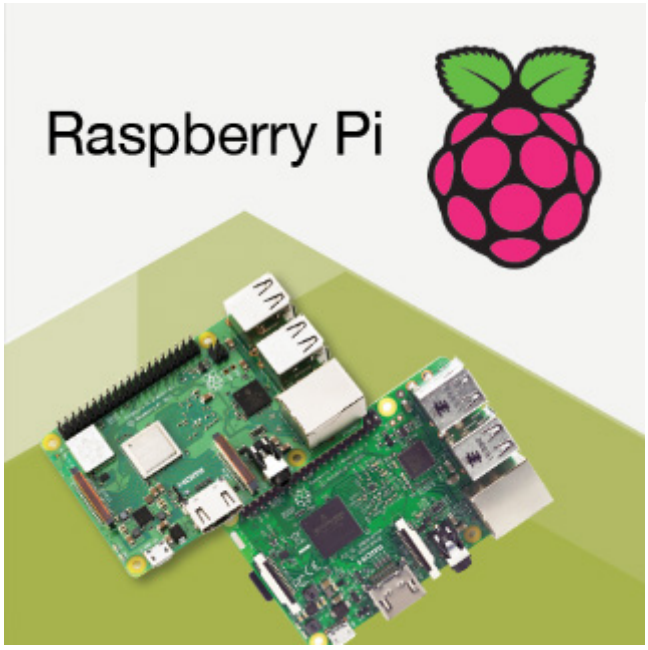
CODESYS для Raspberry Pi

Вопросы и ответы



10.01.2020
версия 1.0

https://oscat.ru/

Оглавление

Введение

Если вы читаете эту статью – то, по всей видимости, являетесь (или планируете стать) счастливым обладателем одноплатного компьютера Raspberry Pi. Как известно, изначально Raspberry Pi разрабатывался в качестве устройства для обучения информатике. Компактность, функциональность и, главное, дешевизна позволили ему стать популярным и в других сферах – например, в задачах домашней автоматизации. Некоторые пользователи даже стали пытаться применять его для управления производственными процессами в различных отраслях промышленности – в качестве предельно бюджетного программируемого логического контроллера (ПЛК).

Популярность устройства привела к тому, что в некоторых средах программирования ПЛК, рассчитанных на работу с «настоящими» контроллерами, появилась возможность программирования и Raspberry Pi. К числу таких сред относятся CODESYS V3, ISaGRAF, MULTIPROG, Beremiz и т.д.

Для авторов этой статьи «родным» инструментом является CODESYS – поэтому мы решили рассказать об особенностях работы с этой средой при программировании Raspberry Pi. Статья написана в формате FAQ, то есть представляет собой набор вопросов и ответов.

Если у вас есть замечания и предложения – пожалуйста, пришлите их нам на электронный адрес: OscatLibRu@gmail.com

Вы можете ознакомиться с другими нашими материалами и переводами по тематике ПЛК и АСУ ТП на сайте www.oscat.ru.

1. О чем эта статья?

Эта статья описывает некоторые специфические моменты, связанные с программированием Raspberry Pi в среде CODESYS V3.

Мы не рассматриваем общие вопросы по Raspberry Pi – в сети неисчислимое множество материалов, примеров и видео по данной тематике.

Мы не рассматриваем общие вопросы по CODESYS – вы можете найти соответствующую информацию в онлайн-справке среды программирования, а также, например, обучающих материалах по CODESYS компании ОВЕН.

Подразумевается, что читатель уже обладает базовыми навыками работы с Raspberry Pi (в частности, с терминалом Linux) и CODESYS V3.

Мы используем Putty в качестве терминала и WinSCP в качестве файлового менеджера. Напомним, в ОС Raspbian по умолчанию для подключения используется логин **pi** и пароль **raspberry**.

При решении многих вопросов потребуется использовать команду **chmod**, которая позволяет изменить права доступа к файлу. Например, команда

sudo chmod 0777 /etc/CODESYSControl_User.cfg

делает файл **CODESYSControl_User.cfg**, расположенный в директории **/etc**, доступным для чтения, записи и запуска на исполнение.

2. Зачем программировать Raspberry Pi в CODESYS?

Raspberry Pi в большинстве случаев используется с ОС, основанной на Linux. Для Linux написано очень много свободно распространяемого и открытого ПО. К тому же, вы можете использовать для программирования Python, C/C++ и другие языки. Возникает резонный вопрос – зачем в этом случае вообще нужен CODESYS?

По нашему мнению, CODESYS будет интересен:

- пользователям с опытом в программировании ПЛК, которые смогут использовать Raspberry Pi для отладки своих проектов и создания тестовых стендов. Например, Raspberry Pi удобнее использовать дома, чем обычный ПЛК – он компактный, ему не нужен блок питания, он основан на архитектуре ARM/Linux (как и большинство «настоящих» ПЛК, программируемых в CODESYS), у некоторых моделей есть Wi-Fi. Наконец, не каждая компания выдаст сотруднику ПЛК (который часто стоит существенную сумму) для домашнего использования;
- пользователям, планирующих использовать Raspberry Pi в домашней автоматизации и IoT. Очевидно, что есть и другие решения в этой области – достаточно упомянуть Node.js – но CODESYS как одно из них тоже имеет право на жизнь;
- пользователям, планирующим на свой страх и риск применять Raspberry Pi в задачах промышленной автоматизации;
- пользователям, стремящихся обучиться программированию ПЛК.

3. Какой функционал предоставляет CODESYS?

Если говорить в общих чертах, то CODESYS V3 содержит:

- редакторы кода для языков стандарта МЭК 61131-3 со встроенными средствами отладки;
- веб-сервер визуализации, редактор которой интегрирован в среду программирования;
- драйвера промышленных протоколов (Modbus, EtherCAT и т.д.);
- плагины для управления движением (SoftMotion).

Поставка системы исполнения для Raspberry Pi включает в себя:

- драйвера I^2C, SPI, 1-Wire (для конкретных устройств, с возможностью разработки своих драйверов для других устройств);
- обработку GPIO;
- драйвер Raspberry Pi Camera;
- протоколы Modbus RTU (Master/Slave), Modbus ASCII (Master), Modbus TCP (Master/Slave), EtherCAT (Master), EtherNet/IP (Scanner/Adapter), PROFINET (Controller/Device), CANopen (Master/Slave, через шлюз EL6751 от Beckhoff), J1939, OPC UA (Server);
- сервер web-визуализации;
- плагины для управления движением SoftMotion, SoftMotion CNC + Robotics (*начиная с версии 3.5.15.0 плагины станут платными*).

У пользователя имеется возможность приобретения дополнительных библиотек и драйверов протоколов (например, MQTT), а также возможность реализации своих протоколов с помощью системных библиотек.

4. Сколько стоит CODESYS для Raspberry Pi?

В первую очередь следует отметить наличие триальной версии. В этой версии время работы системы исполнения ограничено двумя часами, время работы драйверов промышленных протоколов – получасом. Система исполнения может быть перезапущена вручную (средствами CODESYS) или автоматически. Триальная версия доступна только для одноядерной системы исполнения (*релиз триальной версии для многоядерной системы исполнения запланирован для CODESYS V3.5 SP16*).

Для снятия ограничений по времени требуется приобрести лицензию. Ее стоимость – **50 евро** для одноядерной системы исполнения и **75 евро** – для многоядерной (*цены на лето 2019 года без НДС*). Старожилы помнят те времена, когда стоимость одноядерной системы исполнения составляла всего **35 евро**.

После покупки лицензии предоставляется так называемый soft key (ключ активации), который активирует лицензию на конкретном экземпляре Raspberry Pi. Пользователь также может приобрести аппаратный ключ (CODESYS Runtime Key), который стоит **45 евро**. Преимуществами использования аппаратного ключа является возможность его применения с разными экземплярами Raspberry Pi, повышенная надежность (soft key может слететь, например, при обновлении ОС) и возможность привязки к ключу лицензий на дополнительные компоненты, приобретенные в CODESYS Store.

5. Какую версию CODESYS использовать для Raspberry Pi?

Для программирования Raspberry Pi используется CODESYS V3.5. Среда находится в постоянном развитии, и ежегодно выходят пакеты обновлений, называемых сервис-паками (SP). В рамках каждого сервис-пака выходит некоторое количество патчей, исправляющих критические ошибки. На момент написания этих строк самой свежей версией является CODESYS V3.5 SP15 Patch 2 (3.5.15.20).

Поддержка Raspberry Pi впервые была представлена в версии 3.5.4.0. В версии 3.5.8.0 была добавлена поддержка моделей B, B+, 2B, в версии 3.5.9.20 – модели 3B. Начиная с версии 3.5.11.0 поддерживаются все модели. Поддержка моделей 3A+ и 4 осуществляется с версий 3.5.15.0 и 3.5.15.10 соответственно.

Поддержка многоядерной системы исполнения появилась в версии 3.5.14.0

На данный момент в CODESYS Store доступны версии системы исполнения начиная с 3.5.10.0. Более старые версии официально не распространяются.

Так какую же версию лучше использовать? Есть два основных варианта:

- самую свежую версию из доступных – вы получаете максимум функционала и актуальные исправления багов (и, потенциально, новые баги, о которых еще никто не знает);
- привычную для вас (например, ту версию, в который вы программируете ПЛК на работе. Вероятно, вы уже знаете все ее баги и способы их обхода).

Настоятельно рекомендуется использовать одну и ту же версию системы исполнения и среды программирования.

6. Какие ОС поддерживает среда исполнения CODESYS для Raspberry Pi?

Только Raspbian (версию Stretch или выше).

7. Какие модели Raspberry Pi можно программировать в CODESYS?

Начиная с версии CODESYS V3.5 SP11 (3.5.11.0) поддерживаются все модели. Поддержка новых моделей добавляется по мере их выхода (например, в версии 3.5.15.0 была добавлена поддержка 3A+).

Также поддерживаются некоторые устройства, созданные на базе Raspberry Pi – например, PiXtend и UniPi Neuron.

8. Как установить систему исполнения CODESYS на Raspberry Pi?

1. Загрузите пакет для одноядерной или многоядерной[1] системы исполнения Raspberry Pi из CODESYS Store (предварительно потребуется зарегистрироваться). Выбрать нужную версию системы исполнения можно на вкладке **All Versions**. В результате будет загружен файл формата **.package**.

2. В среде программирования CODESYS в меню **Инструменты** выберите команду **Менеджер пакетов** и нажмите кнопку **Установить**. Укажите путь к загруженному файлу формата .package. В процессе установки потребуется указать путь, по которому будет распакован образ системы исполнения, а также примеры и шаблоны для разработки драйверов.

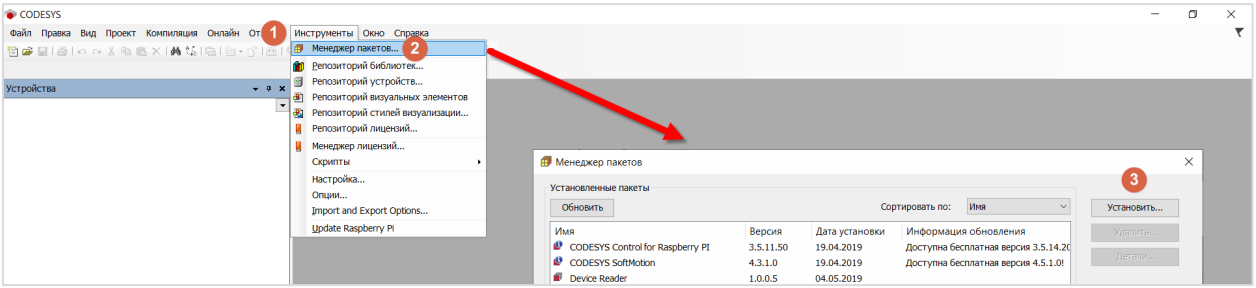


Рис. 1. Установка пакета в CODESYS

3. После установки пакета в меню **Инструменты** появится новая команда: **Update Raspberry Pi**. Используйте ее, чтобы перейти в меню управления Raspberry Pi (см. рис. 2). В нем потребуется указать логин и пароль для подключения к устройству (по умолчанию – **pi/raspberry**) и IP-адрес. Нажмите кнопку **Scan**, чтобы найти ваше устройство. Если все пройдет успешно, то появится окно **Scan Raspberry Pi**, в котором будет отображаться IP-адрес и MAC-адрес вашего устройства. Нажмите кнопку **OK** для подтверждения подключения. В строке **Package Directory** укажите путь, по которому был распакован образ системы исполнения (он был выбран при установке пакета в п. 2). По умолчанию путь выглядит следующим образом:

   C:\Users\<имя_пользователя>\CODESYS Control for Raspberry PI

   Нажмите **Install** для установки системы исполнения.

   В этом же меню во вкладке **Runtime** находятся кнопки **Start** и **Stop**, которые позволяют запустить и остановить системы исполнения. Если вы используете триальную версию системы, то она автоматически прекратит свою работу спустя два часа – и из этого меню вы сможете ее перезапустить.

[1] Обратите внимание, что для многоядерной системы исполнения триальная версия недоступна – вам потребуется сразу приобрести лицензию (*релиз триальной версии для многоядерной системы исполнения запланирован для CODESYS V3.5 SP16).*

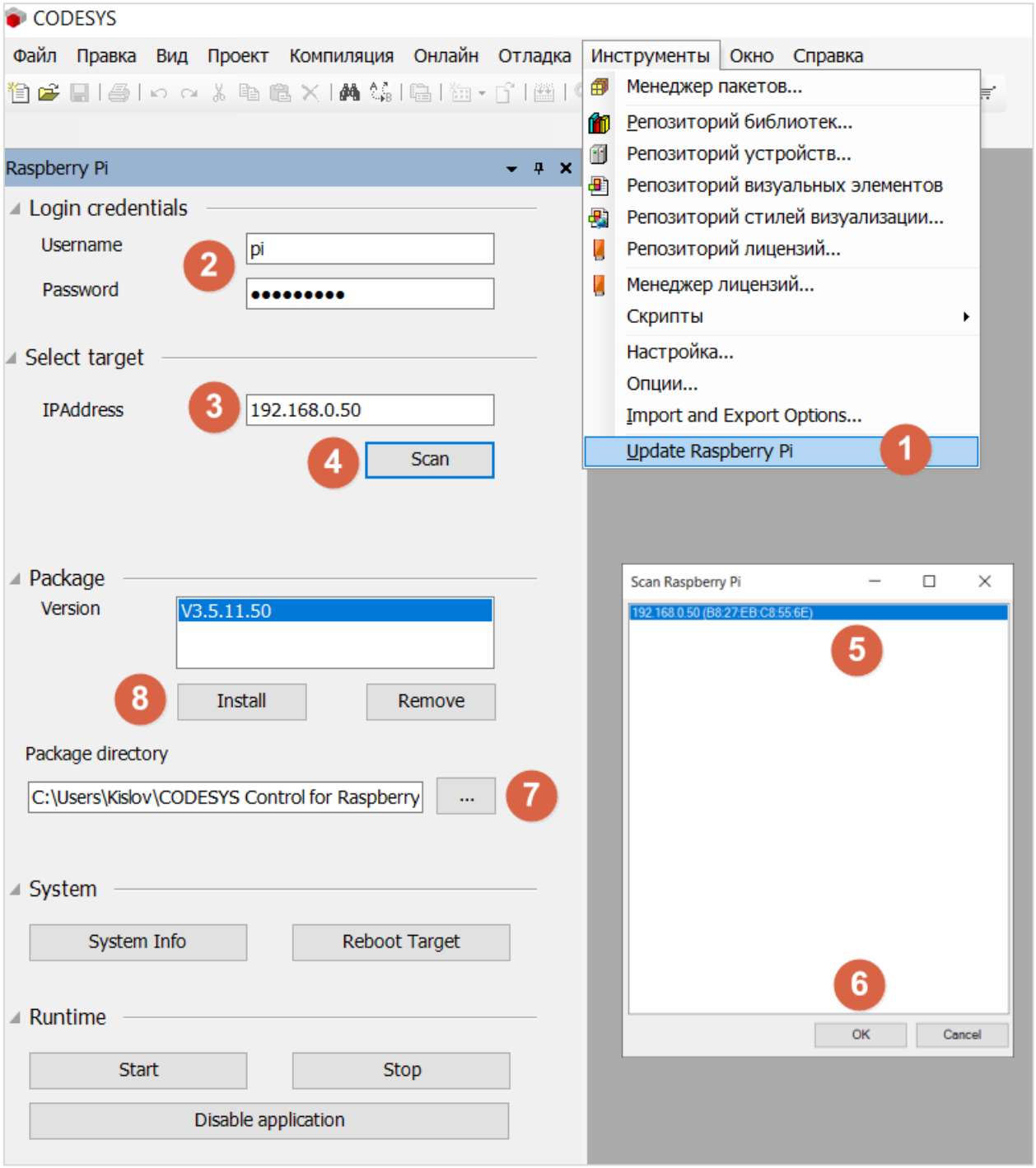


Рис. 2. Меню **Update Raspberry Pi**

4. В меню **Файл** выберите команду **Новый проект**, выберите тип проекта – **Стандартный**, укажите его имя и путь, по которому он будет сохранен. Нажмите **ОК** и выберите устройство **Codesys Control for Raspberry Pi SL** (см. рис. 3). Нажмите **ОК**.

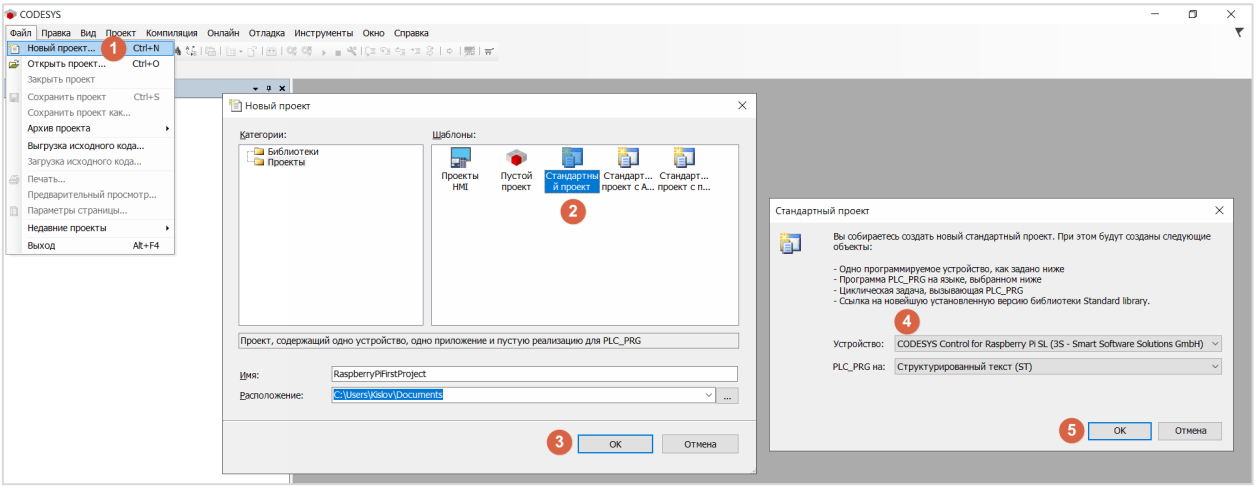


Рис. 3. Создание нового проекта CODESYS

5. Двойным кликом выделите компонент **Device**. Выберите вкладку **Установки соединения** и нажмите кнопку **Сканировать сеть**. В появившемся окне должно отобразиться ваше устройство. Нажмите **ОК** для подключения.

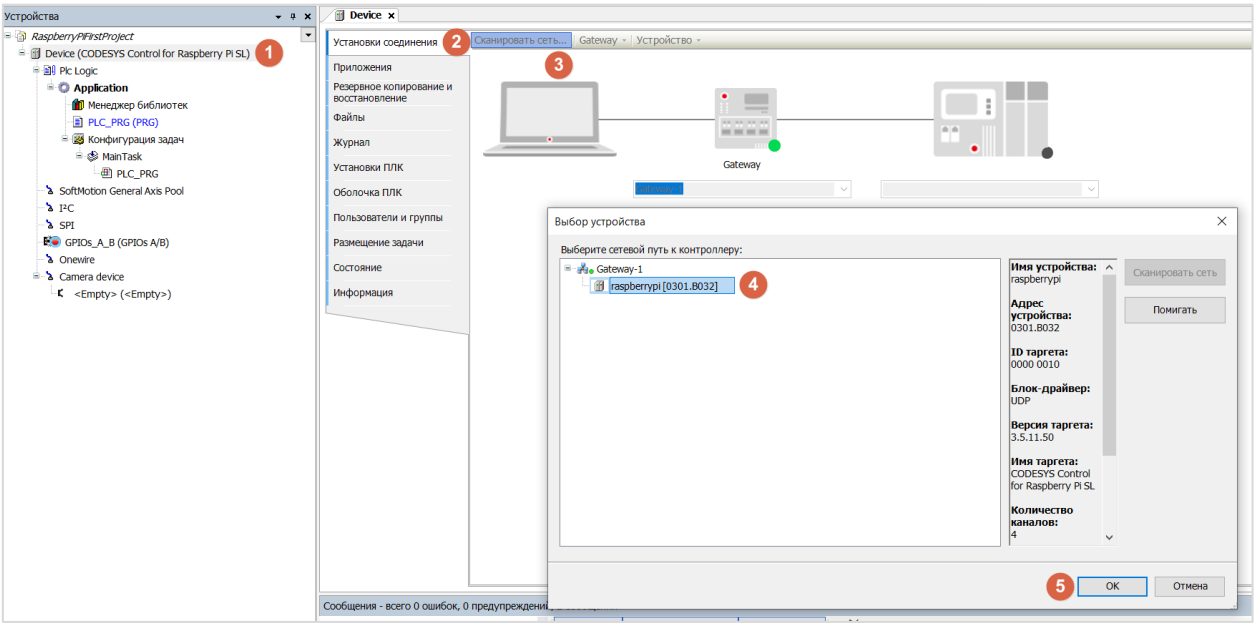


Рис. 4. Подключение к Raspberry Pi из CODESYS

Если ваше устройство не было найдено при сканировании, что попытайтесь:

- перезапустить систему исполнения (см. рис. 2, вкладка **Runtime**);
- перезагрузить Raspberry Pi;
- ввести IP-адрес устройства вручную и нажать **Enter** (см. рис. 5). Последний вариант связан с тем, что сканирование производится с помощью широковещательных UDP-пакетов – которые могут быть заблокированы в некоторых сетях.

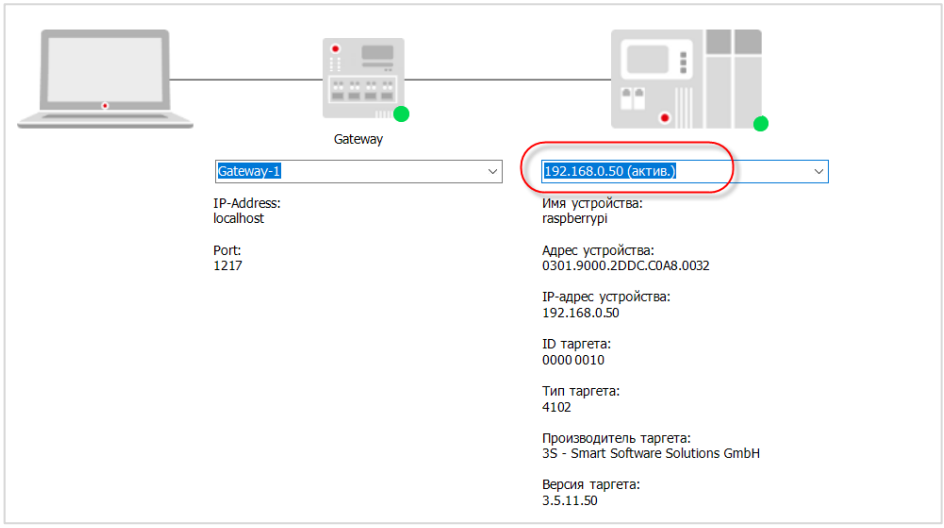


Рис. 5. Информация о подключенном устройстве

6. В меню **Онлайн** выберите команду **Логин**. В появившемся окне загрузки проекта нажмите **ОК** (если в устройстве уже был какой-то проект, то окно может выглядеть иначе). После загрузки (и не отключаясь от устройства) выполните в меню **Онлайн** команду **Создать загрузочное приложение**, чтобы ваш проект сохранился в Raspberry Pi после его перезагрузки.

   Если все прошло успешно, то в дереве проекта отобразится статус подключения (соединен) и статус приложения (запуск). Если приложение не запущено, то в меню **Отладка** выполните команду **Старт**.

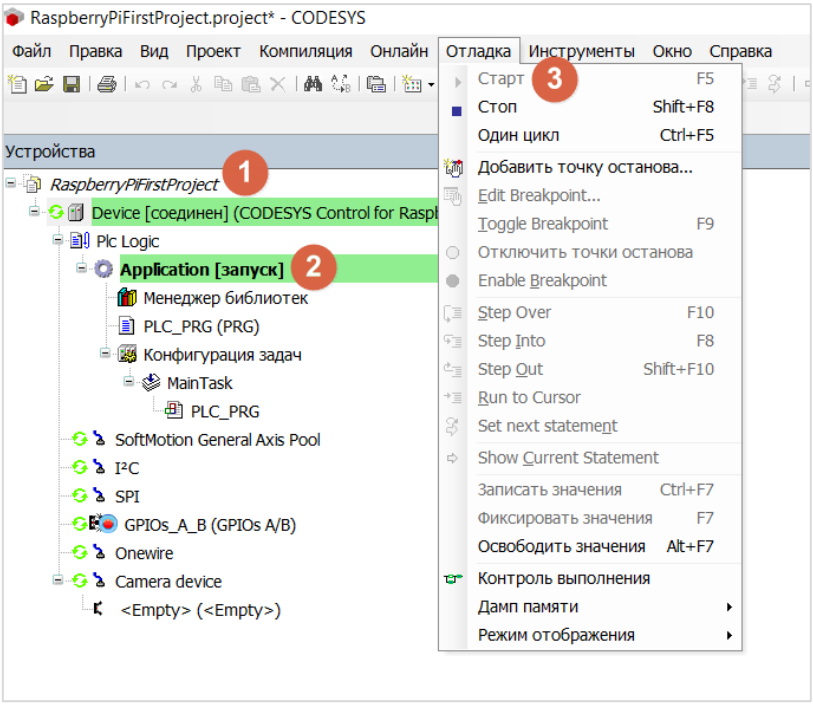


Рис. 6. Запуск приложения

Итак, вы загрузили в Raspberry Pi пустой проект. Теперь вы можете наполнить его кодом, визуализациями и т.д. В п. 9 приведены ссылки, где искать информацию по этим вопросам.

9. Где найти дополнительную информацию?

1. Общая информация о CODESYS V3.5 доступна в онлайн-справке, а также обучающих материалах компании ОВЕН.
2. В онлайн-справке есть отдельный раздел по Raspberry Pi. До создания онлайн-справки содержащая в ней информация была доступна в виде отдельного документа Raspberry PI with Standard CODESYS V3. First Steps.
3. На форуме CODESYS есть отдельный раздел по Raspberry Pi и раздел, в котором выкладываются примеры и драйвера устройства, разработанные пользователями и разработчиками CODESYS.
4. В CODESYS Store есть ряд примеров и продуктов, связанных с Raspberry PI:
    - OpenCV for Raspberry Pi
    - PiXtend for CODESYS
    - PiXtend V2 Professional for CODESYS
    - CODESYS for UniPi
    - Pigeon for CODESYS Control
    - MEGA-IO Library for CODESYS
    - Horter & Kalb - I2C-Support for Raspberry PI
5. На форуме компании ОВЕН есть отдельная тема, в которой обсуждается программирование Raspberry Pi в CODESYS.
6. Разумеется, вы можете найти больше информации в интернете – на YouTube, в специализированных блогах и т.д.

10. Как разрабатывать свои библиотеки и драйвера?

При установке пакета Raspberry Pi в среду CODESYS в выбранную вами директорию ПК будет распакован не только файл системы исполнения, но и примеры, шаблоны драйверов и описаний устройств. По умолчанию используется директория

C:\Users\<имя_пользователя>\CODESYS Control for Raspberry PI

Шаблоны драйверов представляют собой библиотеки CODESYS с открытым исходным кодом (.library). Пользователь с навыками программирования на языке ST и опытом работы в CODESYS сможет без особых проблем по аналогии разрабатывать свои драйвера.

Описания устройств представляют собой файлы формата .devdesc.xml. Они используется для создания GUI, позволяющего настраивать драйвера в среде программирования. Каждое устройство в дереве проекта (например, **GPIOs_A_B**) представляет собой совокупность файла описания и библиотеки (для сложных устройств может использования несколько файлов описаний и библиотек).

Информация о формате файлов описаний и инструкции по их разработке отсутствуют в онлайн-справке CODESYS. Определенная документация входит в поставку (delivery), которую получают разработчики контроллеров и ПО, заключившие договор с CODESYS Group.

«Обычным» пользователям остается только изучать уже готовые описания и данную статью, чтобы путем проб и ошибок создавать свои файлы описаний.

11. Как заменить таргет в уже готовом проекте на таргет Raspberry Pi?

Для начинающих пользователей это может показаться не совсем очевидным, поскольку при выполнении команды изменения таргет-файла в проекте (**Device – Обновить устройство**) требуется открыть папку ПЛК SoftMotion, чтобы найти таргет-файл Raspberry Pi.

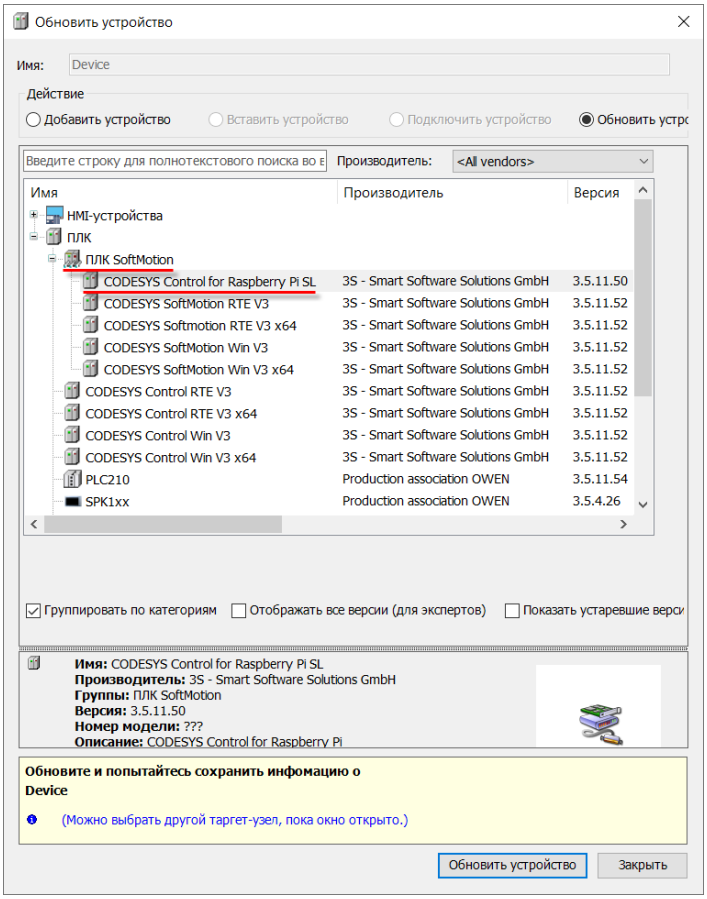


Рис. 7. Выбор таргета Raspberry Pi при замене таргет-файла в проекте

12. Какой путь к рабочей директории CODESYS?

Рабочая директория CODESYS: **/var/opt/codesys**

Рекомендуем сразу установить на нее права для записи и исполнения:

sudo chmod -R 0777 /var/opt/codesys

13. Как перезапустить систему исполнения?

Как упоминалось в п. 4, время работы триальной системы исполнения ограничено двумя часами. После этого систему исполнения можно перезапустить вручную – см. в п. 8 рис. 2 (команда **Runtime/Start**). Если требуется, чтобы система исполнения работала без остановок, то единственный корректный способ добиться этого – приобрести лицензию.

Тем не менее, можно организовать автоматический перезапуск системы исполнения средствами Linux или даже самого CODESYS. Рассмотрим один из примеров реализации второго варианта:

1. Добавьте в файл **/etc/CODESYSControl_User.cfg** следующий блок:

   ```
   [SysProcess]
   Command=AllowAll
   ```

   Не забудьте, что сначала потребуется сделать файл доступным для записи:

   sudo chmod 0666 /etc/CODESYSControl_User.cfg

2. Добавьте в проект библиотеку **SysProcess**.
3. Напишите в проекте следующий код:

PLC_PRG

```
1  PROGRAM PLC_PRG
2  VAR
3      pRes:   POINTER TO SysProcess.SysTypes.RTS_IEC_RESULT;
4      fbTon:  TON;
5  END_VAR
6
```

```
1  fbTon(IN := NOT(fbTon.Q), PT := T#30M);
2
3  IF fbTon.Q THEN
4      // TODO: перевести систему в безопасное состояние
5      // отправляем команду в терминал Linux на перезапуск системы исполнения
6      SysProcess.SysProcessExecuteCommand('sudo service codesyscontrol restart', pRes);
7  END_IF
```

Рис. 8. Пример кода, перезапускающего систему исполнения

Приведенный код спустя 30 минут после старта приложения перезапускает систему исполнения (таким образом, остановки из-за истечения триального периода не происходит никогда). Интервал времени перезапуска выбран исходя из того факта, что время работы драйверов промышленных протоколов (настраиваемых через дерево проекта) ограничено 30 минутами. Разумеется, вы можете указать другой интервал (но он должен быть меньше 120 минут). Перед перезапуском системы исполнения следует перевести программу в безопасное состояние – например, закрыть файлы, переключить GPIO и т.д. Следует помнить, что подобный подход к перезапуску системы исполнения допустим только в тех системах, где периодическая потеря управления на несколько секунд допустима (например, это может быть приемлемым в рамках умного дома)

14. Как вызвать Python-скрипт из программы CODESYS?

В более общем виде этот вопрос звучит так: «Как выполнить команду в терминале Linux из программы CODESYS?».

Это можно сделать с помощью библиотеки **SysProcess**. В первую очередь обязательно ознакомьтесь с п. 13 – там написано, как нужно поправить конфиг-файл CODESYS для использования библиотеки и приведен пример работы с ней.

В библиотеке **SysProcess** содержится более десятка функций, из которых наиболее часто используемыми являются **SysProcessExecuteCommand** (отправить команду в терминал) и **SysProcessExecuteCommand2** (отправить команду и получить результаты выполнения). Следует отметить, что функции библиотеки выполняются в блокирующем режиме.

Рассмотрим пример запуска Python-скрипта из кода.

Для начала создадим файл **test.py** со следующим содержимым:

```
#!/usr/bin/env python
f = open('new_test_file.txt', 'a')
f.write('Hello, world\n')
f.close()
```

Данный скрипт создает в рабочей директории CODESYS файл **new_test_file.txt** и при каждом вызове записывает туда строку «Hello, world». Очевидно, что пример является совершенно синтетическим – ту же самую операцию было бы проще выполнить через библиотеку CODESYS **SysFile** или **CAA File**, но в рамках примера этот код удобен тем, что позволяет легко увидеть результат своего выполнения.

Если вы создали файла скрипта на ПК, то можете загрузить скрипт в Raspberry Pi вручную (например, через WinSCP). Но удобнее прикрепить его к проекту CODESYS: для этого следует нажать ПКМ на узел **Application**, использовать команду **Добавление объекта** и выбрать объект **Внешний файл**, указав путь к файлу скрипта:

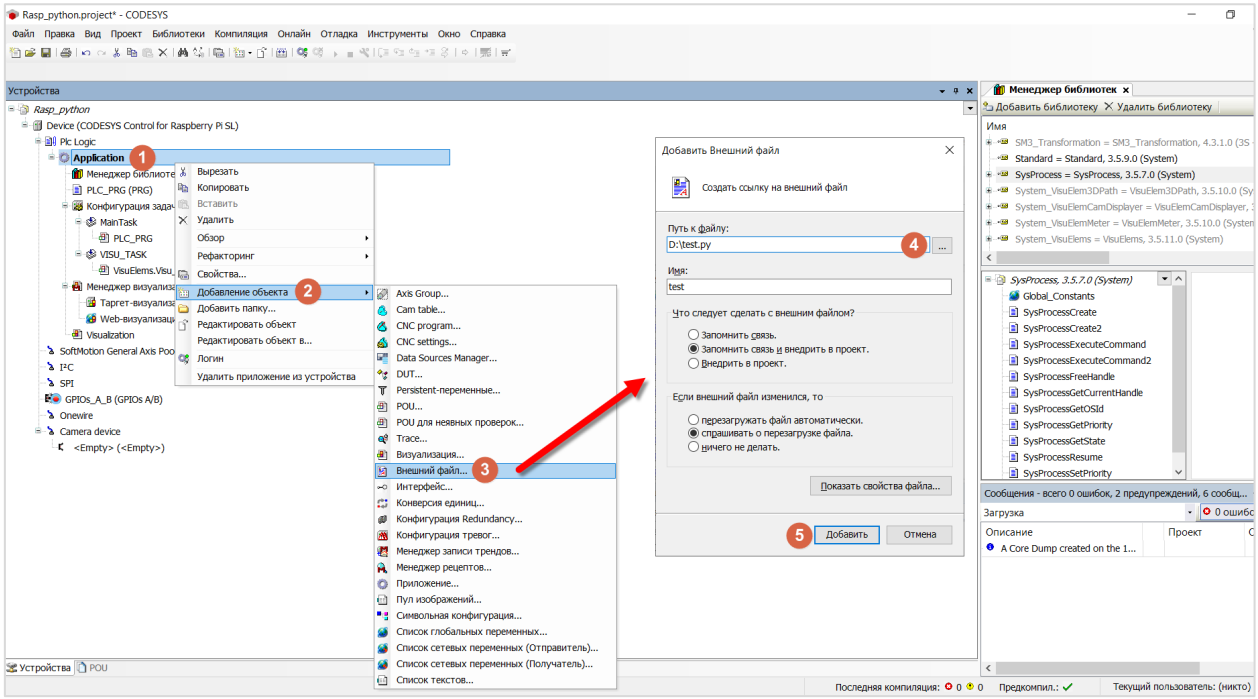


Рис. 9. Прикрепление файла к проекту CODESYS

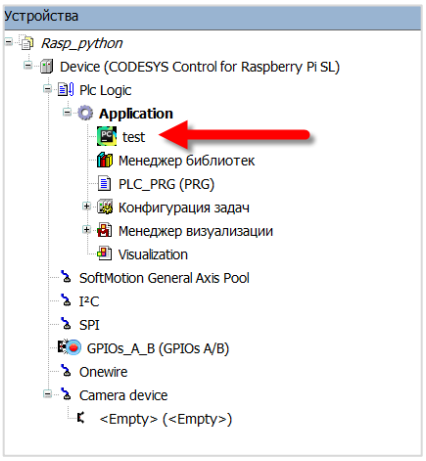


Рис. 10. Прикрепленный файл в дереве проекта

При загрузке проекта прикрепленный файл будет помещен в директорию **/var/opt/codesys/PlcLogic/Application/**

Удостоверьтесь, что файл доступен для запуска на исполнение. При необходимости установите ему нужные права командой

sudo chmod 0777 /var/opt/codesys/PlcLogic/Application/test.py

Для запуска скрипта в программе CODESYS используйте следующий код:

```
PROGRAM PLC_PRG
VAR
    xExecuteScript:  BOOL;
    pResult:         POINTER TO SysProcess.SysTypes.RTS_IEC_RESULT;
END_VAR
```

```
IF xExecuteScript THEN
    SysProcess.SysProcessExecuteCommand('sudo python /var/opt/codesys/PlcLogic/Application/test.py', pResult);
    xExecuteScript:= FALSE;
END_IF
```

Рис. 11. Код вызова скрипта из программы CODESYS

Вызов скрипта будет происходить при каждом присвоении значения **TRUE** в переменную **xExecuteScript**.

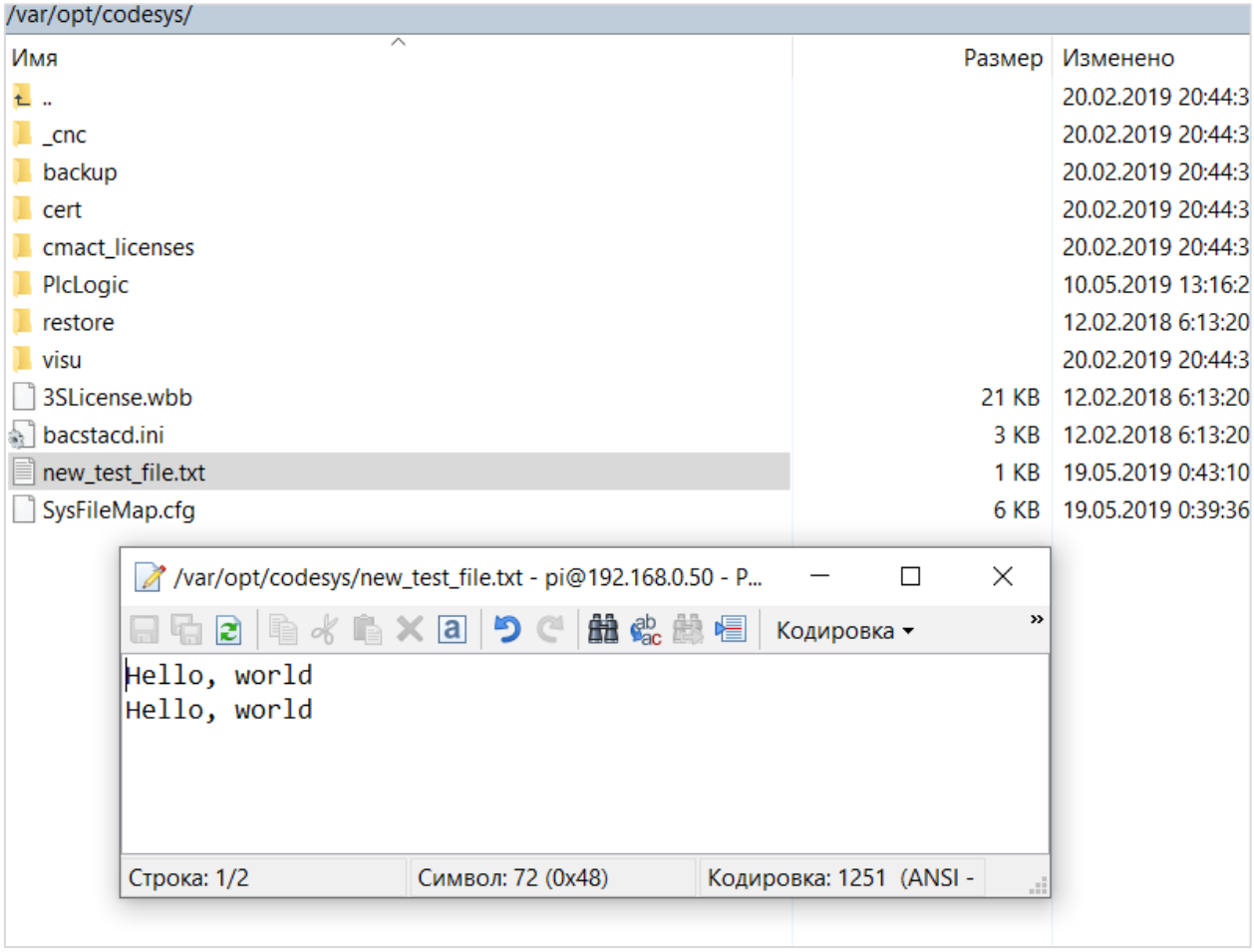


Рис. 12. Результат выполнения скрипта (после двух вызовов)

15. Поддерживает ли система исполнения CODESYS для Raspberry Pi таргет-визуализацию?

Таргет-визуализация позволяет отображать визуализацию CODESYS на дисплее, физически подключенном к устройству (например, по HDMI). Система исполнения CODESYS для Raspberry Pi не поддерживает эту возможность. В качестве альтернативы можно настроить web-браузер на Raspberry Pi в режиме киоска и отображать в нем web-визуализацию CODESYS. В режиме киоска браузер автоматически открывается на полный экран после загрузки устройства. Пример настройки Chromium в таком режиме доступен здесь.

16. Как определить путь к подключенному USB-накопителю?

Предположим, вы хотите подключить к Raspberry Pi USB-накопитель, чтобы средствами CODESYS (например, библиотеками SysFile и CAA File) записывать на него архивы или считывать с него рецепты. В этом случае вам потребуется знать путь к накопителю. Определить его можно, выполнив в терминале команду **cd** и найдя в ее выводе строку типа **/dev/sda1** (если накопителей несколько, то они могут называться sda2, sdb1 и т.п.). В правом столбце этой строки отображается путь к накопителю (в нашем случае – **/media/pi/321A-C127**). Вы можете определить путь из кода программы с помощью библиотеки **SysProcess** (см. пример ее использования в п. 14).

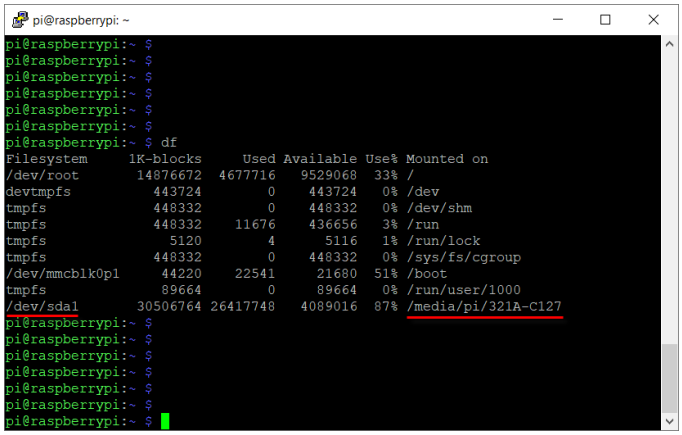


Рис. 13. Определение пути к USB-накопителю

17. Поддерживает ли CODESYS протокол MQTT?

Одна из популярных сфер применения Raspberry Pi – домашняя автоматизация, в рамках которой протокол MQTT де-факто является стандартом. В дистрибутиве CODESYS отсутствует возможность использования этого протокола, но функционал MQTT-клиента может быть добавлен с помощью одной из доступных библиотек. Ниже приведен их обзор.

Библиотека	Автор	Цена	Сложность использования	Поддержка QoS	Поддержка TLS
MQTT Client	Stefan Rossmann Engineering Solution	бесплатно, open-source	очень просто	QoS0	нет
MQTT	Stefan Dreyer	бесплатно, open-source	сложно	QoS0, QoS1, QoS2	есть
Janz Tec MQTT library	Janz Tec	49 евро (без НДС) – лицензия на каждое устройство	просто	QoS0, QoS1	есть
MQTT Client	3S-Smart Software Solutions GmbH	300 евро (без НДС) – лицензия на ПК разработчика	просто	QoS0, QoS1, QoS2	есть

Также вы можете разработать свою библиотеку или использовать внешнюю библиотеку, не связанную с CODESYS (например, paho-mqtt). Чтобы взаимодействовать с внешними библиотеками из программы CODESYS потребуется использовать библиотеку **SysProcess** (см. п. 13 и 14).

Готовые реализации MQTT-брокера для CODESYS отсутствуют, но, опять-таки, вы можете воспользоваться внешними библиотеками.

18. Поддерживается ли для Raspberry Pi функция Visu File Transfer?

Начиная с версии V3.5 SP11 в CODESYS появилась возможность передачи и скачивания файлов с устройства через web-визуализацию. Соответствующий пункт доступен для элементов визуализации на вкладке действий в узле **Input Configuration**:

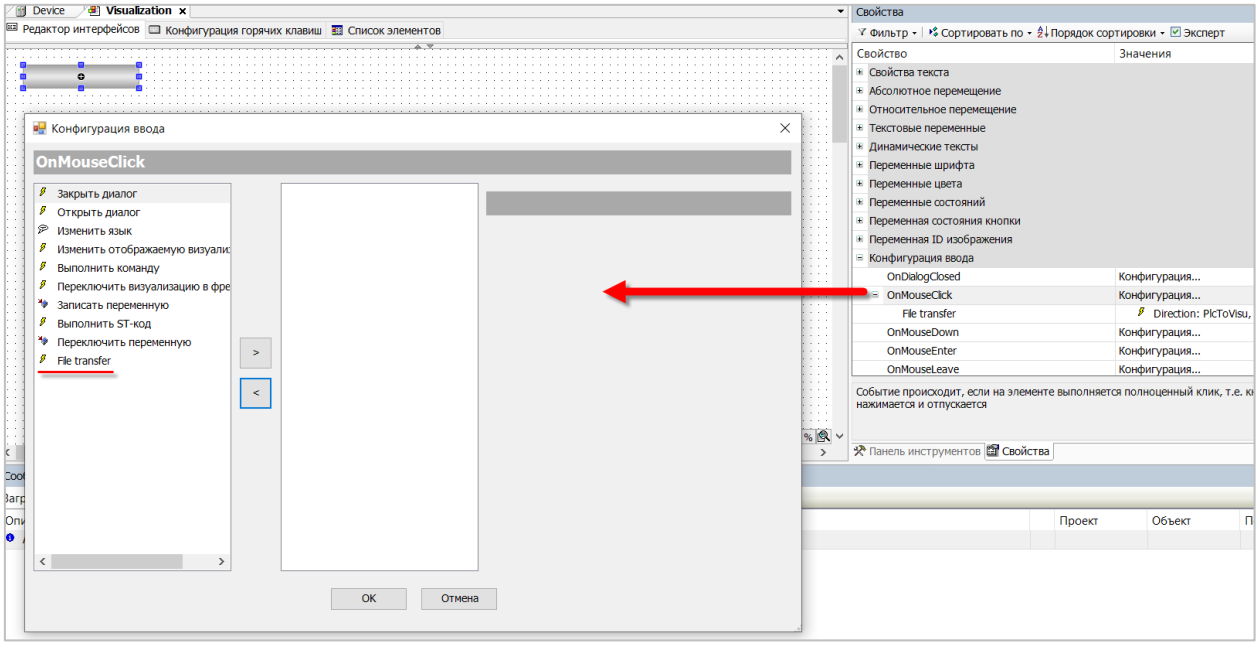


Рис. 14. Настройка действия Visu File Transfer

Описание пункта приведено в онлайн-справке CODESYS.

Этот функционал поддерживается и для Raspberry Pi, но для его активации потребуется отредактировать конфигурационный файл. Добавьте в файл **/etc/CODESYSControl_User.cfg** следующий блок:

```
[CmpWebServerHandlerV3]
AllowFileTransferServices=1
```

Не забудьте, что сначала потребуется сделать файл доступным для записи:

sudo chmod 0666 /etc/CODESYSControl_User.cfg

Чтобы изменения вступили в силу, потребуется перезагрузить устройство командой **sudo reboot**.

19. Как организовать обмен данными между CODESYS и другим приложением?

В некоторых случаях требуется организовать обмен данными между CODESYS и другим приложением, запущенным на Raspberry Pi (базой данных, web-сервером и т.д.). Очевидным вариантом является использование файлов (одно приложение пишет туда данные, другое читает) или сокетов (одно приложение выполняет функции клиента, второе – сервера). Тем не менее, такой подход может быть неприемлем, если требуется передавать большие объемы данных с высокой частотой. В этой ситуации оптимальным вариантом является использование разделяемой памяти (shared memory) с помощью библиотеки **SysShm**. В CODESYS Store доступен пример работы с библиотекой, в котором реализован обмен данными между CODESYS и приложениями, написанными на различных языках (C/Linux, C++/Windows, C#/Windows).

20. Как использовать GPIO?

В дереве проекта расположен узел GPIOs. На вкладке **Конфигурация** можно настроить режим работы каждого GPIO (не используется/вход/выход), на вкладке **Соотнесение входов/выходов** – привязать к ним переменные типа BOOL или DWORD (в случае использования битовой маски).

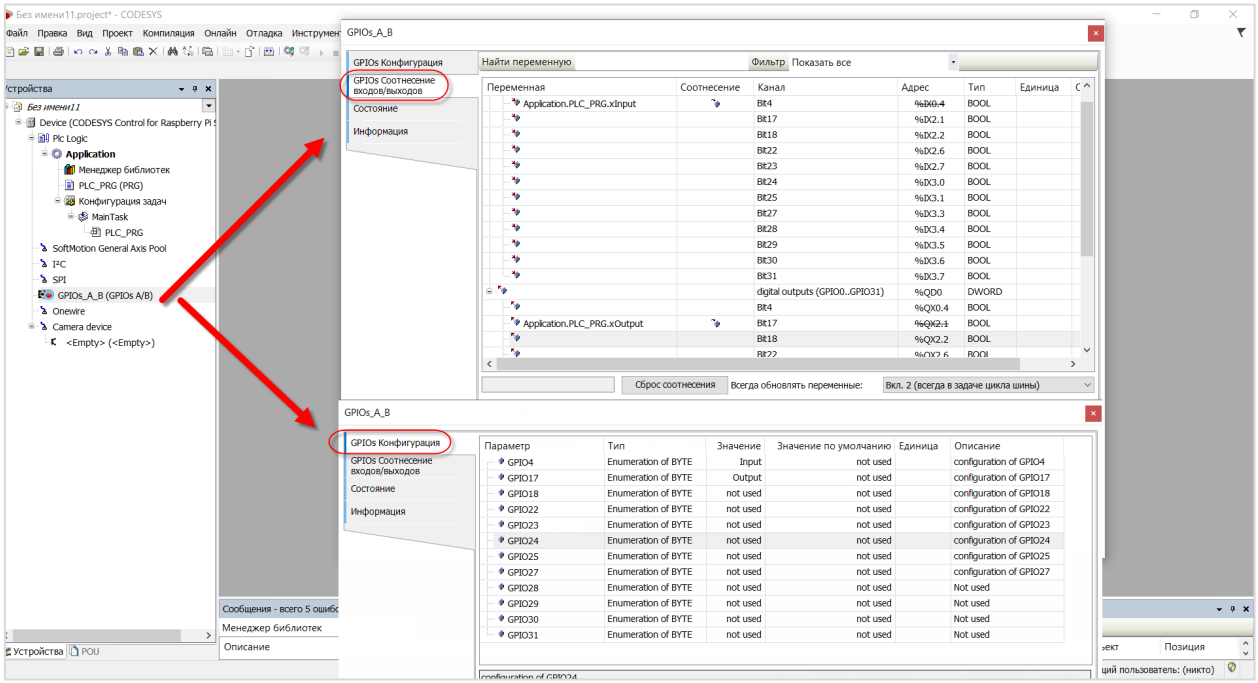


Рис. 15. Узел GPIOs в дереве проекта

По умолчанию узел содержит набор GPIO для моделей Raspberry Pi A и B (в нем присутствует 12 GPIO из имеющихся 17). Если вы используете более старшие модели, то нажмите на узел правкой кнопкой мыши и выберите команду **Обновить устройство**. Установите галочку

Отображать все версии и выберите элемент **GPIOs B+/Pi2**. В нем присутствует 17 GPIO из доступных 28.

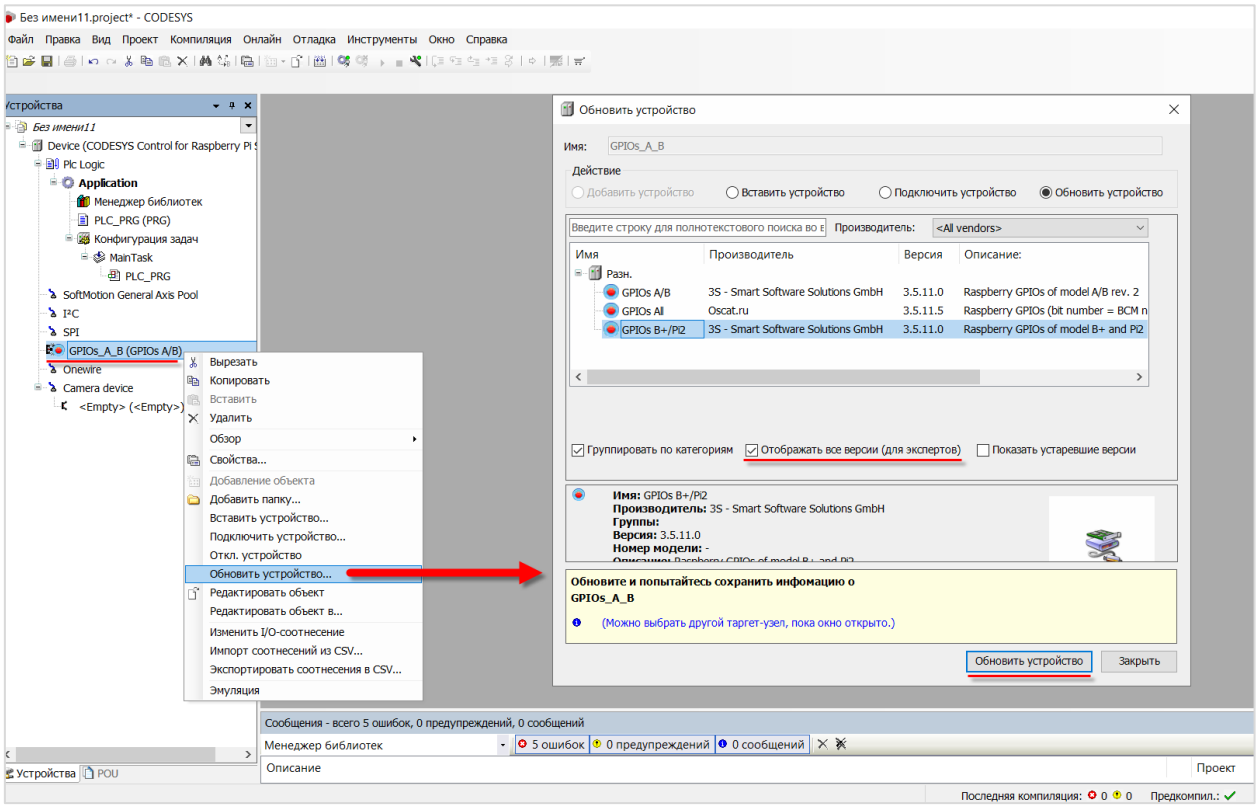


Рис. 16. Замена узла

Если нужно получить доступ ко всем GPIO, то вы можете воспользоваться отредактированным файлом описания **GPIOs All**. Скачайте этот архив, распакуйте его содержимое и в меню **Инструменты** выберите команду **Репозиторий устройств**. Нажмите кнопку **Установить**, выберите формат файла .devdesc.xml и укажите путь к файлу GPIOs_All.devdesc.xml из распакованного архива. После этого по приведенной выше инструкции замените в проекте узел GPIOs на **GPIOs All**.

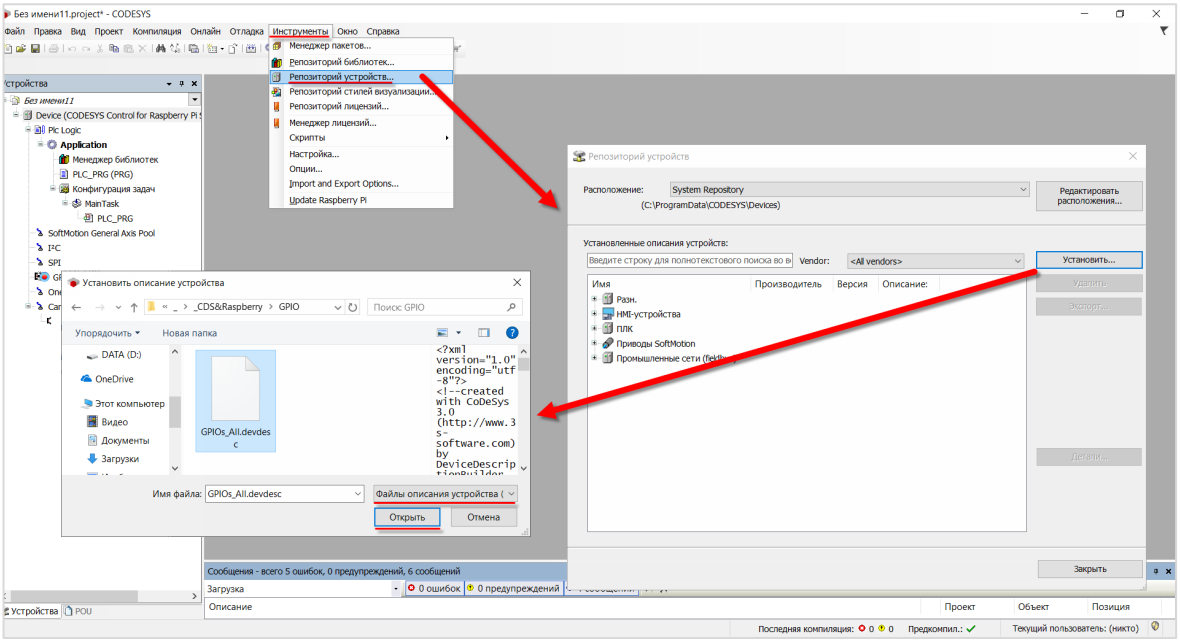


Рис. 17. Установка файла описания устройства

При отладке GPIO удобно использовать в терминале команду **gpio readall**.

```
pi@raspberrypi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 3+---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5v      |     |     |
 |   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT5 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 0 | ALT5 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI | ALT0 | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO | ALT0 | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | OUT  | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | OUT  | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 0 | OUT  | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3+---+---+------+---------+-----+-----+
pi@raspberrypi:~ $
```

Рис. 18. Вывод команды gpio readall

21. Как настроить обмен по последовательному интерфейсу?

Raspberry Pi не имеет встроенных COM-портов (но есть UART), поэтому для работы по последовательному интерфейсу (RS-232 или RS-485) обычно используются преобразователи USB/Serial. Для работы с COM-портом в CODESYS требуется указать его ID (номер).

Предварительно надо выполнить следующие операции:

1. Сделать файл **/etc/CODESYSControl_User.cfg** доступным для записи:

   sudo chmod 0666 /etc/CODESYSControl_User.cfg

   В файле раскомментировать строку в блоке [SysCom], приведя ее к подобному виду:

   ```
   [SysCom]
   Linux.Devicefile=/dev/ttyUSB
   ```

   Вместо ttyUSB может быть указан другой используемый интерфейс (например, ttyAMA и т.п.).
2. Сделать папку **/etc/udev/rules.d** доступной для записи:

   sudo chmod 0666 /etc/udev/rules.d

   Создать в этой папке файл **99-usb-serial.rules**

   sudo nano /etc/udev/rules.d/99-usb-serial.rules

   Пример содержимого файла (вы можете задать свое соответствие devpath и SYMLINK):

   ```
   SUBSYSTEM=="tty", ATTRS{devpath}=="1.2", SYMLINK+="ttyUSB0"
   SUBSYSTEM=="tty", ATTRS{devpath}=="1.4", SYMLINK+="ttyUSB1"
   SUBSYSTEM=="tty", ATTRS{devpath}=="1.3", SYMLINK+="ttyUSB2"
   SUBSYSTEM=="tty", ATTRS{devpath}=="1.5", SYMLINK+="ttyUSB3"
   ```

3. Перезагрузить устройство командой **sudo reboot**
4. В результате USB-порт 1.2 будет связан с символьным именем ttyUSB0 и в CODESYS будет иметь ID = 1. Порт 1.4 будет связан с ttyUSB1 и в CODESYS будет иметь ID=2, и так далее.

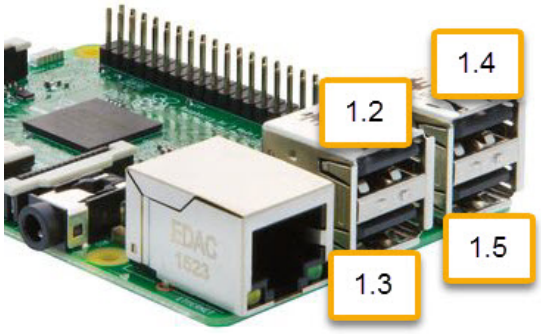


Рис. 19. Обозначения USB-портов Raspberry Pi

Перед началом работы стоит проверить, что преобразователи были определены ОС с помощью команды **lsusb**:

```
pi@raspberrypi:~ $
pi@raspberrypi:~ $ lsusb
Bus 001 Device 004: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x UART Br
idge / myAVR mySmartUSB light
Bus 001 Device 005: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x UART Br
idge / myAVR mySmartUSB light
Bus 001 Device 006: ID 0424:7800 Standard Microsystems Corp.
Bus 001 Device 003: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 001 Device 002: ID 0424:2514 Standard Microsystems Corp. USB 2.0 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
pi@raspberrypi:~ $
pi@raspberrypi:~ $
```

Рис. 20. Вывод команды lsusb (к Raspberry Pi подключены два преобразователя USB/RS-485 – к портам 1.2 и 1.4)

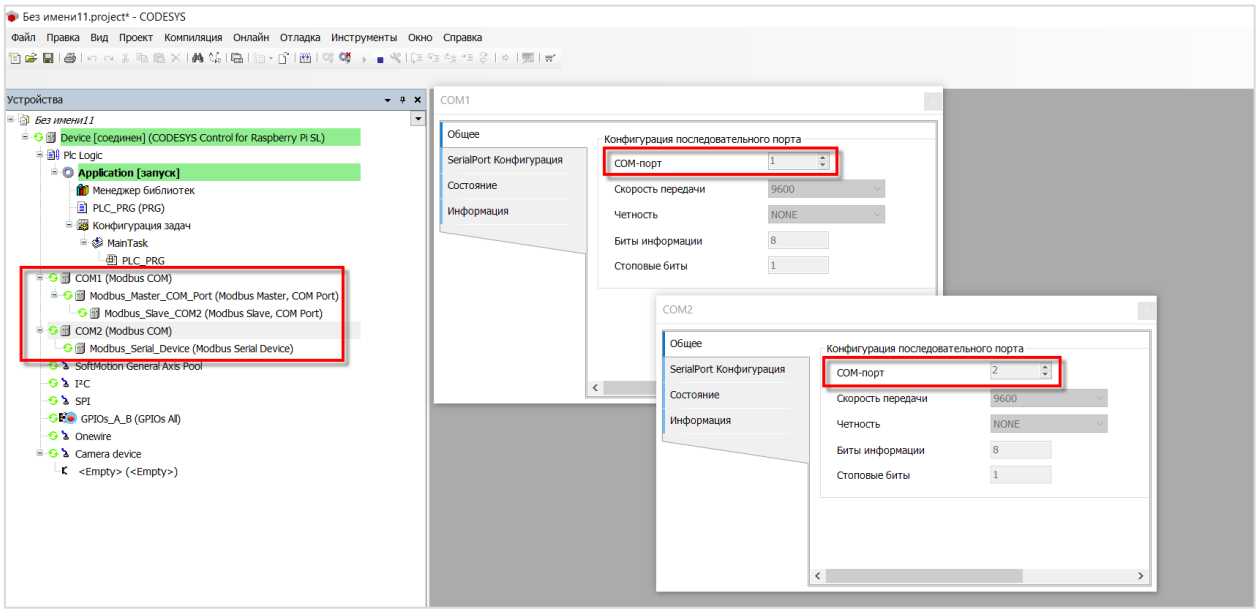


Рис. 21. Пример успешной настройки обмена

Данная статья основана на информации с форума CODESYS.

Информация о настройке обмена по Modbus приведена в онлайн-справке CODESYS.

На youtube можно найти видеопримеры – например, вот этот.

Для работы с портом напрямую (без использования компонентов из дерева проекта) можно использовать библиотеку CAA SerialCom.

22. Зачем нужна библиотека CmpCharDevice?

Библиотека **CmpCharDevice** входит в состав пакета (.package) CODESYS для Raspberry Pi. Она используется для работы с символьными устройствами Linux (например, устройствами, подключаемыми по шинами SPI и I^2C, часами реального времени и т.д.). В библиотеку входят функции открытия файла устройства (CDOpen), записи (CDWrite), чтения (CDRead), закрытия (CDClose) и управления устройством (CDIoctl).

См. пример использования библиотеки на форуме CODESYS и форуме ОВЕН (за последний благодарим Вячеслава Мезенцева).

23. Поддерживаются ли энергонезависимые переменные?

Система исполнения CODESYS для Raspberry Pi поддерживает энергонезависимые переменные (RETAIN и PERSISTENT). Сохранение переменных происходит только при корректном завершении работы системы исполнения (например, при выполнении команды **sudo reboot**). При пропадании питания энергонезависимые переменные не будут сохранены. Для сохранения энергонезависимых переменных по команде из программы пользователя применяются функции из библиотеки **CmpApp**.

По умолчанию под энергонезависимые переменные выделено 4 Кб памяти. Это значение может быть изменено путем редактирования таргет-файла.

Информация о редактировании таргет-файла и управлением сохранения переменных с помощью библиотеки **CmpApp** приведена в блоге Вячеслава Мезенцева.