



# Raspberry PI with Standard CODESYS V3

## First Steps

Version: 2.0

Template: templ\_tecdoc\_en\_V1.0.docx

File name: RaspberryPI\_CodesysV3\_FirstSteps\_EN.doc

# CONTENT

	Page	
<b>1</b>	<b>Product description</b>	<b>3</b>
<b>2</b>	<b>Installation, configuration and licensing</b>	<b>4</b>
2.1	Preparations	4
2.2	Installation	5
2.3	Licensing via the CODESYS Development System	5
2.4	License backup	6
2.5	License Reactivation	6
<b>3</b>	<b>Example applications</b>	<b>7</b>
3.1	Webvisu.project	7
3.2	Camera.project	7
3.3	CameraStream.project	7
3.4	GPIO.project	8
3.5	PiFace.project	9
3.6	PiFaceIoDrv.project	9
3.7	PiFaceDisplayAndControl.project	9
3.8	I2CExample.project	10
3.9	MCP3008_Temperature.project	10
3.10	MCP23S17.project	11
3.11	OneWire.project	11
3.12	SoftMotion Servo Example	12
3.13	EtherCAT.project	13
3.14	OPCUA.project	13
<b>4</b>	<b>Connecting additional devices via I<sup>2</sup>C and SPI</b>	<b>14</b>
<b>5</b>	<b>Screenshots</b>	<b>16</b>

## 1 Product description

This product contains a CODESYS Control application for Raspberry Pi (see <http://www.raspberrypi.org/>) as well as driver support for the extension modules Raspberry PiFace Digital, Raspberry Pi Camera and several devices/breakouts with I<sup>2</sup>C communication interface.

The recommended operating system 'Raspian' can be downloaded here:

Common download page: <https://www.raspberrypi.org/downloads/>

Latest version: [https://downloads.raspberrypi.org/raspbian\\_latest](https://downloads.raspberrypi.org/raspbian_latest)

This product can be installed via CODESYS Update Manager (CODESYS PlugIn RPIUpdate) on a Linux distribution (Raspbian). After starting Raspberry Pi without a valid license, CODESYS Control runs for two hours without functional limitations and shuts down automatically.

The runtime system does not have real-time behavior. Its Jitter depends on many factors, especially on parallel executed Linux applications, and is ideally about 50µs with maximum values of 400µs. This product supports the following functionalities:

- CODESYS EtherCAT Master
- CODESYS Profinet Master
- CODESYS Modbus TCP Master / Slave
- CODESYS Modbus RTU Master / Slave (serial interface must be supported by and installed in the OS)
- CODESYS WebVisu
- CODESYS SoftMotion CNC
- CODESYS OPC/UA Server
- CANopen via EL6751 Gateway
- CODESYS EtherNet/IP Scanner
- CODESYS EtherNet/IP Adapter

This product consists of:

- Debian package with CODESYS Control for Raspberry Pi
- CODESYS Plugin to install and update the package on a Raspberry Pi
- CODESYS device description files for Raspberry Pi, Raspberry PiFace Digital, Raspberry PiFace Control&Display, Raspberry Pi Camera, several devices/breakouts with I<sup>2</sup>C (SRF02, Adafruit PWM, MPU6050, MPU9150, AK8975), SPI (MCP3008, MCP23S17) or 1-wire (DS18B20) communication interface

This product offers amongst other things the possibility to plug and control additional devices via SPI, I<sup>2</sup>C or 1-wire.

This product is provided for testing and training purposes. Usage for industrial purposes is not recommended.



- e. Optional: SPI  
dtparam=spi=on
  - f. Optional: 1-wire  
dtoverlay=w1-gpio-pullup,pullup=1
3. Optional: Prepare camera
    - a. un raspi-config (see 1) and execute "Enable Camera".
    - b. Connect via SSH (e.g. with Putty) to your device (standard user "pi", password "raspberry") and execute the following commands in your shell in order to install the c:
 

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo rpi-update

git clone https://github.com/silvanmelchior/RPi_Cam_Web_Interface.git
cd RPi_Cam_Web_Interface
chmod u+x RPi_Cam_Web_Interface_Installer.sh
./RPi_Cam_Web_Interface_Installer.sh install
```
    - c. Start a web browser and connect to your device at `http://<network address>`  
In case the installation was successful, the UI of RPi Cam Control should open and you can configure your camera.

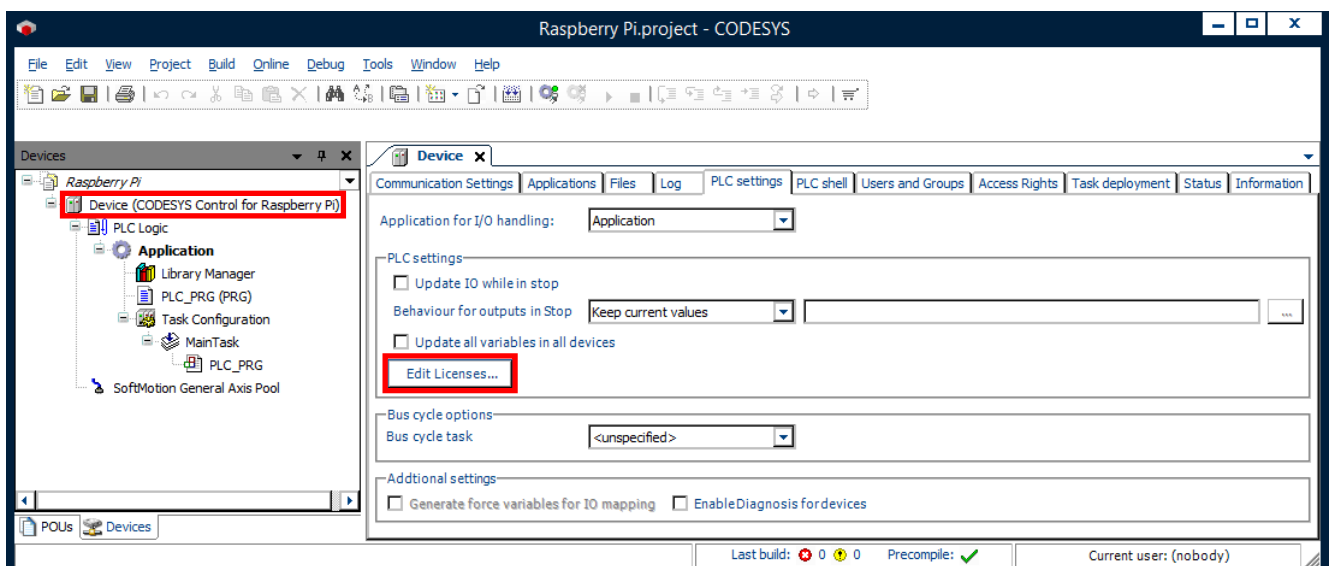
## 2.2 Installation

1. In CODESYS execute the command „Update RaspberryPi (menu Tools).
  - a. Select the desired version
  - b. nter the correct login data (default: `pi/raspberry`)
  - c. Select the IP address of your device
  - d. Click OK and check if the message window prompts that the update/installation has been successfully executed
2. After a restart the device will be ready to be used with CODESYS.

## 2.3 Licensing via the CODESYS Development System

Requirements: PC with CODESYS Development System, internet access and connected Raspberry Pi.

Licensing is done via PC / notebook with the CODESYS Development System and the connected Raspberry Pi. The license entries are edited via double-click on the device under "PLC settings" / "Edit licenses...".



The license activation is done under “Install licenses” / “Activate license” by entering the ticket number and transfer of the license to the CODESYS Software Key (Softcontainer).

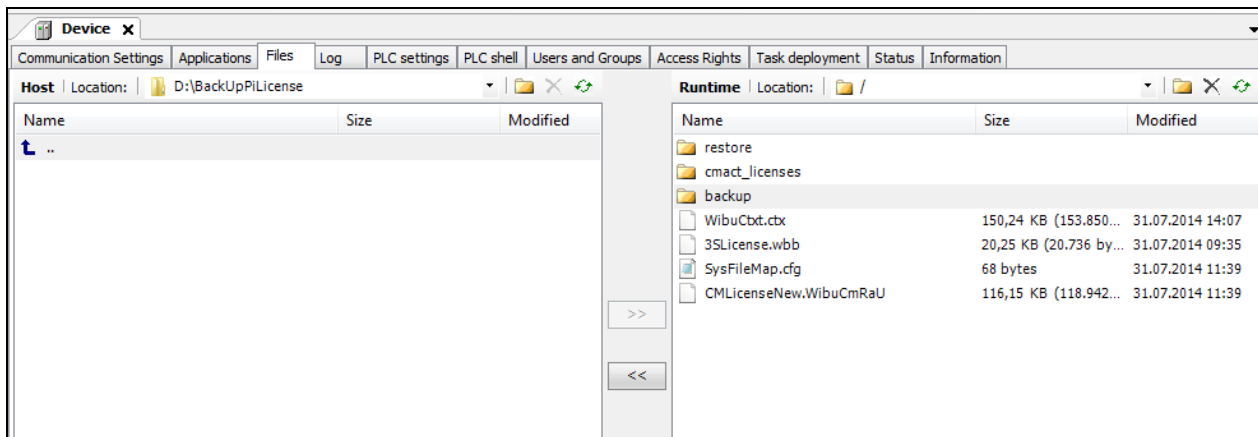
Note: On activation the license is bound to a single Raspberry Pi and can only be reactivated on the same device!

## 2.4 License backup

Different processes (e.g. loss of power) can lead to a corruption of the file system of the Raspberry Pi. To back up the license the following proceeding is recommended:

1. Activation of the license (as described above)
2. Reboot of the Raspberry Pi
3. Backup of the license file on an external storage device

To safe the license file move to the folder “backup” on the Raspberry Pi (accessible in CODESYS via double-click on the device under the tab “Files”).



Store the content of the folder („3SLicenseInfo.tar“) to an external storage device..

## 2.5 License Reactivation

To reactivate the license file has to be copied to the folder “restore”. Subsequently the system has to be rebooted.

### 3 Example applications

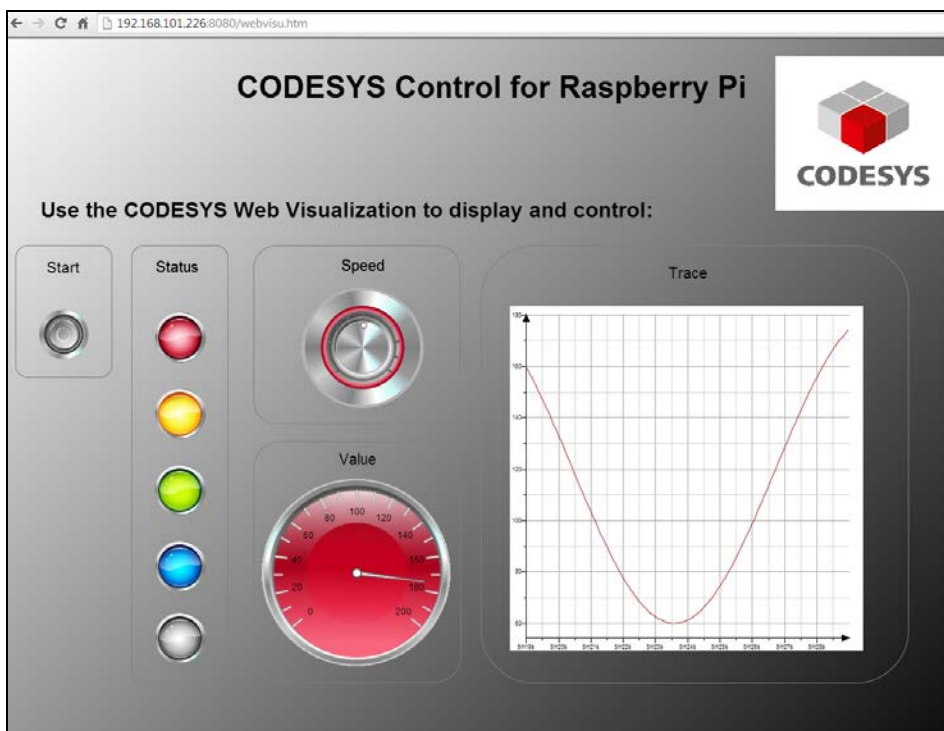
In the installation path (that you have noted down during installation of the package) you will find the following example projects:

#### 3.1 Webvisu.project

This example shows how CODESYS web visualization can be used. Open the project and download it onto your Raspberry Pi by doubleclicking on the node "Device" in the device tree (left hand side). Then press "Scan network" in the communication dialog tab and select your device that should now appear under the name "RaspberryPi", if the device is in the same network with your programming PC. Select it and run "log in" from the menu "Online". Then start the application with F5.

Start an internet browser (possibly also on your smartphone) and connect to <Network Address>:8080/webvisu.htm.

In your browser you will see the visualization that has been designed in the project:



#### 3.2 Camera.project

(Precondition: Raspberry Pi Camera is connected and installed, see 2.1.)

This project shows how you can use Raspberry Pi Camera (extension hardware) to take a picture and save it as file. Please note that on some hardware models the "RPi Cam Control application" must be deactivated. Use your browser to connect to the configuration page <Netzwerk-Adresse>:8080/webvisu.htm > and execute "stop camera".

Download the application to your controller, start the program and set the variable xTakePicture to TRUE. The camera will take a picture and store it in the local file system with the name „Picture.jpg“.

You can copy this file onto your programming PC with the help of the file dialog (doubleclick "Device", tab "Files", Button "Update" on the right side etc.).

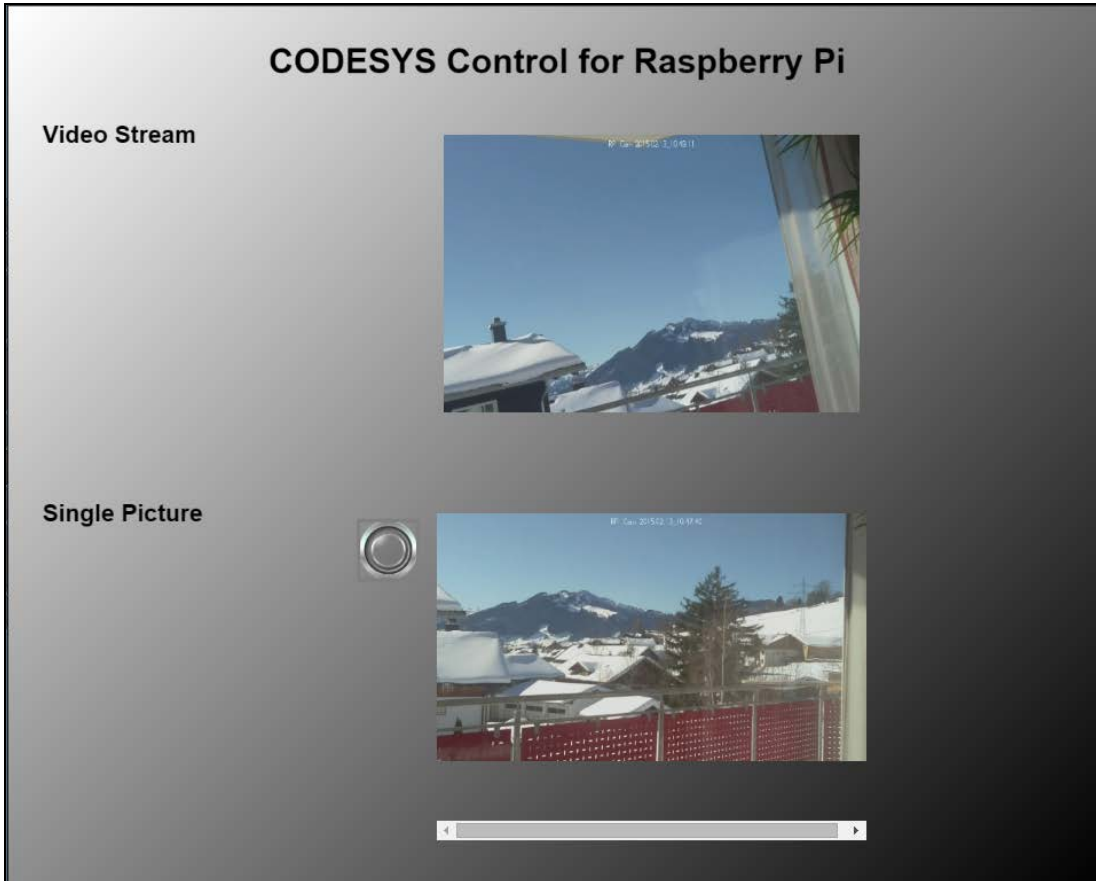
#### 3.3 CameraStream.project

(Precondition: Raspberry Pi Camera ist angeschlossen und installiert, see 2.1.)

This project shows how a camera stream or single camera picture can be included in your web visualization.

Download the application to your controller and start the program. Start an internet browser and connect to <Network Address>:8080\webvisu.htm. There you see the live stream of your cam in the upper window part and the last taken single picture in the lower part; you can update the latter by pressing the button next to it.

Please note that, depending on the version of the „RPi\_Cam\_Web\_Interface“ and the contained Apache installation, the default path changes from „/var/www/“ zu „/var/www/html/“. If applicable, this path must be adapted in PLC\_PRG in line 3.



### 3.4 GPIO.project

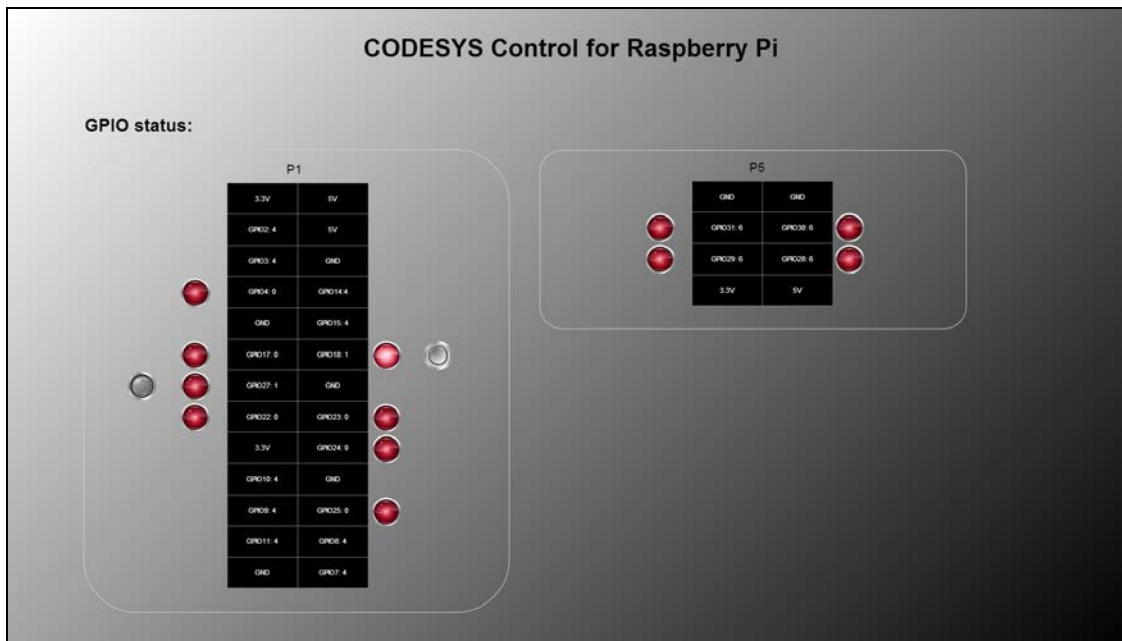
This project shows you can use free GPIOs. In the configuration of the GPIO device in the device tree the function of each GPIO can be defined:

GPIOs Configuration						
Parameter	Type	Value	Default Value	Unit	Description	
GPIO4	Enumeration of BYTE	not used	not used		configuration of GPIO4	
GPIO17	Enumeration of BYTE	not used	not used		configuration of GPIO17	
GPIO18	Enumeration of BYTE	Output	not used		configuration of GPIO18	
GPIO22	Enumeration of BYTE	not used	not used		configuration of GPIO22	
GPIO23	Enumeration of BYTE	not used	not used		configuration of GPIO23	
GPIO24	Enumeration of BYTE	not used	not used		configuration of GPIO24	
GPIO25	Enumeration of BYTE	not used	not used		configuration of GPIO25	
GPIO27	Enumeration of BYTE	not used	not used		configuration of GPIO27	
GPIO28	Enumeration of BYTE	not used	not used		configuration of GPIO28	
GPIO29	Enumeration of BYTE	not used	not used		configuration of GPIO29	
GPIO30	Enumeration of BYTE	not used	not used		configuration of GPIO30	
GPIO31	Enumeration of BYTE	not used	not used		configuration of GPIO31	



The inputs and outputs are available as DWORD in the tab „GPIOs I/O Mapping“. Bit <X> of the DWORD correlates with GPIO <X>.

In this example GPIO18 is used as output and blinks, controlled by a timer FB in PLC\_PRG. A visualization screen displays the input values of the GPIOs and allows setting outputs.



For using the additional GPIOs of the hardware variant Raspberry Pi B+ we provide you with a separate GPIO device, that can be plugged into the slot "GPIOs" in the device tree.

Please note that depending on other drivers that might be loaded, some GPIOs can be blocked and be not available for general purpose.

### 3.5 PiFace.project

(Precondition: Raspberry PiFace Digital is connected)

This example shows the usage of Raspberry PiFace Digital (8 digital inputs and outputs).

Open the project, download it to the controller and start it. The simple application in PLC\_PRG controls the relay output K0 depending on the key button S1 (on-switching of K0 is delayed for 1s) and the relay output K1 depending on button S2 (off-switching is delayed for 500ms).

Please note that this driver allows connecting more than one PiFace devices (hardware address is set with the jumpers JP1, JP2) by setting the corresponding parameter in the PiFace device in the device tree.

The library SPI\_PiFace that operates as driver is provided as source code and can be seen as example how to communicate to other devices via SPI. The communication bases are on the library RaspberryPiPeripherals, for which reference documentation is provided (see online help (F1) -> Libraries).

### 3.6 PiFaceIoDrv.project

(Precondition: Raspberry PiFace Digital is connected)

This example is similar to the one before. Instead of providing the I/O data as inputs of an automatically instanced function block, the example implements the data exchange as standard PLC IO driver via a process image, how it is typically done on PLCs.

The driver library IoDrvPiFace.library is provided as source code.

### 3.7 PiFaceDisplayAndControl.project

(Precondition: Raspberry PiFace Control&Display is connected)

This example shows how the two-lined textual display and the input buttons of this device can be used to program a parameter editor.

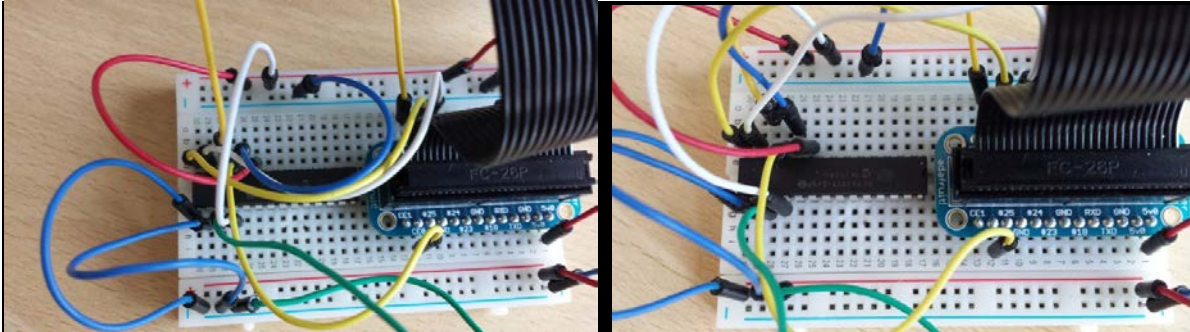
Note that PiFace Control&Display communicates via SPI port 1 ('/dev/spidev0.1') that needs to be set in the SPI master instance in the device tree. By adding a PiFace Control&Display device, a FB instance is automatically



### 3.10 MCP23S17.project

(Precondition: special hardware is connected via SPI)

This example shows how a port expander chip (MCP23S17) can be used. The hardware should be connected to the Pi in the following way:



In the device settings you can specify, the direction (input/output) of the GPIOs and for inputs you can activate a pull-up resistor. Depending on the wiring of the hardware address pins, the address must be set in the configuration screen.

Parameter	Typ	Wert	Standardwert	Einheit	Beschreibung
Hardware address	USINT	0	0		Hardware address of the device (0..7)
Inputs of GPIOA	BYTE	0	0		if bit is set, GPIO is setup as input
Inputs of GPIOB	BYTE	2#11111111	2#11111111		if bit is set, GPIO is setup as input
Pullup active for inputs A	BYTE	2#00000000	2#00000000		if bit is set, pull-up resistor for corresponding input is active
Pullup active for inputs B	BYTE	2#00000000	2#00000000		if bit is set, pull-up resistor for corresponding input is active

### 3.11 OneWire.project

(Precondition: 1-wire temperature sensor is connected)

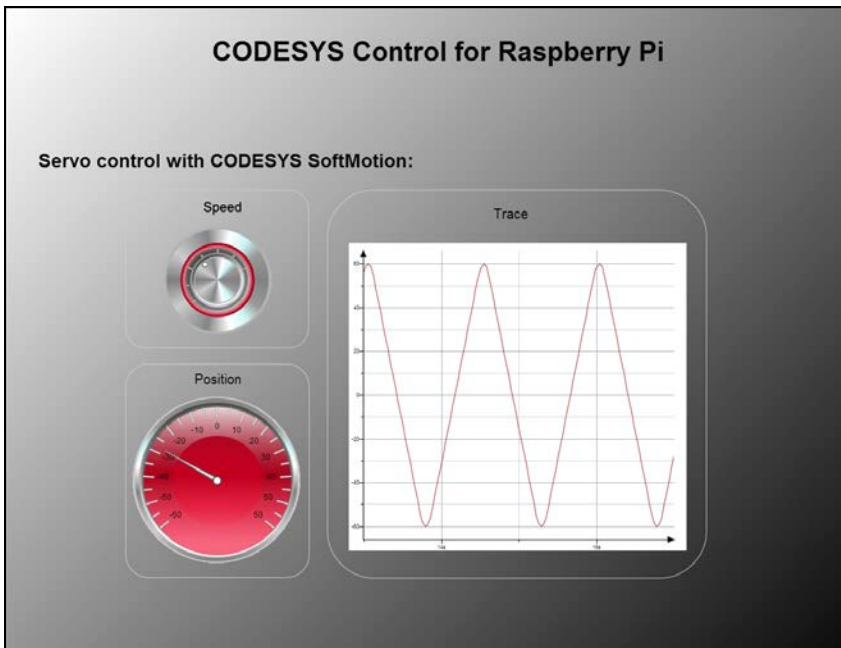
This example shows Raspberry Pi scanning the devices that are connected via 1-wire and reading data from a temperature sensor DS18B20. The 1-wire data line is connected to GPIO4.

Each 1-wire device has its unique ID that is used to address it. Hence one needs to set the ID in the configuration screen of every device before using it.

This example helps to find out the ID of the connected devices. It shows two functions:

1. Display the result of the scan in a visualization screen. This helps to find out the IDs of the connected devices.
2. Show the temperature of a DS18B20 temperature sensor, if it is configured correctly.





### 3.13 EtherCAT.project

(Precondition: The following devices are connected to the Pi's LAN adapter: Beckhoff EK1100 mit Beckhoff EL2008)

This example switches the eight available outputs on the connected hardware implementing an EtherCAT master.

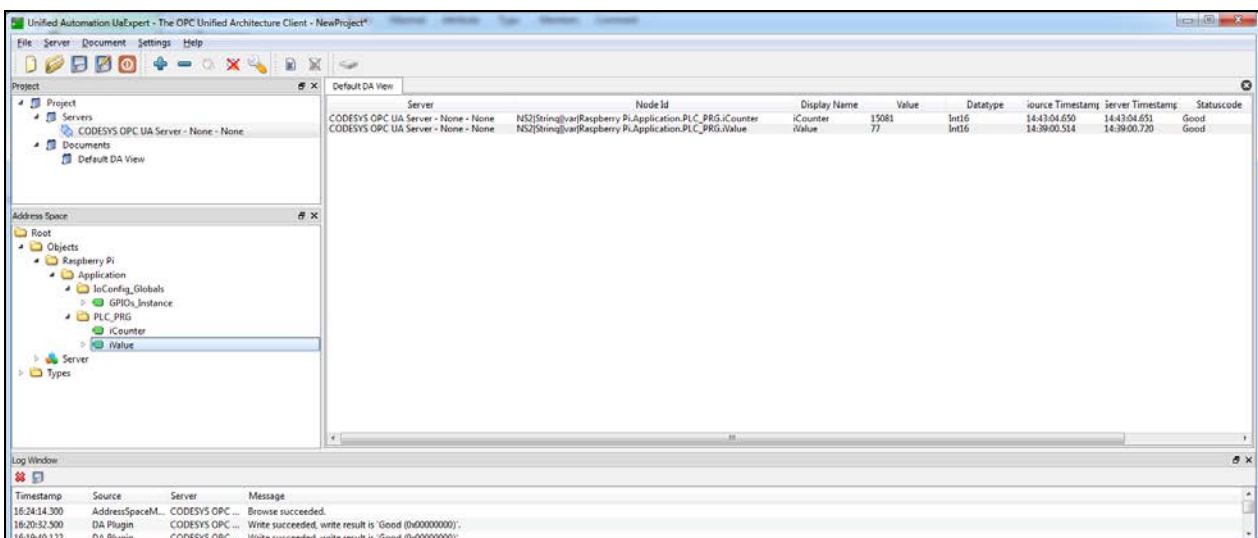
Open the project, download and start it. The outputs of the clamp will change continuously.

Please note that the LAN port of your Raspberry Pi is then blocked for EtherCAT and hence cannot be used as communication and programming interface. It is recommended to use a USB WLAN adapter (e.g. Edimax N150) in this case..

### 3.14 OPCUA.project

This simply example shows how to publish variables of your application and access them with an OPC/UA client. In the object "Symbol Configuration" the two integer variables of PLC\_PRG are marked for external access-

With an appropriate OPC/UA client (e.g. Unified Automation UaExpert) you can now connect to your RaspberryPi via the url „opc.tcp://<Network address>:4840“, browse its objects and monitor or write the variables' values:



## 4 Connecting additional devices via I<sup>2</sup>C and SPI

There are three typical ways to connect and run additional hardware via the i<sup>2</sup>c, spi or 1-wire interface:

1. Program a function block (FB) and declare and call an instance of it in your application
2. Program a function block (FB) and a device description file for a specific hardware that allows to insert and configure this device in the CODESYS device tree
3. Program an IO driver

1 represents the easiest and fastest way, 2 describes a better integrated, more user-friendly possibility and 3 implements the method, which is typical for PLCs, copying the I/O data into or from the process image, which allows mapping of data to new or existing variables.

For better understanding the three options, please compare the example projects PiFace\_FB (implements option 1), PiFace (option 2) und PiFaceloDrv (option 3). All three of them support the PiFace extension board.

Option 1 is explained with this example and with the help of the Library RaspberryPiPeripherals (see Library Manager) and the contained reference documentation of the function block spiMaster (i2cMaster). Other hardware can be accessed and controlled in an analogous manner.

Option 3 requires detailed knowledge of CODESYS and the IO driver concept and is out of the scope of this document.

Option 2 is explained in the following paragraph:

To support a new device, you should generate a new device description (\*.devdesc.xml) and a new library (\*.library). As templates you can use one of the example device drivers (s. file path of example projects) that uses the same communication method (spi, i<sup>2</sup>c). Execute the following steps:

### (A) Device description

- Generate a copy of an existing device description that is installed to the installation path and rename it to <myDeviceName>.devdesc.xml.

- Modify the ID of this device. Set the high word to FFFF, the low word set locally unique:<sup>1</sup>

```
<Device hideInCatalogue="false">
  <DeviceIdentification>
    <Type>501</Type>
    <Id>FFFF 4711</Id>
    <Version>1.0.0.0</Version>
  </DeviceIdentification>
```

- Adapt the device information:

```
<DeviceInfo>
  <Name name="local:ModelName">MCP3008</Name>
  <Description name="local:DeviceDescription">MCP3008</Description>
  <Vendor name="local:VendorName">3S - Smart Software Solutions GmbH</Vendor>
  <OrderNumber>-</OrderNumber>
```

- Enter the name, vendor and version of the library you are going to generate later:

```
<RequiredLib libname="Raspberry SPI MCP3008" vendor="3S - Smart Software Solutions GmbH" version="1.0.0.0" identifier="deviceLib">
```

- Set the name of the FB in the library, which will do the communication and represent the device:

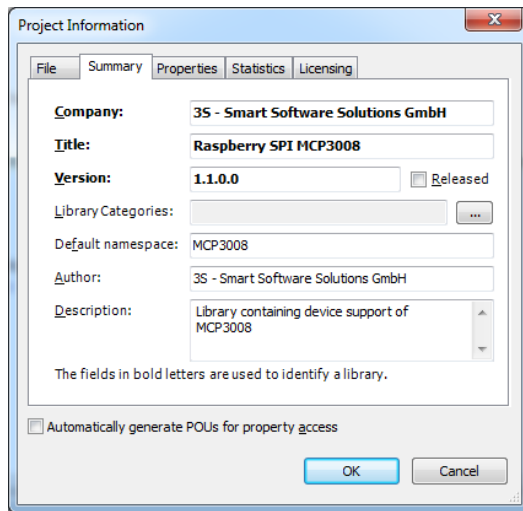
```
<FBInstance basename="$(DeviceName)" fbname="MCP3008">
  <Initialize methodName="Initialize" />
  <CyclicCall methodname="AfterReadInputs" task="#buscycletask" whentocall="afterReadInputs" />
  <CyclicCall methodname="BeforeWriteOutputs" task="#buscycletask" whentocall="beforeWriteOutputs" />
</FBInstance>
```

- Install the device description in your device repository in CODESYS. From now on you will be able to add your new device below „I<sup>2</sup>C devices“ or „SPI devices“.

<sup>1</sup> Please note, that if you want to distribute your new device support you need a registered customer ID at 3S - Smart Software Solutions GmbH

**(B) Library**

- Generate a copy of the example library. Rename it to <myDeviceName>.library.
- Open the library in CODESYS and adapt the project information:



Company, title and version must match with the device description.

- Rename the FB in the library. The new name must match with the one set in the device description.
- You can install a state machine in the body of the FB. `iState=0` represents the init state, `_iState=10` normal operation, `_iState=1000` a mistake. Intermediate steps may be added if needed.
- In the method `AfterReadInputs` you should read the inputs from your device. For the communication use the standard methods of the base FBs `i2c` (`read8`, `write8`, etc.) or `spi` (`transfer`)
- In the method `BeforeWriteOutputs` you should write the outputs to your device
- Save the library and install it in your library repository.

**(C) Utilization**

- You can now generate a project and add the new device under the correct communication interface. This will automatically generate a FB instance that you can use in your application.

## 5 Screenshots

