin

### Requirements

The HID Actuator CIM (CIM ID 0x0101) has to be plugged into a free USB port of the target computer and the cable has to be connected to the AsTeRICS platform.



the HID Actuator plugs into the target computer and connects via Bluetooth wirelessly to the ARE

- joystickX [integer]: The desired X-Position of the Joystick.
- joystickY [integer]: The desired Y-Position of the Joystick.
- joystickZ [integer]: The desired Z-Position of the Joystick.
- joystickR [integer]: The desired R-Position of the Joystick.
- joystickPov [integer]: The desired POV-angle of the Joystick.

### **Event Listener Description**

- **pressButton1 pressButton13:** An incoming event at this port causes the corresponding button of the Joystick to be pressed.
- releaseButton1 releaseButton13: An incoming event at this port causes the corresponding button of the Joystick to be released.

### **Properties**

- **refreshInterval [integer]:** This property value sets a minimum time (in milliseconds) between two updates of the remote joystick device. If set to 0, no limit is given to the update rate which should be okay unless very high update rates (>100 Hz) of the joystick angles are performed.
- uniqueld: unique number of the CIM if more than one CIMs of the same type are used. The module flashes a LED for identification when the ID is selected. Supports value suggestions from ARE (dynamic property)

### **Remote Keyboard**

# **Component Type: Actuator (Subcategory: Input Device Emulation)**

The RemoteKeyboard component interfaces the AsTeRICS Personal Platform to a second computer via the HID actuator CIM (USB dongle, plugged into the target computer). The HID actuator emulates a standard USB keyboard device on the target computer (no special driver software is needed). The component supports key press/release actions and sending key sequences and special keycodes. Note that multiple instances of the Remote-components (RemoteJoystick, RemoteKeyboard and RemoteMouse) can be used concurrently with one HID actuator USB dongle, e.g. to provide different actions for up to three

different input devices on the target computer.



RemoteKeyboard plugin

### Requirements

The HID Actuator CIM (CIM ID 0x0101) has to be plugged into a free USB port of the target computer and the cable has to be connected to the AsTeRICS platform.



the HID Actuator plugs into the target computer and connects via Bluetooth wirelessly to the ARE

• **keyCodes [string]:** An incoming string which consists of alphanumeric characters and special key codes. The keys are sequentially sent to the target computer via the HID actuator as the string is received and as the elp\_sendkey(s) events of the component are being triggered.

### **Event Listener Description**

- **sendKeys:** An incoming event at this port sends the whole keycodes string to the target computer. The key will be released immediately after it has been pressed.
- **pressKey:** An incoming event at this port sends the next keycode of the keycode string to the target computer. After the last character, the send position will be reset to the first character.
- **holdKey:** An incoming event at this port holds the current key of the keycode string on the target computer (the key is pressed but not released). The holdKey function can be used together with the releaseKey function to create long key presses of single keys (e.g. of the cursor keys) to allow game control etc.
- **releaseKey:** An incoming event at this port releases the current key of the keycode string on the target computer.

### **Properties**

- keyCodeString [string]: A string containing keys and keycodes. Please note that this string will be replaced by an incoming string at the keycodes input port. The keycode-string can contain alphanumeric characters and special characters. Special characters are written in parentheses, for example {SHIFT}, {CTRL}, {ALT}, {BACKSPACE}, {ENTER} etc. Modifier keys like {SHIFT} or {ALT} are combined as they appear consecutively in the keystring, and are sent with the next a printable character. For example, the keystrings "{SHIFT}a" or "{CTRL}{ALT}{DEL}" are sent as single key-values to the target computer.
- **uniqueld:** unique number of the CIM if more than one CIMs of the same type are used. The module flashes a LED for identification when the ID is selected. **Supports value suggestions from ARE** (dynamic property)

### **Remote Mouse**

## **Component Type: Actuator (Subcategory: Input Device Emulation)**

The RemoteMouse component interfaces the AsTeRICS Personal Platform to a second computer via the HID actuator CIM (USB dongle, plugged into the target computer). The HID actuator emulates a standard USB mouse on the target computer (no special driver software is needed). The mouse x-position, y-position, press/release actions of three mouse buttons and mouse-wheel movements can be controlled via desired input values and event triggers. Note that multiple instances of the Remote components (RemoteJoystick, RemoteKeyboard and RemoteMouse) can be used concurrently with one HID actuator USB dongle, e.g. to provide different key actions for up to three different input devices on the target computer.



RemoteMouse plugin

### Requirements

The HID Actuator CIM (CIM ID 0x0101) has to be plugged into a free USB port of the target computer and the cable has to be connected to the AsTeRICS platform.



the HID Actuator plugs into the target computer and connects via Bluetooth wirelessly to the ARE

- mouseX [integer]: The desired X-Position of the mouse. This input port supports synchronization
- mouseY [integer]: The desired Y-Position of the mouse. This input port supports synchronization
- **action [string]:** Input port for a command string. This command string allows to modify the action of the next left mouse click. It can be set to trigger other types of mouse clicks. A command string may be composed of several items that are delimited by ',' or ' '. Following command strings are accepted:
  - "@MOUSE: nextclick, right": next left click event will create a right mouse button click.
  - "@MOUSE: nextclick, double": next left click event will create a double click.
  - "@MOUSE: nextclick, middle": next left click event will create a middle button click.
  - "@MOUSE: nextclick, drag": next left click event will hold the left mouse button.
  - "@MOUSE: nextclick, release": next left click event will release the left mouse button.
  - "@MOUSE: action, enable": enables all mouse actions.
  - "@MOUSE: action, disable": disables all mouse actions.
  - "@MOUSE: action, toggle": enables / disables all mouse actions.

### **Event Listener Description**

- **leftClick:** An incoming event at this port creates a mouse button click. A left click will be generated, unless a valid "nextclick ..." command has been received at the cmd input which changed the type of the click (see above).
- **middleClick:** A click with the middle mouse button is generated.
- rightClick: A click with the right mouse button is generated.
- **doubleClick:** A double click with the left mouse button is generated.
- **dragPress:** The left mouse button is pressed (but not released again).
- **dragRelease:** The left mouse button is released.
- wheelUp: The mouse wheel is turned one position from the user.
- wheelDown: The mouse wheel is turned one position to the user.
- **nextClickRight** next left click event will create a right mouse button click.
- nextClickDouble next left click event will create a double click.
- nextClickMiddlenext left click event will create a middle button click
- nextClickDrag next left click event will hold the left mouse button.
- nextClickRelease next left click event will release the left mouse button.

### **Properties**

- **absolutePosition [boolean]:** Currently not supported. All mouse position input values are interpreted as relative changes of the position (X- and Y-movement).
- **uniqueld:** unique number of the CIM if more than one CIMs of the same type are used. The module flashes a LED for identification when the ID is selected. **Supports value suggestions from ARE** (dynamic property)

### **Remote Tablet**

## **Component Type: Actuator (Subcategory: Input Device Emulation)**

The RemoteTablet component interfaces the AsTERICS Personal Platform to a second computer via the HID actuator CIM (USB dongle, plugged into the target computer). This plugins performs similar like the RemoteMouse plugin, with the difference that the coorindates are set absolutely via the table devie, not relatively like a mouse device works. The HID actuator emulates a standard USB Tablet on the target computer (no special driver software is needed). The Tablet x-position, yposition, press/release actions of Tablet buttons and wheel movements can be controlled via desired input values and event triggers. Note that multiple instances of the Remote components (RemoteJoystick, RemoteKeyboard, RemoteMouse and RemoteTablet) can be used concurrently with one HID actuator USB dongle, e.g. to provide different key actions for up to four different input devices on the target computer.

Deployment (Ctrl-D) GUI Designer (Ctrl-G)	Ŧ	Properties (Ctrl-P)		₩ џ
		Component: RemoteTa	iblet.1	
		€ AL Search		
		<ul> <li>Internal Properties</li> </ul>		
RemoteTablet.1		Component Descriptio	Remote tablet emulation via HID actuator	
		Component Type	asterics.RemoteTablet	
mouseX		Component Name	RemoteTablet.1	
mouseY		Component Class	actuator	-
action		<ul> <li>Properties</li> </ul>		
		absolutePosition	X	
	-			

RemoteTablet plugin
#### Requirements

The HID Actuator CIM (CIM ID 0x0101) has to be plugged into a free USB port of the target computer and the cable has to be connected to the AsTeRICS platform.



the HID Actuator plugs into the target computer and connects via Bluetooth wirelessly to the ARE

- mouseX [integer]: The desired X-Position of the cursor. This input port supports synchronization
- mouseY [integer]: The desired Y-Position of the cursor. This input port supports synchronization
- action [string]: Input port for a command string. This command string allows to modify the action of the next left Tablet click ¡V it can be set to trigger other types of Tablet clicks. Following command strings are accepted:
  - "@Mouse: nextclick, right": next left click event will create a right Mouse button click.
  - "@Mouse: nextclick, double": next left click event will create a double click.
  - "@Mouse: nextclick, middle": next left click event will create a middle button click.
  - "@Mouse: nextclick, drag": next left click event will hold the left Mouse button.
  - "@Mouse: nextclick, release": next left click event will release the left Mouse button.
  - "@Mouse: action, enable": enables all Mouse actions.
  - "@Mouse: action, disable": disables all Mouse actions.
  - "@Mouse: action, toggle": enables / disables all Mouse actions.

#### **Event Listener Description**

- **leftClick:** An incoming event at this port creates a Mouse button click. A left click will be generated, unless a valid "nextclick ..." command has been received at the cmd input which changed the type of the click (see above).
- **middleClick:** A click with the middle Mouse button is generated.
- rightClick: A click with the right Mouse button is generated.
- **doubleClick:** A double click with the left Mouse button is generated.
- **dragPress:** The left Mouse button is pressed (but not released again).
- dragRelease: The left Mouse button is released.
- wheelUp: The Mouse wheel is turned one position from the user.
- wheelDown: The Mouse wheel is turned one position to the user.
- **nextClickRight** next left click event will create a right mouse button click.
- nextClickDouble next left click event will create a double click.
- nextClickMiddlenext left click event will create a middle button click
- nextClickDrag next left click event will hold the left mouse button.
- nextClickRelease next left click event will release the left mouse button.

- **absolutePosition [boolean]:** Currently not supported. All Mouse position input values are interpreted as relative changes of the position (X- and Y-movement).
- **uniqueld:** unique number of the CIM if more than one CIMs of the same type are used. The module flashes a LED for identification when the ID is selected. **Supports value suggestions from ARE** (dynamic property)

### RemoteWindow

# **Component Type: Actuator (Subcategory: Graphical User Interface)**

This component allows moving or setting the state of a specific window (which is active and can be identified via the window title) to desired locations on the screen. Several default locations can be selected via incoming events (top, left, bottom, right or center of the screen). X- and Y- offset values can be defined - thus it becomes possible to move the remote window e.g to a second screen.

GUI Designer (Ctrl-G) = ×	Properties (Ctrl-P)	<b>▼</b> ↓	
^	Component: RemoteV	Vindow.1	
	Search		
	<ul> <li>Internal Properties</li> </ul>		
RemoteWindow.	Component Class	actuator	
	Component Name	RemoteWindow.1	
name	Component Type	asterics.RemoteWindow	
XPos	Component Description	provides basic Window manipulation for anothe	
YPos V	<ul> <li>Properties</li> </ul>		
	windowName	notepad	
	mode	window title contains text	
	xPos	0	
	yPos	0	
	autoSetPosition		
$\sim$			

RemoteWindow plugin

- **xPos [integer]:** The x offest value for positioning the window
- **yPos** [integer]: The y offest value for positioning the window

#### **Event Listener Description**

- **moveToTop:** moves the window to the top of the screen (y offset will be applied). The x position will not be changed.
- **moveToBottom:** moves the window to the bottom of the screen (y offset will be applied). The x position will not be changed.
- **moveToLeft:** moves the window to the left side of the screen (x offset will be applied). The y position will not be changed.
- **moveToRight:** moves the window to the right side of the screen (x offset will be applied). The y position will not be changed.
- **moveToCenter:** moves the window to the center of the screen (x and y offsets will be applied).
- minimize: minimizes the window to the taskbar
- **restore:** restores the window from the taskbar
- bringToFront: places the window on top of other windows
- moveNow: moves the window to the selected xPos and yPos positions

- windowName [string]: the window title / name
- mode [integer, combobox selection]: the search mode for finding the window. If "exact match" is selected, the window title must be exactly the same as the given name. If "contains text" is selected, the any window title which contains the given text will be found. Note that both modes are case-sensitive !
- **xPos [integer]:** default value for the x offset
- **yPos** [integer]: default value for the y offset
- **autoSetPosition [boolean]:** if selected, the position of the window will be set to xPos/yPos at startup

## SerialSender

# **Component Type: Actuator (Subcategory: Serial Communication)**

The Serialsender can be used to send structured data to Serial devices. It has 16 data slots. Whenever a send event occurs the plugin sends every slot which is active to the device, beginning with slot0

Component Class	actuator	
Component Name	serialSender.1	
Component Type	asterics.serialSender	
Component Descrip	tio My Plugin description	
<ul> <li>Properties</li> </ul>		
COMPort	COM0	
debug		
BaudRate	600	
slot0	255	
slot1	0	
slot2	0	
slot3	0	
slot4	0	
slot5	0	
slot6	0	
slot7	0	
slot8	0	
slot9	0	
slot10	0	
slot11	0	
slot12	0	
slot13	0	
slot14	0	
slot15	0	
slot0Active	X	
slot1Active	×	
slot2Active		

SerialSender plugin

• **slot0 - slot15 [int]:** Input data for each slot. The lower 8 Bit of the input will be sent when the slot is set Active and a send event occurs

- **COMPort [string]:** Defines the COM Port of the target serial device. e.g. COM0
- **BaudRate [integer]:** Defines the Baudrate for the communication. It must match the baudrate of the target device
- **Slot[0-15] [int]:**Defines the default value of a slot. This value will be overridden if there is data available at the corresponding input slot
- Slot[0-15]Active [boolean]: Activate a Slot. Whenever a send event occurs the SerialSender will iterate over all Slots beginning with slot 0 and send the data of every Active Slot
- **Slot[0-15]Delay [int]:**Defines the delay the plugin should wait before sending data to a slot.

## **Synthetic Voice**

#### **Component Type: Actuator (Subcategory: Audio and Voice)**

The Synthetic Voice component uses the SAPI 5 technology to generate synthetic voice.

Deployment (Ctrl-D) GUI Designer (Ctrl-G) 🗧	Properties (Ctrl-P)	-	<b></b>	
A	componentType: SyntheticVoice.2			
_	All Search			
	<ul> <li>Internal Properties</li> </ul>			
	Component Descriptic Uses SAPI technology to generate synthetic vo			
SyntheticVoice.2	Component Name	SyntheticVoice.2		
	ComponentType	actuator		
input	<ul> <li>Properties</li> </ul>			
	volume	100		
	speed	0		
	voice			
	xmlTags			
< 🗋 🔶 🔸	Troperties (Ctrl-P)	Tinput Ports		
ARE Status: Disconnected 10	00%	Ctrl +/-		

SyntheticVoice plugin

### Requirements

The appropriate voice shold be installed on the platform.

• **input [string]:** The text sentence, which will be converted into speech.

- **volume [integer]:** Defines the volume of the voice. The volume propety values should be in range from 0 to 100.
- **speed [integer]:** Defines the speed of the voice. The speed propety values should be in range from -10 to 10.
- voice [string]: Specifies the voice used for the speech synthesis.
- **xmlTags [boolean]:** Defines if the XML tags in the input text will be suported.

## **SkyWatcherMount**

#### **Component Type: Actuator (Subcategory: Others)**

The SkyWatcher plugin can be used to control telescope mounts using the Nexstar 5 protocol, e.g. Skywatcher AllView, Skywatcher Merlin via Synscan hand control.



SkyWatcherMount plugin

#### Requirements

A Nexstar5 compatible telescope mount connected to the target computer via a serial connection

- **speed [integer]:** defines the movement speed of the mount (range: 0-9)
- **panPosition [integer]:** defines the target position in steps for the pan axis for the goto command
- **tiltPosition [integer]:** defines the target position in steps for the tilt axis for the goto command
- endPointLeft [integer]: defines the maximum position the pan axis can move to the left
- **endPointRight [integer]:** defines the maximum position the pan axis can move to the right
- endPointUp [integer]: defines the maximum position the tilt axis can move up
- endPointDown [integer]: defines the maximum position the tilt axis can move down

#### **Output Port Description**

- **posX [integer]:** the actual position of the pan axis in steps
- **posY [integer]:** the actual position of the tilt axis in steps

#### **Event Listener Description**

- **goLeft:** moves the mount endless to the left
- **goRight:** moves the mount endless to the right
- goUp: moves the mount endless upwards
- **goDown:** moves the mount endless downwards
- **stopPan:** stops the movement of the pan axis
- stopTilt: stops the movement of the tilt axis
- stop: stops the movement of the pan and tilt axis

• **goToPanPosition:** move the pan axis to the step position specified by the panPosition input port

• **goToTiltPosition:** move the tilt axis to the step position specified by the tiltPosition input port

• **triggerOn:** if the mount has a shutter for a digital camera, the shutter is set to 1

• **triggerOff:** if the mount has a shutter for a digital camera, the shutter is set to 0

- SerialPort [string]: The serial port of the mount, e.g. COM1
- EndPointsActive [boolean]: defines if the movement boundaries set by the endpoints are active or not.

## **SSVEP File Writer**

# **Component Type: Actuator (Subcategory: Brain Computer Interface)**

This plugin writes to a text file the 4 EEG channels along with a software trigger received through the event listener ports. This file is lately analyzed by the <u>ProtocolSSVEPTrain</u> plugin to obtain the optimus frequencies to be used on the SSVEP detection.



SSVEPFileWriter plugin

- filename [string]: Name of the file to be saved.
- **channel1 to channel4 [integer]:** Input EEG signal from channels 1 to 4.
- **StimulationFrequency [integer]:** If a stimulation frequency value is received before the StarStimulation Event the stimulation frequency is appended to the name of the output file to be saved.

#### **Event Listener Description**

- **StartTrial:** An incoming event at this port starts the file writer process.
- **StopTrial:** An incoming event at this port stops the file writer process.
- **StartStimulation:** An incoming event at this port sets the trigger channel to the corresponding stimulation frequency value in Hz.
- **StopStimulation:** An incoming event at this port sets the trigger channel to zeros.

• **DefaultFileName [string]:** The default file name.

## **SSVEP Stimulator**

# **Component Type: Actuator (Subcategory: Brain Computer Interface)**

This plug-in allows the user to interact with the SW-generated flickering surfaces (panels) for SSVEP stimulation. The stimulation frequency of the panels can be modified before the stimulation is started.



SSVEPStimulator application

### Requirements

A recent version of DirectX has to be installed.

• **frequency [integer]:** the stimulation frequency the SW-generated panel in Hertz. The value ranges from 1 to 20 Hertz.

#### **Event Listener Description**

- **startStim:** starts the stimulation of the SW-generated flickering panels when receiving an event.
- **stopStim:** stops the stimulation of the SW-generated flickering panels when receiving an event.

#### **Event Trigger Description**

• **stimPeriodFinished:** an event is emitted through this port when the stimulation period of N miliseconds has finished.

- onBitmapFile [string]: filename of a bitmap file which is used in the on-phase of the stimulation. The file is expected in the data/SSVEPStimulator subfolder of the ARE. The filename is given without extension (e.g. "arrow\_up" for the file "ARE/data/SSVEPStimulator/arrow\_up.bmp")
- **offBitmapFile [string]:** filename of a bitmap file which is used in the off-phase of the stimulation. (same filname format as above)
- **xPosition [integer]:** x-position of the flickering surface on the desktop/screen.
- **yPosition [integer]:** y-position of the flickering surface on the desktop/screen.
- **frequency [integer]:** default stimulation frequency of in Hertz. This property ranges from 1 to 20 Hertz.
- **msec [integer]:** duration in miliseconds of the stimulation started by the startStimPeriod event.

## TeensyRCprototype

#### **Component Type: Actuator (Subcategory: Others)**

The TeensyRCprototype component allows to use radio-controlled toys from within the AsTeRICS framework. As a prerequisite, the Teensy\_RC\_CIM.hex firmware (see /CIMS/Teensy\_RC\_CIM) must be installed on a teensy microcontroller (see <a href="http://www.pjrc.com">http://www.pjrc.com</a>), and the teensy must be connected to a Walkera MTC-01 "MagicCube" or a similar remote control unit with PPM-signal input, which allows sending up to 8 channel values to a connected RC-receiver:



TeensyRC Remot control gear

#### Application

The channel values which are sent into the plugin are transformed into so-called "PPM-signals" which are supported by several toy remote controls. For example to use a RC-model car, only 2 channels are needed. For more information about the firmware, the needed modules and their application, please refer to the Bachelor-thesis by Alexander Frimmel in the documentation section

(/Documenation/DIYGuides/RC\_ToyControl\_Frimmel.pdf).



remote control of a toy car

- **channel1 [integer]:** This input port accepts the level for channel1 of the RC remote control (values 0-1000 are allowed, center:500)
- **channel2 [integer]:** This input port accepts the level for channel2 of the RC remote control (values 0-1000 are allowed, center:500)
- **channel3 [integer]:** This input port accepts the level for channel3 of the RC remote control (values 0-1000 are allowed, center:500)
- **channel4 [integer]:** This input port accepts the level for channel4 of the RC remote control (values 0-1000 are allowed, center:500)
- **channel5 [integer]:** This input port accepts the level for channel5 of the RC remote control (values 0-1000 are allowed, center:500)
- **channel6 [integer]:** This input port accepts the level for channel6 of the RC remote control (values 0-1000 are allowed, center:500)
- **channel7 [integer]:** This input port accepts the level for channel7 of the RC remote control (values 0-1000 are allowed, center:500)
- **channel8 [integer]:** This input port accepts the level for channel8 of the RC remote control (values 0-1000 are allowed, center:500)

currently, this plugin does not have any properties

### **TextArea**

# **Component Type: Actuator (Subcategory: Graphical User Interface)**

The TextArea is a GUI component which displays (optionally usereditable) text information. It features multiple rows and a scrollbar, and event for appending string text, deleting characters or sending the text content

GUI Designer (Ctrl-G) Deployment (Ctrl-D) =	Properties (Ctrl-P)		<b>•</b> ‡	
	Component: TextArea.1			
	Search			
	<ul> <li>Internal Properties</li> </ul>			
	Component Class	actuator	*	
TextArea.1	Component Name	TextArea.1		
setText	Component Type	asterics.TextArea		
appendText	Component Descriptio	GUI component, which displays	ext.	
	<ul> <li>Properties</li> </ul>			
	caption	Text Display		
	default			
	editable			
	fontSize	14	0	
	textColor	black	•	
	backgroundColor	white	-	
·				
	🕈 Pro 🔰 Eve 🔰	Eve 📽 Out 📽 Inp		

TextArea plugin
#### **Input Port Description**

- **setText [string]:** The text which will be displayed. It replaces the prior content of the Text Area.
- **appendText [string]:** This string input will be appended to te current content of the Text Area.

#### **Output Port Description**

• **text [string]:** The current content of the text area. Sent on incoming events at the "send" or "sendAndClear" event listener ports

#### **Event Listener Description**

- **delete:** Removes the last charater from the text area content.
- clear: Clears the text area content.
- send: Sends the text area content to te output port.
- **sendAndClear:** Sends the text area content to te output port and clears the text area.

## **Event Trigger Description**

• **clicked:** The event is trigger when the user click on the component.

- caption [string]: Caption of the component.
- **default [string]:** Default content of the text area.
- editable [bolean]: Defines if the text area can be edited by the user.
- fontSize [integer]: The font size in pixels.
- textColor [integer]: Defines color of the text.
- **backgroundColor** [integer]: Defines background color.
- **displayGUI [boolean]:** If selected, the GUI of this component will be displayed if not, the GUI will be hidden and disabled.

# **Text Display**

# **Component Type: Actuator (Subcategory: Graphical User Interface)**

The Text Display is the GUI component, which displays the text information.



TextDisplay plugin

### **Input Port Description**

• input [string]: The text which will be displayed. This input port supports synchronization

## **Event Listener Description**

• **clear:** Removes the text from the component.

## **Event Trigger Description**

• **clicked:** The event is trigger when the user click on the component.

- caption [string]: Caption of the component.
- default [string]: The text, which is in the box after start.
- **textPosition [integer]:** Position of the text in the component: left, center or right.
- textColor [integer]: Defines color of the text.
- **backgroundColor** [integer]: Defines background color.
- **displayGUI [boolean]:** If selected, the GUI of this component will be displayed if not, the GUI will be hidden and disabled.

## ToneGenerator

#### **Component Type: Actuator (Subcategory: Audio and Voice)**

This plugin generates a stereo sound. The frequency of the tone for the left and right channel can be set individually.



ToneGenerator plugin

## Requirements

This software component requires an audio output device to percept the sound

#### **Input port Description**

- **frequencyLeft[double]:** specifies the frequency of the tone on the left channel
- **frequencyRight[double]:** specifies the frequency of the tone on the right channel

• **sampleRate[integer]:** sets the sampleRate of the tonegenerator

### **Eventlistener Description**

- start: starts the playback of the tones
- stop: stops the playback of the tones

## **Wavefile Player**

#### **Component Type: Actuator (Subcategory: Audio and Voice)**

This plugin plays a wave file specified in the properties on the platforms sound output.

GUI Designer (Ctrl-G) Deployment (Ctrl-D) =	Properties (Ctrl-P) 🔹 🖡
WavefilePlayer.1 wavefileName	componentType: WavefilePlayer.1   Image: Search Search   Internal Properties Component Descriptic   Component Descriptic Simple SAPI component. It uses d   Component Name WavefilePlayer.1   ComponentType actuator   Properties filename
< >	Tinput Ports / Event Listener
WavefilePlayer plugin	

#### **Input port Description**

• **wavefileName[string]:** this port will change the property filename to the string on the input. Thus it is possible to have other plugins change the file to be played.

#### **Event Listener Description**

- **Start:** An incoming event at this port initiates the playback of the wave file indicated in the property *filename*.
- **Stop:** An incoming event at this port stops the playback of the wave file indicated in the property *filename*.
- VolumeUp: An incoming event increases the volume of the current playback.
- **VolumeDown:** An incoming event decreases the volume of the current playback.

• filename[string]: The value of this property specifies the file name of the wave file to be played. Supports value suggestions from ARE (dynamic property)

# **CSV** Writer

#### **Component Type: Actuator (Subcategory: File System)**

This plugin writes incoming strings into an .csv file.

The data gets live recorded, so it can be viewed simultaneously in an oscilloscope.



### **Event Listener Description**

- **StartWriting:** Creates a new file to save the data.
- **StopWriting:** Stops a saving process.

#### **Input port Description**

• **Input[string]:** The string contains the data. Example of one string: "data1;data2;data3". To get data values and separators into this string, the StringExpander plugin can be used. After each string, a line separator (newline) is appended.

- FileName[string]: Specifies the name of the file (without extension), in which the data will be recorded. Current date and time information are added to the filename.
- **FilePath[string]:** Defines the Path were the File shoud be saved (relative from the ARE folder). If not existing, the path will be created.