CODESYS Control for Raspberry Pi SL

1 General information

Order number: 603001	**Supplier information** 3S-Smart Software Solutions GmbH Memminger Straße 151 87439 Kempten Germany Support: Tel: +49 831 54031 66 support@codesys.com
Version: 2.2.0.0	
Short description CODESYS Control for Raspberry Pi SL is an adapted CODESYS Control runtime system for Raspberry Pi.	

2 Requirements and restrictions

Programming system	CODESYS Development System Version 3.5.7.0 or higher
Target system	CODESYS Control Version 3.5.7.0 (this is part of this product)
Supported Platforms / Devices	Raspberry Pi (s. http://www.raspberrypi.org/) - Pi Model B - Pi Model B+ - Pi2 Model B
Additional requirements	SD-card (minimum 4GB)
Restrictions	-

3 Price

35 EUR plus 19% VAT

This product is subject to license conditions. The product can be licensed on a software base (the license container is part of the CODESYS Control for Raspberry Pi SL) or via the CODESYS Runtime Key (not part of the product range).

After starting the Raspberry Pi without a valid license CODESYS Control runs for two hours without functional limitations and shuts down automatically (demo).

4 Optional accessory, purchasable in the CODESYS Store

CODESYS Runtime Key, compact Order number: 601003

5 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 I²C communication interface. 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 I²C (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, I²C or 1-wire.

This product is provided for testing and training purpose and may not be used in the field.

6 Technical description

Start CODESYS and the PackageManager, which you can find in the menu “Tools”. Install the package file that you have downloaded from CODESYS Store.

For the installation of CODESYS Control onto your Raspberry Pi you will need an operative Raspbian OS. If this is not yet installed on your device, please execute the steps described in the section *Preparations*. Else continue with *Installation*.

Preparations

Follow the steps described on https://www.raspberrypi.org/downloads/ in order to install an OS onto the SD card of your device. We recommend the use of Raspbian.

Reboot Raspberry Pi with this SD card afterwards. In order to obtain a valid IP address, the device should be connected to a network with a DHCP server.

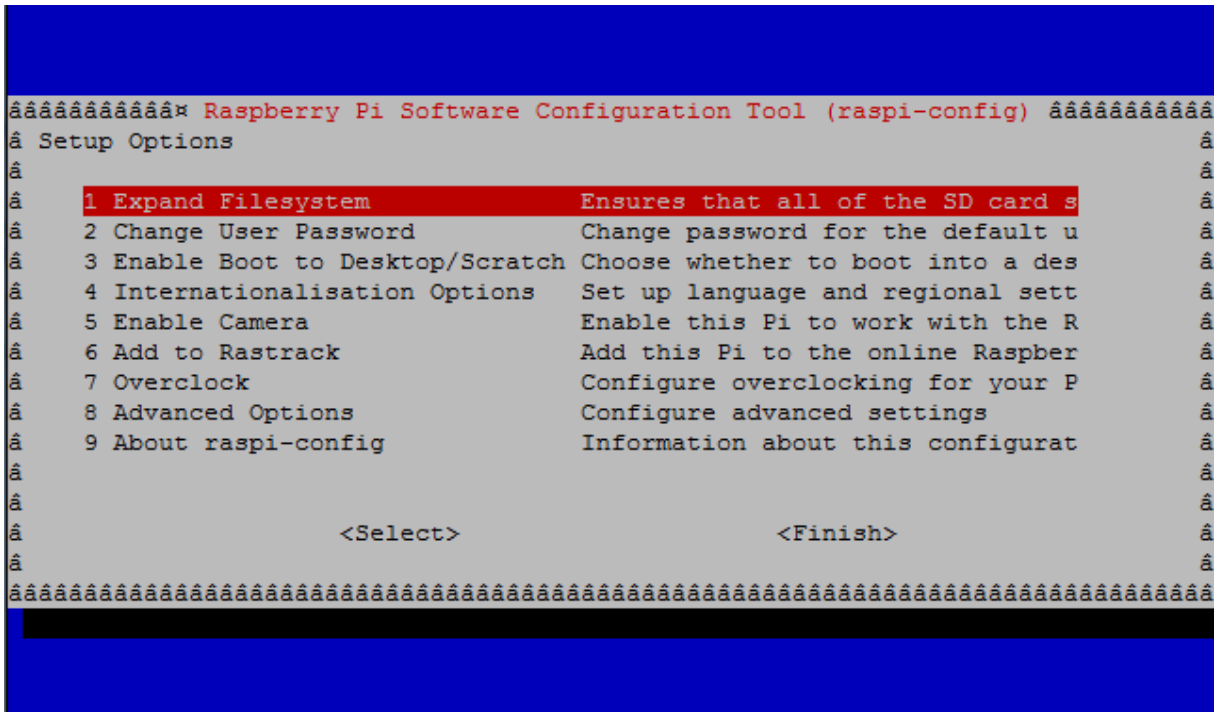
To find out the network address, you may either plug a keyboard, mouse and a monitor to your device, log in (user: `pi`, password: `raspberry`), open a console and issue the command `ifconfig`. Or you can ping the device from the windows console with its network name (default: `RaspberryPi`) and obtain the IP address. Alternatively, you can start CODESYS and execute the command “Update Raspberry Pi” in the menu Tools, where you execute “Scan” and find the list of IP addresses of Raspberry devices in your network.

With the IP address you can also log in via SSH using tools like Putty (user: `pi`, password: `raspberry`).

1. Basic configuration

From a console run `sudo raspi-config` and execute the following action:

a) Expand Filesystem



b) Optional: Rename the device (8 Advanced Options)

c) Optional: Activate the camera (5 Enable Camera)

d) Exit and restart the device

2. Activate the interfaces i²c, spi, 1-wire

   a. Connect via SSH (e.g. using Putty) with your device

   b. Edit the file /boot/config.txt e.g. using
   `sudo nano /boot/config.txt`

   c. Make sure that the following lines are in this file (and not deactivated with #):

      i. Optional: i²c
      `dtparam=i2c_arm=on`

      ii. Optional: SPI
      `dtparam=spi=on`

      iii. Optional: 1-wire
      `dtoverlay=w1-gpio-pullup,pullup=1`

3. Option: prepare camera

   a. Run 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.

Installation

4. In CODESYS execute the command „Update RaspberryPi (menu Tools).

   a. Select the desired version

   b. Enter 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.

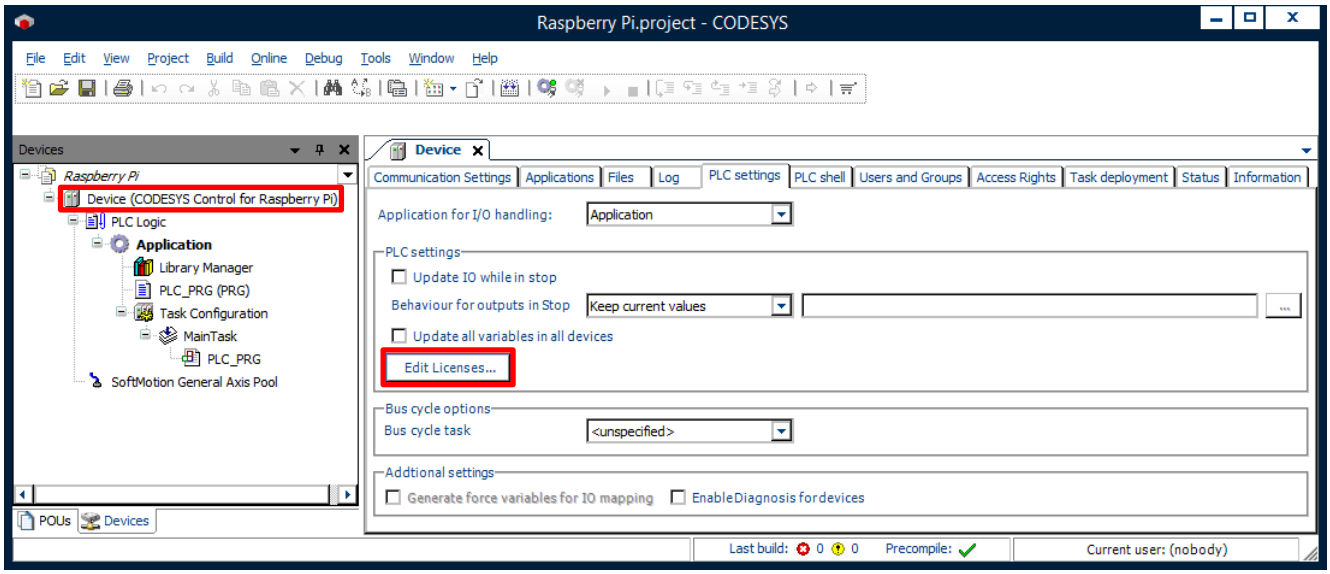
5. After a restart the device will be ready to be used with CODESYS.

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 licence entries are edited via double-click on the device under “PLC settings” / “Edit licenses…”.



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

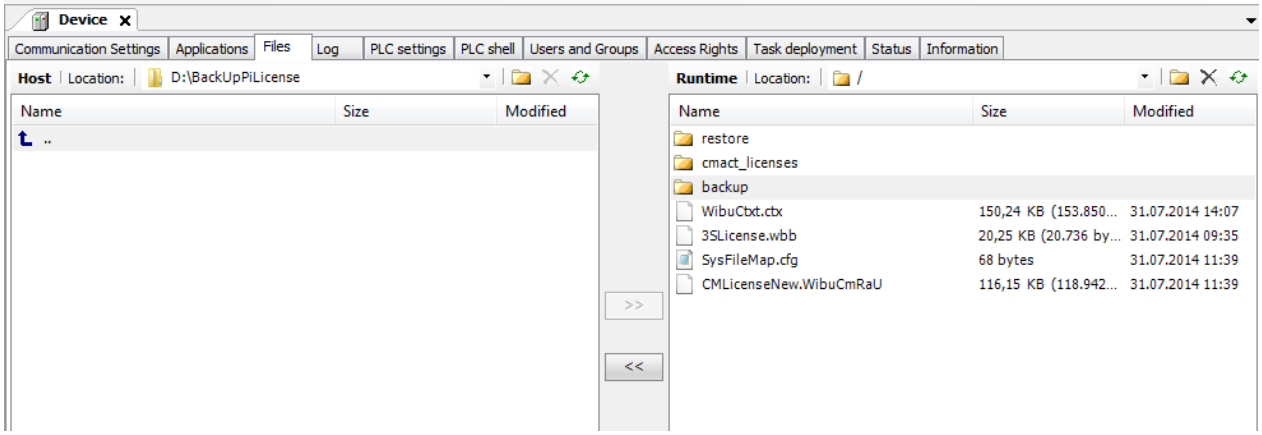
Note: On activation the licence is bound to a single Raspberry Pi and can only be reactivated on the same device!

Backup of the licence

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

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

To safe the licence 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.

Licence Reactivation

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

Example applications

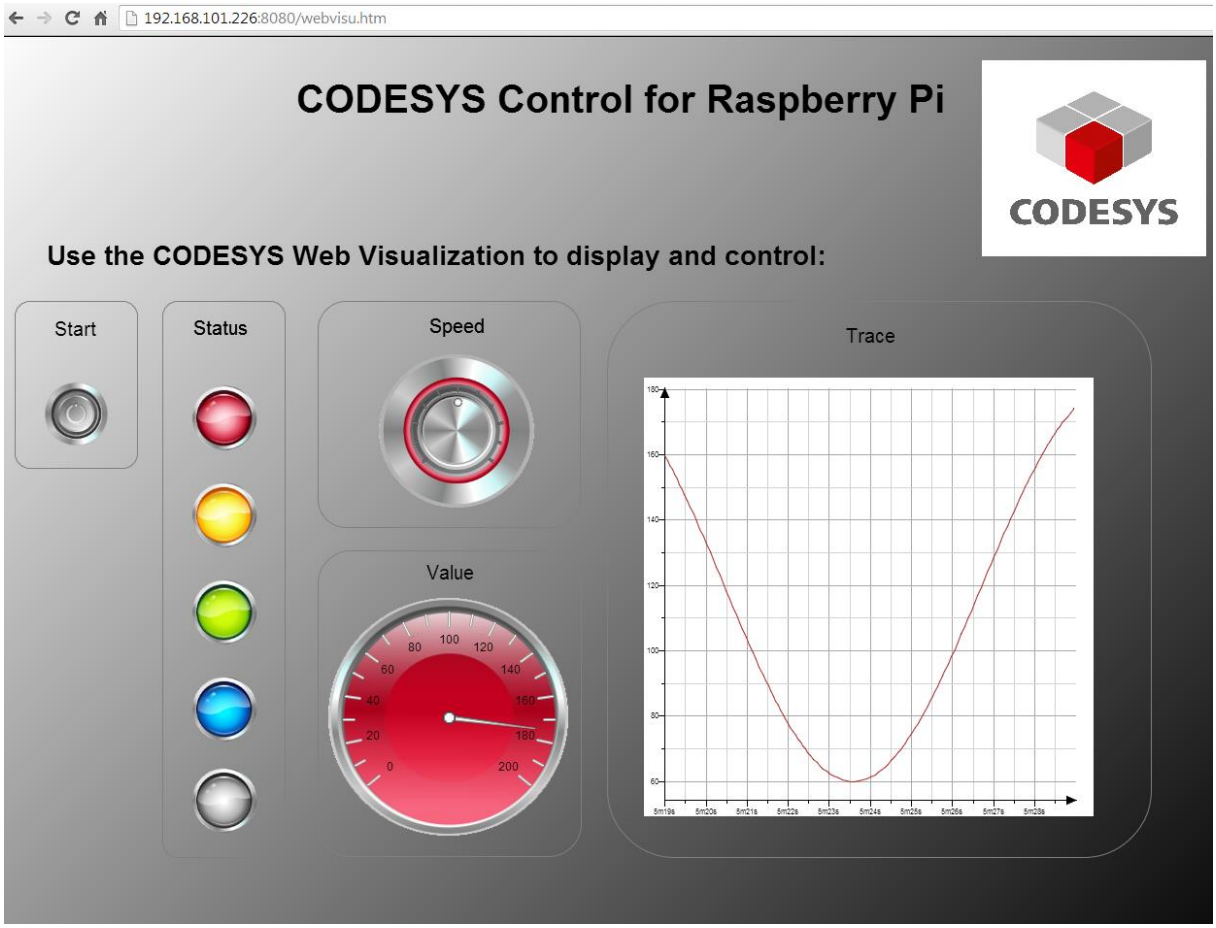
In the installation path (that you have noted down during installation of the package) you will find the following example projects:

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:



2. **Camera.project (precondition: Raspberry Pi Camera is connected and installed, see 6.3)**

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 (http://<Netzwerk-Adresse>) and execute “stop camera”.

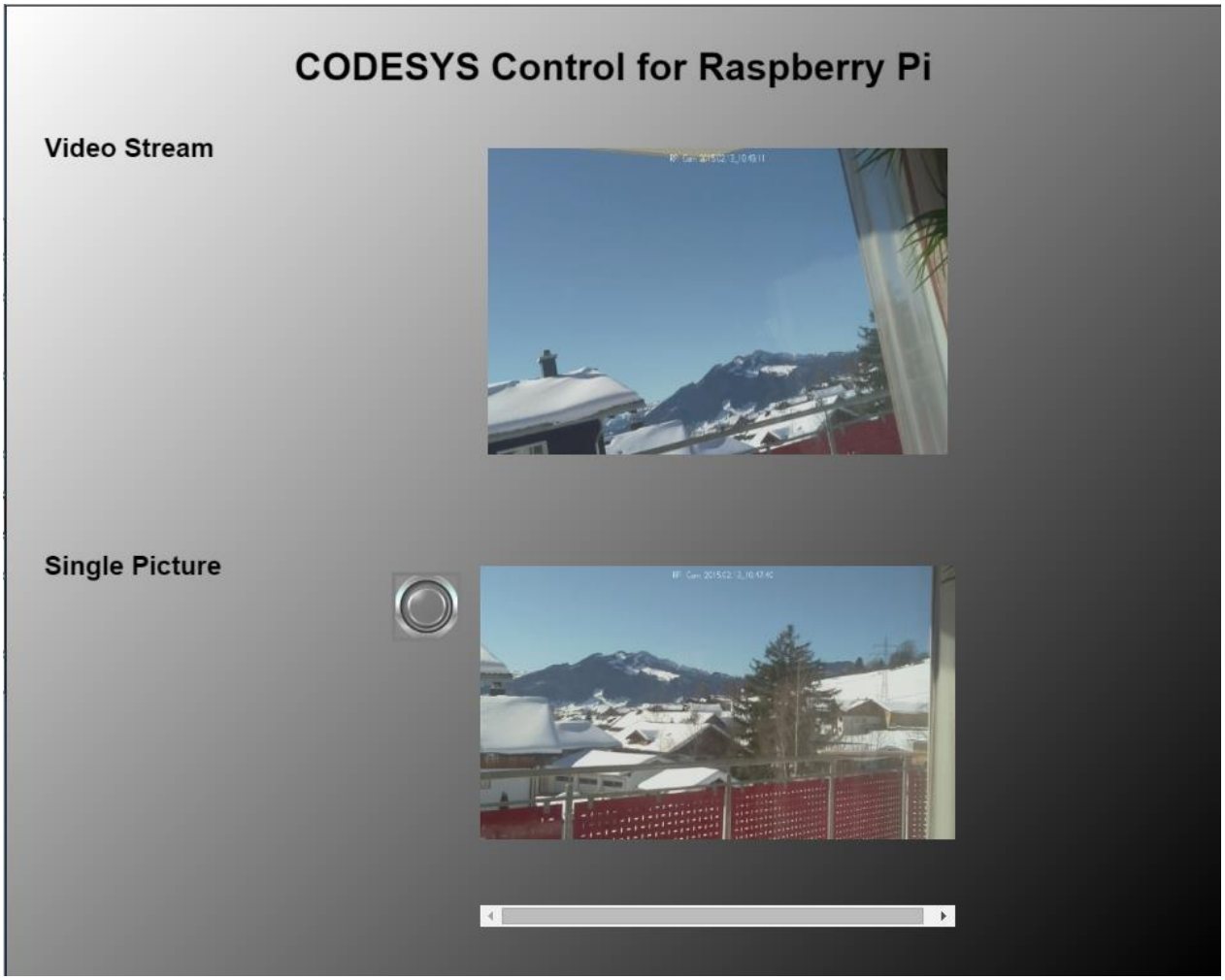
Download the application to your controller, start the program and set the variable `xTakePicture` to `TRUE`. The camera will take a picture and store it in the local file system with the name „Picture.jpg“.

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. **CameraStream.project (precondition: Raspberry Pi Camera is connected and installed, see 6.3)**

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.



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:

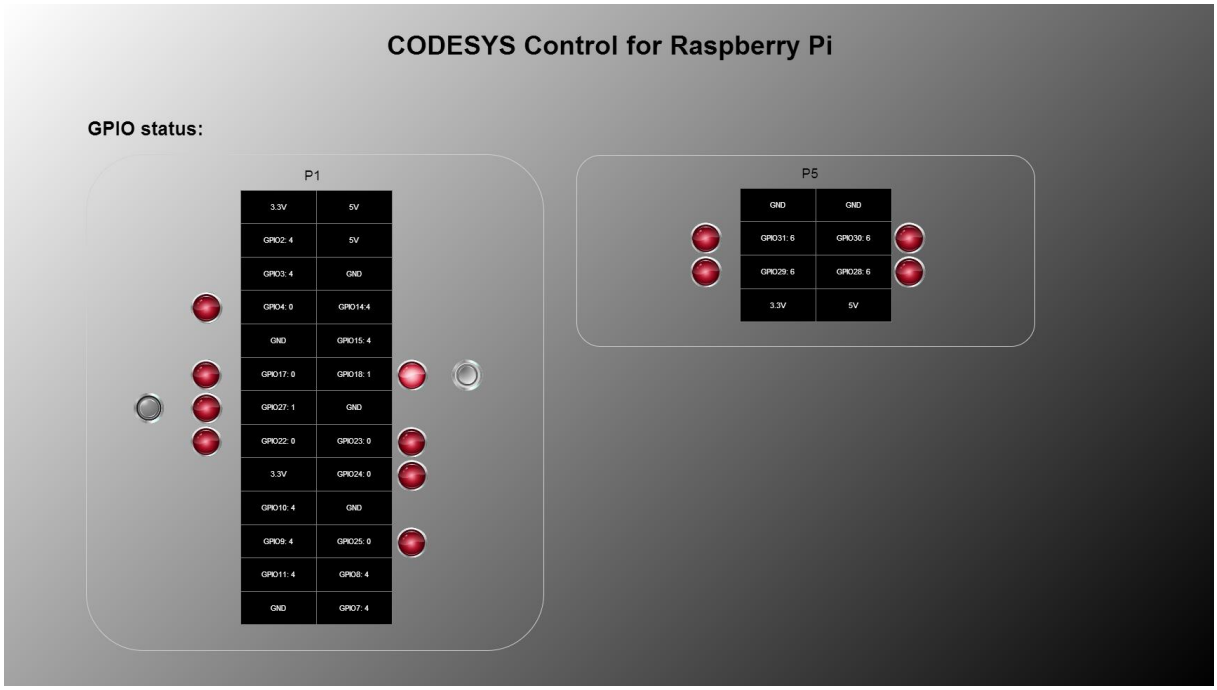
GPIOs

GPIOs Configuration | GPIOs I/O Mapping | Status | Information

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.

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).

6. PiFaceIoDrv.project (precondition: Raspberry PiFace Digital is connected)

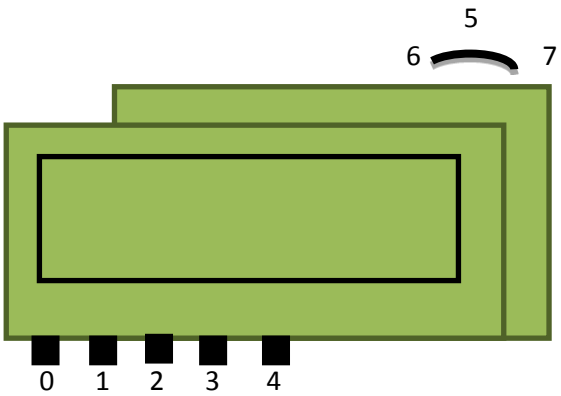
This example is similar to the one before. Instead of providing the I/O data as inputs of an automatically instanced function block, the example implements the data exchange as standard PLC IO driver via a process image, how it is typically done on PLCs.

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

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 generated, which offers several methods and properties and lets the application control the device and returns the actual values of the switches, where the single bits (0..7) of the output variablke bySwitches represent the following switches:



In the example application the FB instance is passed over to an instance of the FB ParamListPiFace, that implements the logic of a parameter editor. In the view mode you can step through the parameter list using the navigator switch (6/7). Pressing it (5) shows details of the parameter of the first line. This detailed view can be exited with 4. By pressing 5 once again you enter the editing mode. Now you can modify the value of the parameter with the switches 0 and 1. Button 2 stores the parameter, button 4 exits the editing mode and goes back to the detailed view.

8. I2CExample.project (precondition: special hardware is connected via I²C)

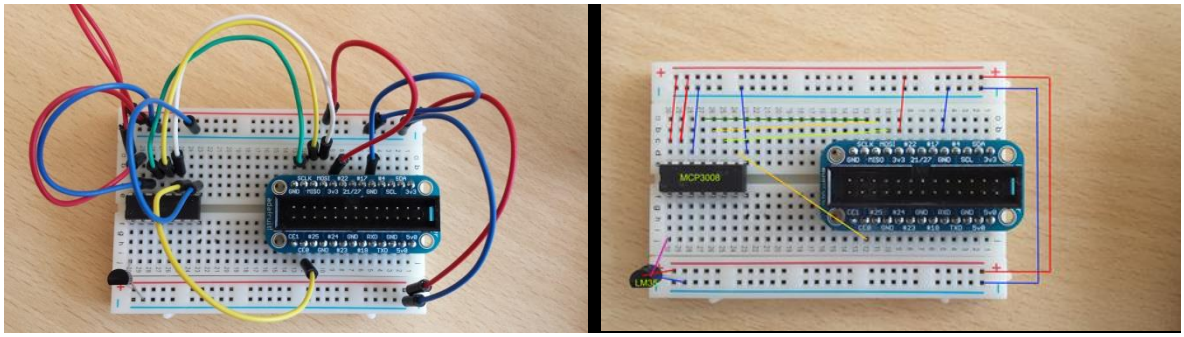
This example communicates with the following breakouts via I²C:

- Adafruit 16-channel/12-Bit PWM
- SRF02 (supersonic distance sensor)
- Drotek IMU 9DOF - MPU9150 (gyroskope, accelerometer, compass)

The libraries I2C_* that implement the data exchange are provided in source code and can be used as example for additional interface connections. The communication bases on the library RaspberryPiPeripherals, for which reference documentation is provided (see online help (F1) -> Libraries).

9. MCP3008_Temperature.project (precondition: special hardware is connected via SPI)

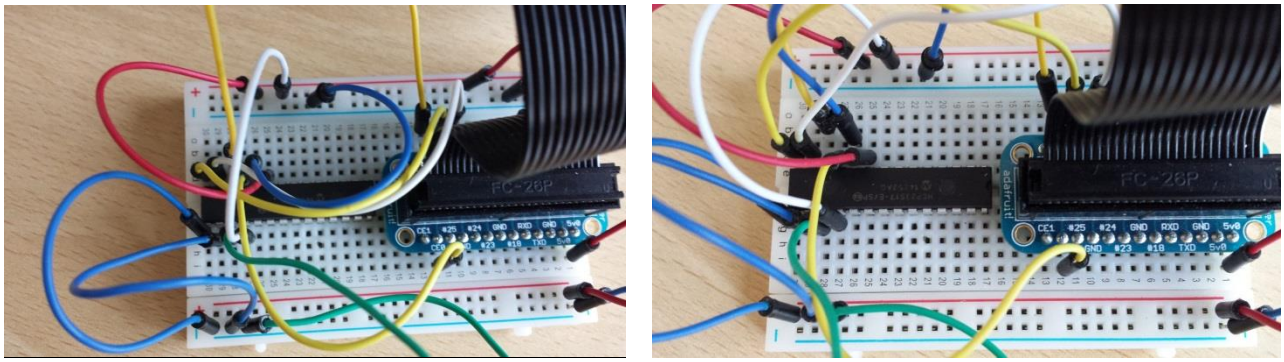
This example shows how the value of an analog temperature sensor (LM35), that is connected to an A/D converter chip (MCP3008), can be read in CODESYS via SPI. MCP3008 can process 8 analogue channels. In this example we use only one of these. The following test installation is used:



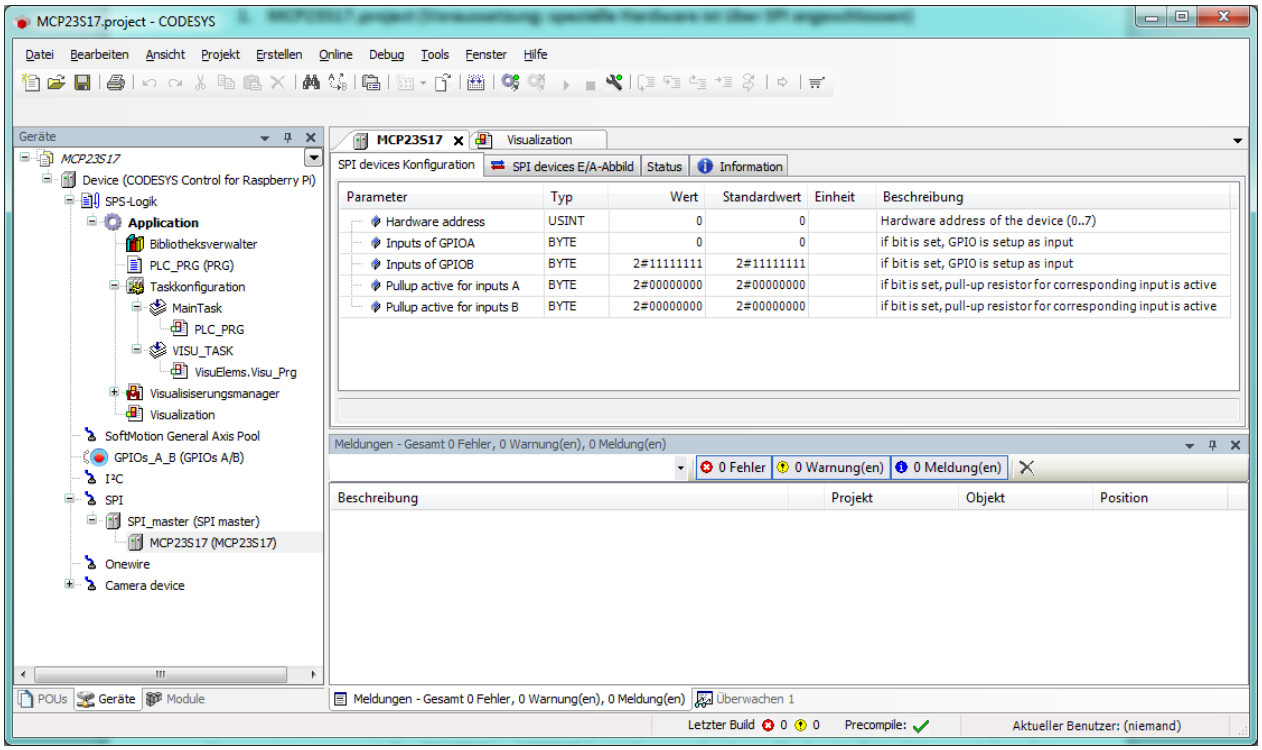
The library SPI_MCP3008.library that implements the data exchange is provided in source code and can be used as example for additional interface connections. The communication bases on the library RaspberryPiPeripherals, for which reference documentation is provided (see online help (F1) -> Libraries).

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:



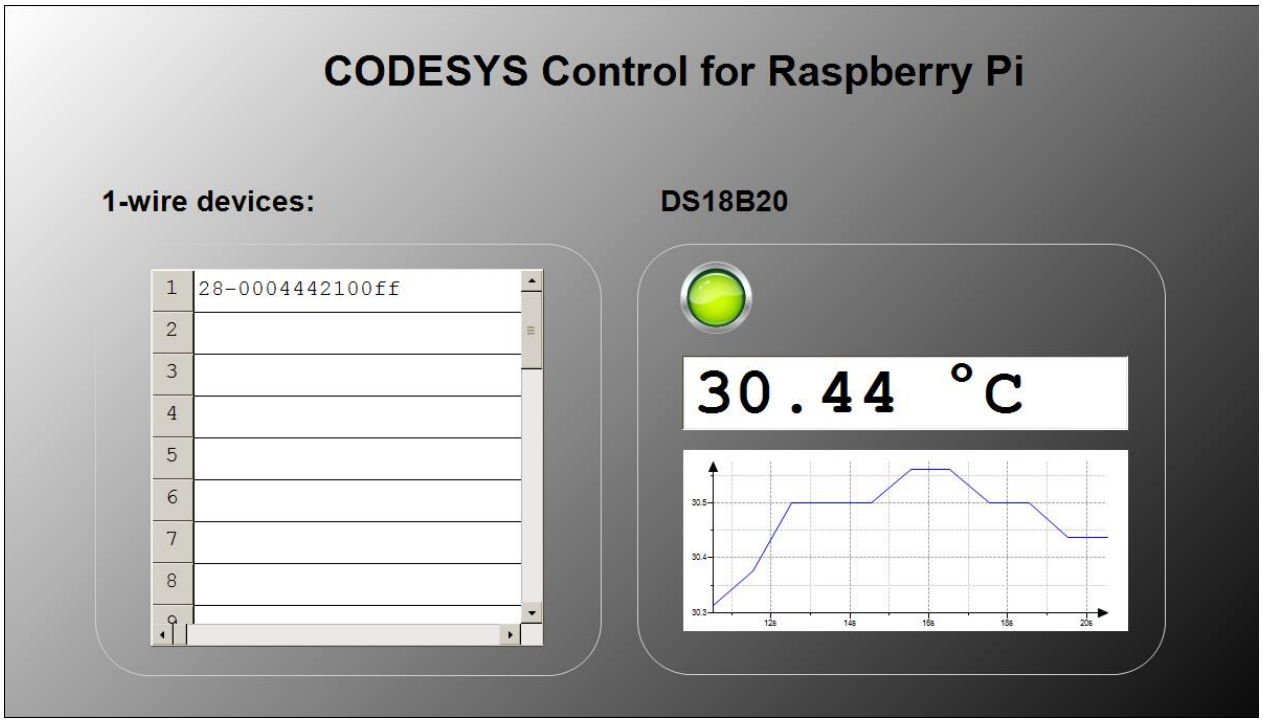
In the device settings you can specify, the direction (input/output) of the GPIOs and for inputs you can activate a pull-up resistor. Depending on the wiring of the hardware address pins, the address must be set in the configuration screen.



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



Note that the data exchange via 1-wire takes quite a lot of calculation time in this implementation, during which the task is blocked. Therefore it is recommended for time critical applications to assign the 1-wire devices to a task of low priority (see device configuration, tab I/O exchange, buscycle options).

12. SoftMotion Servo Example (precondition: an Adafruit 16-channel/12Bit PWM circuit board is connected via I²C. On its first PWM channel a servo motor is connected)

This example shows, how CODESYS SoftMotion can be used with modelbuilding servo motors. An additional circuit board, Adafruit ID 815, acts as communication interface.

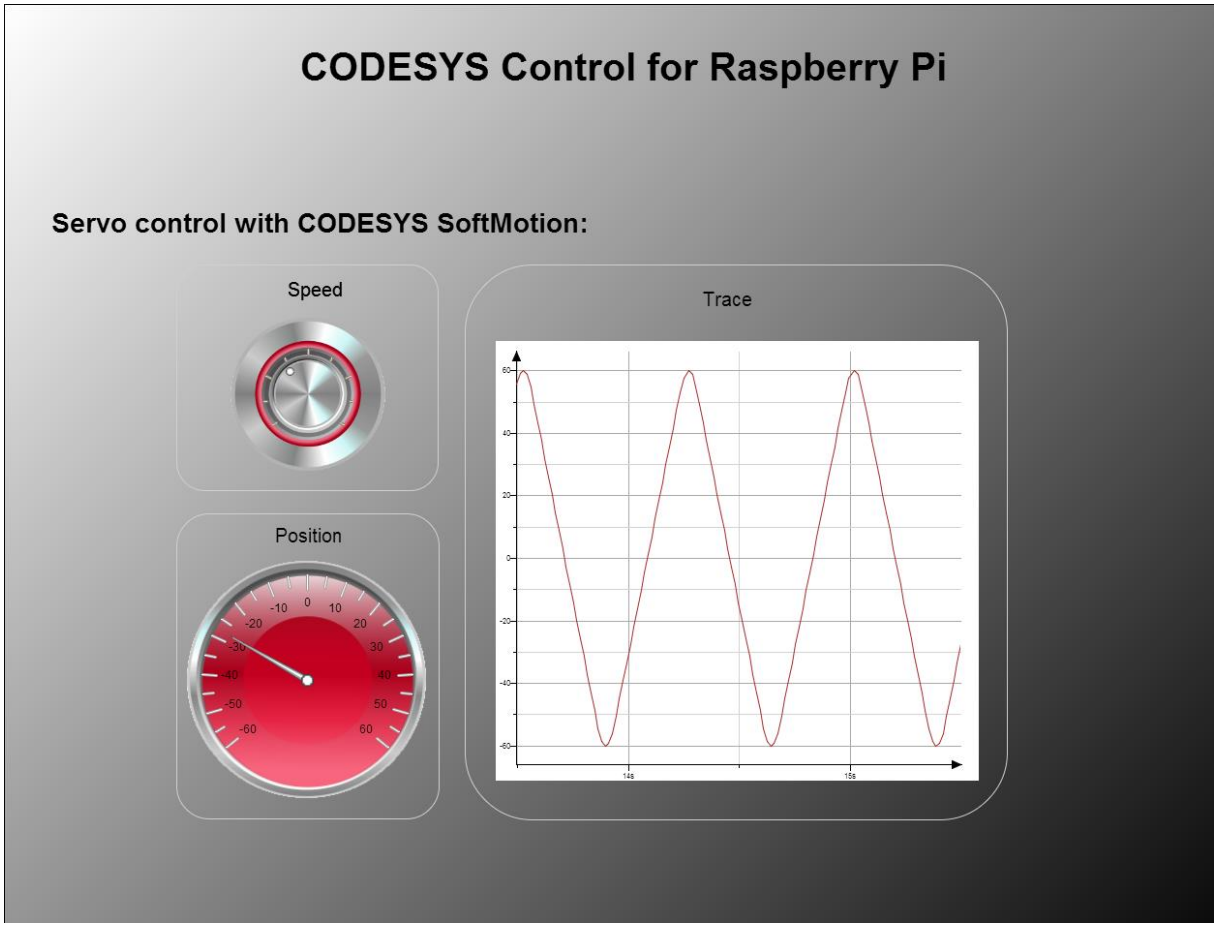
Open the project and download it to your Raspberry Pi. The motor starts continuously to turn. This has been programmed in SFC in PLC_PRG, which first enables the axis and then moves it between the positions -60 and +60, which had been configured as limit positions.

The set position command is transmitted to the servo via a PWM interface. In a fixed cyclic period (default: 50Hz, parameter of device Adafruit PWM Softmotion) a HIGH pulse is generated with a duration between 1ms and 2ms. 1ms represents the lower, 2ms the upper position limit. The movement range differs between servo types. To control the drive positions e.g. in degrees, it is necessary to measure the position range by moving the servo to its limits and enter the measured positions in the configuration screen of the axis device:

SoftMotion Drive: Basic | SM_Drive_Servo: Configuration | SM_Drive_Servo: I/O Mapping | Status | Information

Parameter	Type	Value	Default Value	Unit	Description
AXIS_REF: Standard					
wDriveID	WORD	1	1		ID of drive
bVirtual	BOOL	FALSE	FALSE		drive is simulated
dwRatioTechUnitsDenom	DWORD	1	1		conversion inc./tech.units denominator
iRatioTechUnitsNum	INT	1	1		conversion inc./tech.units numerator
iMovementType	INT	1	1		movement type: 0: rotary/modulo, 1: linear
fPositionPeriod	LREAL	360.0	360.0		modulo value for rotary drives
eRampType	INT	0	0		selects the velocity ramp type used by the FBs
fSWMaxVelocity	LREAL	1e3	1e3		maximum velocity value used for limit check at SMC_ControlAxisByPos
fSWMaxAcceleration	LREAL	1e5	1e5		maximum acceleration value used for limit check at SMC_ControlAxisByPos
fSWMaxDeceleration	LREAL	1e5	1e5		maximum deceleration value used for limit check at SMC_ControlAxisByPos
fRampJerk	LREAL	0	0		jerk used for bringing acceleration to 0 when sin² ramp is interrupted
fSWLimitPositive	LREAL	1000.0	60.0		software limit position in positive direction
fSWLimitNegative	LREAL	0.0	-60.0		software limit position in negative direction
fSWLimitDeceleration	LREAL	0	0		deceleration value used to stop when a software error has been detected
bSWLimitEnable	BOOL	FALSE	TRUE		activate/deactivate software limit
fSWErrorMaxDistance	LREAL	0	0		maximum distance that may be travelled for ramping down after a software error has been detect...
Servo: Configuration					
negative position [units]	LREAL	-60	-60.0		
positive position [units]	LREAL	60	60.0		
startup position [units]	LREAL	0.0	0.0		

Connect with a web browser to <network address>:8080/webvisu.htm and see the generated set positions and influence the velocity:



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.

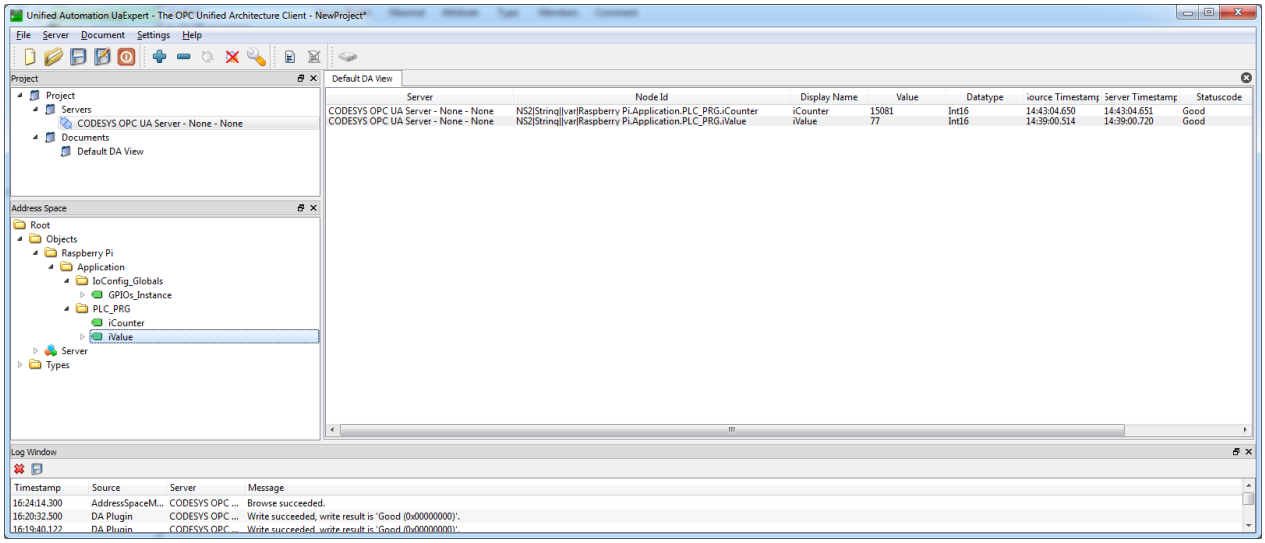
Open the project, download and start it. The outputs of the clamp will change continuously.

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.

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:



7 Connect additional devices via I²C and SPI

There are three typic ways to connect and run additional hardware via the i²c, 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 PiFaceIoDrv (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, i²c). Execute the following steps:

A. Device descirption

- 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:[1]

```
<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>
```

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

- 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">
```

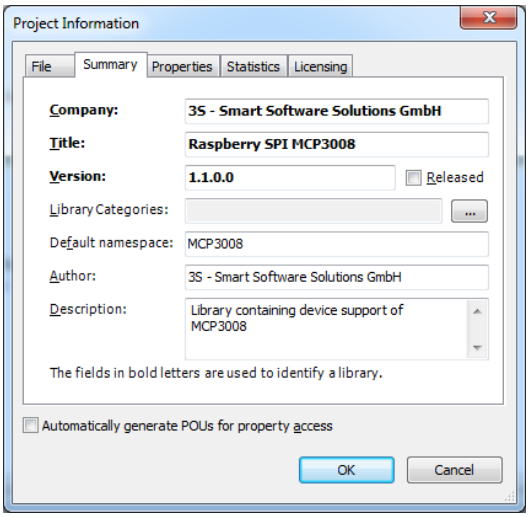
- Set the name of the FB in the library, which will do the communication and represent the device:

```
<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 „I²C devices“ or „SPI devices“.

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.

8 Screenshots