Визуальное программирование (FBD) для микропроцессорных систем & IoT

В.Д. Мунистер

Издание третье

Учебно-практическое издание

«Визуальное программирование (FBD) для микропроцессорных систем и IoT»

Практикум

2021 г.

GLÜCKSRITTER
MUNISTER

§ INSCRIPTUM

Устойчивое желание обрести как можно больше профессиональных навыков и повысить общий уровень компетенций по тому или иному вопросу требуют более высокого уровня функциональности и комплексности в методологии преподаваемых дисциплин и междисциплинарных модулей.

Лабораторный практикум - существенный элемент учебного процесса в профессиональном учебном заведении, в ходе которого обучающиеся фактически впервые сталкиваются с самостоятельной практической деятельностью в конкретной области. Лабораторные занятия, как и другие виды практических занятий, являются средним звеном между углубленной теоретической работой обучающихся на лекциях, семинарах и применением знаний на практике. Эти занятия удачно сочетают элементы теоретического исследования и практической работы. Выполняя лабораторные работы, студенты лучше усваивают программный материал, так как многие определения и формулы, казавшиеся отвлеченными, становятся вполне конкретными, происходит соприкосновение теории с практикой, что в целом содействует уяснению сложных вопросов науки и становлению обучающихся как будущих специалистов.

Само значение слов «лаборатория», «лабораторный» (от латинского labor труд, работа, а laboro - трудиться, стараться, хлопотать, преодолевать затруднения) указывает на сложившиеся понятия, связанные с применением умственных и физических усилий к изысканию ранее неизвестных путей и средств для разрешения научных и жизненных задач.

Неслучайно слово «практикум», применяемое для обозначения определенной системы практических (преимущественно лабораторных) учебных работ, и выражает ту же основную мысль (греческое - praktikos), означает «деятельный», это значит, что предполагаются такие виды учебных занятий, которые требуют от обучающихся усиленной деятельности.

В целях создания интегрированного курса, связывающего между собой основы алгоритмизации и программирования (а также реверс-инжиниринга), электронику и электротехнику, был создан данный лабораторный практикум.

Полагаю, что разработанный мной практикум, базирующийся на концепции обучения посредством визуального программирования и встроенным программам интродукции в архитектуру аппаратных средств, широко используемых во встроенных компьютерных системах, помогут вам в профессиональной деятельности.

Мунистер В.Д.

§ СОДЕРЖАНИЕ

«Визуальное программирование (FBD) для микропроцессорных систем и IoT»

§ СОДЕРЖАНИЕ КУРСА

Как говорил еще Конфуций: задача учителя — открывать новую перспективу размышлениям ученику.

Раскрытие перспективы современных, многофункциональных и доступных к применению в практической деятельности микропроцессорных систем может достигаться за счет вовлечения в результат удивительного сплава мысли в области программной инженерии – визуального программирования.

Визуальное (графическое) программирование — способ создания программы для ЭВМ путём манипулирования графическими объектами вместо написания её текста. Его часто представляют, как следующий этап развития текстовых языков программирования.

В последнее время визуальному программированию стали уделять больше внимания — в связи с развитием мобильных сенсорных устройств и средств, обеспечивающих Human-machine interface (человеко-машинный интерфейс) на уровне взаимодействия оператора с органами управления системы. Визуальное программирование в основном используется для создания программ с графическим интерфейсом для операционных систем с графическим интерфейсом пользователя.

Среда визуального программирования позволяет написать Web-приложение для браузера; создать консольное приложение для программирования микроконтроллеров, программируемых микросхем.

В учебно-практическом пособии «Визуальное программирование (FBD) для микропроцессорных систем и IoT» уделено большое внимание изучению и применению в прикладных задачах распространенных микропроцессорных систем (MCU/SoC) посредством их конфигурирования (данных управляющих устройств) на графическом языке программирования FBD (Function Block Diagram) под конкретные целевые задачи.

Издание предназначено для студентов, изучающих дисциплины «Алгоритмизация и программирование», «Микропроцессорные системы», «Проектирование микропроцессорных систем», «Аппаратно-программные комплексы», «Вычислительные машины системы и сети», «Вычислительная техника и сети в отрасли», «Основы визуального программирования».

Практикум является элементом интегрированного курса «Интернет вещей. Межмашинное взаимодействие. Программирование в компьютерных системах и сетях» в который входит учебно-теоретическое издание (хрестоматия): «Компьютерные сети. IoT и межмашинное взаимодействие», и учебно-практическое издание «Дом, который построил сам себя. Сетевой практикум. IoT»

Данное учебно-практическое издание должно выработать у студента навыки осознанного применения теоретических знаний о беспроводных технологиях взаимодействия и идентификации, изученных в ранее указанных изданиях рассматриваемого интегрированного курса по сетевому взаимодействию вместе с рассматриваемыми технологиями и средствами со смежных учебных курсов.

Потенциал рассматриваемой здесь предметной области (в сфере программирования) наиболее сильно раскрывается в прикладной деятельности и может рекомендоваться в качестве дополнения к дисциплинам:

«Эксплуатация информационных систем на транспорте/производстве (по отраслям)», «Инструментальные средства информационных систем (по отраслям)» по направлениям подготовки ВПО ОКУ «Бакалавр» 09.03.01 – Информатика и вычислительная техника, 09.03.02 – «Информационные системы и технологии», 15.03.04 – «Автоматизация технологических процессов и производств», 19.06.02 – «Эксплуатация транспортно-технологических машин и комплексов»,

Материалы издания безо всех ограничений могут быть использованы, дополнены инженерами и аспирантами, занимающимися задачами автоматизации технологических процессов, автоматизированного управления и мониторинга, синтеза дискретных систем, так как являются адаптированными версиями материалов (стандартов и иной документации), изложенных на открытых источниках. Автор данного учебного издания несет ответственность за корректность перевода и адаптации стандартов IEEE/IETF, которые на момент публикации не были русифицированы.

Содержание данного учебного курса было апробировано на рассматриваемых микропроцессорных устройствах в условиях эксперимента и тестирования.

§ ЯЗЫК ФУНКЦИОНАЛЬНЫХ БЛОКОВ ДИАГРАММ (FBD)

FBD (англ. Function Block Diagram) — графический язык программирования стандарта **МЭК 61131-3**.

ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р МЭК 61131-3— 2016

КОНТРОЛЛЕРЫ ПРОГРАММИРУЕМЫЕ

Часть 3

Языки программирования

(IEC 61131-3:2013, IDT)

Издание официальное

Стандарт **МЭК 61131-3** описывает 5 языков программирования, явившихся результатом изучения наиболее удачных фирменных разработок мировых лидеров рынка ПЛК. Языки ПЛК весьма оригинальны и существенно отличаются от известных языков программирования для компьютеров. Рассмотрим один из языков, получивший признание инженеров в области автоматизации технологических процессов: FBD.

Развитие идеи программной реализации электрических схем привело к появлению языка функциональных блоков и диаграмм (ФБД), FBD (Function Block Diagram).

Диаграмма FBD очень **напоминает принципиальную схему электронного устройства на микросхемах.** Выходы экземпляров функциональных блоков могут быть поданы на входы других блоков либо непосредственно на выходы ПЛК. Сами блоки, представленные на схеме как функциональные модули, могут выполнять стандартные и специальные функции.

FBD схемы ясно отражают взаимосвязь входов и выходов диаграммы. **Если алгоритм хорошо описывается с позиции сигналов, то его FBD-представление всегда получается значительно нагляднее, чем в текстовых языках.**

FBD предназначен для программирования программируемых логических контроллеров (ПЛК). Программа образуется из списка цепей, выполняемых последовательно сверху вниз. Цепи могут иметь метки.

Инструкция перехода на метку позволяет изменять последовательность выполнения цепей для программирования условий и циклов.

При программировании используются наборы библиотечных блоков и собственные блоки, также написанные на FBD или других языках **МЭК 61131-3.**

Блок (элемент) — это подпрограмма, функция или функциональный блок (И, ИЛИ, НЕ, триггеры, таймеры, счётчики, блоки обработки аналогового сигнала, математические операции и др.).

Каждая отдельная цепь представляет собой выражение, составленное графически из отдельных элементов. К выходу блока подключается следующий блок, образуя цепь.

Внутри цепи блоки выполняются строго в порядке их соединения. Результат вычисления цепи записывается во внутреннюю переменную либо подается на выход ПЛК (рис.1):

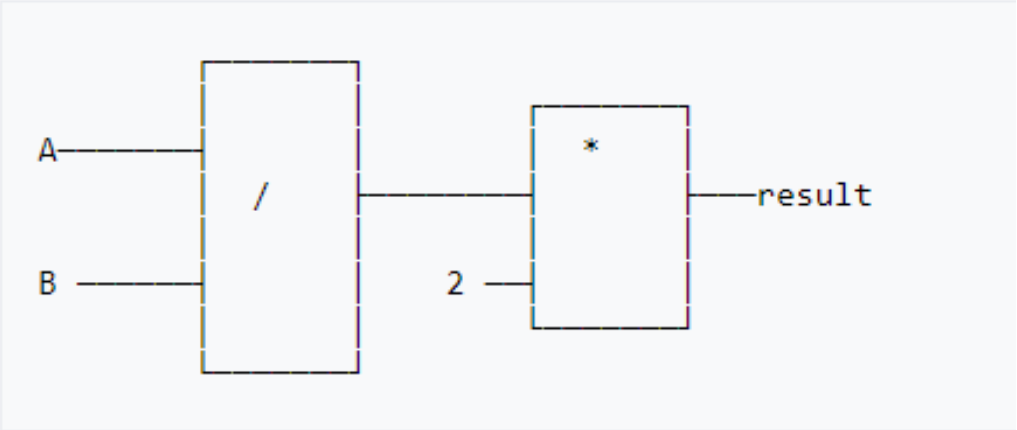


Рис. 1 — Пример фрагмента программы на FBD

Пример формализации исходного предиката: «A поделить на B, умножить на 2 и записать в переменную result» предоставлен выше. Та же самая функция на псевдокоде: result:= 2*A/B.

При необходимости управления вызовом блоков в них добавляются специальные входы EN (enable) и выходы ENO. Логический ноль на входе EN запрещает вызов блока. Выход ENO используется для индикации ошибки в блоке и позволяет прекратить вычисление остатка цепи.

Язык FBD прост в изучении, нагляден и удобен для прикладных специалистов, не имеющих специальной подготовки в области информатики. Жесткая последовательность выполнения приводит к простой внутренней структуре команд, которая транслируется в быстрый и надежный код. Существует много практических реализаций языка FBD с определенными расширениями или ограничениями.

Одним из вариантов FBD является язык программирования CFC (Continuous Function Chart). Он позволяет произвольно задавать порядок выполнения блоков. Диаграммы CFC дают программисту большую свободу действий, но платой за это является несколько большая вероятность допустить ошибку и более объемный код.

Язык функциональных блоковых диаграмм (FBD)
и его применение
Онлайн-журнал: "Электрик Инфо".

§ ОБЗОР ТИПОВ СРЕД ПРОГРАММИРОВАНИЯ СИСТЕМ

Среды программирования (среды разработки) — это программы, в которых программисты пишут свои программы.

Иными словами, среда программирования служит для разработки (написания) программ и обычно ориентируется на конкретный язык или несколько языков программирования (в этом случае языки, обычно, принадлежат одной языковой группе, например, Си-подобные).

Интегрированная среда программирования (IDE) содержит в себе все необходимое для разработки программ:

Редактор — в нем программист пишет текст программы, так называемый программный код;

Компилятор — он, как мы уже с вами знаем, транслирует программу, написанную на высокоуровневом языке программирования в машинный язык (машинный код), непосредственно понятный компьютеру. Язык С++ относится к компилируемым языкам, поэтому для обработки текстов его программ служит компилятор, иногда вместо компилятора (либо вместе с ним) используется интерпретатор, для программ, написанных на интерпретируемых языках программирования;

Отладчик — служит для отладки программ. Как мы все знаем, ошибки в программах допускают абсолютно все: и новички, и профессионалы - они могут быть синтаксическими (обычно они выявляются еще на стадии компиляции) и логическими. Для тестирования программы и выявления в ней логических ошибок служит отладчик.

Мы рассмотрели базовую комплектацию среды программирования, но иногда в них присутствуют еще и такие компоненты, как система управления версиями, различные инструменты для конструирования графического интерфейса программы, браузер классов, инспектор объектов и другие.

Рис. 2 — Состав системы программирования

Таким образом, инструментальная среда (система) программирования включает, прежде всего, текстовый редактор, позволяющий конструировать программы на заданном языке программирования, а также инструменты, позволяющие компилировать или интерпретировать программы на этом языке, тестировать и отлаживать полученные программы.

Кроме того, могут быть и другие инструменты, например, для статического или динамического анализа программ. Взаимодействуют эти инструменты между собой через обычные файлы с помощью стандартных возможностей файловой системы.

Различают следующие классы инструментальных сред программирования (см. рис.2):

· среды общего назначения,

· языково-ориентированные среды.

Инструментальные среды программирования общего назначения содержат набор программных инструментов, поддерживающих разработку программ на разных языках программирования (например, текстовый редактор, редактор связей или интерпретатор языка целевого компьютера) и обычно представляют собой некоторое расширение возможностей используемой операционной системы. Для программирования в такой среде на каком-либо языке программирования потребуются дополнительные инструменты, ориентированные на этот язык (например, компилятор).

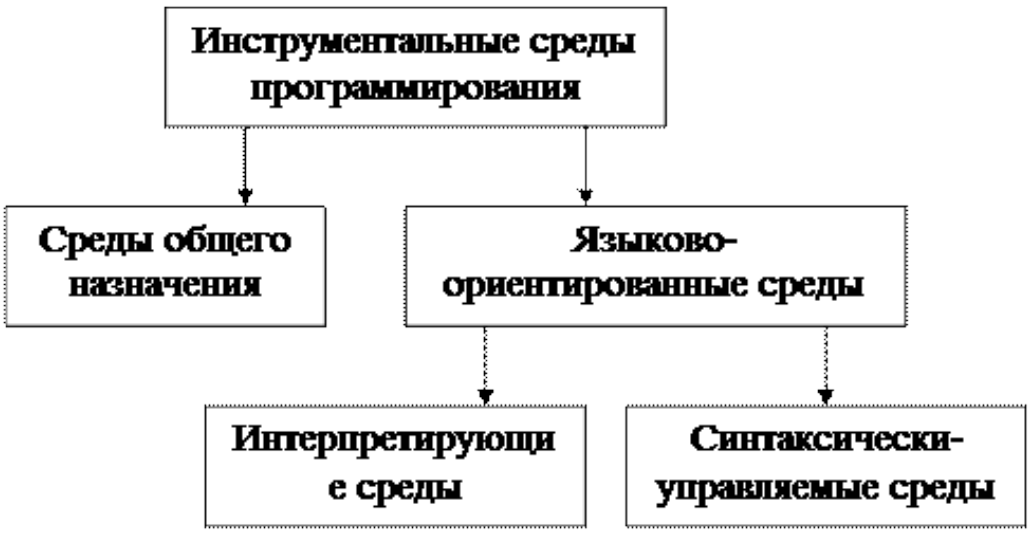


Рис. 3 — Классификация инструментальных сред программирования.

Языково-ориентированная инструментальная среда программирования предназначена для поддержки разработки ПС на каком-либо одном языке программирования и знания об этом языке существенно использовались при построении такой среды.

Вследствие этого в такой среде могут быть доступны достаточно мощные возможности, учитывающие специфику данного языка. Такие среды разделяются на два подкласса:

· интерпретирующие среды,
· синтаксически-управляемые среды.

Интерпретирующая инструментальная среда программирования обеспечивает интерпретацию программ на данном языке программирования, т.е. содержит, прежде всего, интерпретатор языка программирования, на который эта среда ориентирована. Такая среда необходима для языков программирования интерпретирующего типа (таких, как Лисп), но может использоваться и для других языков.

Синтаксически-управляемая инструментальная среда программирования базируется на знании синтаксиса языка программирования, на который она ориентирована. В такой среде вместо текстового используется синтаксически-управляемый редактор, позволяющий пользователю использовать различные шаблоны синтаксических конструкций (в результате этого разрабатываемая программа всегда будет синтаксически правильной). Одновременно с программой такой редактор формирует (в памяти компьютера) ее синтаксическое дерево, которое может использоваться другими инструментами.

В данном практикуме в основном мы будем использовать две среды общего назначения, **связь которых между собой** несет в себе характерные свойства интерпретирующих инструментальных сред, так и синтаксически-управляемых инструментальных сред (связь станет очевидной при комплексном анализе).

Основной средой разработки в практикуме станет IDE **FLProg**, результат работы в которой будет интерпретировано и транслировано в формальную форму, применяемую в другой среде разработки – **Arduino IDE**.

Arduino — это прежде всего торговая марка аппаратно-программных средств для построения простых систем автоматики и робототехники, ориентированная на начинающих пользователей.

Программная часть состоит из бесплатной программной оболочки (IDE) (об этом пойдет чуть позже) для написания программ, их компиляции и программирования аппаратуры. Аппаратная часть представляет собой набор смонтированных печатных плат, продающихся как официальным производителем, так и сторонними производителями. Полностью открытая архитектура системы позволяет свободно копировать или дополнять линейку продукции Arduino.

Arduino используется как для создания автономных объектов, так и подключения к программному обеспечению через проводные и беспроводные интерфейсы.

Среды программирования Arduino-совместимых плат можно разделить на следующие типы:

- Интегрированные среды разработки
- Графические среды, визуализирующие структуру кода.
- Графические среды, отображающие код в виде графики.
- Визуальные среды программирования, не использующие кода.

Рассмотрим каждый тип – начав по порядку:

Arduino IDE — интегрированная среда разработки для Windows, MacOS и Linux, разработанная на C и C ++, предназначенная для создания и загрузки программ на Arduino-совместимые платы, а также на платы других производителей.

Исходный код для среды выпущен под общедоступной лицензией GNU версии 2. Поддерживает языки Си и C ++ с использованием специальных правил структурирования кода.

Arduino IDE предоставляет библиотеку программного обеспечения из проекта Wiring, которая предоставляет множество общих процедур ввода и вывода. Для написанного пользователем кода требуются только две базовые функции для запуска эскиза и основного цикла программы, которые скомпилированы и связаны с заглушкой программы main () в исполняемую циклическую программу с цепочкой инструментов GNU, также включённой в дистрибутив IDE. Использует программу avrdude для преобразования исполняемого кода в текстовый файл в шестнадцатеричной кодировке, который загружается в плату Arduino программой-загрузчиком во встроенном программном обеспечении платы.

Проектирование программы для контроллера в ней происходит на языке Processing/Wiring, который является диалектом языка Си (скорее Си++). Эта среда представляет собой, по сути, обычный текстовый редактор с возможностью загрузки написанного кода в контроллер

Альтернативой Arduino IDE является среда разработки от производителя микроконтроллеров Atmel — AVRStudio.

Программирование в ней ведётся на чистом C, и она уже имеет намного больше возможностей и более похожа на серьёзные IDE (Visual Studio).

Эти два типа рассчитаны на опытных программистов, которые хорошо знают язык и могут с помощью них создавать серьёзные проекты.

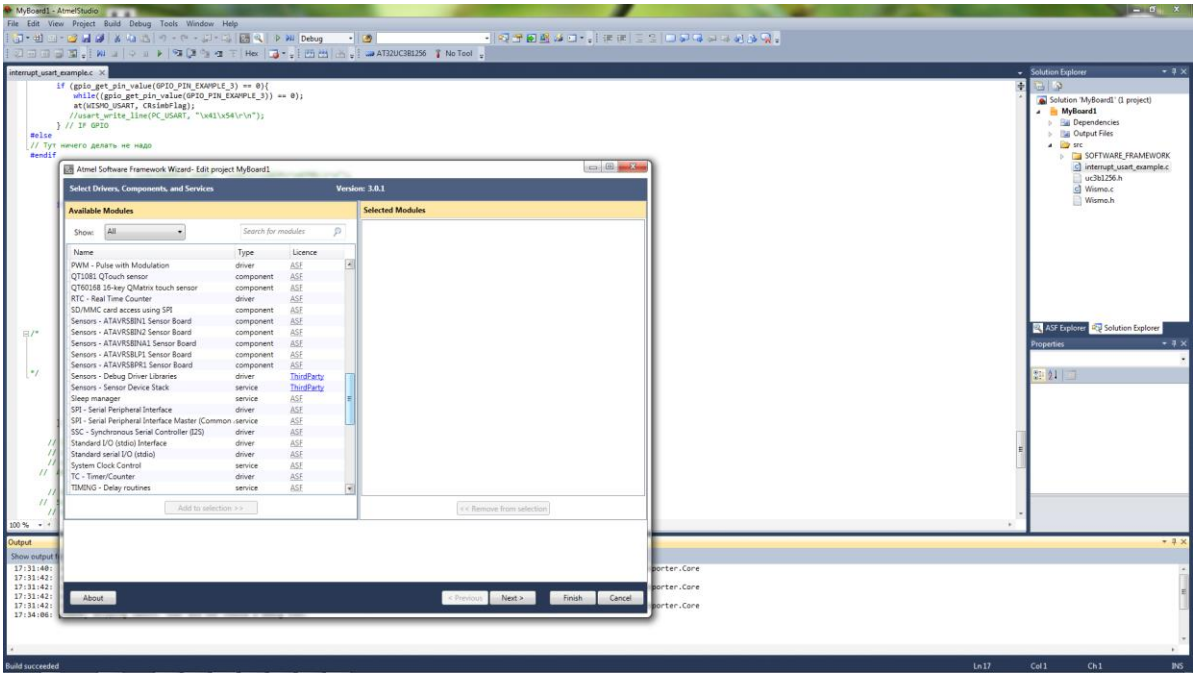


Рис. 4 — UI IDE AVRstudio

Графические среды, визуализирующие структуру кода.

Это программы, которые, по сути, являются расширением форматирования для обычного текстового редактора кода. В нем программа так же пишется на языке C, но в более удобном варианте. Сейчас таких сред очень много, самые яркие примеры: Scratch, S4A, Ardublock (рис.5).

Они очень хорошо подходят для начального обучения программированию на языке C, поскольку отлично показывают структуру и синтаксис языка.

Но для больших серьёзных проектов программа получается громоздкой.

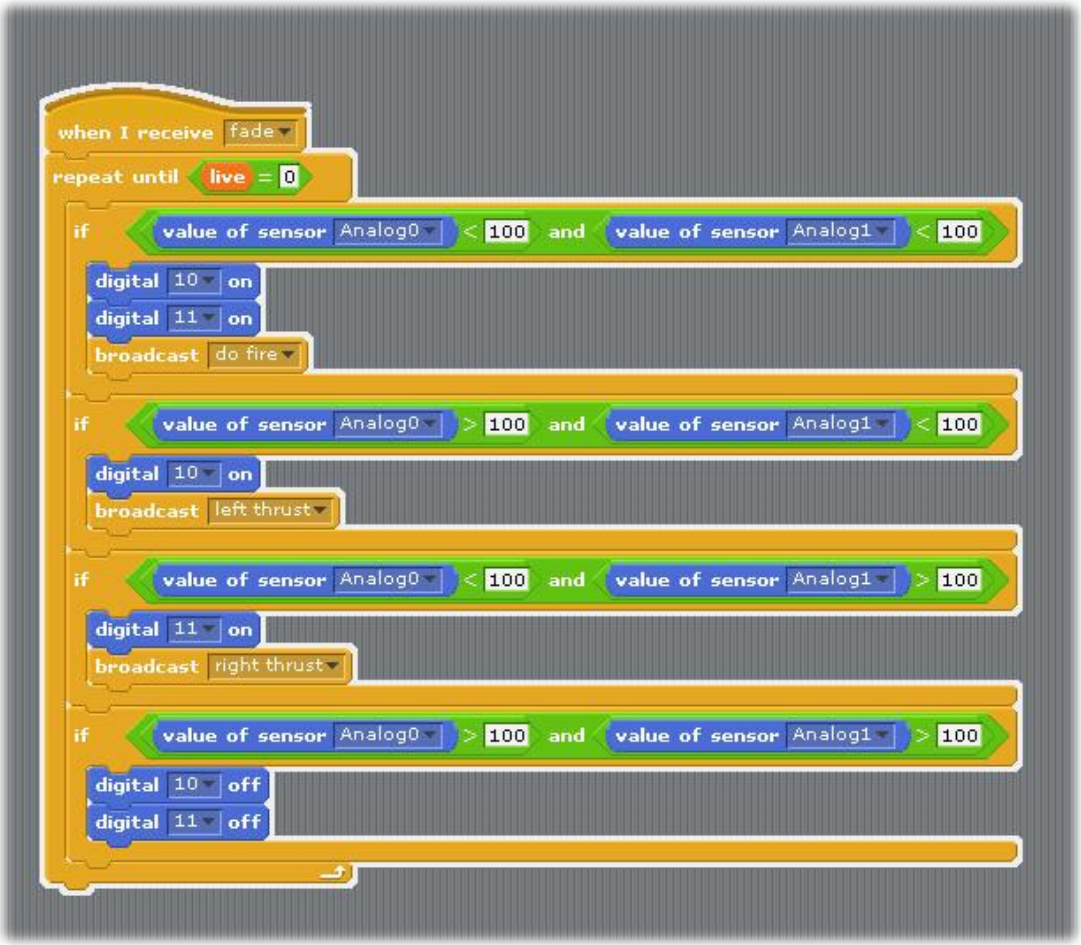


Рис. 5 — Графическая среда, визуализирующая структуру кода.

Графические среды, отображающие код в виде графики.

Это программы, скрывающие код и заменяющие его графическими аналогами. В них так же повторяется структура языка, формируются циклы, переходы, условия. Так же очень хорошо подходят для обучения построению алгоритмов, с последующим переходом на программирование на классических языках. И так же не подходят для построения больших проектов ввиду громоздкости получаемого отображения. Пример таких программ: **MiniBlog, Algorithm Builder, Flowcode.**

Описанные выше типы программ рассчитаны на full-stack программистов или на тех, кто решил изучать классическое программирование. Но для изготовления конечного устройства кроме непосредственно программирования контроллера обычно требуется разработка внешней обвязки платы, разработка и расчет силовой части, входных развязок и многого другого. С этим у программистов часто возникают проблемы. Зато с этим прекрасно справляются электрики и электронщики.

Но среди них мало программистов, которые смогли бы составить программу для контроллера. Сочетание программиста и электронщика – достаточно редкий случай. В результате такой ситуации реальных, законченных проектов на основе плат Arduino (да и других контроллеров) единицы. Для решения этой проблемы и служат программы последнего типа:

Визуальные среды программирования, не использующие кода.

Данные программы реализуют принцип, который уже много лет применяется практически всеми производителями контроллеров промышленного применения. Он заключается в создании программ для контроллера на языках FBD или LAD. Собственно говоря, как таковыми языками они не являются. Это, скорее, графические среды для рисования принципиальных или логических схем. Вспомним, что процессоры далеко не всегда были микропроцессорами, а создавались на базе цифровых микросхем. Поэтому тем, кто привык работать с цифровой техникой, больше понравится работа на них, чем написание кода на классических языках программирования. Примером таких программ являются проекты Horizont и **FLProg**. Программы этого типа хорошо подходят как для изучения построения импульсной и релейной техники, так и для создания серьезных проектов.

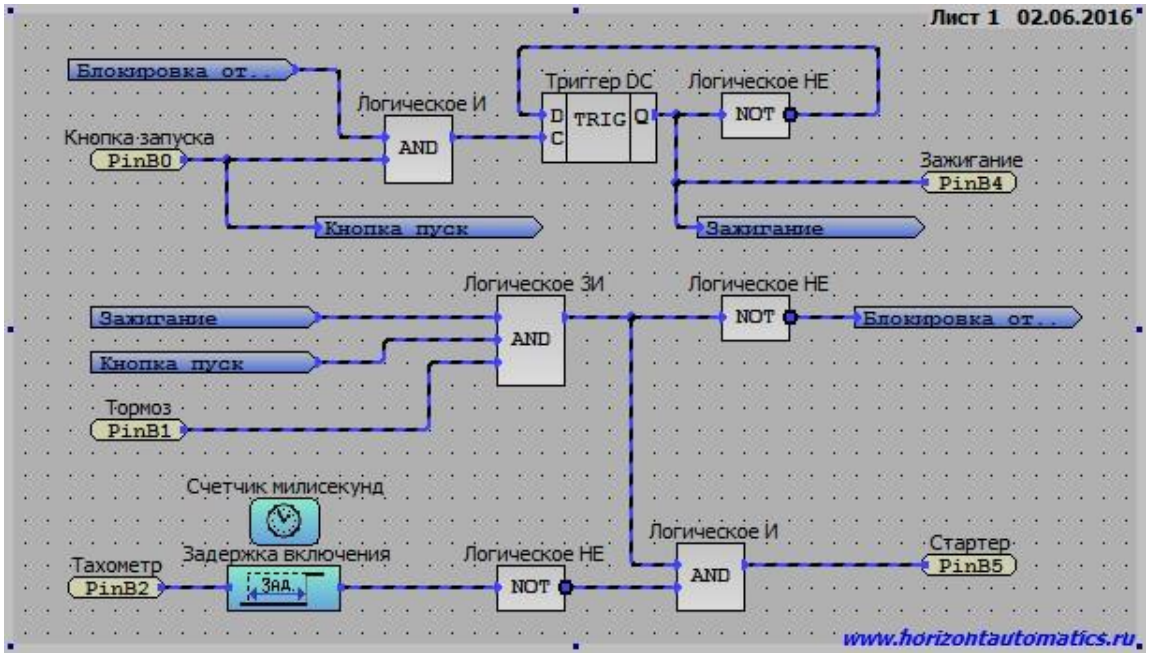


Рис. 6 — Horizont Configurator, визуальная среда построения алгоритмов работы устройств, в том числе и микроконтроллеров. объединяются друг с другом непосредственно линиями связи – графическими связями.

§ СРЕДА ВИЗУАЛЬНОГО ПРОГРАММИРОВАНИЯ FLProg

FLProg IDE — среда разработки, предназначенная для создания транслируемого в C/C++ и компилируемоего кода посредством методологии графического (визуального) программирования.

С помощью этой программы можно запрограммировать контроллер не зная текстовых языков программирования, а выглядит это как рисование электронной или электрической схемы.

Визуальные языки программирования FBD и Ladder, с помощью которых пишется программа, используются для программирования практически всех логических реле, и части промышленных контроллеров во всем мире.

Для создания FLProg был использован опыт программистов фирм Siemens, ABB, Schneider Electric и наработки в их средах программирования. При этом был несколько расширен классический функционал языков для работы с промышленными контроллерами путём добавления функциональных блоков, отвечающих за работу с внешними устройствами. Программа работает на компьютерах под управлением ОС Windows и Linux.

Пользовательский интерфейс FLProg устроен так, что проект представляет собой набор виртуальных плат, на каждой из которых собран законченный модуль разрабатываемой системы.

Каждая плата имеет наименование и снабжена комментариями. Для экономии места в рабочей зоне её можно свернуть, если работа над ней закончена, а при необходимости вновь развернуть и внести коррективы.

Разработанную «принципиальную схему» FLProg переводит на язык Processing/Wiring. По завершении компиляции автоматически открывается программа Arduino IDE с загруженным скетчем проекта. В Arduino IDE необходимо указать COM-порт компьютера, к которому подключён микроконтроллерный модуль, выбрать тип модуля и загрузить программу в его микроконтроллер.

Первый урок по работе с программой FLProg (обзор)
https://flprog.ru/uchebnyj-centr/articles/znakomstvo-s-flprog/pervyj-urok-po-rabote-s-programmoj-flprog/

§ ОБЗОР АППАРАТНО-ПРОГРАММНЫХ СРЕДСТВ ARDUINO

Петин В. А. — Проекты с использованием контроллера Arduino.

Появление первых микроконтроллеров ознаменовало начало новой эры в развитии микропроцессорной техники. Наличие в одном корпусе большинства системных устройств сделало микроконтроллер подобным обычному компьютеру. В отечественной литературе они даже назывались однокристальными микроЭВМ. Соответственно и желание использовать микроконтроллеры как обычные компьютеры появилось практически с их появлением.

Но желание это сдерживалось многими факторами. Например, чтобы собрать устройство на микроконтроллере, необходимо знать основы схемотехники, устройство и работу конкретного процессора, уметь программировать на ассемблере и изготавливать электронную технику. Потребуются также программаторы, отладчики и другие вспомогательные устройства.

В итоге без огромного объема знаний и дорогостоящего оборудования не обойтись. Такая ситуация долго не позволяла многим использовать микроконтроллеры в своих проектах. Сейчас, с появлением устройств, дающих возможность работать с микроконтроллерами без наличия серьезной материальной базы и знания многих предметов, все изменилось. Примером такого устройства может служить проект Arduino итальянских разработчиков.

Arduino и его клоны представляют собой наборы, состоящие из готового электронного блока и программного обеспечения. Электронный блок здесь — это печатная плата с установленным микроконтроллером и минимумом элементов, необходимых для его работы.

Фактически электронный блок является аналогом материнской платы современного компьютера. На нем имеются разъемы для подключения внешних устройств, а также разъем для связи с компьютером, по которому и осуществляется программирование микроконтроллера. Особенности используемых микроконтроллеров ATmega фирмы Atmel позволяют производить программирование без применения специальных программаторов. Все, что нужно для создания нового электронного устройства, — это плата Arduino, кабель связи с компьютером.

Второй частью проекта Arduino является программное обеспечение для создания управляющих программ.

Оно объединило в себе простейшую среду разработки и язык программирования, представляющий собой вариант языка C/C++ для микроконтроллеров. В него добавлены элементы, позволяющие создавать программы без изучения аппаратной части. Так что для работы с Arduino практически достаточно знания только основ программирования на C/C++. Создано для Arduino и множество библиотек, содержащих код, работающий с различными устройствами.

В чем преимущество Arduino?

Пользователь современного компьютера не задумывается о функционировании отдельных частей ПК. Он просто запускает нужные программы и работает с ними.

Точно так же и Arduino позволяет пользователю сосредоточиться на разработке проектов, а не на изучении устройства и принципов функционирования отдельных элементов. Нет надобности и в создании законченных плат и модулей. Разработчик может использовать готовые платы расширения или просто напрямую подключить к Arduino необходимые элементы. Все остальные усилия будут направлены на раз работку и отладку управляющей программы на языке высокого уровня.

В итоге доступ к разработке микропроцессорных устройств получили не только профессионалы, но и просто любители что-то сделать своими руками. Наличие готовых модулей и библиотек программ позволяет непрофессионалам в электронике создавать готовые работающие устройства для решения своих задач. А варианты использования Arduino ограничены только возможностями микроконтроллера и имеющегося варианта платы, ну и, конечно, фантазией разработчика.

История создания Arduino

В 2002 году программист Массимо Банци (Massimo Banzi) был принят на работу в должности доцента в Институт проектирования взаимодействий города Иврея (Interaction Design Institute Ivrea, IDII) для продвижения новых способов разработки интерактивных проектов. Однако крошечный бюджет и ограниченное время доступа к лабораторной базе сводили его усилия практически на нет.

В проектах Банци использовал устройство BASIC Stamp, разработанное калифорнийской компанией Parallax. Stamp представлял собой небольшую печатную плату с размещенными на ней источником питания, микроконтроллером, памятью и портами ввода/вывода для соединения с различной аппаратурой. Программирование микроконтроллера осуществлялось на языке BASIC.

BASIC Stamp имел две проблемы: недостаток вычислительной мощности и достаточно высокую цену — плата с основными компонентами стоила около 100 долларов. И команда Банци решила самостоятельно создать плату, которая удовлетворяла бы всем их потребностям.

Банци и его сотрудники поставили себе целью создать устройство, представляющее собой простую, открытую и легкодоступную платформу для разработки, с ценой — не более 30 долларов — приемлемой для студенческого кармана. Хотели они и выделить чем-то свое устройство на фоне прочих.

Поэтому в противовес другим производителям, экономящим на количестве выводов печатной платы, они решили добавить их как можно больше, а также сделали свою плату синей, в отличие от обычных зеленых плат. Продукт, который создала команда, состоял из дешевых и доступных компонентов — например, базировался он на микроконтроллере ATmega328. Но главная задача состояла в том, чтобы гарантировать работу устройства по принципу plug-and play, чтобы пользователь, достав плату из коробки и подключив к компьютеру, мог немедленно приступить к работе.

Первый прототип платы был сделан в 2005 году, она имела простейший дизайн и еще не называлась Arduino. Чуть позже Массимо Банци придумал назвать ее так — по имени принадлежащего ему бара, расположенного в городе Ивреа. Бренд "Arduino" без какой-либо рекламы и привлечения средств маркетинга быстро приобрел высокую популярность в Интернете.

С начала распространения продано более 250 тыс. комплектов Arduino, и это, не учитывая множества клонов. В мире насчитывается более двухсот дистрибьюторов продукции Arduino — от крупных фирм, таких как SparkFun Electronics, до мелких компаний, работающих на местный рынок.

На сегодня платформа Arduino представлена не одной платой, а целым их семейством.

В дополнение к оригинальному проекту, называемому Arduino Uno, новые модели, имеющие на плате более мощные средства, носят название Arduino Mega, компактные модели — Arduino Nano, платы в водонепроницаемом корпусе — LilyPad Arduino, а новая плата с 32-разрядным процессором Cortex-M3 ARM — Arduino Due.

Своим успехом проект Arduino обязан существовавшим до него языку Processing и платформе Wiring. От этих проектов Arduino унаследовал одну сильную черту — удобную для пользователя среду разработки.

Due — плата на базе 32-битного ARM микропроцессора Cortex-M3 ARM SAM3U4E;

Leonardo — плата на микроконтроллере ATmega32U4;

Uno — самая популярная версия базовой платформы Arduino;

Duemilanove — плата на микроконтроллере ATmega168 или ATmega328;

Diecimila — версия базовой платформы Arduino USB;

Nano — компактная платформа, используемая как макет. Nano подключается к компьютеру при помощи кабеля USB Mini-B;

Mega ADK — версия платы Mega 2560 с поддержкой интерфейса USB-host для связи с телефонами на Android и другими устройствами с интерфейсом USB;

Mega2560 — плата на базе микроконтроллера ATmega2560 с использованием чипа ATMega8U2 для последовательного соединения по USB-порту;

Mega — версия серии Mega на базе микроконтроллера ATmega1280;

Arduino BT — платформа с модулем Bluetooth для беспроводной связи и программирования;

LilyPad — платформа, разработанная для переносимых изделий, может зашиваться в ткань;

Fio — платформа разработана для беспроводных применений. Fio содержит разъем для радио XBee, разъем для батареи LiPo и встроенную схему подзарядки;

Mini — самая маленькая платформа Arduino;

Pro — платформа, разработанная для опытных пользователей, может являться частью большего проекта;

Pro Mini — как и платформа Pro, разработана для опытных пользователей, которым требуется низкая цена, меньшие размеры и дополнительная функциональность.

Рассмотрим более подробно некоторые из этих плат.

Arduino Uno R3 — флагманская платформа для разработки на базе микроконтроллера ATmega328P.

На Arduino Uno предусмотрено всё необходимое для удобной работы с микроконтроллером: 14 цифровых входов/выходов (6 из них могут использоваться в качестве ШИМ-выходов), 6 аналоговых входов, кварцевый резонатор на 16 МГц, разъём USB, разъём питания, разъём для внутрисхемного программирования (ICSP) и кнопка сброса.

Микроконтроллер ATmega328P

Сердцем платформы Arduino Uno является 8-битный микроконтроллер семейства AVR — ATmega328P. А микроконтроллер ATmega16U2 обеспечивает связь микроконтроллера ATmega328P с USB-портом компьютера. При подключении к ПК Arduino Uno определяется как виртуальный COM-порт. Прошивка микросхемы 16U2 использует стандартные драйвера USB-COM, поэтому установка внешних драйверов не требуется.

Порты ввода/вывода

Цифровые входы/выходы: пины 0–13;

Логический уровень единицы — 5 В, нуля — 0 В.

Максимальный ток выхода — 40 мА. К контактам подключены подтягивающие резисторы, которые по умолчанию выключены, но могут быть включены программно.

ШИМ: пины 3,5,6,9,10 и 11

Позволяют выводить 8-битные аналоговые значения в виде ШИМ-сигнала.

АЦП: пины A0–A5

6 аналоговых входов, каждый из которых может представить аналоговое напряжение в виде 10-битного числа (1024 значений). Разрядность АЦП — 10 бит.

TWI/I²C: пины SDA и SCL

Для общения с периферией по синхронному протоколу, через 2 провода. Для работы — используйте библиотеку Wire.

SPI: пины 10(SS), 11(MOSI), 12(MISO), 13(SCK).

Через эти пины осуществляется связь по интерфейсу SPI. Для работы — используйте библиотеку SPI.

UART: пины 0(RX) и 1(TX)

Эти выводы соединены с соответствующими выводами микроконтроллера ATmega16U2, выполняющей роль преобразователя USB-UART. Используется для коммуникации платы Arduino с компьютером или другими устройствами через класс Serial.

Имя светодиода	Назначение
RX и TX	Мигают при обмене данными между Arduino Uno и ПК.
L	Светодиод вывода `13`. При отправке значения HIGH светодиод включается, при отправке LOW – выключается.
ON	Индикатор питания на плате.

Рис. 7 — Светодиодная индикация на плате Arduino UNO R3

Характеристики Arduino UNO R3

Микроконтроллер: ATmega328
Тактовая частота: 16 МГц
Напряжение логических уровней: 5 В
Входное напряжение питания: 7–12 В
Портов ввода-вывода общего назначения: 20
Максимальный ток с пина ввода-вывода: 40 мА
Максимальный выходной ток пина 5V: 800 мА
Портов с поддержкой ШИМ: 6 Портов, подключённых к АЦП: 6 Разрядность АЦП: 10 бит Flash-память: 32 КБ EEPROM-память: 1 КБ Оперативная память: 2 КБ Габариты: 69×53 мм

Arduino Due — плата микроконтроллера на базе процессора Atmel SAM3X8E ARM Cortex-M3.

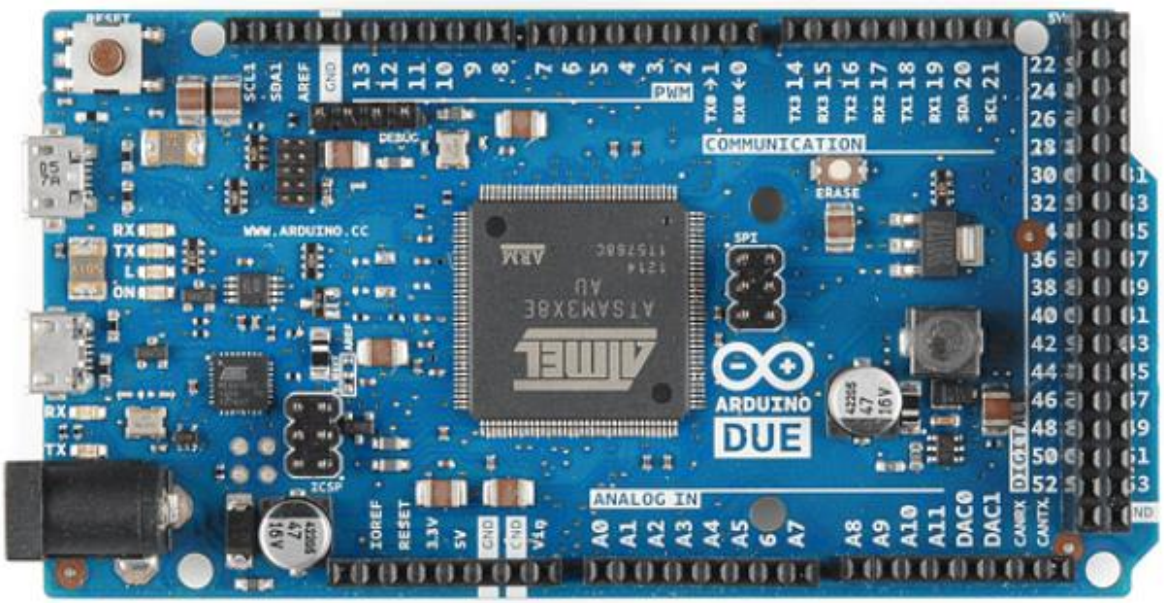


Рис. 8 — Arduino Due

Это первая плата Arduino на основе 32-битного микроконтроллера с ARM ядром. На ней имеется 54 цифровых вход/выхода (из них 12 можно задействовать под выходы ШИМ), 12 аналоговых входов, 4 UARTa (аппаратных последовательных порта), а генератор тактовой частоты 84 МГц, связь по USB с поддержкой OTG, 2 ЦАП (цифро-аналоговых преобразователя), 2 TWI, разъем питания, разъем SPI, разъем JTAG, кнопка сброса и кнопка стирания.

Arduino Due работает от 3,3 В. Максимальное напряжение, которое выдерживают вход/выходы составляет 3,3 В. Подав более высокое напряжение, например, 5 В, на выводы Arduino Due, можно повредить плату.

Плата содержит все, что необходимо для поддержки микроконтроллера. Чтобы начать работу с ней, достаточно просто подключить её к компьютеру кабелем микро-USB, либо подать питание с AC/DC преобразователя или батарейки. Due совместим со всеми платами расширения Arduino, работающими от 3,3 В, и с цоколевкой Arduino 1.0.

Порты ввода/вывода

В отличие от других плат Arduino, Arduino Due работает от 3,3 В. Максимальное напряжение, которое могут выдержать вход/выходы составляет 3,3 В. Подав напряжение, например: 5 В, на выводы Arduino Due, можно вывести плату из строя.

Цифровые входы/выходы: контакты 0–53. Работают на напряжении 3,3 В. В режиме выхода могут выдавать ток 3 или 15 мА (в зависимости от контакта); в режиме входа — принимать ток 6 или 9 мА (в зависимости от контакта). К контактам также подключены подтягивающие резисторы по 100 кОм, которые по умолчанию выключены, но могут быть включены программно.

Аппаратные последовательные порты (RX/TX): 0/1, 19/18, 17/16, 15/14. Передача данных осуществляется на уровне 3,3 В. Первая пара также соединена с чипом ATmega16U2, отвечающим за подключение через USB к компьютеру.

Широтно-имульсная модуляция (ШИМ/PWM): контакты 2–13. Дают возможность выдавать аппаратный шим с разрешением 8 бит (256 градаций).

SPI — отдельная группа контактов 2×3. На Arduino Due используется только для общения по SPI-интерфейсу с другими устройствами. Он не может быть использован для программирования контроллера, как на других Arduino. По расположению он в точности совпадает с расположением на Arduino Uno, Arduino Mega 2560, Arduino Leonardo, а следовательно даёт возможность работы с платами расширения его использующими, таких как Ethernet Shield.

CAN-шина: контакты CANRX и CANTX. Позволяют использовать Arduino Due в промышленных сетях. Поддержка с программной стороны пока не реализована производителем.

Встроенный светодиод: контакт 13 (L). Для простой индикации. В отличии от Arduino Uno и Mega, он поддерживает ШИМ.

Шины **TWI/I²C**: 20(SDA)/21(SCL), SDA1/SCL1. Для общения с периферией по синхронному протоколу, через 2 провода.

Аналоговые входы: контакты A0–A11. Принимают сигнал до 3,3 В. Большее напряжение может вывести процессор из строя. Аналоговые входы предоставляют разрешение до 12 бит (4096 градаций), хотя по умолчанию настроены на разрешение в 10 бит для совместимости со скетчами для других моделей Arduino.

Цифро-аналоговый преобразователь: контакты DAC1 и DAC2. Позволяют выдавать настоящий аналоговый сигнал с 12-битным разрешением (4096 градации), например, для устройств, связанных с обработкой звука.

Сброс процессора: RESET. Позволяет аппаратно перезагружать плату.

Входное напряжение: Vin. Выдаёт напряжение, поданное внешним источником, либо может являться входом для внешнего питания.

Стабилизированные 5 В: контакт 5V. Позволяет получать ровные 5 В и ток до 800 мА.

Стабилизированные 3,3 В: контакт 3.3V. Позволяет получать ровные 3,3 В и ток до 800 мА.

Общая земля: GND.

Опорное напряжение для плат расширения: IOREF. Платы расширения должны «советоваться» с этим контактом, чтобы правильно определять родное напряжение родительской платы. Arduino Due выдаёт на IOREF 3,3 В.

Характеристики Arduino Due

Микроконтроллер: AT91SAM3X8E
Тактовая частота: 84 МГц
Портов ввода-вывода общего назначения: 54
Максимальный ток с пина ввода-вывода: 3 или 15 мА (в зависимости от вывода)
Максимальный ток с пина ввода-вывода: 40 мА
Максимальный выходной ток пина 5V: 800 мА
Портов с поддержкой ШИМ: 12 Портов, подключённых к АЦП: 12 Разрядность АЦП: 12 бит Flash-память: 512 КБ Оперативная память: 96 КБ Габариты: 101×53 мм

http://arduino.ru/Hardware/ArduinoDue

§ ПРАКТИКУМ: ПЕРВЫЙ ПРОЕКТ В FLPROG и ARDUINO IDE

Оборудование: ПК, кабель AM/microBM 5p Cablexpert Pro (CCP-mUSB2-AMBM-1.0M или аналоги), Arduino UNO R3.

Программное обеспечение: IDE FLProg, IDE Arduino, интерактивный справочник (exeBook) FLProg_Старт.exe

Цель: ознакомиться с интерфейсом инструментальных средств, настроив их для работы соответствующим образом, получить навыки работы в интегрированных средах разработки, получить готовый к реализации FBD-скетч (прошивку), запрограммировать микропроцессорную систему.

Ход работы:

1. Открыть руководство FLProg_старт (рис.9) и программу FLProg. Изучить содержимое руководства (стр. 57 – 67).
2. Выполнить по инструкции содержание урока (1) (стр. 67 – 85). В качестве отчета предоставить видео-ролик продолжительностью до 25 секунд с демонстрацией работы устройства.

- Качество – не менее 720p.
- Платформа размещения – youtube.
- При размещении видео-ролика, демонстрирующего работу **светодиода определенного цвета** по заданному сценарию, не **соответствующему выбранному** студентом/преподавателем на начальном этапе работа не принимается.

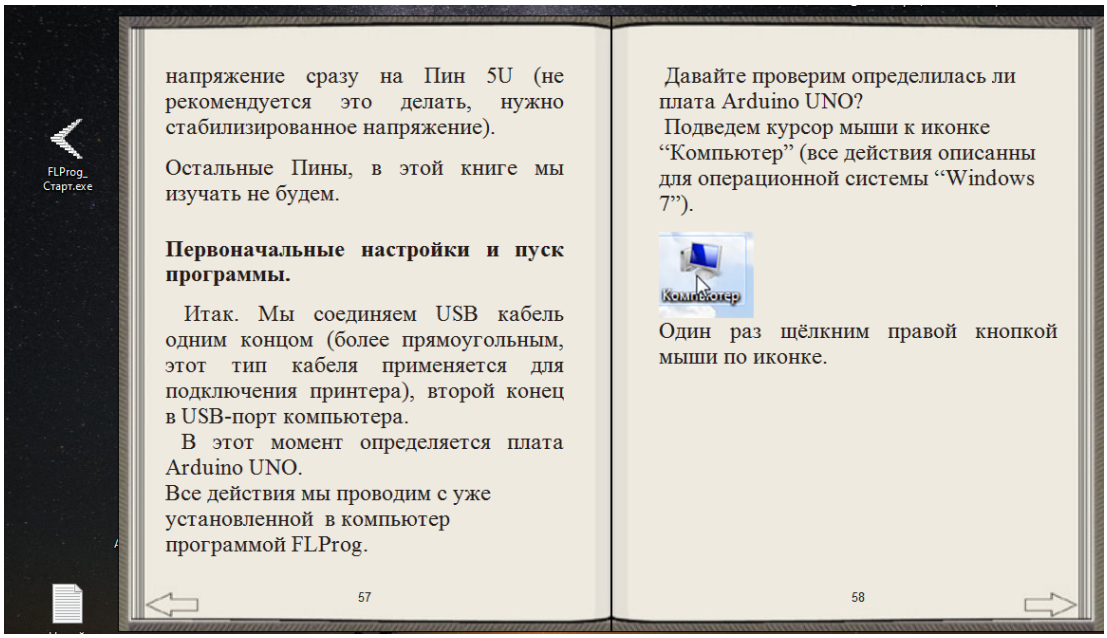


Рис. 9 — FLProg_старт.

3. У программистов в качестве первого урока принято использовать“Hello World”, у программистов микроконтроллеров помигать светодиодом (что мы уже сделали), а у электриков и электронщиков собрать схему управления контактором. Поскольку основными пользователями программы FLProg как раз они и являются, собирать на первом уроке будем как раз данную схему.

Контактор (лат. contāctor «соприкасатель») — двухпозиционный электромагнитный аппарат, предназначенный для частых дистанционных включений и выключений силовых электрических цепей в нормальном режиме работы. **Контактор** является разновидностью **электромагнитного** реле — важнейшего переключательного компонента во всей **электронике**, в том числе в **IoT**.

Принцип работы контактора любой модели заключается в том, что группа подвижных контактов постоянно сходится и расходится с неподвижными фиксированными контактами, пропуская и ограничивая течение электрического тока. Можно показаться, что это простой переключатель, но с рядом особенностей.

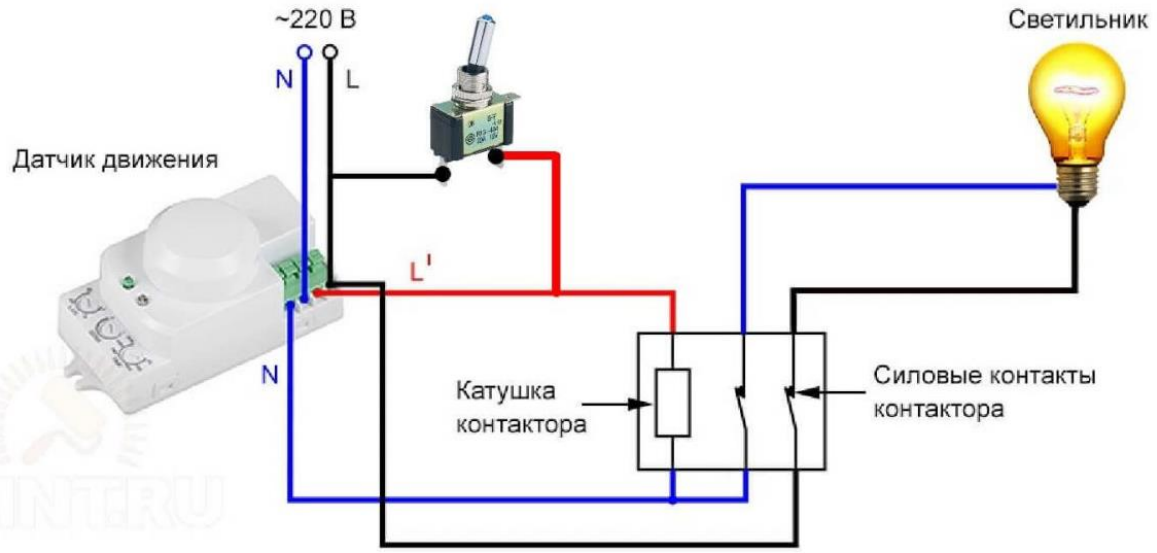


Рис. 11 — Схема управления освещением на контакторе

Первое и самое главное — для обеспечения безопасности нормальное положение контактных групп (режим покоя) разомкнутое. В таких приспособлениях нет никаких механических функций для воздействия на контакты, чтобы они всегда были во включенном состоянии. Они смыкаются лишь тогда, когда на них подают напряжение.

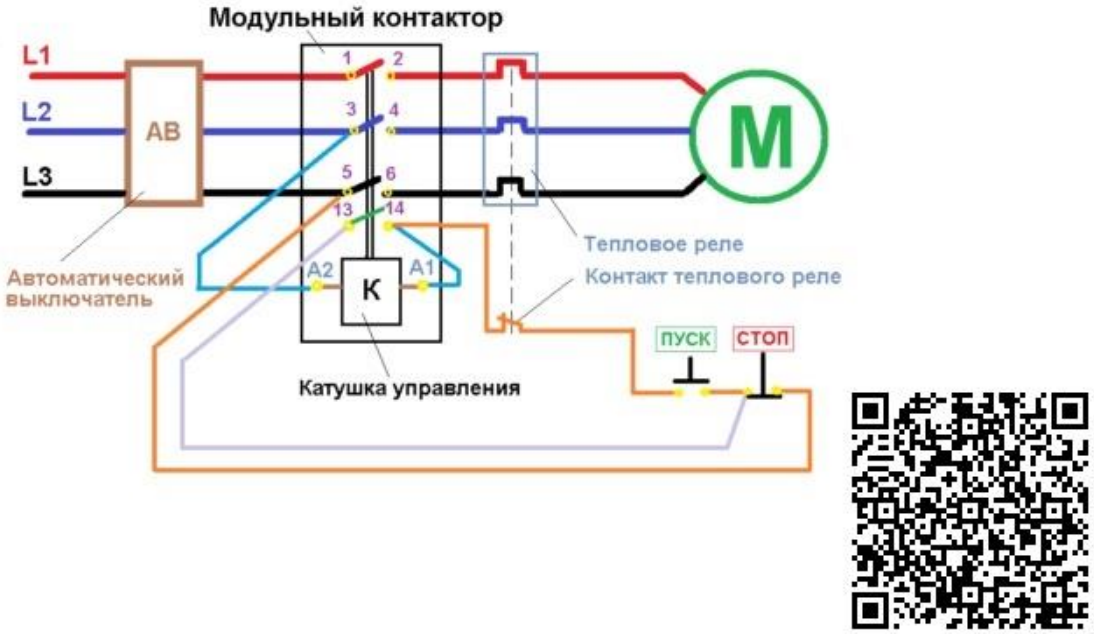


Рис. 11 — Стандартная схема управление контактором

Заменим эту схему контроллером Ардуино. Оставим в стороне вопросы помехозащищённости и экранирования устройства. Наша цель — создать в программе FLProg соответствующую дискретную логику. Поэтому оформим схему подключения.

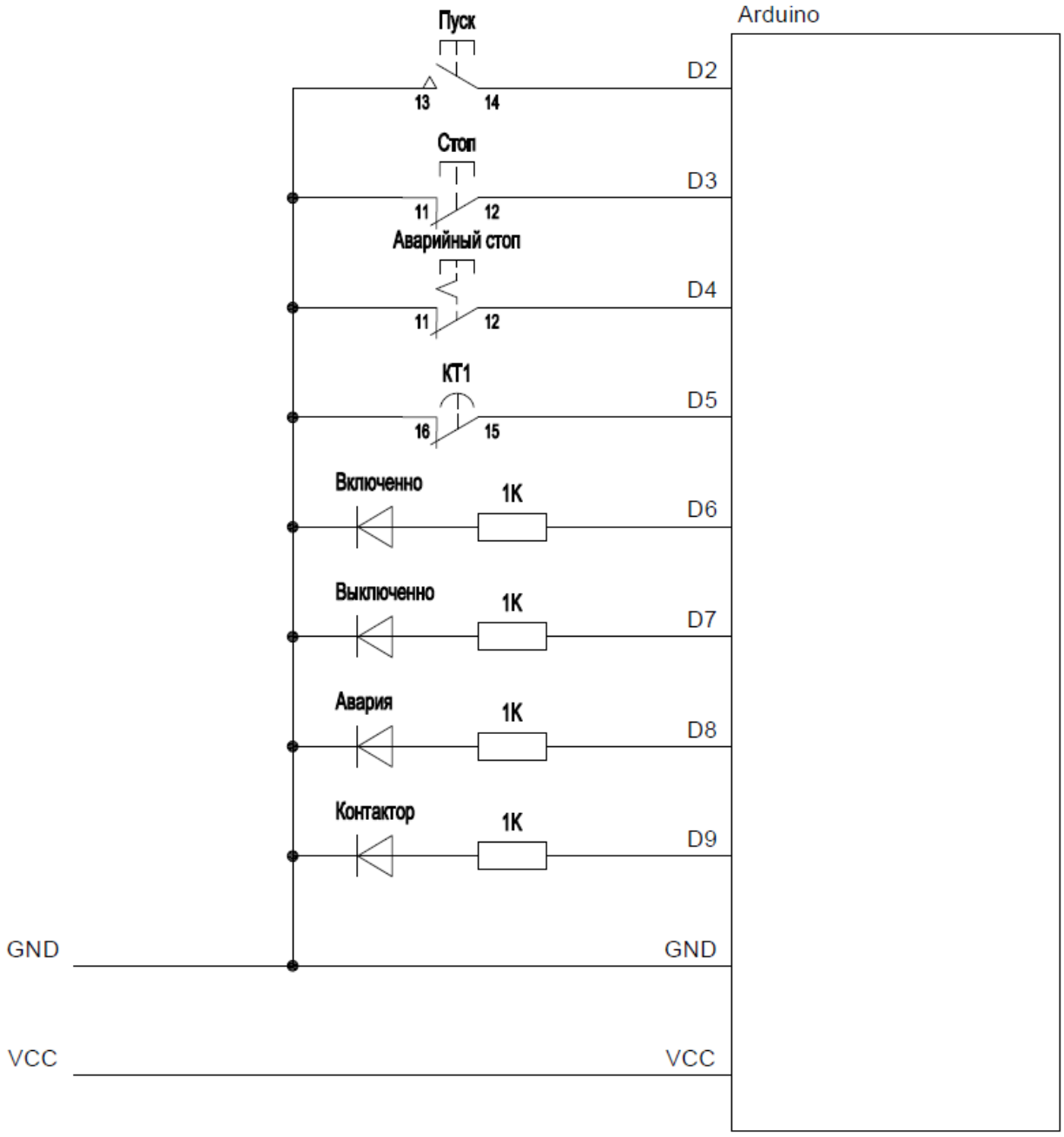


Рис. 11 — Тестовая схема управления под Arduino

Разберем схему чуть подробнее: D0…D9 – цифровые порты выбранной платы. Под обозначением 1К мы подразумеваем резисторы с таким номиналом. Индикация на портах вывода D6…D9 светодиодная, о чем свидетельствует надписи. GND – заземление. VCC – питание.

То есть, роль контактора в данной тестовой схеме выполняет светодиод «Контактор». Теперь попробуем запрограммировать контроллер.

Запускаем программу FLProg, нажимаем кнопку «Создать новый проект».

Откроется окно выбора контроллера и языка программирования проекта.

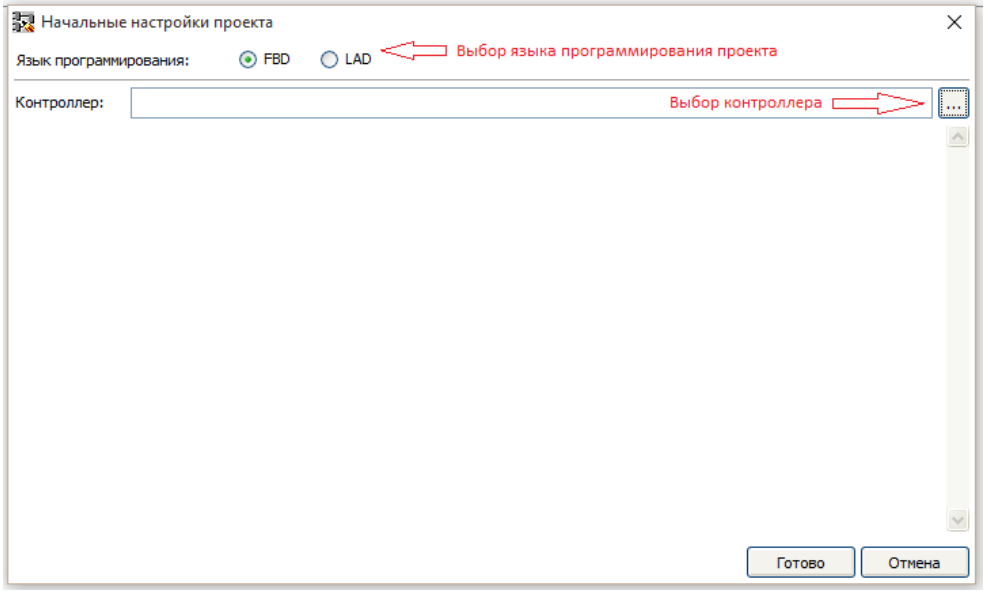


Рис. 12 — Начальные настройки проекта

Для создания проекта можно использовать любой из двух языков программирования (FBD и LAD) являющимися стандартами в области программирования промышленных контроллеров. **Выбираем язык FBD и контроллер Arduino UNO.**

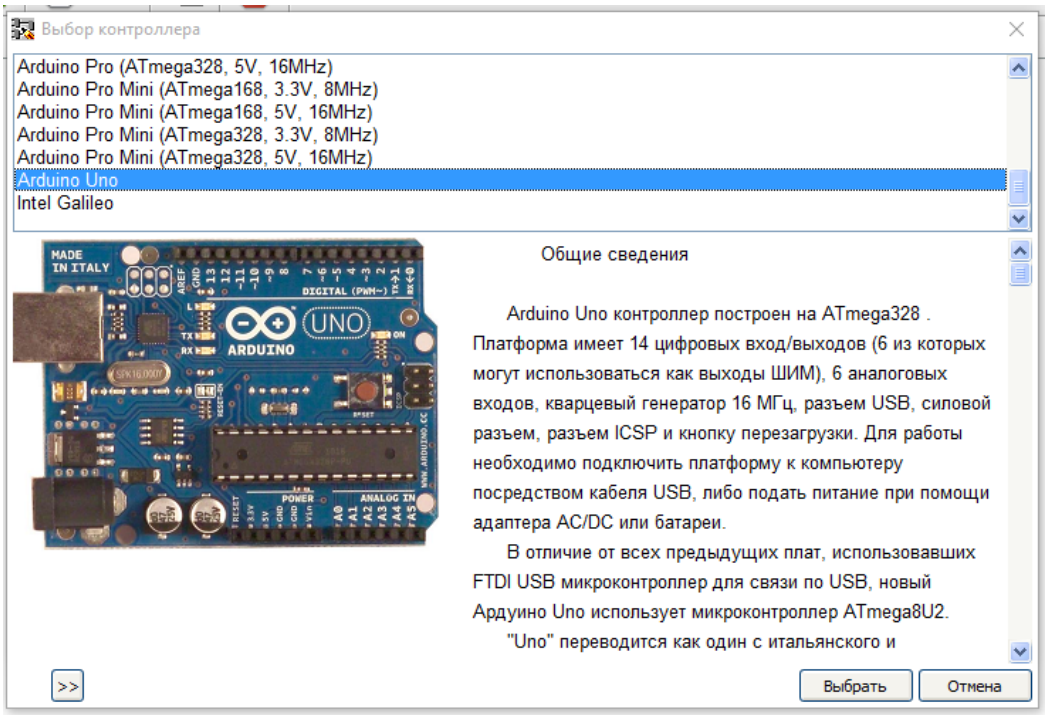


Рис. 13 — Окно выбора контроллера в FLProg

Рабочее окно программы FLProg (рис.14) на языке FBD состоит из нескольких полей:

1. Основное меню программы
2. Дерево проекта
3. Дерево установленного оборудования (**дерево тэгов**, если речь идет о разработке в FBD). В нём представлено оборудование (промежуточные реле, реле времени, генераторы…), которое используется в проекте. В новом проекте в нём присутствуют только входы и выходы контроллера.
4. Библиотека блоков. В ней находится оборудование, которое возможно применить в проекте. В данном уроке нас будет интересовать только папка «Базовые блоки»
5. Область схемы, в которой и будет собственно рисоваться схема. Схема в FLProg представляет собой набор плат с оборудованием.

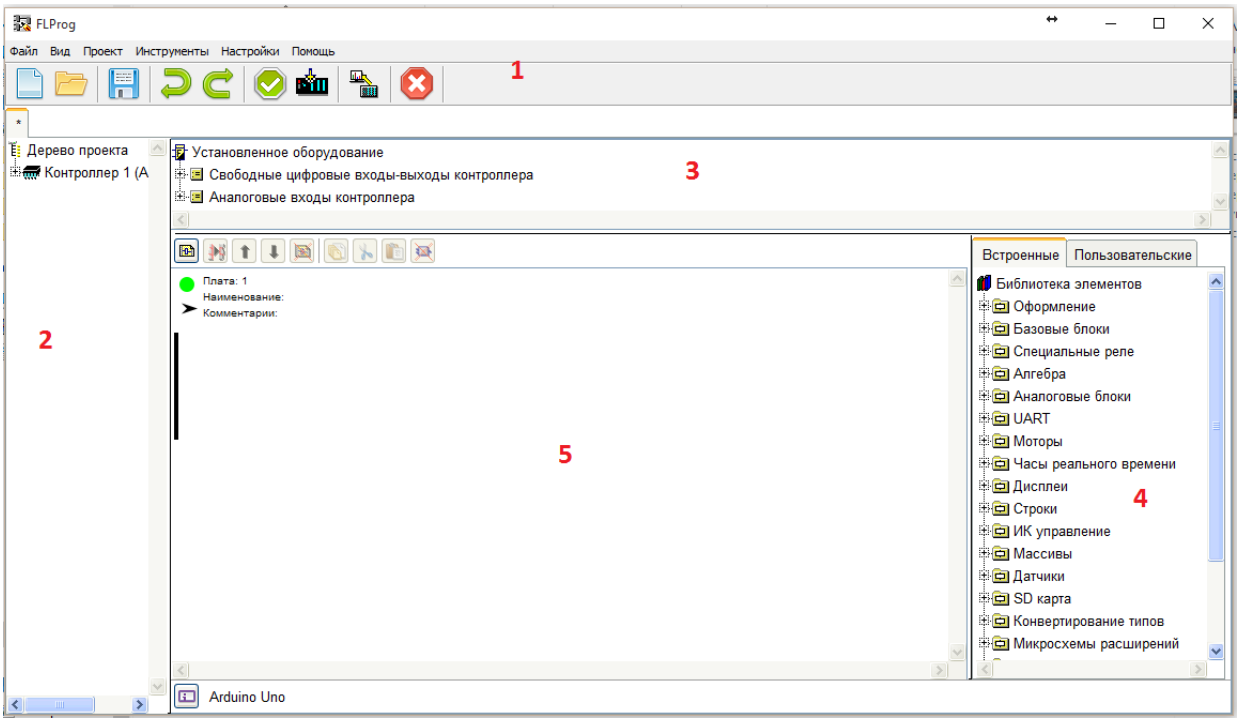


Рис. 14 — Интерфейс IDE FLProg

Для начала вытащим на область схемы контакты кнопок. Это возможно сделать двумя путями.

Перетащить соответствующий вход из папки (рис.15) «Свободные входы–выходы контроллера» дерева установленного оборудования на область схемы:

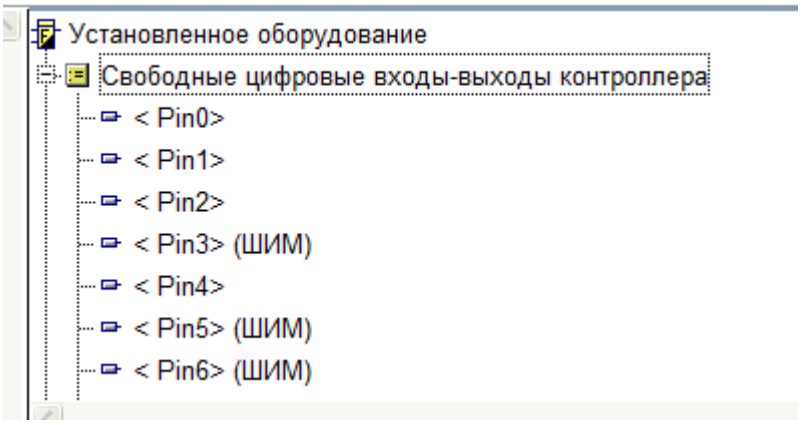


Рис. 15 — «Свободные входы–выходы контроллера»

Или (рис.16) перетащив блок «Контакт» из папки «Базовые элементы» библиотеки блоков:

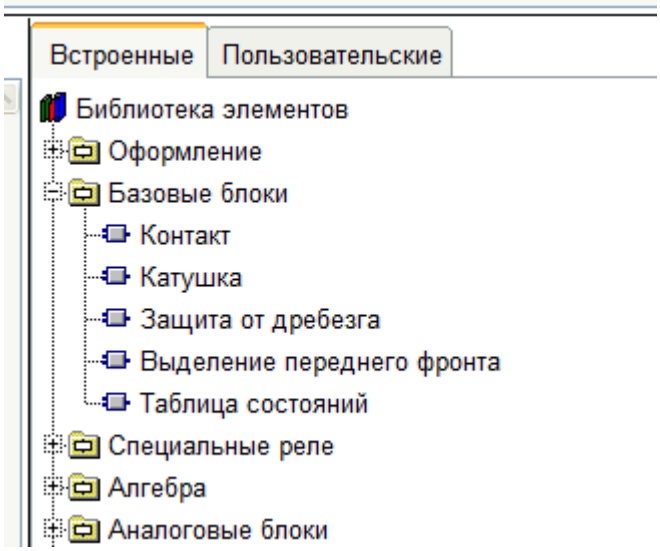


Рис. 16 — «Библиотека элементов»

В результате на схеме появится УГО (условно – графическое обозначение) контакта. В случае перетаскивания его из дерева установленного оборудования контакт окажется сразу привязанным к цифровому входу – выходу платы.

Если блок контакта был вытащен из библиотеки элементов, он будет абстрактным контактом без какой – либо привязки (рис.17):

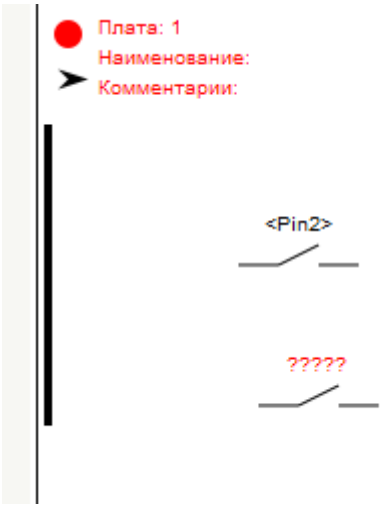


Рис. 17 — Абстрактный контакт

И любом случае контакты необходимо параметрировать. Для этого делаем двойной клик на контакте. Открывается окно редактирования блока (рис.18):

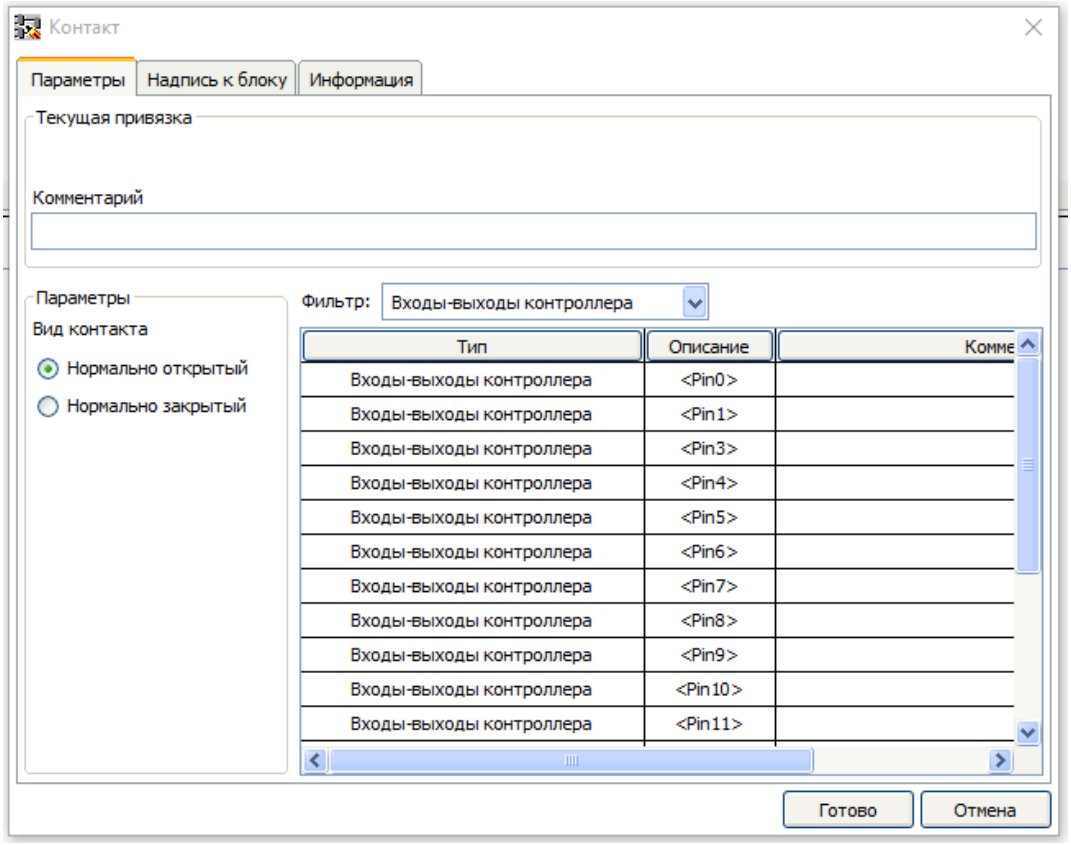


Рис. 18 — Параметризация элемента (контакта)

Вернее к заданию – в первую очередь «создадим» входы:

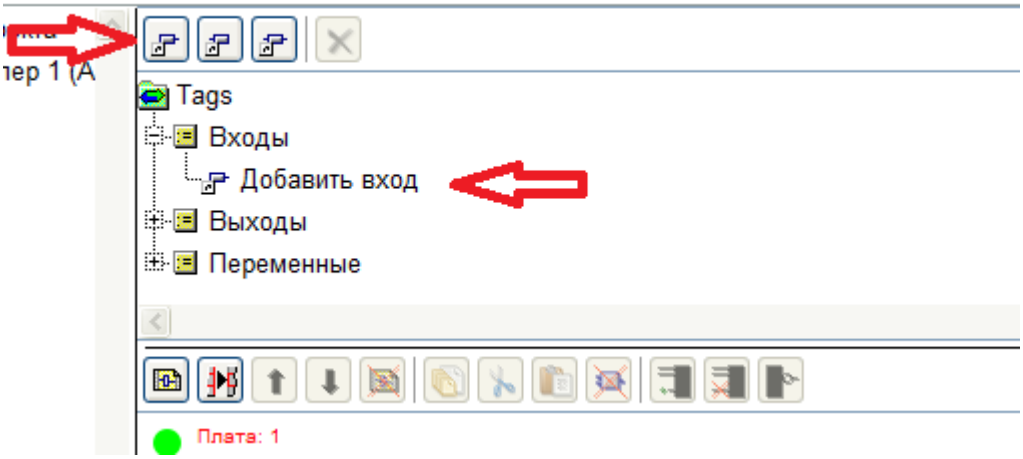


Рис. 19 — Панель создания входов в дереве тэгов.

Выбираем цифровой вход, появляются новые параметры. Записываем название входа, выбираем нужный вход платы Ардуино, и ставим галочку «Включить подтягивающий резистор».

Таким же образом добавляем все необходимые входы:

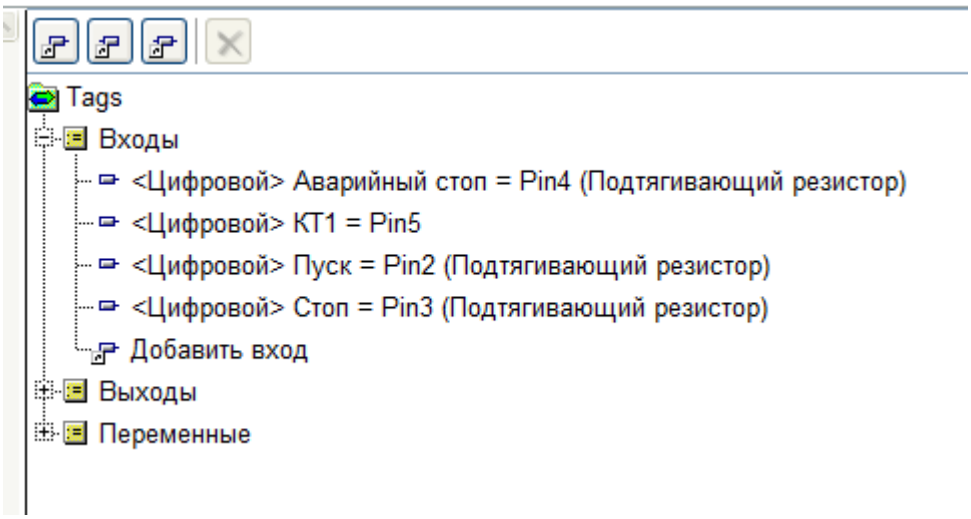


Рис. 20 — Созданные входы

Затем создаем переменную, отвечающую за состояние контактора. Для этого либо нажимаем на кнопку «Добавить переменную», либо делаем двойной клик на пункте «Добавить переменную» в дереве тэгов.

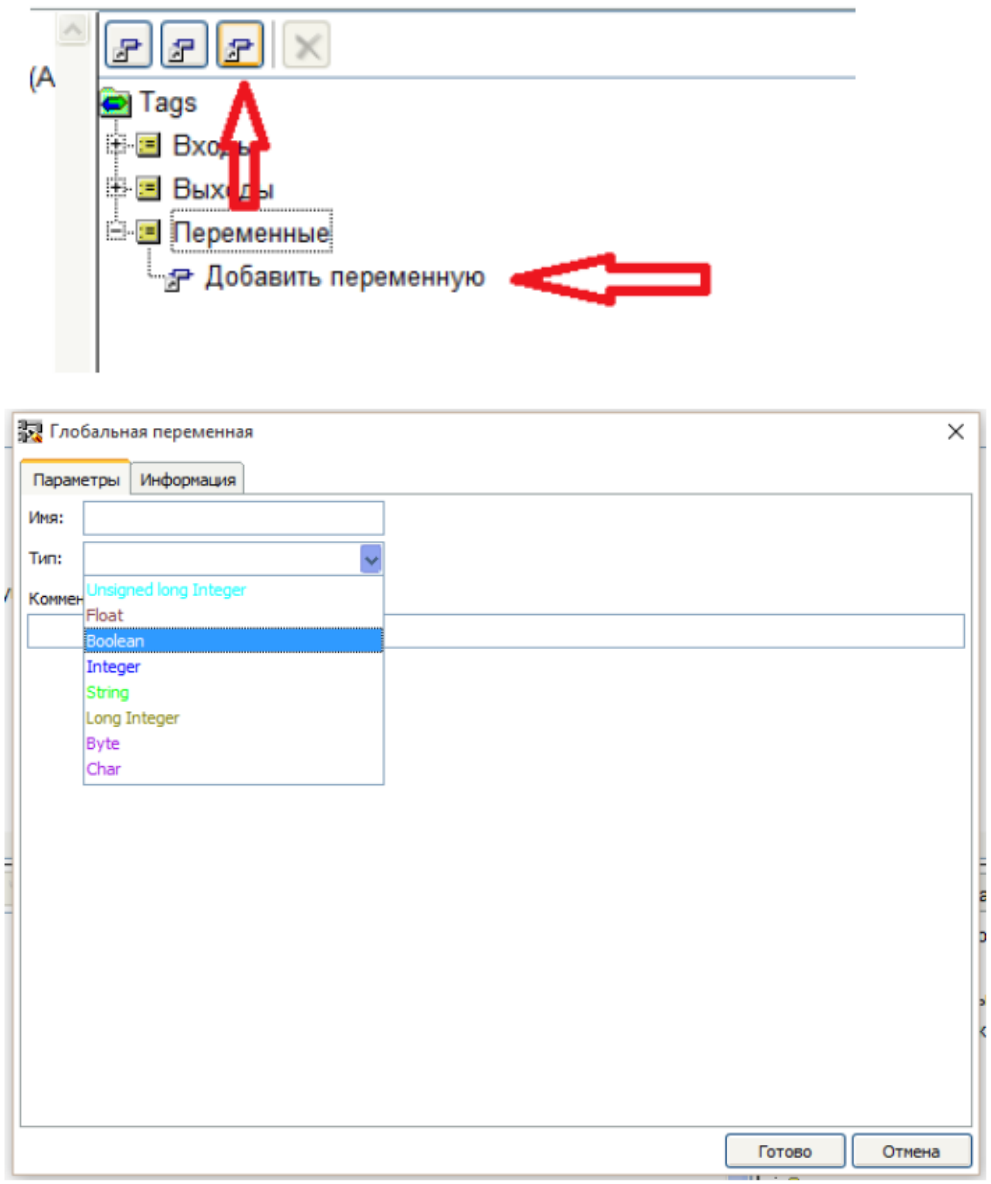


Рис. 20 — Окно настройки типа переменной (выбираем – Boolean)

Блоки входа на языке FBD соответствуют реальным выходам платы, следующим образом. Когда на реальном входе 0 – на выходе блока – False, когда на входе платы 5В на выходе блока True.

Для запоминания состояния контактора используем **RS триггер**. Его надо перетащить из папки «Триггеры» библиотеки блоков на рабочее поле схемы.

RS-триггер, или SR-триггер — триггер, который сохраняет своё предыдущее состояние при нулевых входах и меняет своё выходное состояние при подаче на один из его входов единицы.

При подаче единицы на вход S (от англ. Set — установить) выходное состояние становится равным логической единице.

А при подаче единицы на вход R (от англ. Reset — сбросить) выходное состояние становится равным логическому нулю.

При логическом нуле на обоих входах на выходе удерживается последнее состояние. При логических единицах на обоих входах в случае RS триггера выход устанавливается в логический ноль, а в случае SR триггера в логическую единицу.

Для того что бы включился контактор необходимо подать на вход S сигнал со входа «Пуск». Для этого перетаскиваем из дерева тэгов вход «ПУСК» на рабочую область схемы. Если вспомнить о том, что при нажатии кнопки Пуск на вход платы подаётся логический 0, то понятно, что необходимо инвертировать сигнал с кнопки. Для этого наведём курсор на вход S триггера и кликнем правой кнопкой мыши (рис.21):

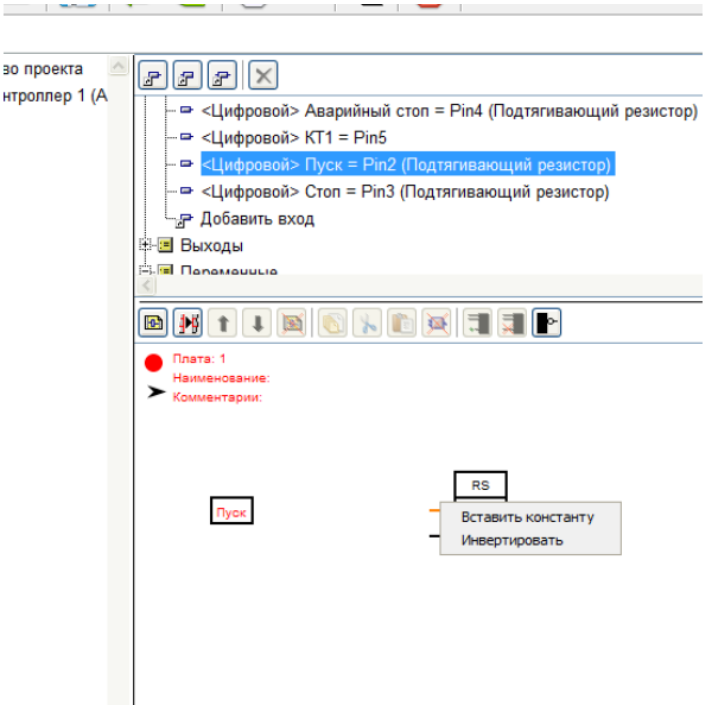
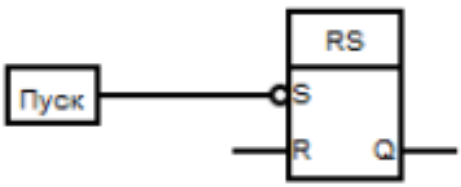


Рис. 21 — В открывшимся меню в пункт выберем «Инвертировать»

После чего соединяем вход S триггера с выходом блока входа «Пуск».



Остановка контактора происходит если:

Нажата кнопка «СТОП» (лог.1 на блоке входа «Стоп») ИЛИ нажата кнопка «АВАРИЙНЫЙ СТОП» (лог.1 на блоке входа «Аварийный стоп») ИЛИ сработало тепловое реле (лог.1 на блоке входа «КТ1»).

Значит, нам нужен блок ИЛИ с тремя входами. Перетаскиваем его из библиотеки блоков из папки «Базовые блоки».

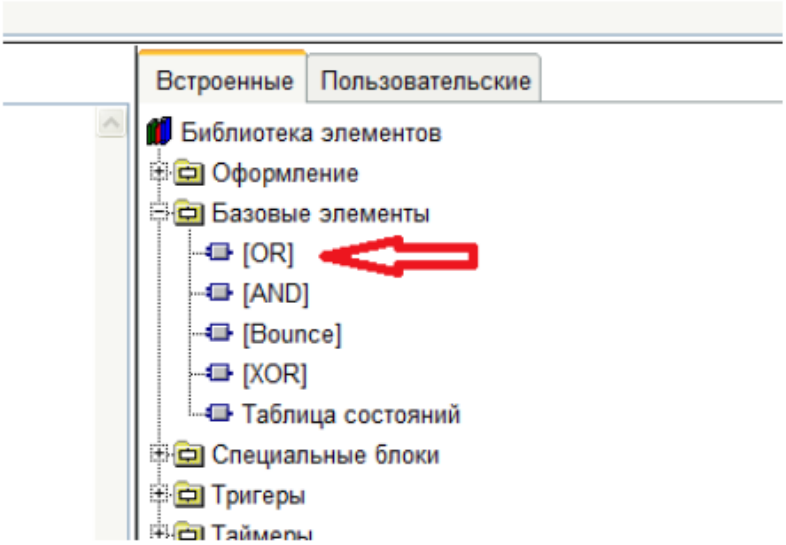


Рис. 22 — Базовый блок OR (ИЛИ)

По умолчанию у блока ИЛИ два входа. Для того что бы добавить третий, выделяем блок и нажимаем кнопку «Добавить вход» (рис.23):

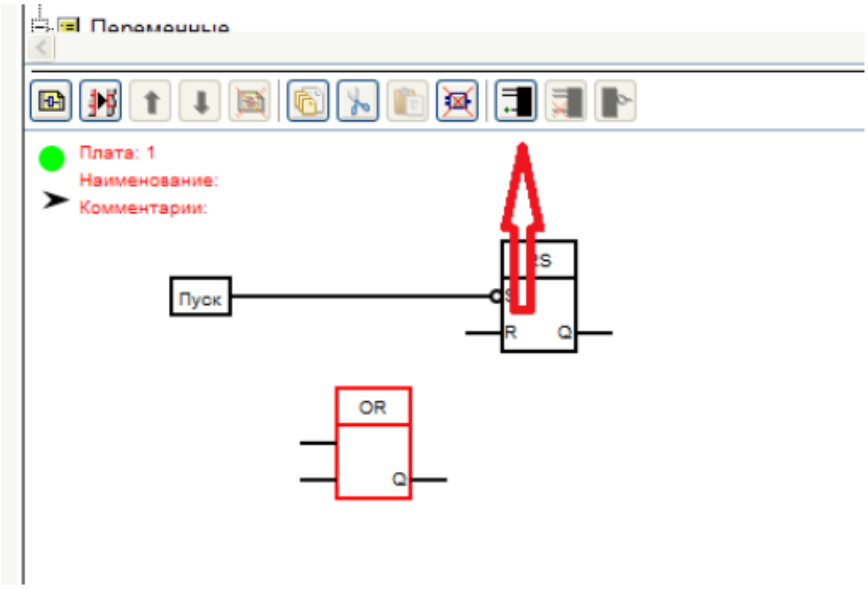


Рис. 23 — Увеличение ходов у логического блока ИЛИ

Переносим необходимые входы из дерева тэгов и соединяем со входами блока ИЛИ. А выход блока ИЛИ соединяем с входом R триггера.

Затем забираем из дерева тэгов переменную «Состояние контактора» и выход триггера соединяем со входом этой переменной:

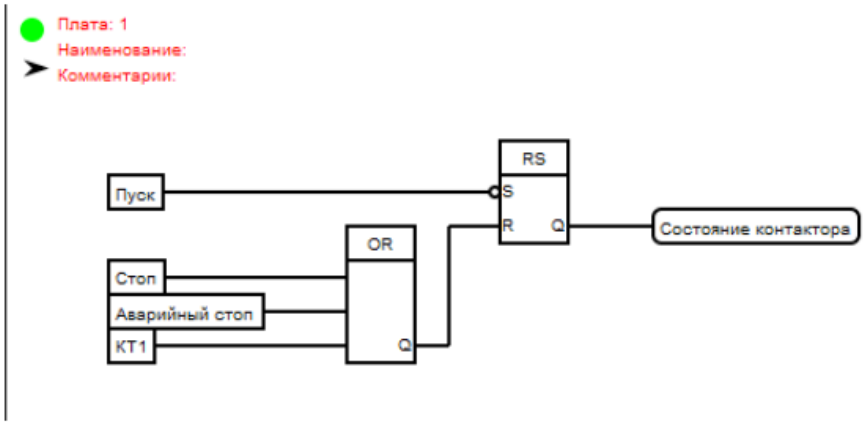


Рис. 24 — «Плата управления»

На этом закончим первую плату – «Плата управления».

После чего создадим новую плату и сразу назовём её «Управление выходами».

Далее создадим выходы платы в соответствии со схемой. Для этого надо кликнуть на кнопку «Добавить выход» дли сделать двойной клик на пункте «Добавить выход» в дереве тэгов. Выходы создаём цифрового типа.

Перетащим на вторую плату созданные выходы, вход КТ1 и переменную «Состояние контактора» Затем соединим блоки в соответствии со схемой. Необходимые входы блоков инвертируем.

Рис. 25 — «Плата управление выходами (выводами)»

Обратите внимание, что при перетаскивании на схему блоков входа, выхода или переменной изначально у них нет входов или выходов. Они появляются при подведении курсора к блоку в месте их будущего расположения.

С созданием схем закончили. Теперь надо залить программу в контроллер. Для этого нажимаем кнопку «Компилировать проект».

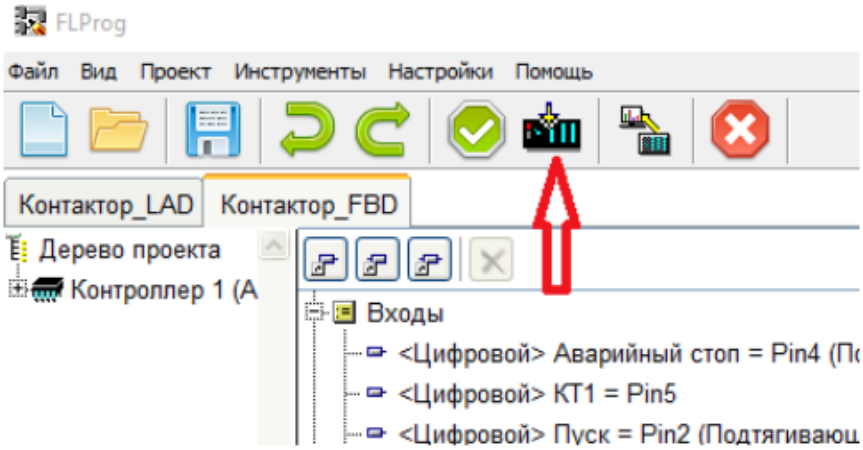


Рис. 26 — Кнопка компиляции проекта в FLProg

После компиляции должна открыться IDE Arduino. Изучаем транслированный в привычный текстовый вид код на языке С.

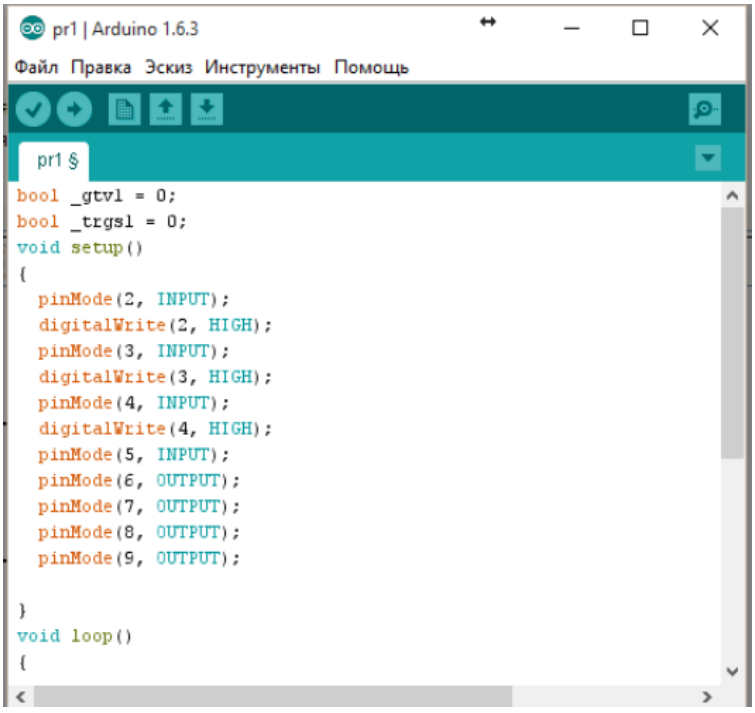


Рис. 27 — Arduino-IDE с открытым скетчем, в который была преобразована созданная схема.

После проделанных шагов вам необходимо перейти к настройке интегрированной среды разработки Arduino.

Подключите плату Arduino UNO R3 к любому из доступных и удобных вам портов USB. Определите номер виртуального COM-порта (в реальности - USB), на котором находится плата (левый нижний угол или диспетчер устройств, там должен быть драйвер CH340g.

Микросхема CH340G – преобразователь интерфейса USB в UART (мост USB-UART). Характеристики, условия эксплуатации, типовые схемы включения.

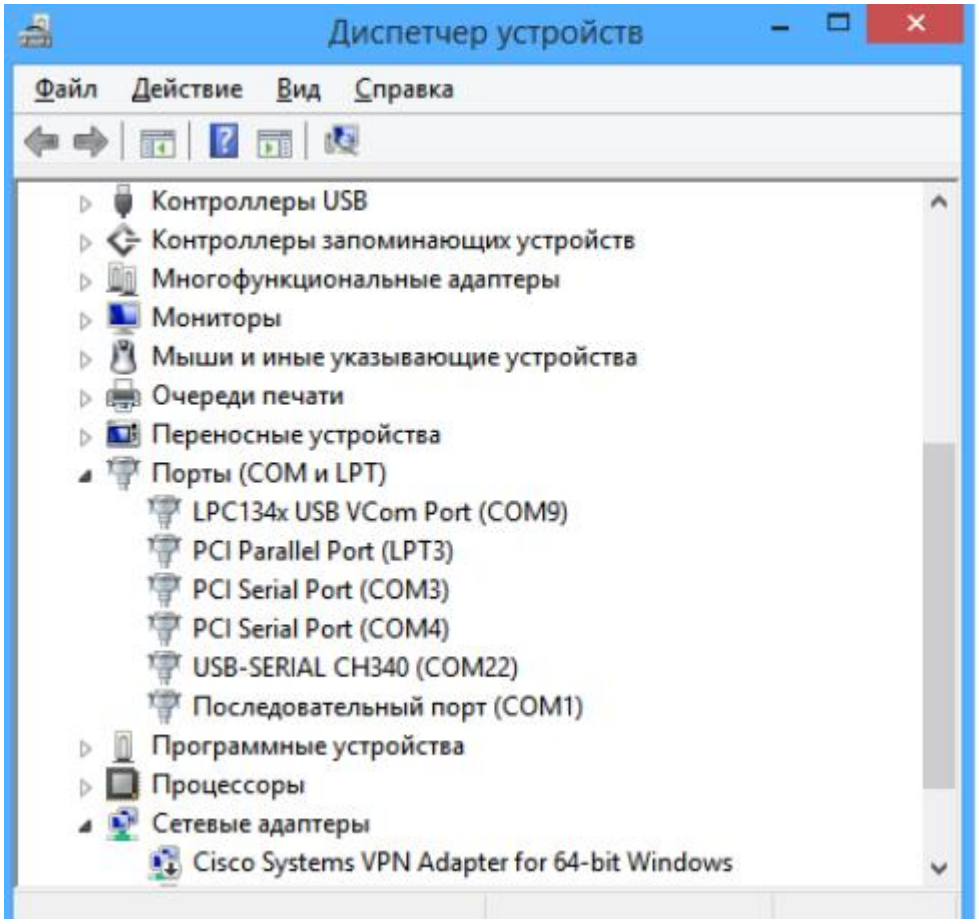


Рис. 27 — Определение номера COM-порта (в данном случае COM22)

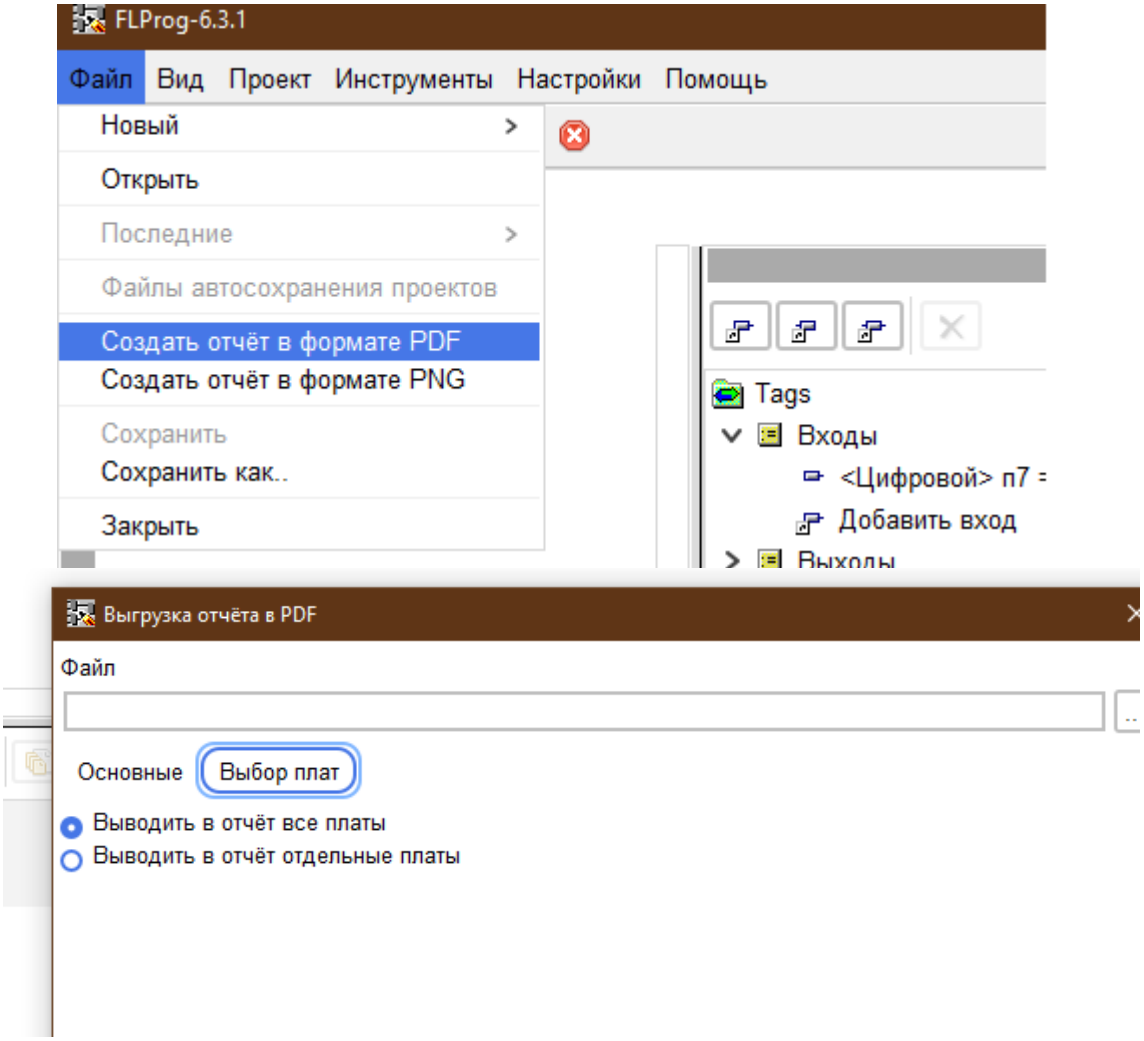
Вкладка: **Инструменты** – **Порт** (последовательный порт): выбираем в нашем случае 22 (обычно цифры меньше – 1-9).

Тип платы: **Arduino UNO**.

Перед компиляцией в Arduino IDE (процессом программирования самого устройства) необходимо удостовериться в правильности настройки (воспользуйтесь обязательно помощью преподавателя или ассистента (лаборанта).

Требования к отчету:

1. Ссылка на предоставляемый открытый доступ к видео-демонстрации по заданным требованиям.
2. Копия готового проекта в FLProg в расширении .flp («Сохранить как» с именованием следующего вида: Пр1_ФАМИЛИЯ_ГРУППА)
3. Отчет в PDF по всем платам (с аналогичным именованием);
4. Код программы в оформленном в Microsoft Word/LibreOffice виде, в типологии отчета с титульной страницей.
5. Скриншот (в отчет) описания базового элемента OR, которая находится в встроенной справке программы FLProg.



§ ПРАКТИКУМ: СОЗДАНИЕ СИСТЕМЫ ОГРАНИЧЕНИЯ ДОСТУПА С ПРИМЕНЕНИЕМ RFID.

Оборудование: ПК, кабель AM/microBM 5p Cablexpert Pro (CCP-mUSB2-AMBM-1.0M или аналоги), Arduino UNO R3, RFID модуль RFID RC522, соединительные провода.

Программное обеспечение: IDE FLProg, IDE Arduino.

Цель: научиться реализовывать простую систему контроля и управления доступом, получить навыки работы с радиочастотной идентификацией посредством FBD.

Теоретические сведения:

Система контроля и управления доступом, СКУД (англ. Physical Access Control System, PACS) — совокупность программно-аппаратных технических средств контроля и средств управления, имеющих целью ограничение и регистрацию входа-выхода объектов (людей, транспорта) на заданной территории через «точки прохода»: двери, ворота, КПП.

Основная задача — управление доступом на заданную территорию (кого пускать, в какое время и на какую территорию), включая также:

ограничение доступа на заданную территорию;

идентификацию лица, имеющего доступ на заданную территорию.

Дополнительные задачи:

учёт рабочего времени;

расчет заработной платы (при интеграции с системами бухгалтерского учёта);

ведение базы персонала / посетителей;

интеграция с системой безопасности, например:

с системой видеонаблюдения для совмещения архивов событий систем, передачи системе видеонаблюдения извещений о необходимости стартовать запись, повернуть камеру для записи последствий зафиксированного подозрительного события;

с системой охранной сигнализации (СОС), например, для ограничения доступа в помещения, стоящие на охране, или для автоматического снятия и постановки помещений на охрану.

с системой пожарной сигнализации (СПС) для получения информации о состоянии пожарных извещателей, автоматического разблокирования эвакуационных выходов и закрывания противопожарных дверей в случае пожарной тревоги.

Идентификатор Основные типы исполнения — карточка, брелок, метка. Является базовым элементом системы контроля доступа, поскольку хранит код, который служит для определения прав («идентификации») владельца.

Это может быть Touch memory, бесконтактная карта (например, **RFID-метка**), или устаревающий тип карт с магнитной полосой. В качестве идентификатора могут выступать также коды, вводимый на клавиатуре, или отдельные биометрические признаки человека — отпечаток пальца, рисунок сетчатки или радужной оболочки глаза, трехмерное изображение лица.

Надежность (устойчивость к взлому) системы контроля доступа в значительной степени определяется типом используемого идентификатора: например, наиболее распространенные бесконтактные карты proximity могут подделываться в мастерских по изготовлению ключей на оборудовании, имеющемся в свободной продаже. Поэтому для объектов, требующих более высокого уровня защиты, подобные идентификаторы не подходят. Принципиально более высокий уровень защищенности обеспечивают RFID-метки, в которых код карты хранится в защищённой области и шифруется.

Кроме непосредственного использования в системах контроля доступа, RFID-метки широко применяются и в других областях. Например, в локальных расчетных системах (оплата обедов в столовой и других услуг), системах лояльности и так далее.

Контроллер - это «мозг» системы: именно контроллер определяет, пропустить или нет владельца идентификатора в дверь, поскольку хранит коды идентификаторов со списком прав доступа каждого из них в собственной энергонезависимой памяти. Когда человек предъявляет (подносит к считывающему устройству) идентификатор, считанный из него код, сравнивается с хранящимся в базе, на основании чего принимается решение об открытии двери.

Сетевой контроллер объединяется в единую систему с другими контроллерами и компьютером для возможности централизованного контроля и управления. В таком случае решение о предоставлении доступа может приниматься как контроллером, так и программным обеспечением головного компьютера.

Чаще всего объединение контроллеров в сеть осуществляется посредством промышленного интерфейса RS-485 или локальной сети Ethernet.

В случаях, когда необходимо обеспечить работу контроллера при авариях электросети, блок контроллера обеспечивается собственным аккумулятором, либо внешним блоком резервного питания. Время работы от аккумулятора может составлять от нескольких часов до нескольких суток.

Ход работы:

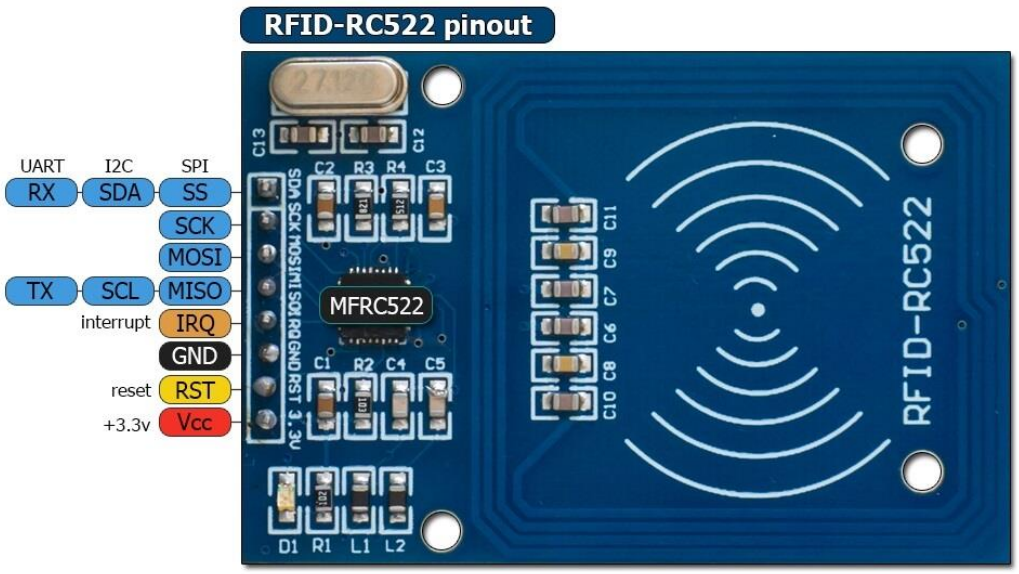
1. Подключить Arduino UNO, определив номер отождествляемого COM-порта.
2. Запустить и настроить Arduino IDE на соответствующий порт, плату.
3. Изучить характеристики модуля RFID RC522:

Технические характеристики RFID-модуля RC522

Напряжение питания: **3.3V**;
Потребляемый ток :13-26mA;
Рабочая частота: 13.56MHz;
Дальность считывания: 0 - 60 мм;
Интерфейс: SPI;
Скорость передачи: максимальная 10МБит/с;
Размер: 40мм x 60мм;

Назначение выводов интерфейса SPI:

SDA – выбор ведомого; SCK –сигнал синхронизации;
MOSI – передача от master к slave; MISO – передача от slave к master;
RST – вывод для сброса; IRQ – вывод прерывания;
GND – земля; Vcc –питание **3.3 В.**



4. Сканер бесконтактных меток RFID-RC522 без меток позволяет обнаружить и считать идентификаторы бесконтактных карт, меток, пропусков стандарта 13,56 МГц на расстоянии до 6 см. Благодаря данному сканеру можно сделать ряд интересных проектов: пропускные системы, электронные замки, складской учету и много другое. Предложить прикладной вариант применения данного модуля.
5. Запустить IDE FLprog. Создать проект в FBD для Arduino Uno (Atmega328). Перейти к изучению функциональных блоков:

В программе FLProg реализовано два вида функциональных блоков для работы со сканером.

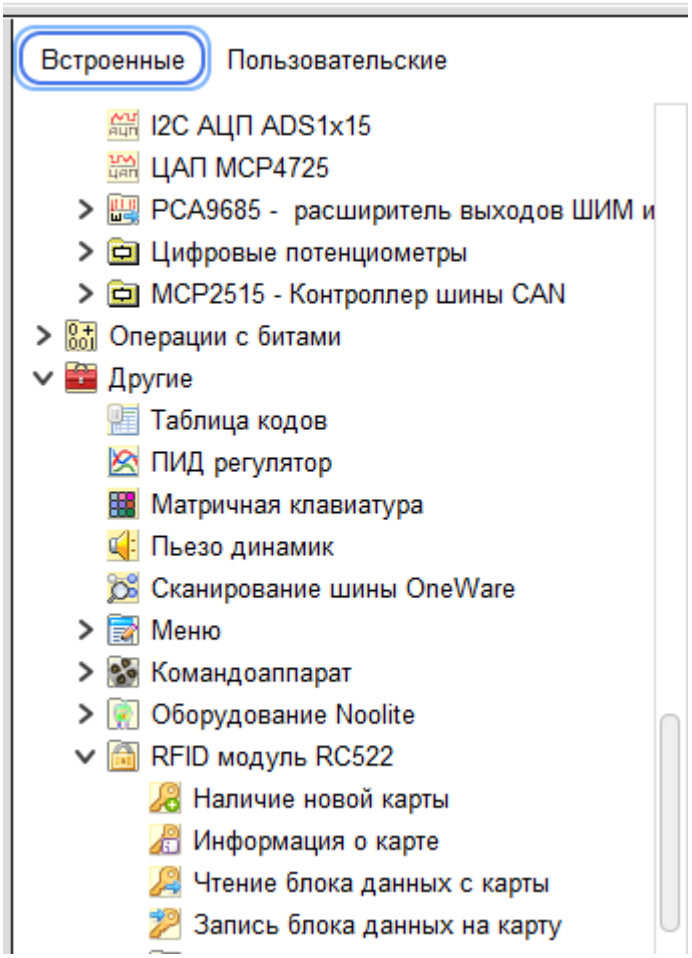
Блоки непосредственно связанные со сканером

«Наличие новой карты»

«Информация о карте»

«Чтения бока данных на карту»

«Запись блока данных на карту»



Блоки, связанные со сканером косвенно и предназначенные для работы с хранилищем UUID карт.

В программе реализовано хранилище UUID карт, представляющее собой набор ячеек каждая из которых хранит непосредственно UUID карты, и статус ячейки.

Три статуса зарезервировано программой:
0x00 – ячейка свободна;
0X01 – в ячейке хранится UUID но она заблокирована;
0x02 — в ячейке хранится UUID и она активна;

UUID (англ. universally unique identifier «универсальный уникальный идентификатор») — это стандарт идентификации, используемый в создании программного обеспечения, стандартизированный Open Software Foundation (OSF) как часть DCE — среды распределённых вычислений.

Остальные коды статуса (3 … 255) пользователь может использовать по своему усмотрению.

Хранилище может располагаться в оперативной памяти контроллера (**при снятии питания или перезагрузке оно очистится**), или в **EEPROM**. Хранилищ может быть несколько, и они могут быть разных типов. При расположении хранилища в EEPROM размеры его ограниченны. Для Arduino Uno – это максимум 85 ячеек (во всех хранилищах EEPROM в сумме), для Arduino Mega – 341 ячейка.

Блоки для работы с хранилищем
«Сохранить UUID карты в хранилище»
«Прочитать UUID карты из хранилища»
«Статус ячейки в хранилище»
«Записать статус ячейки в хранилище»
«Блокировка/разблокировка ячейки»
«Поиск UUID в хранилище»
«Свободные ячейки хранилища»
«Очистка ячейки в хранилище»
«Очистка всего хранилища»

Техническое задание:

Пусть в устройстве будут два хранилища расположенных EEPROM. В первом хранятся так называемые Master Card. С помощью них можно записывать обычные карты во второе хранилище.

Для записи Master Card существует «Секретная кнопка». Так же допускается дисплей для отображения необходимой информации, и обычная управляющая кнопка. Создать систему разграничения доступа по следующей схеме (рис.28):

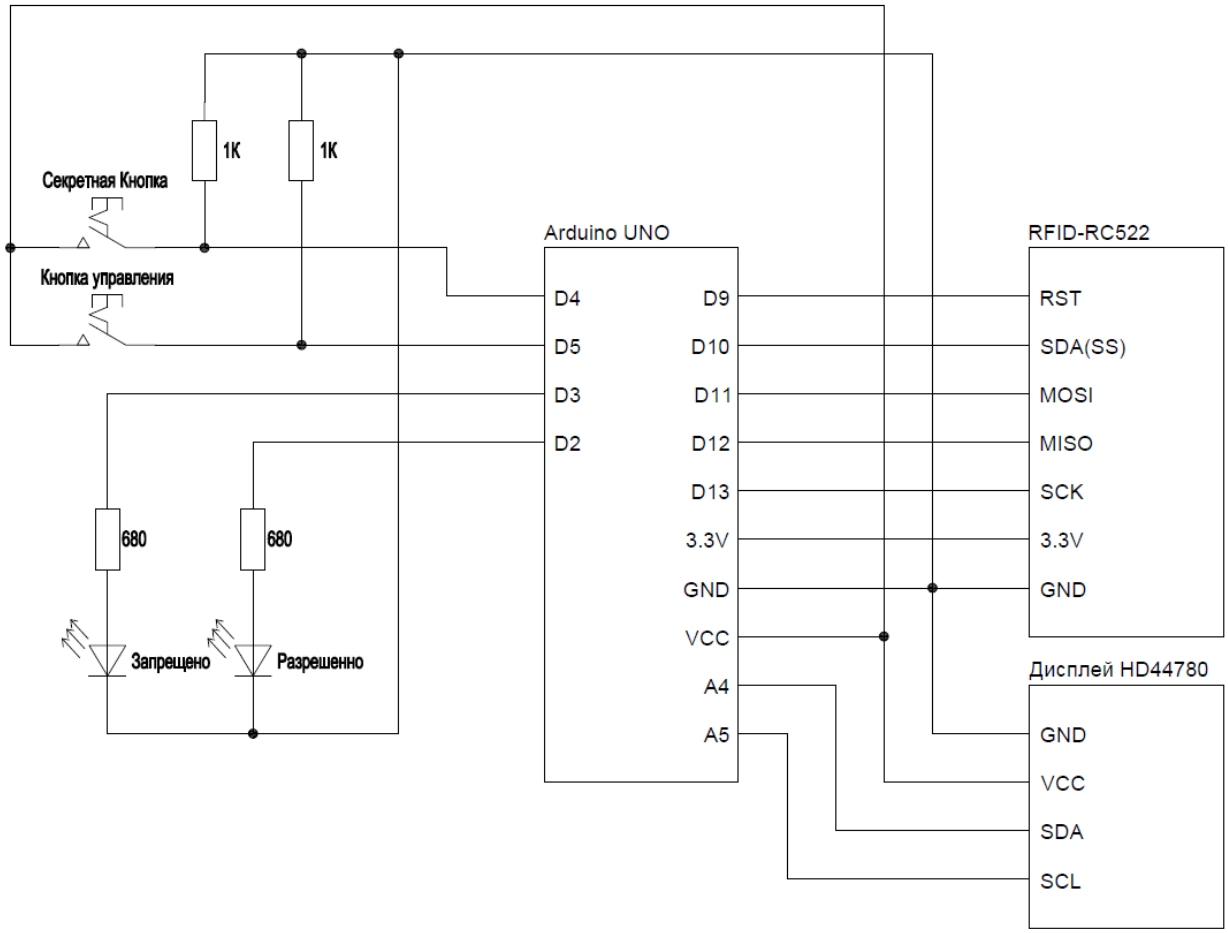
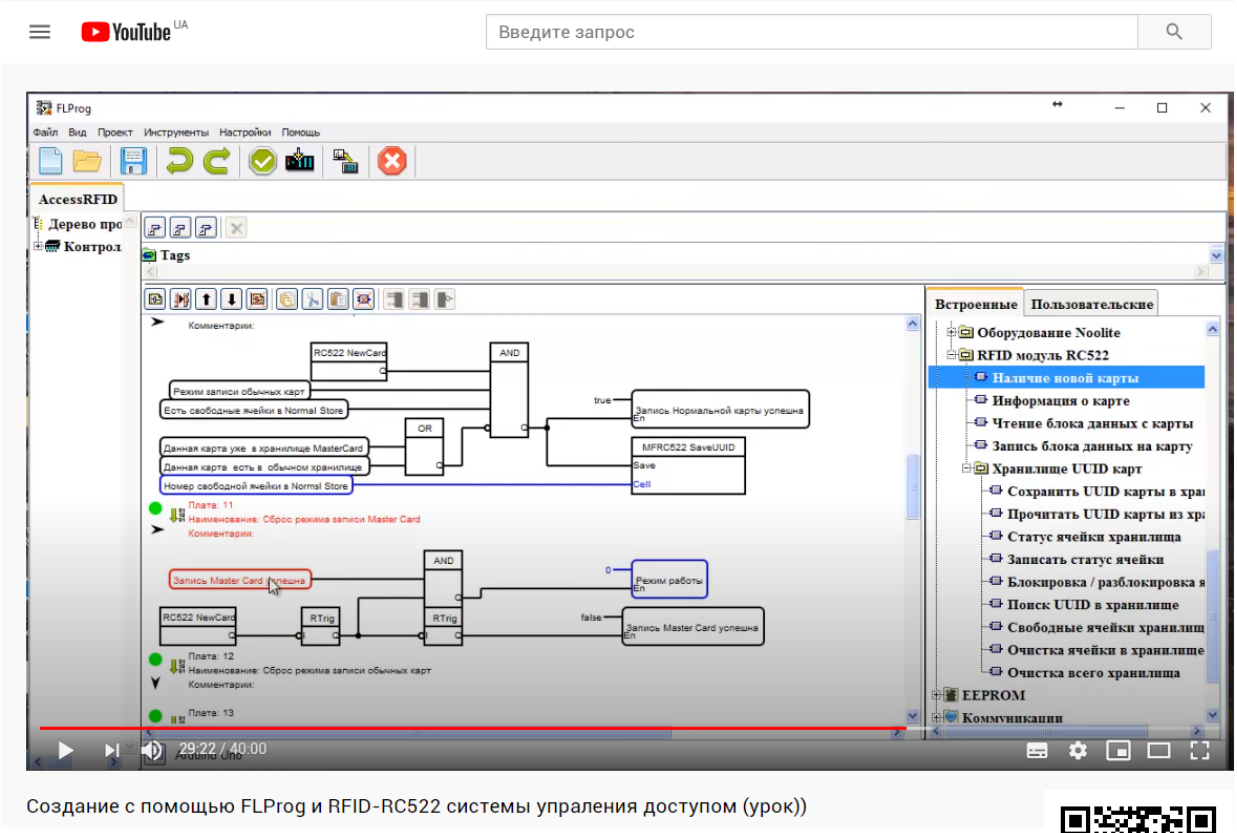


Рис. 28 — Схема разрабатываемого устройства

Карты и брелки используемые вместе со сканером имеют «на борту» 1 кБ памяти которые можно использовать в собственных целях. Давайте расширим техническое задание и будем записывать, какую-либо информацию.

Пошаговая видео-демонстрация выполнения практического задания в открытом доступе по ссылке: https://youtu.be/AXWDZEUY-6I



Создание с помощью FLProg и RFID-RC522 системы упраления доступом (урок))

Требования к отчету:

1. Привести код программы и FBD-грамму.
2. Описать все выходы и входы устройства по вышеизложенной схеме.
3. Привести имена типов карт (RFID-типы), поддерживающих данный модуль, указать частоту. Дать кратких обзор типов RFID-меток.
4. Подготовить реферат на защиту по темам (для лиц, упомянутых преподавателем по списку):

    1. Дальняя идентификация (Идентификация на пассивных метках дальнего считывания);

    2.Альтернативные методы автоматической идентификации;

    3. Информационная безопасность RFID. Вирусы и атаки в радиочастотной идентификации;

    4. RFID и IoT/IoE.

§ ПРАКТИКУМ: СОЗДАНИЕ ВСТРАИВАЕМОЙ СИСТЕМЫ ПО ТЕХНИЧЕСКОМУ ЗАДАНИЮ

Оборудование: персональный компьютер.

Программное обеспечение: IDE FLProg, IDE Arduino.

Цель: научиться реализовывать конечные проекты встраиваемых компьютерных систем посредством среды визуального программированию по техническому заданию.

Теоретические сведения:

Встраиваемая система (встроенная система, англ. embedded system) — специализированная микропроцессорная система управления, контроля и мониторинга, концепция разработки которой заключается в том, что такая система будет работать, будучи встроенной непосредственно в устройство, которым она управляет.

Шаблон задания:

Разработать встраиваемую систему на базе любого микроконтроллера семейства Arduino/ESP, которая будет считывать и записывать в постоянную память показания температуры и влажности воздуха. Обеспечить звуковую или светодиодную индикацию при достижении уровня влажности (относительной или абсолютной – на выбор) более 79%.

Ход работы:

1. Задание выбирается индивидуальным образом – должен соблюдаться подход равноценности и равнозначности выбираемого задания с учетом предпочтений студента. Для этого рекомендуется провести семинар по теме анализа компонентов (датчиков, исполнительных устройств, средств идентификации и аутентификации, специальных устройств ввода-вывода и обработки информации, представляемых в среде FLProg.
2. Студент предоставляет Datasheet по каждому из необходимых устройств (за исключением главного микроконтроллера) в печатном виде, в которой обязательно должна быть оформлена первая титульная печатная страница для рецензирования преподавателем.
3. Студент предоставляет собранную схему в FLProg в электронном формате (flp), код программы на C++ на защиту.

§ RemoteXY и Bluetooth ВЗАИМОДЕЙСТВИЕ С ПРОГРАММОЙ FLPROG.

RemoteXY - сервис для организации удаленного беспроводного и проводного управления устройствами на базе arduino, esp8266, esp32 и chipKIT. Сервис состоит из онлайн-редактора мобильного интерфейса, приложения для устройств Android и iOS, сервера для удаленного управления устройствами, а также библиотеки для ArduinoIDE.

Приложений для Android устройств два - бесплатное и платное. Бесплатное приложение имеет ограничение в 5 элементов интерфейса. Если же элементов интерфейса больше, то приложение позволяет протестировать работу устройства в течении ограниченного времени (всего 30 секунд за сеанс).

Работа с сервисом RemoteXY организована следующим образом:

- в онлайн-редакторе необходимо выбрать программируемое устройство и способ его подключения, затем настроить способ подключения;
- при помощи онлайн-редактора необходимо создать интерфейс программируемого устройства, для этого предусмотрены элементы управления и индикации, такие как кнопки, переключатели, поля ввода и вывода текста, графики;
- получить сгенерированный онлайн-редактором исходный код, перенести его в ArduinoIDE (также поддерживаются FLProg и MPIDE), дополнить необходимым кодом и загрузить в программируемое устройство;
- при помощи приложения подключиться к устройству.

Отметим, что идентификатор интерфейса хранится непосредственно в программируемом устройстве в виде массива чисел, и загружается в приложение для смартфона при каждом подключении. Таким образом, отпадает необходимость настройки интерфейса на каждом подключаемом к устройству смартфоне, необходимо лишь настроить связь между смартфоном и запрограммированном устройстве.

Редактор интерфейса разделен на 3 части. В центре основное окно **(рис.29)**, в котором отображается внешний вид созданного интерфейса. Слева раскрывающиеся списки с элементами интерфейса, сгруппированные по категориям. Справа расположены вкладки конфигурации подключения устройства, настройки экрана и контекстно-зависимая вкладка настройки выбранного элемента интерфейса.

Работа с онлайн-редактором строится по принципу drag-and-drop, также, как и в проводнике компьютера. Просто перетаскивайте необходимый элемент с левой области в центральную, на макет смартфона.

При клике на элемент на макете смартфона он выделяется синей рамкой и теперь его можно перемещать по макету, а также масштабировать, хватая и перемещая маркеры по углам рамки. Обратите внимание, над рамкой выбранного элемента отображается имя переменной, по которому в дальнейшем будет осуществляться доступ к выбранному элементу.

рис. 29 — Основное окно RemoteXY

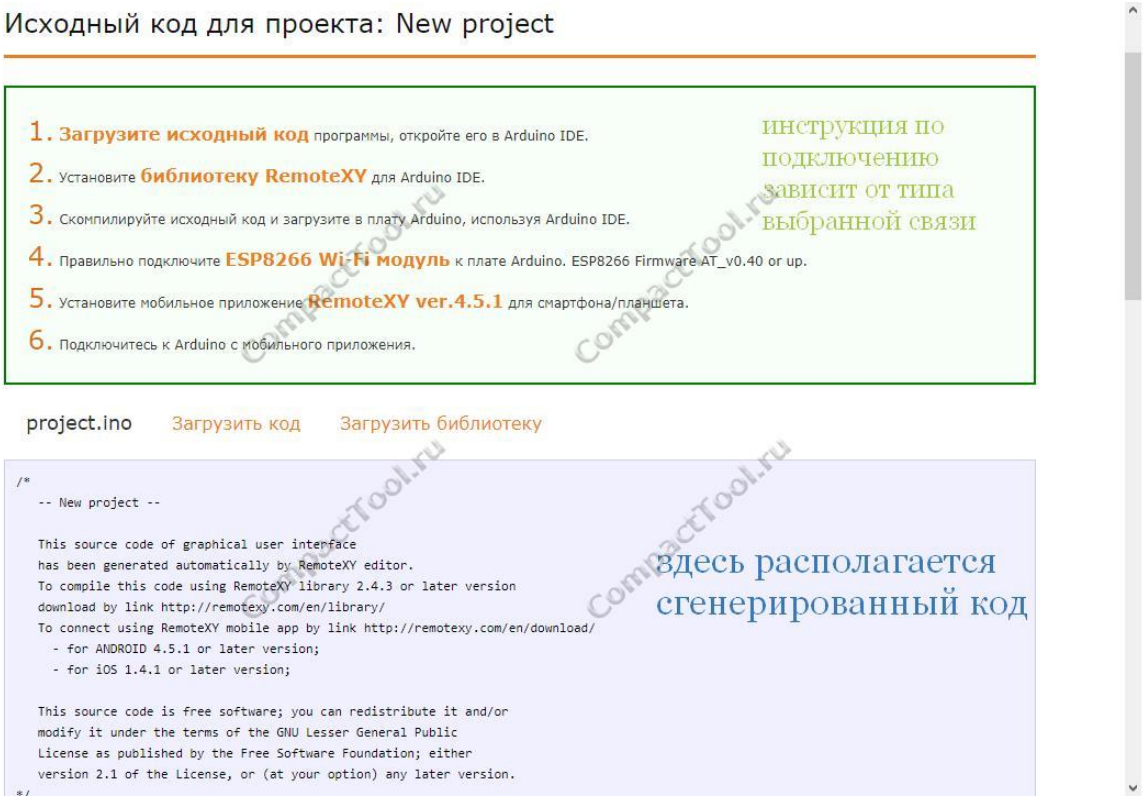
Начать знакомство с RemoteXY разработчиками программы рекомендуется с изучения способов связи приложения и устройств, которыми необходимо управлять. Но перед этим создадим простейший интерфейс следующим образом:

- перетащите на макет смартфона элемент “выключатель”;
- убедитесь, что в настройках выключателя в выпадающем списке “привязать к выводу” выбран вывод с надписью “LED”;

Выглядеть это должно так (рис.30):

рис. 30 — Созданный простейший интерфейс

Для получения сгенерированного кода необходимо нажать большую зеленую кнопку “Получить исходный код”:



Для доступа к вариантам совместимых соединений необходимо раскрыть вкладку “конфигурация” в правой части онлайн редактора и нажать мышкой на любой из появившихся значков.

Список всех возможных соединений приведен на рис.31:

рис. 31 — Совместимые соединения в среде RemoteXY

Соединение при помощи кабеля требует **наличие функции OTG** в смартфоне. К сожалению, не все смартфоны могут похвастаться наличием такой функции. Соединение такого типа доступно для всех плат прототипирования, которые имеют на плате распаянный переходник USB-UART. При отсутствии распаянного переходника USB-UART (как в случае с arduino pro mini) можно подключить внешний.

Продемонстрируем соединение «через кабель» на примере Arduino Mega2560.

Используем следующую конфигурацию подключения:

- соединение: USB OTG
- устройство: Arduino Mega2560
- модуль: USB-UART
- среда разработки: ArduinoIDE.

Далее необходимо выбрать программный или аппаратный протокол, выберем аппаратный (Hardware Serial). Выбираем Serial port к которому подключен переходник USB-UART, который размещен на плате **Mega2560.**

Напоследок устанавливаем скорость соединения 9600 бод. Напомним, что в интерфейсе сейчас находится один элемент управления - переключатель, который зажигает расположенный на плате светодиод. После загрузки скомпилированного кода в микропроцессорную систему мы подключаем её к своему смартфону.

Процесс подключения показан на следующих скриншотах.

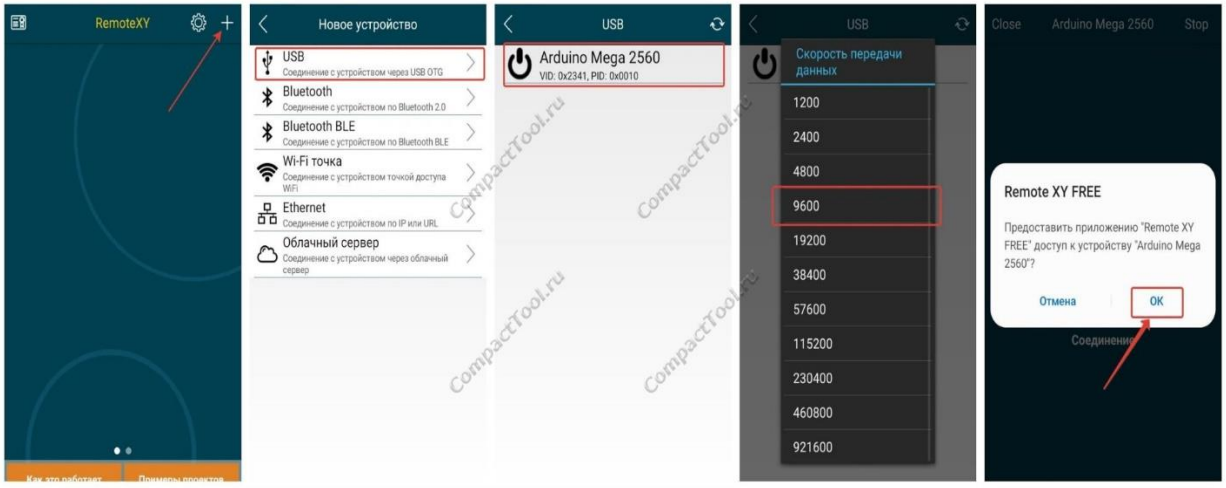


рис. 32 — Подключение Arduino Mega 2560 в RemoteXY

Это очень специфичный способ связи смартфона и управляемого устройства. Для **постоянного соединения годится только** в некоторых случаях, но очень подходит для не частого подключения, например, для настройки управляемого устройства.

Примечание: этот способ соединения не работает с Arduino Leonardo.

Следующий **тип связи** - **BlueTooth**. Возможна работа с модулями **HC-05, HM-10** и встроенным в ESP32 bluetooth on chip.

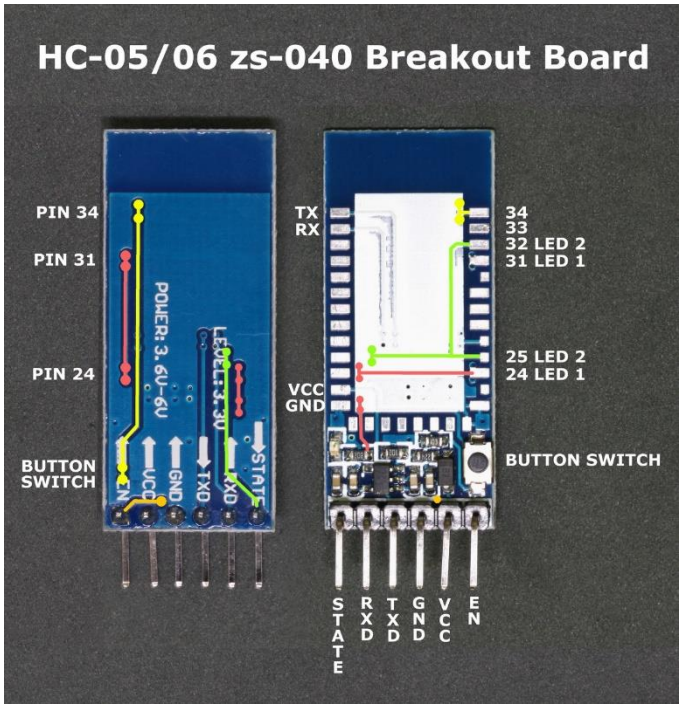
Данный способ связи годится для использования на небольших расстояниях, ограниченных дальностью связи модуля bluetooth. Отлично подойдет для устройств умного дома без доступа к сети интернет, отдельных “умных” устройств, управляемых со смартфона.

Особенностью данного способа связи является то, что смартфон может поддерживать связь с несколькими устройствами одновременно, что не достижимо при помощи следующего еще одного способа взаимодействия - точки доступа (AP).

Arduino UNO + Bluetooth HC-05(06) в RemoteXY

К контроллеру **Arduino UNO** необходимо подключить модуль Bluetooth HC-05, HC-06 (рис. или совместимый. Смартфон или планшет должен поддерживать Bluetooth.

Примечание: Устройства с ОС iOS не поддерживают модули HC-05(06). Вместо них вы можете использовать модуль Bluetooth BLE HM-10.



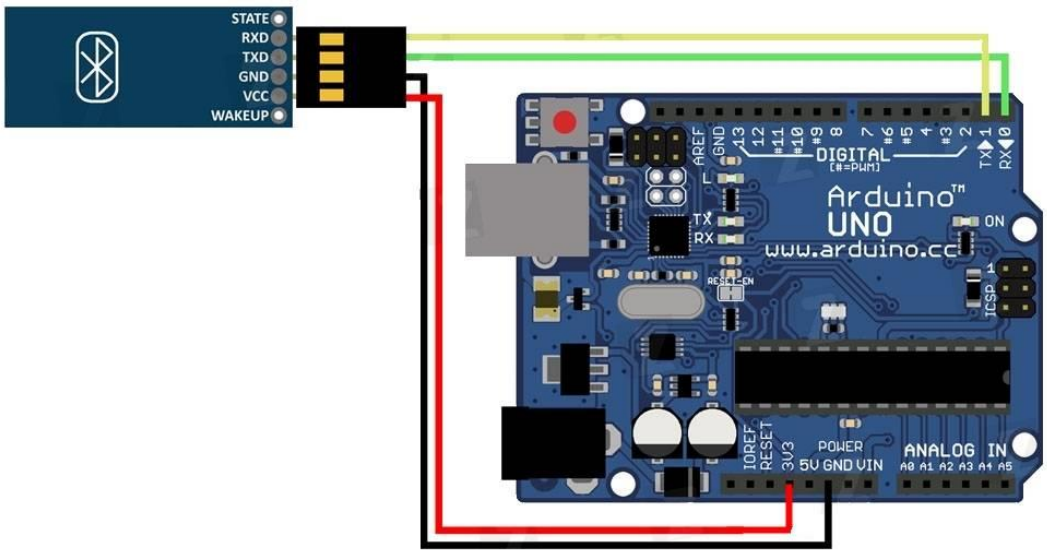


рис. 32 — Bluetooth HC-05(06), breakout и схема подключения.

Шаг 1. Создайте проект графического интерфейса.

Войдите в редактор RemoteXY. Установите в поле смартфона одну кнопку. Выделите эту кнопку, затем в правой панели во вкладке «Элемент» выберите свойство «Привязать к выводу» в значение 13(LED).

Шаг 2. Настройте конфигурацию проекта

В правой панели во вкладке «Конфигурация», выберите следующие настройки.

В правой панели во вкладке «Подключение модуля» установите следующие настройки. В настройках указывают, что модуль HC-05(06) подключается к Arduino через программный последовательный порт SoftwareSerial используя контакты 2 и 3 на скорости 9600. Это стандартная скорость передачи для модулей HC-05(06).

§ ПРАКТИКУМ: БЕСПРОВОДНАЯ КЛАВИАТУРА ДЛЯ КОМПЬЮТЕРА НА СМАРТФОНЕ.

Оборудование: персональный компьютер, смартфон с ОС Android версии 6.0 и выше, доступ к Internet.

Программное обеспечение: RemoteXY, IDE Arduino, IDE FLProg.

Цель: научиться реализовывать конечные проекты встраиваемых компьютерных систем посредством среды визуального программированию по техническому заданию.

Туториал и весь дидактический материал (в виду его большого объема) находится располагается по ссылке:

https://habr.com/ru/company/flprog/blog/400655/

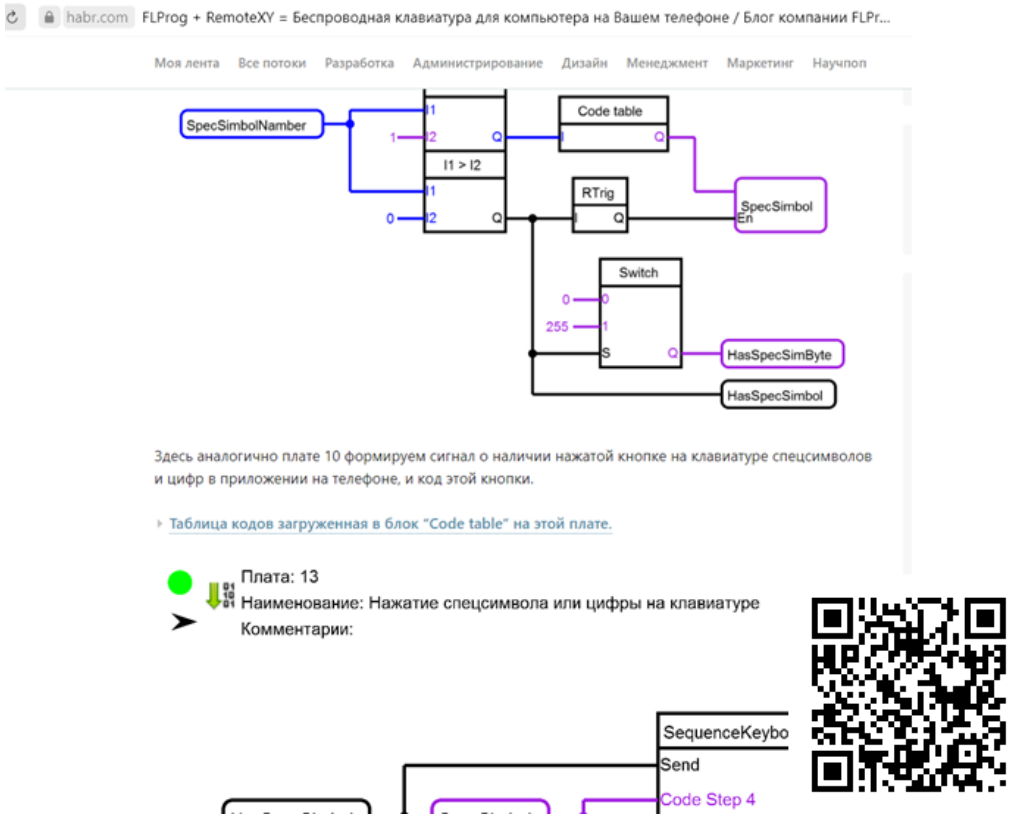


рис. 33 — Статья-туториал на портале Хабрахабр «FLProg + RemoteXY. Беспроводная клавиатура для компьютера»

В виду сложностей реализации проекта, связанных, прежде всего с временными рамками в учебном процессе, задание с реализацией беспроводного управления компьютером имеет демонстрационный характер (достаточно показать его реализацию и обсудить ее в рамках семинара).

На фактическое выполнение предлагается следующее задание. Целевое предназначение, которого направление на освоение последнего из рассматриваемых способов связи (взаимодействий), т.е, посредством IEEE 802.11:

ПРАКТИЧЕСКОЕ КОМПЛЕКСНОЕ ЗАДАНИЕ
«Интернет вещей с RemoteXY: конфигурации подключения.»
Ход работы: https://cxem.net/arduino/arduino213.php
Форма отчетности: скриншоты на всех этапах заданий

Первая часть задания: создать соединение со следующими параметрами:

- Соединение - Wi-Fi access point;
- Устройство – на ваш выбор (я буду использовать Mega);
- Модуль – ESP8266 Wi-Fi module;
- Среда программирования – ArduinoIDE;
- Интерфейс подключения – Hardware Serial;
- Порт – Serial (владельцы Mega могут попробовать любой другой доступный порт, я буду использовать Serial2);
- Скорость обмена – 115200 бод;
- Имя (SSID) – оставить по умолчанию;
- Пароль – оставить по умолчанию;
- Порт – оставить по умолчанию.
- Разместить на рабочем поле редактора кнопку, свойства кнопки не менять. Исходный код скомпилировать в ArduinoIDE и загрузить в контроллер.
- Далее необходимо подключить смартфон к созданной точке доступа, подключить программу RemoteXY к устройству. Если при нажатии на кнопку загорается светодиод на плате контроллера, значит всё сделано правильно.
- **Вторая часть задания:** изменить проект так, чтобы он использовал Software Serial, использовать выводы на свое усмотрение.
- **Третья часть задания:** изменить проект для подключения платы NodeMCU.

ОБЗОР SoC ESP8266/ESP32 как аппаратной основы IoT

ESP32 и ESP8266 — это недорогие микропроцессорные системы идеально подходящие для проектов в области интернета вещей (IoT) и домашней автоматизации.

ESP8266 — микроконтроллер китайского производителя Espressif Systems с интерфейсом Wi-Fi. Помимо Wi-Fi, микроконтроллер отличается отсутствием флеш-памяти в SoC, программы пользователя исполняются из внешней флеш-памяти с интерфейсом SPI.

Характеристики:

- 80 MHz 32-bit процессор Tensilica Xtensa L106.
- IEEE 802.11 b/g/n Wi-Fi. Поддерживается WEP и WPA/WPA2.
- 14-17 портов (варируется от модификации) ввода-вывода(из них возможно использовать 11), SPI, I²S, UART, 10-bit АЦП. I²C возможен только через bit-banging.
- Питание 2,2…3,6 В. Потребление до 215 мА в режиме передачи, 100 мА в режиме приема, 70 мА в режиме ожидания. Поддерживаются три режима пониженного потребления, все без сохранения соединения с точкой доступа: Modem sleep (15 мА), Light sleep (0.4 мА), Deep sleep (15 мкА).

Микроконтроллер не имеет на кристалле пользовательской энергонезависимой памяти. Исполнение программы ведется из внешней SPI ПЗУ путём динамической подгрузки требуемых участков программы в кэш инструкций.

Подгрузка идет аппаратно, прозрачно для программиста. Поддерживается до 16 МБ внешней памяти программ. Возможен Standard, Dual или Quad SPI интерфейс.

Производитель не предоставляет документации на внутреннюю периферию микроконтроллера. Вместо этого он дает набор библиотек, через API которых программист получает доступ к периферии.

Рис. 34 — ESP8266 NodeMCU

Рассмотрим одну из наиболее знаковых и распространенных модификаций ESP 8266 – ESP 12E. Данный модуль имеет 128 КБ ОЗУ и 4 МБ флеш-памяти (для хранения программ и данных), достаточных, чтобы справиться с большими строками, которые составляют веб-страницы, данными в JSON/XML и всем, что мы сегодня добавляем на устройства IoT.

ESP8266 содержит встроенный приемопередатчик Wi-Fi 802.11b/g/n HT40, поэтому он может не только подключаться к сети Wi-Fi и взаимодействовать с интернетом, но и устанавливать собственную сеть, позволяя другим устройствам подключаться напрямую к нему. Это делает ESP8266 NodeMCU еще более универсальным.

Требования к питанию:

Поскольку диапазон рабочего напряжения ESP8266 составляет **от 3 В до 3,6 В**, данная плата для поддержания постоянного напряжения на уровне 3,3 В поставляется с LDO стабилизатором напряжения. Он может надежно обеспечивать ток до 600 мА, чего должно быть более чем достаточно, поскольку ESP8266 во время радиочастотных передач потребляет до 80 мА.

- Рабочее напряжение: от 2,5 до 3,6 В
- Встроенный стабилизатор: 3,3 В, 600 мА
- Рабочий ток: 80 мА
- Потребление в спящем режиме: 20 мкА

Выход стабилизатора также выводится на выводы на сторонах платы и обозначен как 3V3. Эти выводы можно использовать для подачи питания на внешние компоненты.

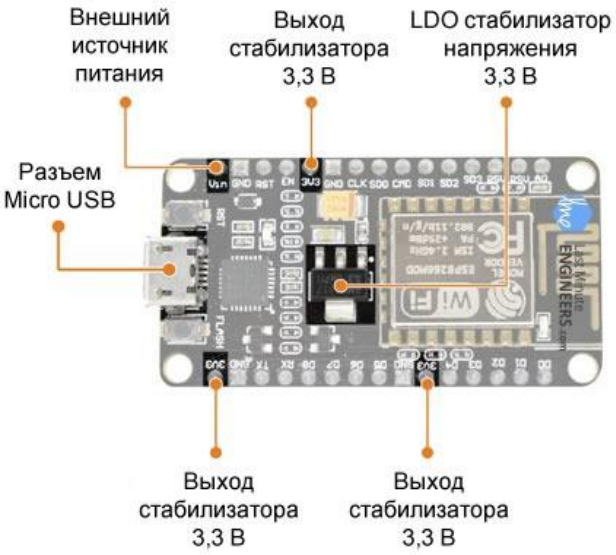


Рис. 35 — Элементы питания ESP8266 NodeMCU

Питание к ESP8266 NodeMCU подается через встроенный USB-разъем MicroB. В качестве альтернативы, если у вас есть стабилизированный источник напряжения 5 В, можно использовать вывод VIN для непосредственного питания ESP8266 и его периферии.

ESP8266 требует 3,3 В для питания и логические уровни 3,3 В для связи. Контакты GPIO не допускают напряжение 5 В! **Если вы хотите соединить плату со схемами 5 В (или выше), то необходимо реализовать согласование логических уровней.**

Периферия и ввод/вывод

ESP8266 NodeMCU имеет в общей сложности 17 выводов GPIO, выведенных на разъемы с обеих сторон отладочной платы. Эти выводы могут использоваться для выполнения различных периферийных задач, в том числе:

- вход АЦП – канал 10-разрядного АЦП;
- интерфейс UART – интерфейс UART используется для загрузки кода по последовательной связи;
- выходы ШИМ – выводы ШИМ могут использоваться для регулировки яркости светодиодов или управления двигателями;
- интерфейсы SPI, I2C – интерфейсы используются SPI и I2C для подключения всевозможных датчиков и периферийных устройств;

Мультиплексируемые выводы ввода/вывода

- 1 канал АЦП
- 2 интерфейса UART
- 4 выхода ШИМ
- Интерфейсы SPI, I2C и I2S

Рис. 36 – Мультиплексируемые выводы GPIO платы ESP8266 NodeMCU

В ESP8266 используется функция мультиплексирования выводов (несколько периферийных устройств мультиплексируются на один вывод GPIO). Это означает, что один вывод GPIO может действовать как PWM/UART/SPI.

Кнопки и светодиодный индикатор на плате:

На плате ESP8266 NodeMCU находятся две кнопки. Одна из них, помеченная как RST, расположенная в верхнем левом углу, представляет собой кнопку сброса, которая, конечно же, используется для сброса микросхемы ESP8266. Другая кнопка, FLASH, в левом нижнем углу – это кнопка загрузки, используемая при обновлении прошивки.

Кнопки и индикаторы

- RST – сброс чипа ESP8266
- FLASH – загрузка новой программы
- Синий светодиод - программируется пользователем

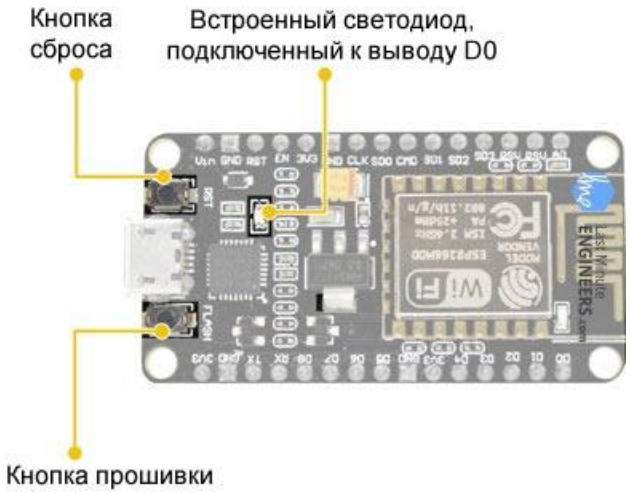


Рис. 37 – Кнопки и светодиоды на плате ESP8266 NodeMCU

На плате также имеется светодиодный индикатор, который программируется пользователем и подключен к выводу D0 платы.

Последовательная связь

На плате установлен контроллер USB-UART CP2102/или CH340 от Silicon Labs, который преобразует USB сигнал в сигнал последовательного порта и позволяет компьютеру программировать и взаимодействовать с микросхемой ESP8266.

- USB-UART преобразователь CP2102/или CH340
- Скорость связи 4,5 Мбит/с
- Поддержка управления потоком

Для простоты мы сгруппируем выводы с аналогичными функциями.

Выводы питания – на плате расположено четыре вывода питания, а именно: один вывод VIN и три вывода 3.3V. Если у вас есть стабилизированный источник напряжения 5 В, вывод VIN можно использовать для непосредственного питания ESP8266 и его периферии. Выводы 3.3V – это выходы встроенного стабилизатора напряжения. Эти выводы могут использоваться для подачи питания на внешние компоненты. GND – это вывод земли отладочной платы ESP8266 NodeMCU.

Выводы I2C используются для подключения всех видов датчиков и периферийных устройств на шине I2C в вашем проекте. Поддерживаются и I2C Master, и I2C Slave.

Работа интерфейса I2C может быть реализована программно, а тактовая частота составляет максимум 100 кГц. Следует отметить, что тактовая частота I2C должна быть выше самой низкой тактовой частоты из ведомых устройств.

Выводы GPIO. На ESP8266 NodeMCU имеется 17 выводов GPIO, которые можно назначать программно на различные функции, такие как I2C, I2S, UART, PWM, дистанционное инфракрасное управление, светодиодный индикатор и кнопка. Каждый включенный вывод GPIO может быть настроен либо на внутреннюю подтяжку к земле или к шине питания, либо установлен на высокоимпедансное состояние. При конфигурировании на вход для генерирования прерываний процессора он может быть настроен на срабатывание либо по фронту, либо по спаду.

Вывод ADC подает сигнал на имеющийся в NodeMCU, встроенный 10-разрядный прецизионный аналого-цифровой преобразователь последовательного приближения (SAR ADC). С помощью этого АЦП могут быть реализованы две функции: проверка напряжения питания на выводе VDD3P3 и проверка входного напряжения на выводе TOUT (но не одновременно).

Выводы UART ESP8266 NodeMCU имеет 2 интерфейса UART, то есть UART0 и UART1, которые обеспечивают асинхронную связь (RS232 и RS485) и могут обмениваться данными со скоростью до 4,5 Мбит/с. Для связи можно использовать UART0 (выводы TXD0, RXD0, RST0 и CTS0), который поддерживает управление потоком. UART1 (вывод TXD1) поддерживает только сигнал передачи данных, поэтому он обычно используется для печати журнала событий.

Выводы SPI ESP8266 имеет два интерфейса SPI (SPI и HSPI), поддерживающих и ведомый (slave), и ведущий (master) режимы. Эти интерфейсы SPI также поддерживают следующие функции SPI:

- 4 режима синхронизации передачи SPI;
- до 80 МГц и тактовые частоты, полученные делением 80 МГц;
- до 64 байт FIFO.

Вывод**ы SDIO ESP8266** имеет защищенный цифровой интерфейс ввода/вывода (SDIO, Secure Digital Input/Output Interface), который используется для прямого подключения карт SD. Поддерживаются 4-битный 25 МГц SDIO v1.1 и 4-битный 50 МГц SDIO v2.0.

Выводы PWM. На плате имеется 4 канала широтно-импульсной модуляции (PWM). Выход ШИМ может быть реализован программно, и использован для управления двигателями и светодиодами. Частотный диапазон ШИМ регулируется от 1000 мкс до 10000 мкс, то есть от 100 Гц до 1 кГц.

Выводы управления используются, как ни странно, для управления ESP8266. Эти выводы включают в себя вывод включения микросхемы EN, вывод сброса RST и вывод пробуждения WAKE.

Вывод EN – микросхема ESP8266 включена, когда на вывод EN подается высокий логический уровень. При низком логическом уровне микросхема работает на минимальной мощности.

Вывод RST используется для сброса микросхемы ESP8266.

Вывод WAKE используется для вывода чипа из глубокого сна.

"Обзор платы NodeMCU ESP8266 и ее использование в Arduino IDE"
Электронный портал: radioprog.ru

Сетевая инфраструктура

Типовое применение ESP8266 как аппаратной основы Internet of Things чаще всего подразумевает установку в домах или офисах. При этом сетевое подключение осуществляется к домашней/офисной локальной сети с выходом в интернет через роутер. Пользователь устройства может контролировать его с помощью планшета или компьютера через свою локальную сеть либо удаленно, через Интернет.

ESP8266 может работать как в роли точки доступа так и оконечной станции. При нормальной работе в локальной сети ESP8266 конфигурируется в режим оконечной станции. Для этого устройству необходимо задать SSID Wi-Fi сети и, в закрытых сетях, пароль доступа. Для первоначального конфигурирования этих параметров удобен режим точки доступа. В режиме точки доступа устройство видно при стандартном поиске сетей в планшетах и компьютерах.

Остается подключиться к устройству, открыть HTML страничку конфигурирования и задать сетевые параметры. После чего устройство штатно подключится к локальной сети в режиме оконечной станции.

В случае исключительно местного использования возможно всегда оставлять устройство в режиме точки доступа, что снижает необходимые усилия пользователя по его настройке.

После подключения к Wi-Fi сети устройство должно получить IP-параметры локальной сети. Эти параметры можно задать вручную вместе с параметрами Wi-Fi либо активизировать какие-либо сервисы автоматического конфигурирования IP-параметров (например, DHCP).

После настройки IP параметров обращение к серверу устройства в локальной сети обычно осуществляется по его IP адресу, сетевому имени (в случае если имена поддержаны какой-либо технологией, например, NBNS) или сервису (в случае если поддержан автоматический поиск сервисов, например через протокол SSDP).

Зачастую доступ к устройству требуется из Интернета.

Например, пользователь с мобильного телефона удаленно проверяет состояние своего «умного дома», обращаясь напрямую к устройству. В этом случае устройство работает в режиме сервера, к которому обращается внешний клиент.

Как правило, устройство на основе ESP8266 находится в локальной сети офиса или дома. Выход в Интернет обеспечивает роутер, подключенный с одной стороны к локальной сети, а с другой, к сети провайдера интернета. Провайдер назначает роутеру свой статический или динамический IP адрес и роутер осуществляет трансляцию адресов локальной сети в сеть провайдера. По умолчанию правила этой трансляции обеспечивают свободную видимость интернет-адресов из локальной сети, но не позволяют обратиться к локальным адресам со стороны Интернета. Есть несколько способов обойти это ограничение.

Конфигурирование NAT

Большинство современных роутеров позволяют задать дополнительные правила трансляции сетевых адресов между локальной и глобальной сетями. Как правило для этого используются технологии Virtual server или DMZ. Обе технологии позволяют обратиться к серверу в локальной сети из глобальной сети, зная лишь IP адрес, выданный роутеру провайдером.

В случае статического IP адреса роутера – это, зачастую, может быть удовлетворительным решением для ограниченного круга пользователей системы. Однако такой подход не всегда удобен: необходимо вручную конфигурировать роутер и выяснять IP-адрес роутера, который может регулярно меняться. Относительно легко решить проблему неизвестного IP адреса можно с помощью механизма DDNS.

DDNS

Чтобы обратиться к серверу устройства конечный пользователь должен знать IP адрес, по которому находится устройство. Однако получить у провайдера Интернета для устройства статический IP адрес не всегда возможно, да и пользоваться таким адресом неудобно. Для решения этой проблемы были созданы специальные интернет-сервисы под общим наименованием динамический DNS. Эти сервисы работают как специальные серверы с фиксированными именами в интернете. Разработчик заводит на таком сервисе свой аккаунт с уникальным именем. Параметры этого аккаунта он прописывает в устройстве. Устройство в режиме клиента периодически обращается к серверу сервиса, сообщая ему имя своего аккаунта и свой текущий IP адрес. Конечный пользователь в интернете обращается к этому же сервису и получает от него текущие IP параметры устройства.

В таком случае устройство в сети видно с доменным именем третьего уровня, например, esp8266.ddns.org. Существует множество DDNS сервисов. Основная проблема DDNS сервисов это гарантии существования конкретного сервиса. Как правило, гарантируется только коммерческий сервис, когда за его использование взимается плата.

Внешние IoT сервисы для ESP8266

Чтобы облегчить проблему доступности устройства в Интернете и сделать инсталляцию устройства легкой для пользователя были разработан ряд решений. Механизм этих решений базируется на существовании в Интернете специального сервера, к которому может подключиться как IoT устройство, так и планшет/компьютер пользователя.

При этом устройство работает в режиме клиента, никаких специальных настроек роутера или особых навыков от инсталлятора и пользователя устройства не требуется. Обмен данными с устройством осуществляется при посредничестве этого специального сервиса, параметры которого в устройство должен заложить разработчик.

Распространение использования таких сервисов сдерживается необходимостью длительно поддерживать свой сервис в Интернете или пользоваться чужими сервисами с непонятными перспективами длительного существования бесплатных возможностей или регулярной оплатой коммерческих вариантов.

Internet of Things

Основное применение ESP8266 находит в управлении разнообразными бытовыми приборами через беспроводные сети. Концепцию такого управления часто называют «Internet of Things» (IoT, «интернет вещей»). **Верхний уровень IoT** представлен разнообразными приложениями под популярные платформы (Android, iOS, Windows, …). Эти приложения позволяют разработчику прибора адаптировать приложение под управление его прибором и передать пользователю готовое решение. Существует несколько популярных реализаций концепции IoT в плане обмена данными по сети:

HTML сервер на ESP8266: Контроль и управление устройством ведется через браузер. Тяжеловесное решение, подходит автономным устройствам автоматики.

AllJoyn — набирающий популярность открытый IoT протокол крупного альянса производителей цифровой техники «Allseen». Поддержка встроена в Windows 10. На русском можно почитать здесь.

HTTP запросы с использованием протоколов типа REST, XML-RPC (SOAP). Для этого на ESP8266 запускают упрощенный HTTP сервер, без HTML. Достоинство метода — отсутствие проблем с настройкой файрволлов, HTTP обычно открыт всегда.

MQTT. Это простой протокол поверх TCP/IP. Очень популярное решение. Существует большое количество IoT приложений верхнего уровня для Android, iOS и других платформ, поддерживающих этот протокол.

SNMP. Расширяемый протокол управления сетевыми устройствами. Основной недостаток в том, что в большинстве сетей файрволлы блокируют прохождение SNMP.

ModBus и другие протоколы промышленной автоматизации.

Интересные проекты ПО **верхнего уровня** с решениями на базе ESP8266:

Majordomo — русскоязычный открытый проект домашней автоматизации.

Blynk — облачная платформа для IoT, которая имеет приложения для iOS и Android и поддерживает управление микроконтроллерами ESP8266, Arduino, Raspberry Pi, SparkFun и д.р. через Интернет.

SUPLA — открытый проект систем автоматизации зданий, использующий ESP8266.

В рамках данного курса мы будем использовать плату **Witty Cloud (ESP8266)** на базе модуля ESP-12F.

Для работы Witty Cloud не нужен внешний микроконтроллер или другое управляющее устройство, так как помимо Wi-Fi модуля в ESP-12F уже встроен 32-битный микроконтроллер с тактовой частотой 80 МГц, а также чип флеш-памяти на 4МБ.

Модуль Witty Cloud - это отличное решения для использования в "интернете вещей", системах удаленного мониторинга или управления и т.д. Плата предлагает несколько вариантов работы с Wi-Fi сетями, в том числе может выступать и как клиент Wi-Fi сети, и сам создавать точку доступа.

Использование Witty Cloud **вместо "голого" ESP8266 модуля существенно упрощает работу с платформой**, так как в него уже встроены USB-UART конвертер CH340, стабилизатор напряжения, припаяна PLS-планка со стандартным шагом 2.54 мм, а также, с программой стороны - в чип ESP8266 залита прошивка со встроенным интерпретатором Lua.

Witty Cloud состоит из двух отдельных плат. На верхней расположен сам Wi-Fi модуль ESP8266, стабилизатор на 3.3В, RGB светодиод, фоторезистор и программируемая кнопка. На нижней - USB-UART конвертер на чипе CH340 с обвязкой.

С помощью программатора (нижней платы) можно прошивать и другие платы из серии ESP-12, установив их на совместимый по распиновке адаптер, и припаяв на него стабилизатор.

Технические характеристики: ESP32 против ESP8266

ESP32 является преемником **ESP8266**. Он имеет дополнительное ядро процессора, более быстрый Wi-Fi, больше GPIO и поддерживает Bluetooth 4.2 и Bluetooth с низким энергопотреблением. Кроме того, ESP32 поставляется с сенсорными контактами, которые можно использовать для пробуждения ESP32 из глубокого сна, встроенным датчиком эффекта Холла и встроенным датчиком температуры (последние версии ESP32 больше не поставляются со встроенным датчиком температуры).

Обе платы очень дешевы, но ESP32 стоит немного дороже. В то время как ESP32 может стоить от 6 до 12 долларов, ESP8266 может стоить от 4 до 6 долларов (но это зависит от того, где вы их приобретаете).

В следующей таблице показаны основные различия между чипами ESP8266 и ESP32:

Specifications	ESP8266	ESP32
MCU	Xtensa® Single-Core 32-bit L106	Xtensa® Dual-Core 32-bit LX6 600 DMIPS
802.11 b/g/n Wi-Fi	Yes, HT20	Yes, HT40
Bluetooth	None	Bluetooth 4.2 and below
Typical Frequency	80 MHz	160 MHz
SRAM	160 kBytes	512 kBytes
Flash	SPI Flash , up to 16 MBytes	SPI Flash , up to 16 MBytes
GPIO	17	36
Hardware / Software PWM	None / 8 Channels	1 / 16 Channels
SPI / I2C / I2S / UART	2/1/2/2	4/2/2/2
ADC	10-bit	12-bit
CAN	None	1
Ethernet MAC Interface	None	1
Touch Sensor	None	Yes
Temperature Sensor	None	Yes
Working Temperature	-40℃ ~ 125℃	-40℃ ~ 125℃

Рис. 38 – Основные различия между чипами ESP8266 и ESP32

Использовать просто чипы ESP32 или ESP8266 сложно и непрактично, особенно при тестировании и создании прототипов. В большинстве случаев вы захотите использовать платы разработки ESP32 и ESP8266.

Эти платы поставляются со всеми необходимыми схемами для питания чипа, подключения его к компьютеру, схемой для легкой загрузки кода, контактами для подключения периферийных устройств, встроенными светодиодами питания и управления и другими полезными функциями.

Платы для разработки ESP32 и ESP8266, которые используют чаще всего — это плата разработки ESP32 DEVKIT DOIT и комплект ESP8266 ESP-12E NodeMCU Kit. Однако есть много других моделей плат для разработки, из которых вы можете выбирать.

В систему интегрирован радиочастотный тракт: балун (симметрирующий трансформатор), встроенные антенные коммутаторы, радиочастотные компоненты, малошумящий усилитель, усилитель мощности, фильтры и модули управления питанием.

ESP32 создан и разработан компанией Espressif Systems, китайской компанией, расположенной в Шанхае, а производится компанией TSMC по техпроцессу 40 нм.

ESP32 имеет большее количество GPIO чем ESP8266. Вы можете сами решать, какими выводами будут UART, I2C, SPI – для этого вам просто нужно прописать их в коде. Это возможно благодаря функции мультиплексирования микросхемы ESP32, которая позволяет назначать несколько функций одному и тому же выводу.

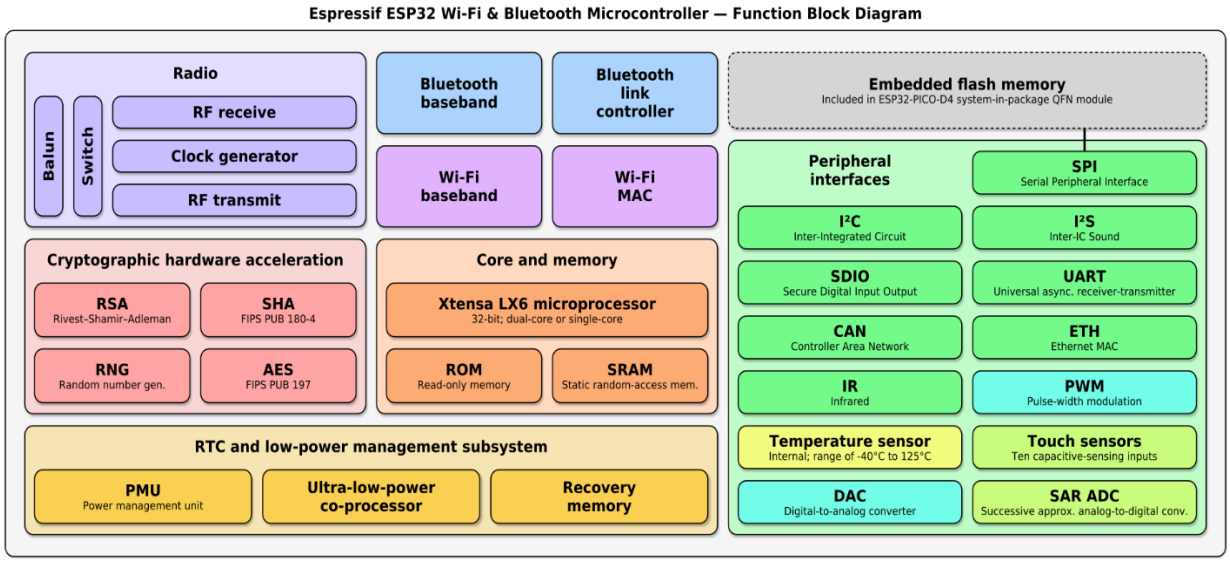


Рис. 39 – Функциональная блок-схема ESP32

§ ПРАКТИКУМ: СОЗДАНИЕ WEB-ИНТЕРФЕЙСА НАСТРОЙКИ ACCESS POINT ESP8266 В FLProg.

Оборудование: персональный компьютер, кабель AM/microBM 5p Cablexpert Pro (CCP-mUSB2-AMBM-1.0M или аналоги), микропроцессорная система ESP8266 (Witty Cloud ESP12F).

Программное обеспечение: IDE FLProg, IDE Arduino.

Цель: научиться реализовывать конечные проекты встраиваемых компьютерных систем посредством среды визуального программированию, овладеть навыками визуального программирования и конфигурирования микропроцессорных систем посредством Wi-fi.

Ход работы:

Работа над проектом начинается с выбора контроллера.

ESP8266 -> Boards -> ESP8266 NodeMCU v3

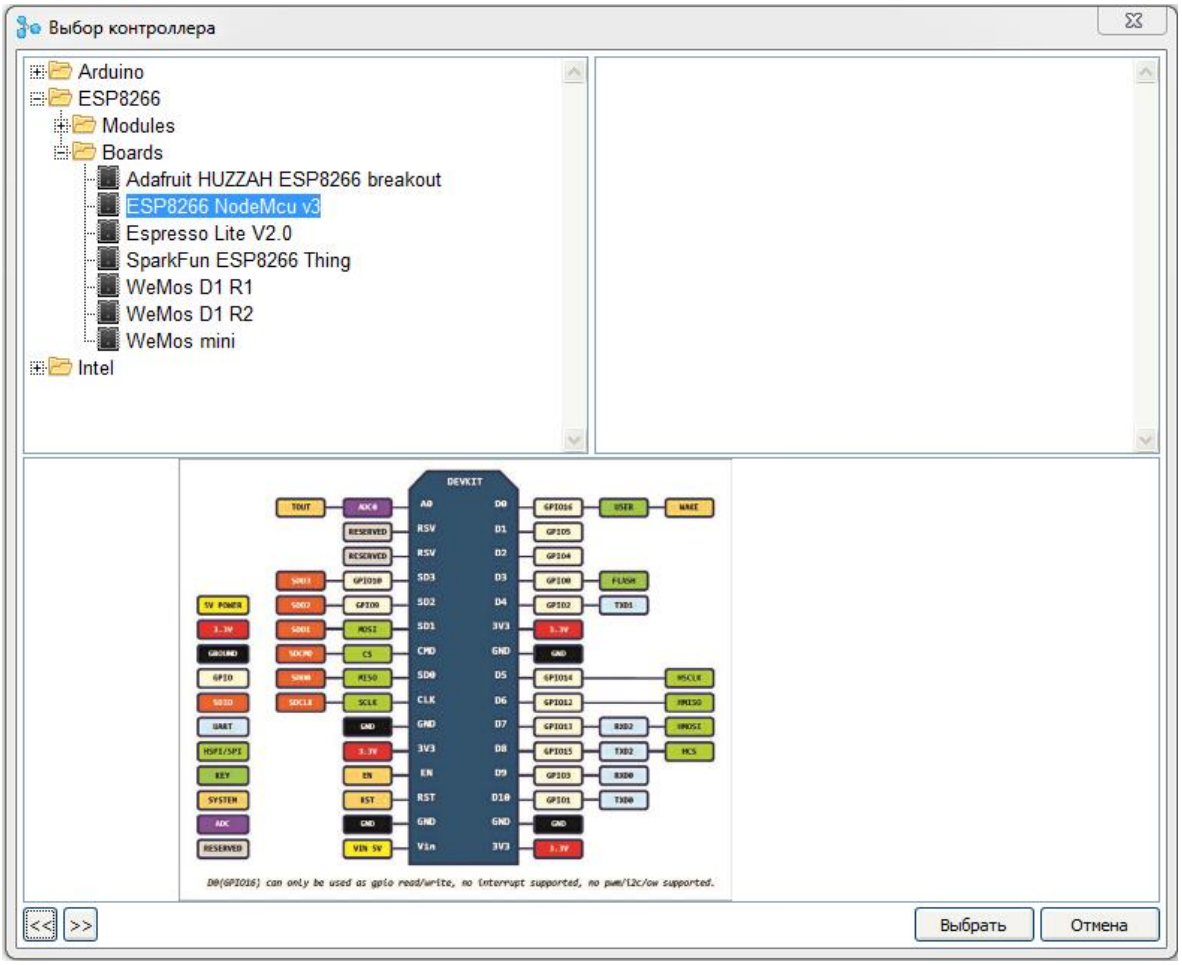


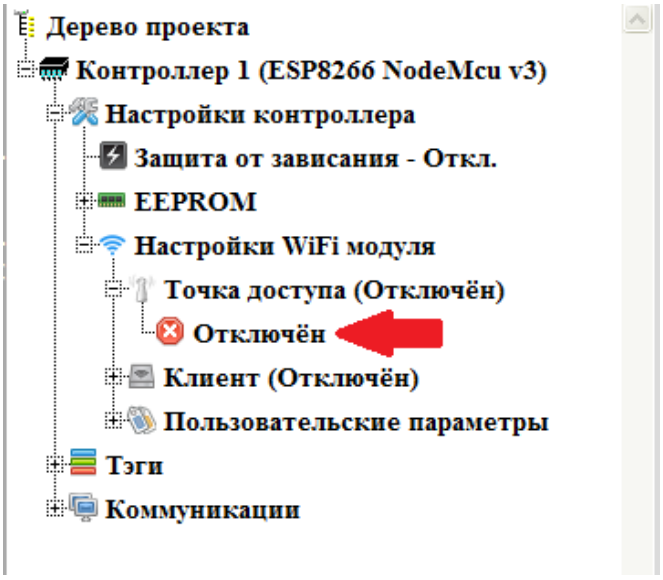
Рис. 40 – плата Witty Cloud является копией ESP8266 NodeMCU v3

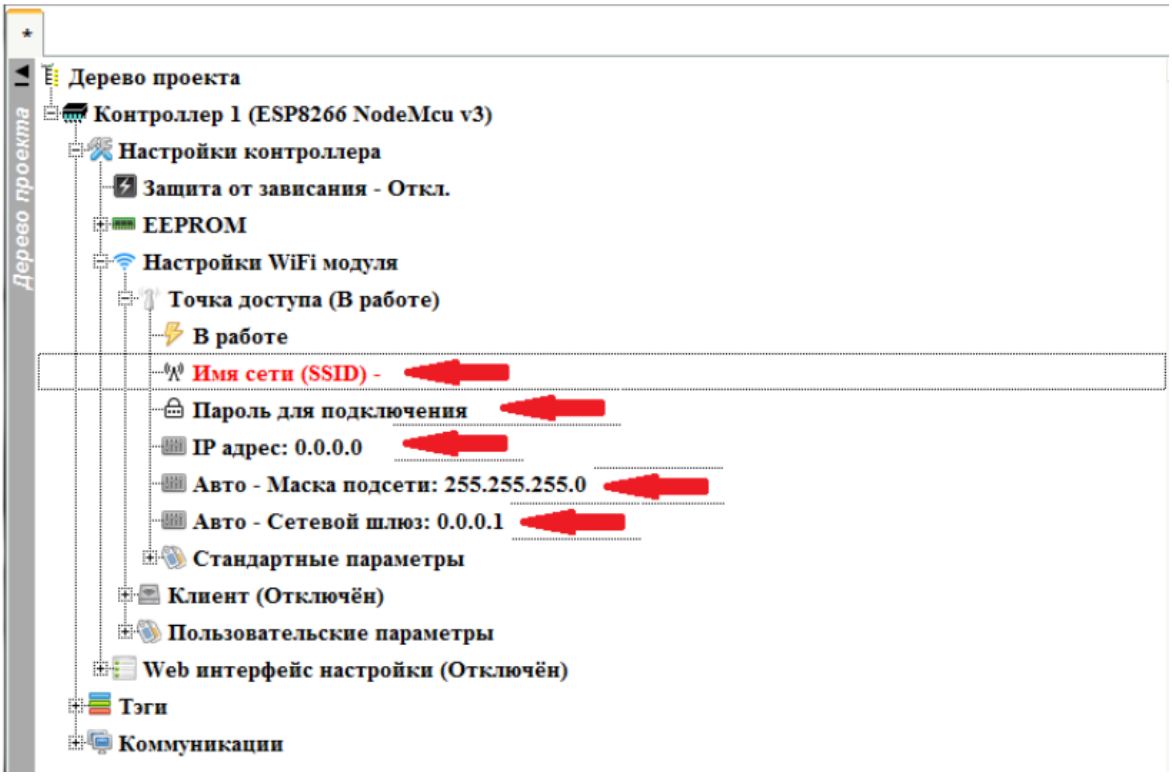
При выборе конкретного контроллера или платы можно посмотреть его изображение, распиновку, а также технические характеристики.

Основная работа при создании web-интерфейса настройки производится в дереве проекта.

Для начала настроим точку доступа. Раскрываем дерево проекта до пункта «Точка доступа» включительно и двойным кликом по ветке «Отключён» включаем точку доступа в работу.

В открывшихся ветках настраиваем параметры точки доступа. Для изменения необходимого параметра производим двойной клик на соответствующей ветке.





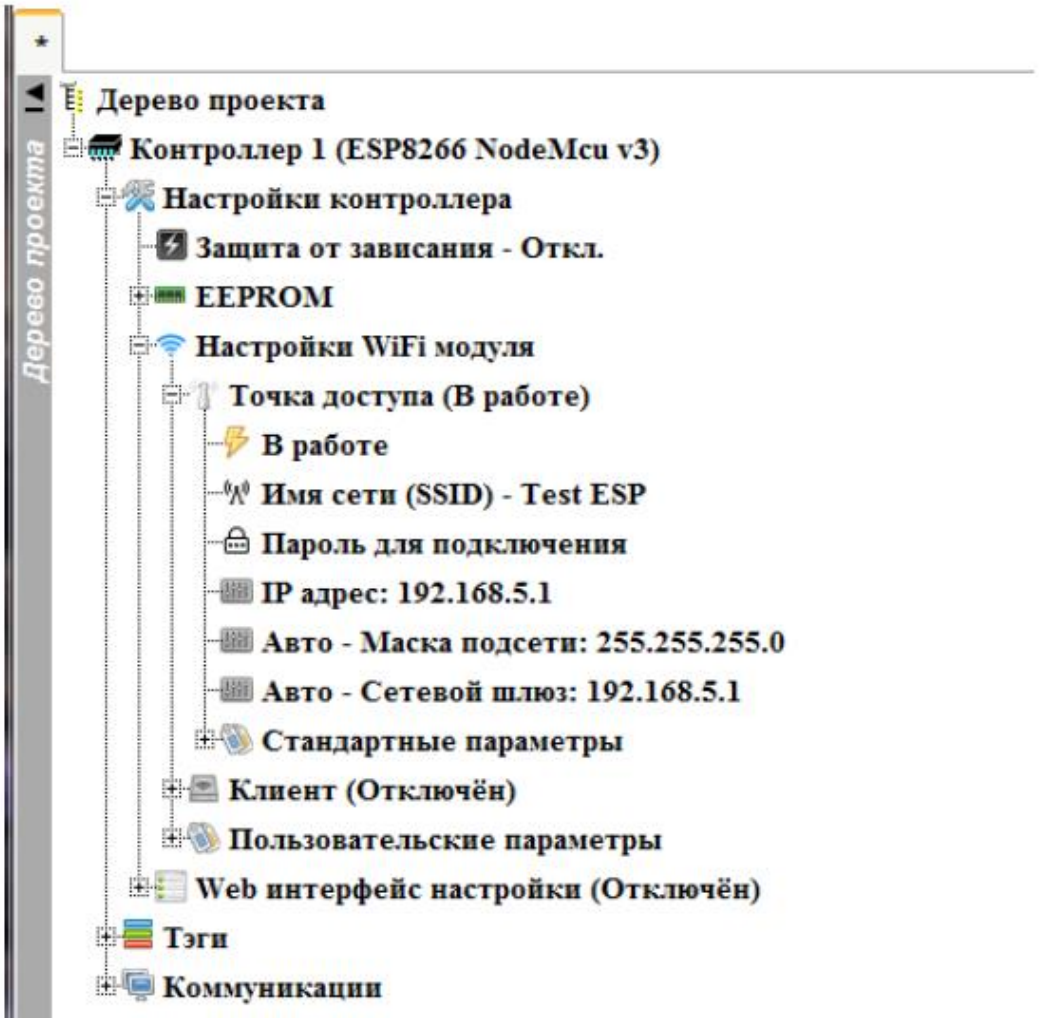
Имя сети (SSID) – Имя сети которую будет организовывать точка доступа.

Пароль для подключения – пароль для подключения к точке доступа. Если оставить пустым, то точка доступа будет без пароля со свободным подключением.

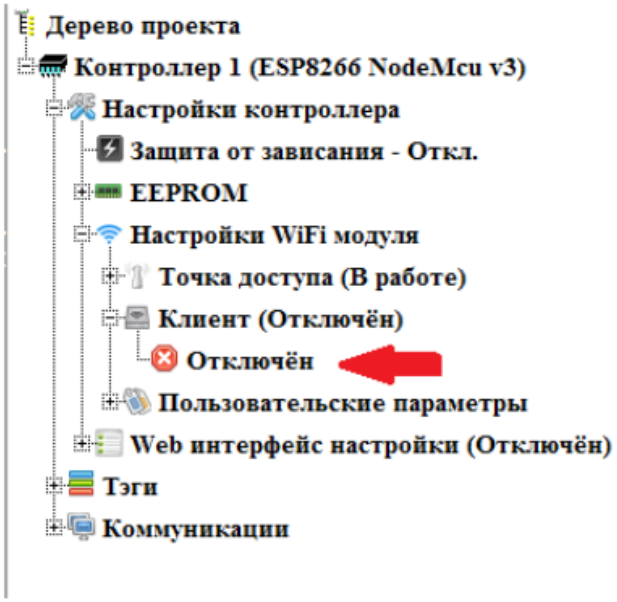
IP адрес – IP адрес который будет иметь контроллер в сети созданным точкой доступа. По этому адресу потом можно будет подключиться к контроллеру.

Остальные параметры (Маска подсети и Сетевой шлюз) заполнятся автоматически после установки IP адреса, но при необходимости их можно изменить, если требуются нестандартные значения.

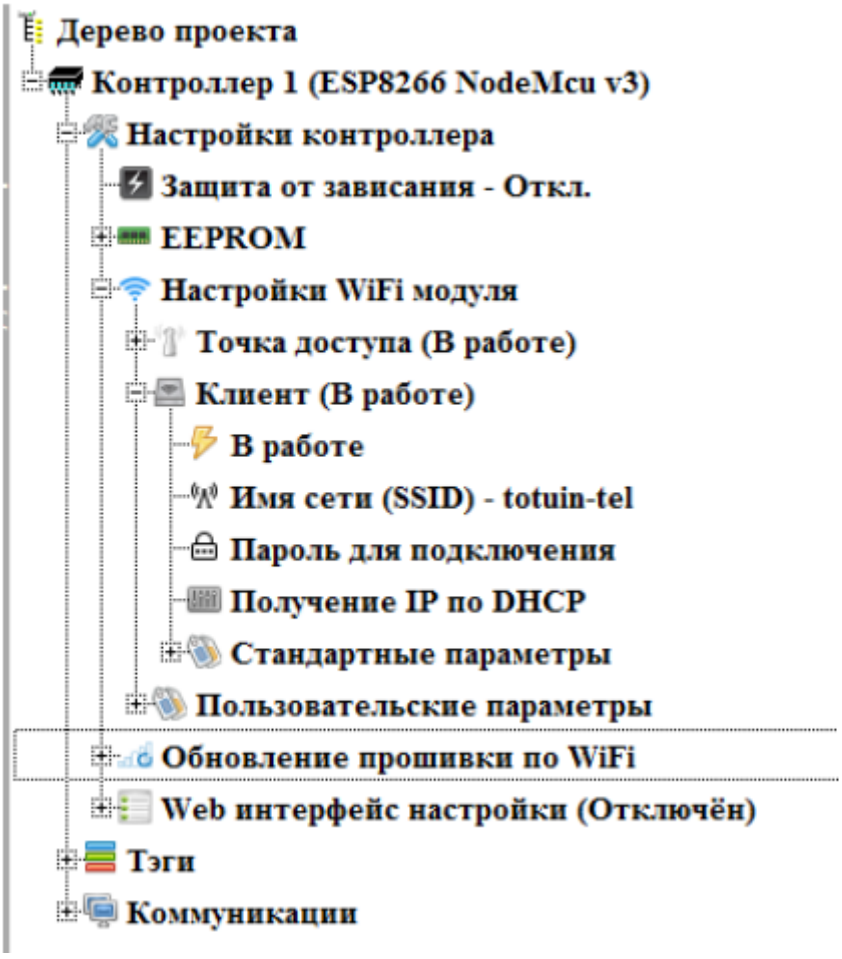
В результате должно получиться, что-то схожее:



С точкой доступа закончили, можем свернуть этот узел, и переходим к клиенту. Так же разворачиваем его узел, и включаем его в работу двойным кликом по ветке «Отключён».



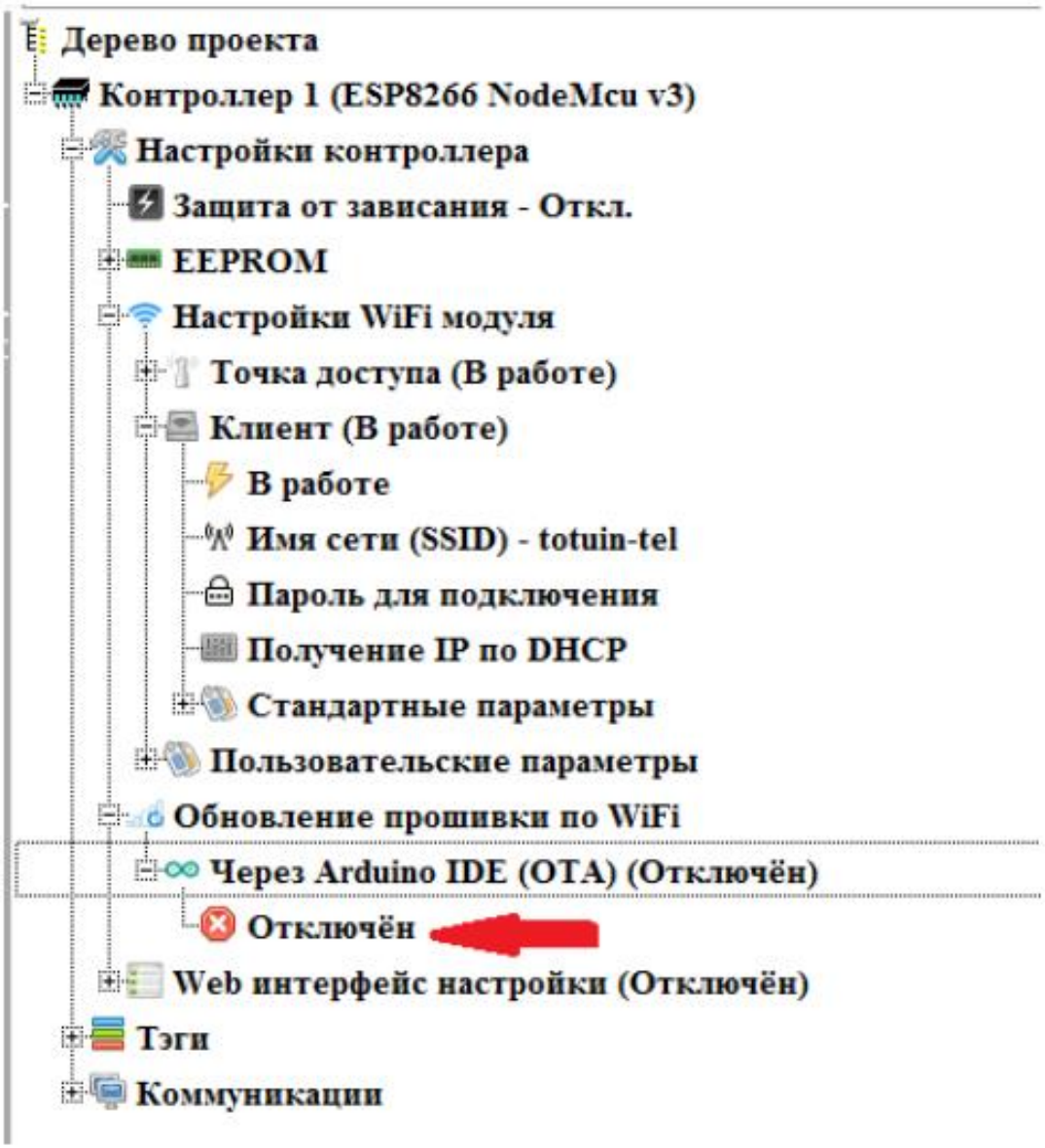
Настраиваем клиента. Возможны два варианта настройки клиента. Непосредственное задание настроек сети, и получение настроек по DHCP. Для начала используем второй вариант.



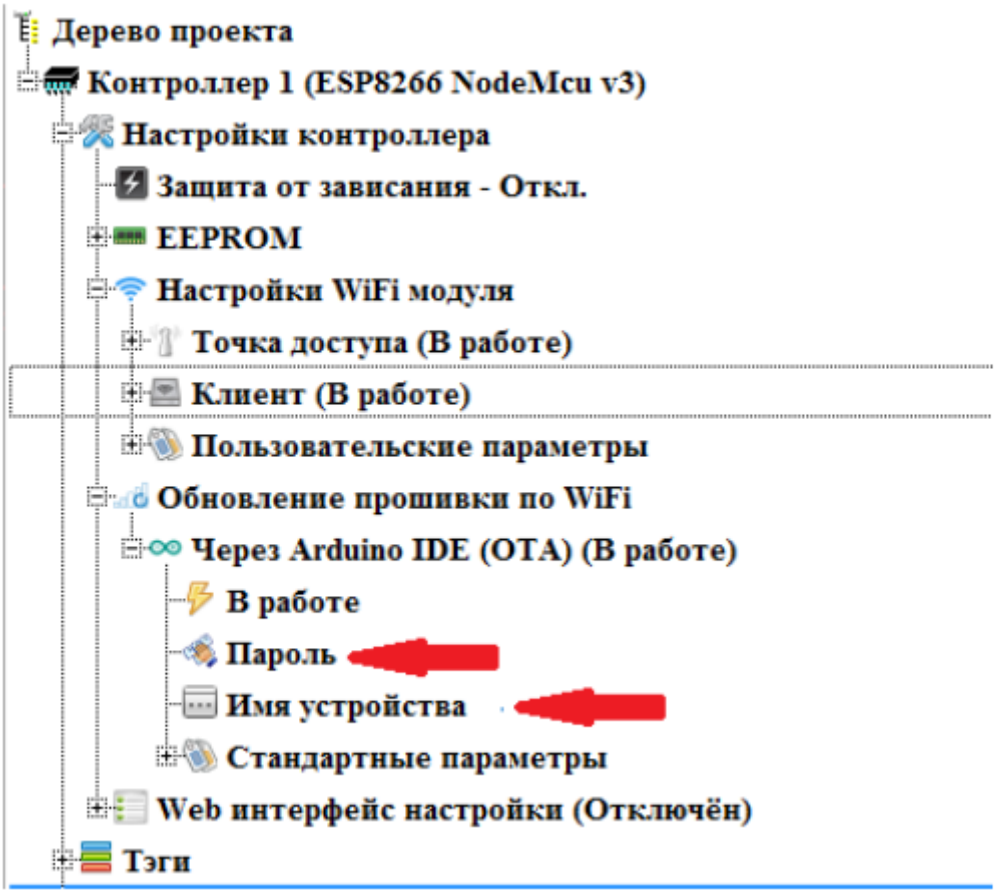
Обратите внимание, что в целях безопасности пароль подключения в дереве проекта не показывается.

С настройкой Wi-Fi интерфейсов закончили. Сворачиваем (если нужно) узел клиента и переходим к настройке режима обновления прошивки по Wi-Fi (если это требуется). Данный узел появляется только при включённом в работу клиенте.

Включаем этот режим двойным кликом по ветке “**Отключён**”



Задаём необходимые параметры (изменение значения параметров производится с помощью двойного клика на соответствующей ветке).

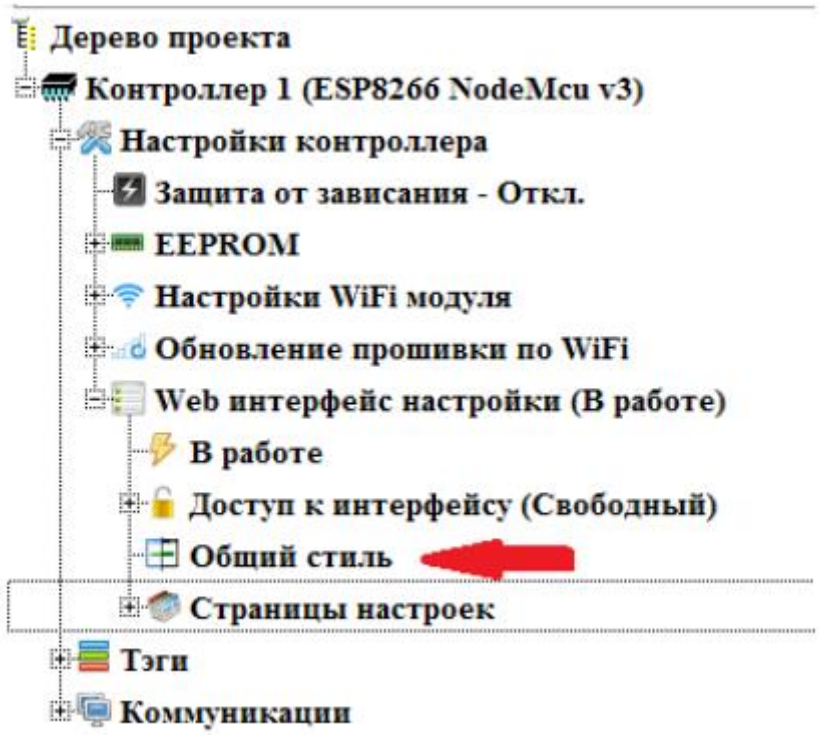


Пароль – при задании пароля перед заливкой новой прошивки потребуется его ввод.

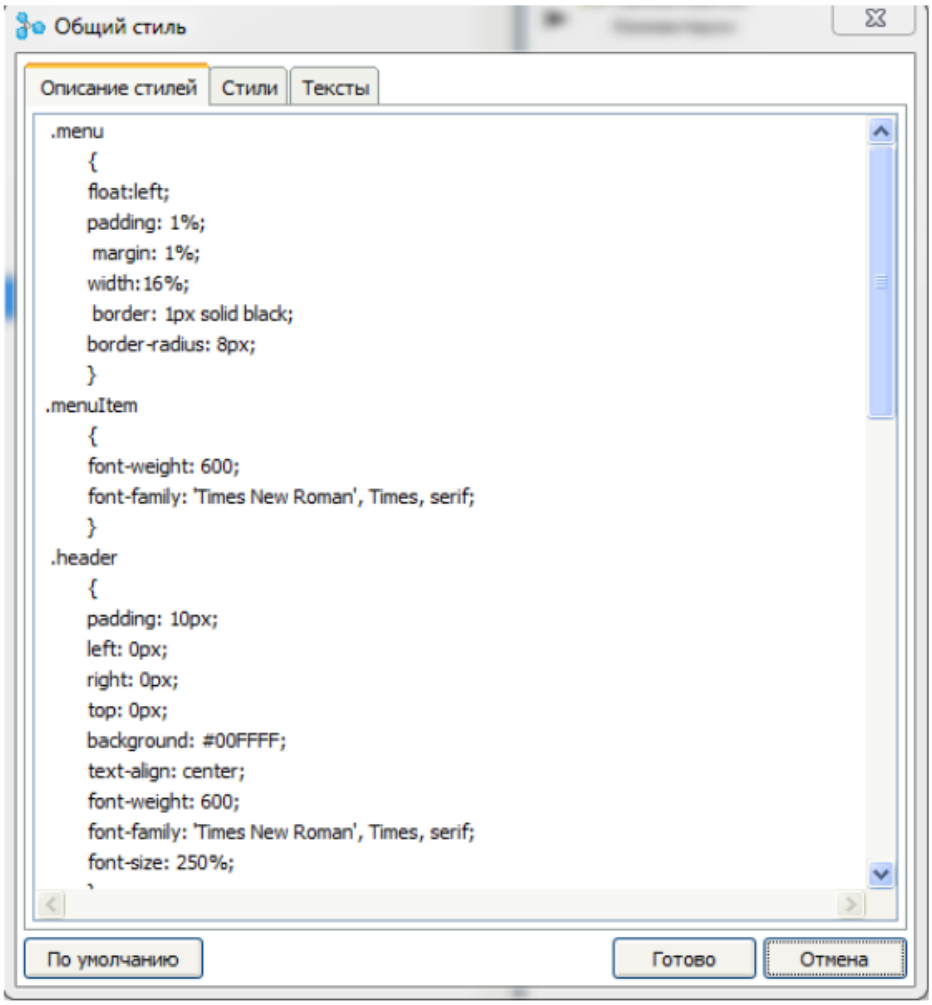
Имя устройства – Это имя будет фигурировать в названии порта соединения в Arduino IDE.

Теперь переходим непосредственно к созданию web интерфейса настройки. Открываем узел “Web интерфейс настройки” и включаем его двойным кликом по ветке “Отключён”.

Web интерфейс настроек представляет собой набор страниц с параметрами. Если страниц более одной, автоматически формируется меню для доступа к ним. Для каждой страницы можно задать собственные стили CSS, если использовать общие стили для всего Web интерфейса настройки.
Для настройки общих стилей CSS для всего web интерфейса совершаем двойной клик по ветке «Общий стиль».

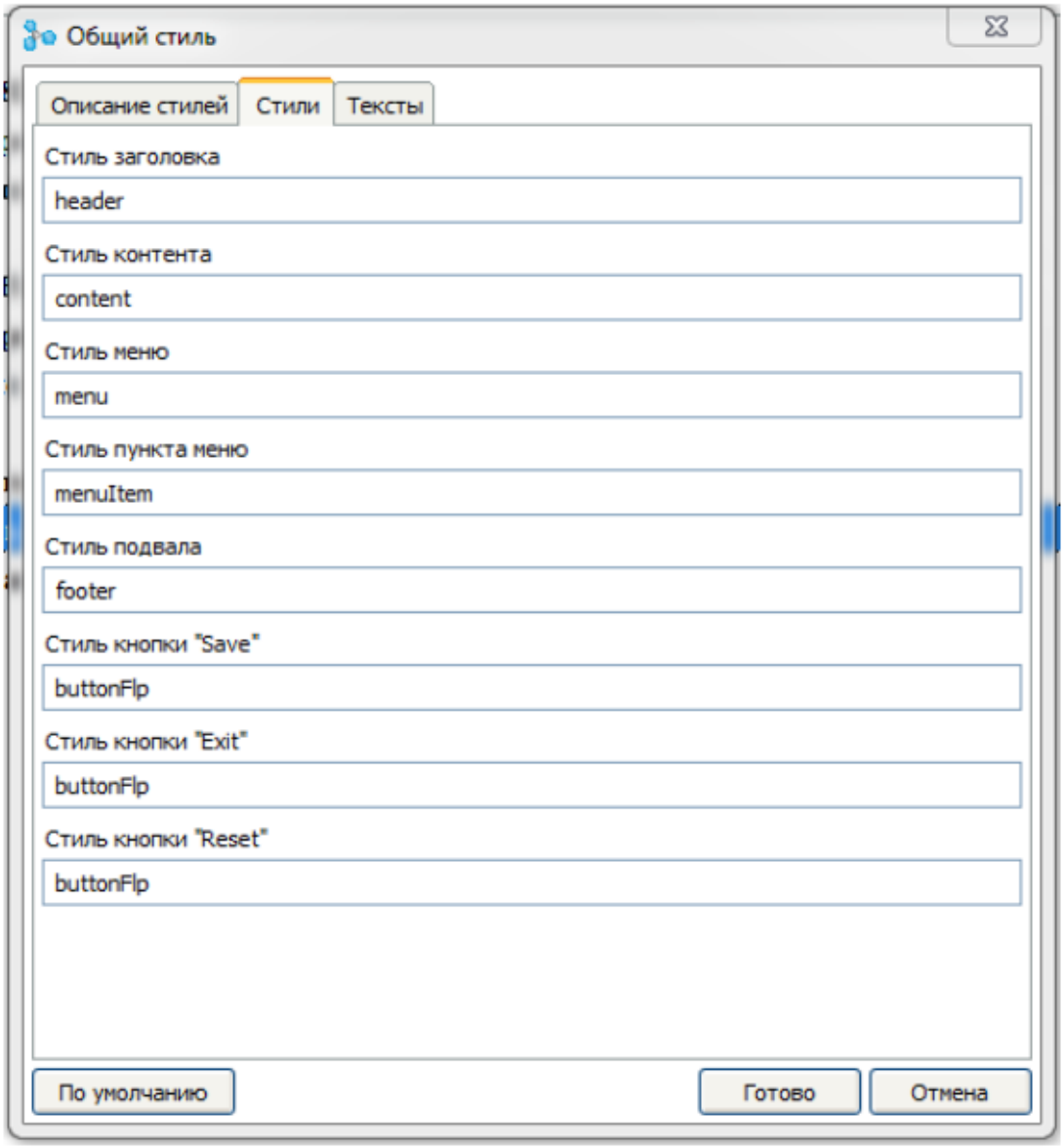


Откроется окно настроек общего стиля.

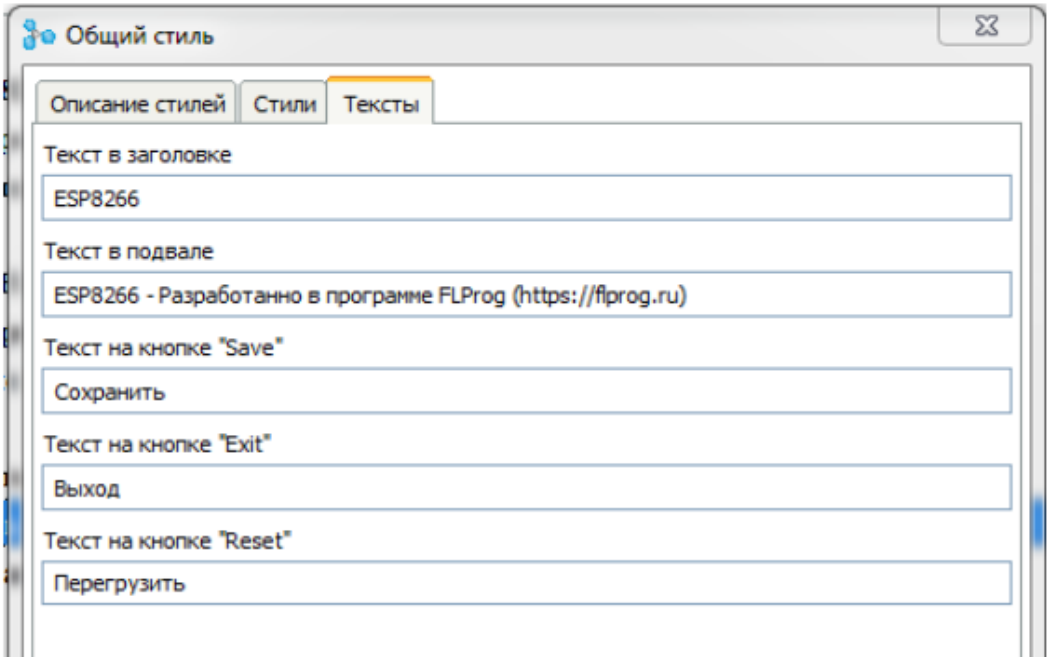


На закладке «Описание стилей» находится поле ввода непосредственно описания стилей применяемых для всех страниц настройки. По умолчанию это поле уже заполнено стилями для создания стандартного интерфейса. Но если есть желание изменить дизайн страниц, то можно их изменить.

На закладке «Стили» можно задать названия стилей, используемых для конкретных элементов страницы.

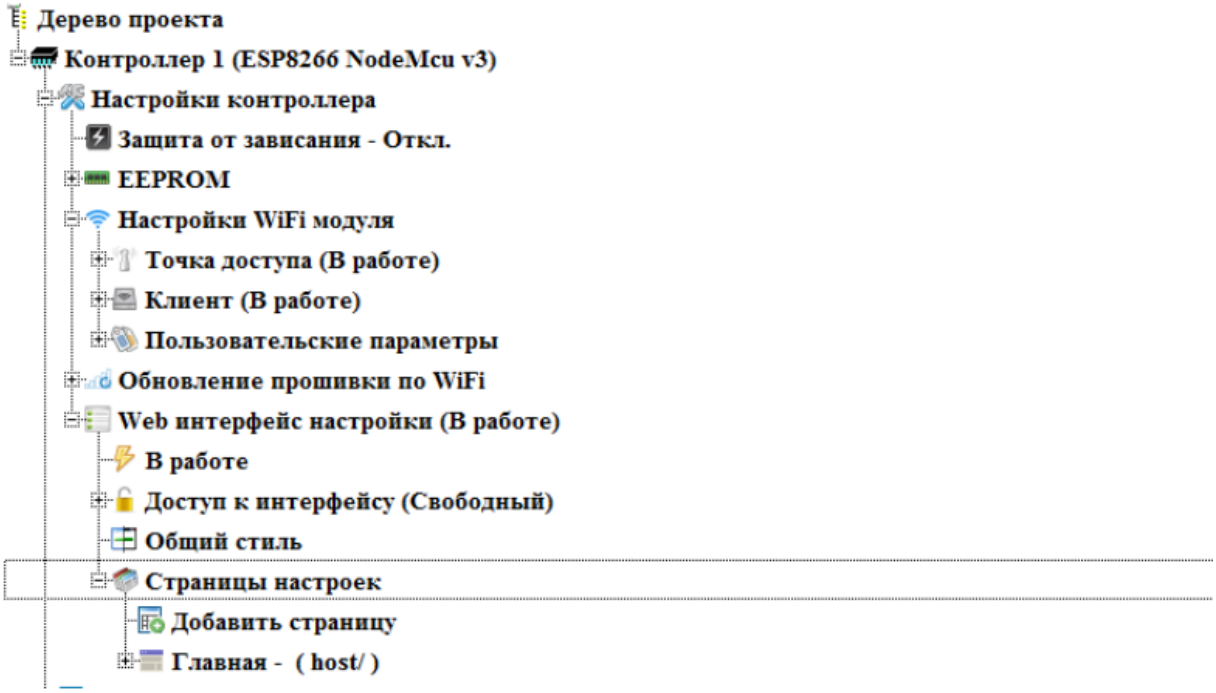


Эта закладка так же заполнена по умолчанию. На закладке «Тексты» можно ввести тексты основных элементов, используемых на странице.

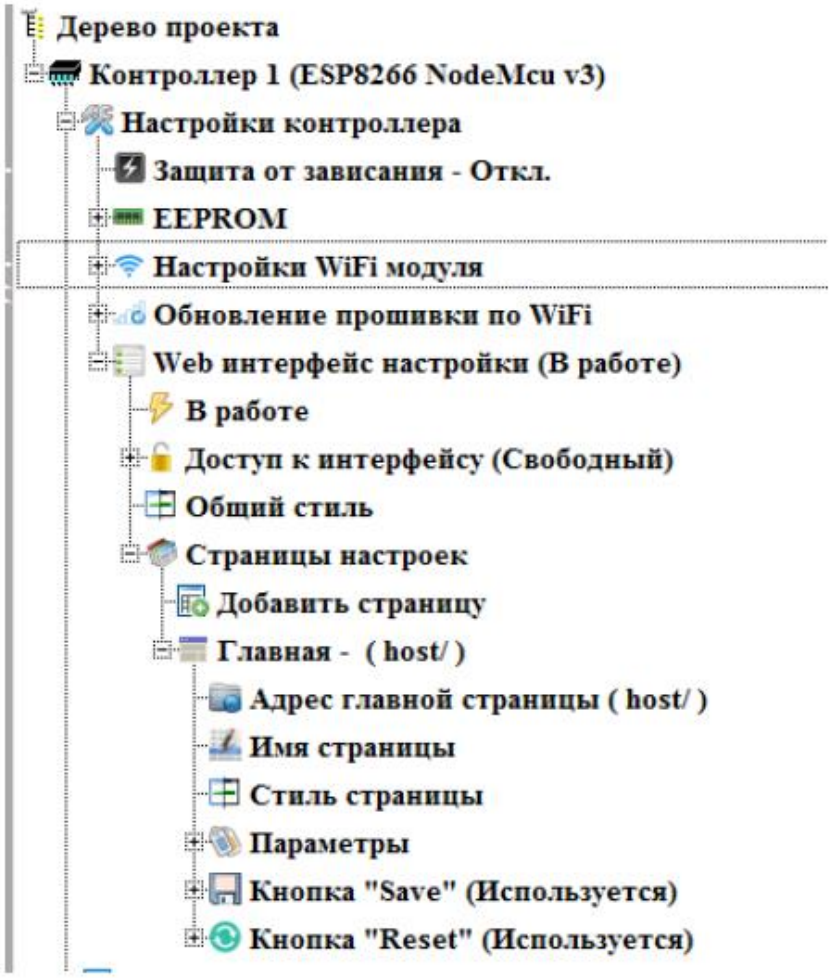


Для восстановления всех значений данного диалога значениями по умолчанию, можно воспользоваться кнопкой «По умолчанию». Настройки стилей и текстов, заданные в общих стилей, применяются на всех страницах настроек, если они не перекрыты настройками стилей конкретной страницей (это рассмотрим ниже)

Страницы показаны в узле «Страницы настроек». Раскрываем его:



По умолчанию всегда присутствует одна страница. Раскрыв её узел, получаем доступ к её настройкам.



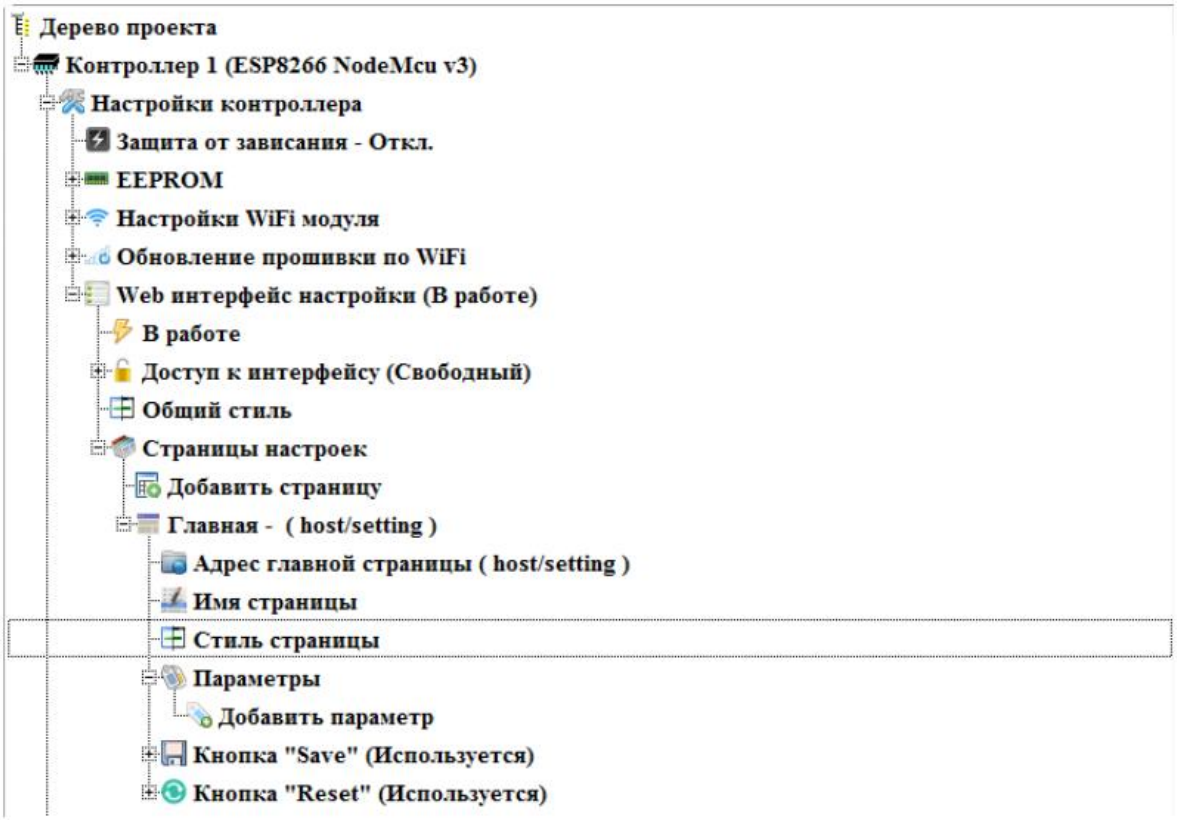
Адрес главной страницы – адрес основной страницы настроек. По умолчанию – host – то есть адрес контроллера в сети. При необходимости можно сменить. Сменим его на адрес host/setting (двойной клик по данной ветке).

Имя страницы – название страницы в меню. Оставим ей название – "Главная".

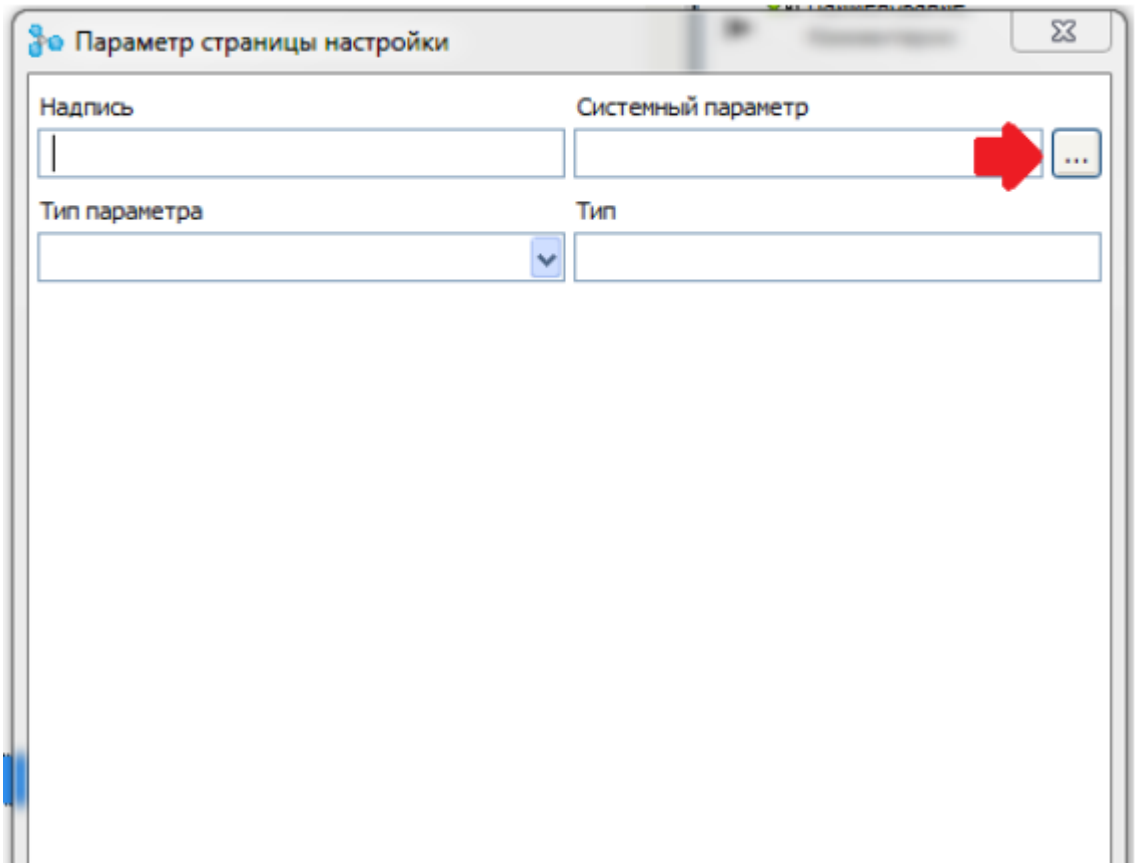
Стиль страницы — двойной клик по данной ветке вызывает диалог настройки стиля для конкретно этой страницы.

В этом диалоге можно дописать дополнительные стили CSS для данной страницы и назначить стили и тексты для элементов дизайна. Так же можно переопреописать стили, описанные в диалоге общих стилей.

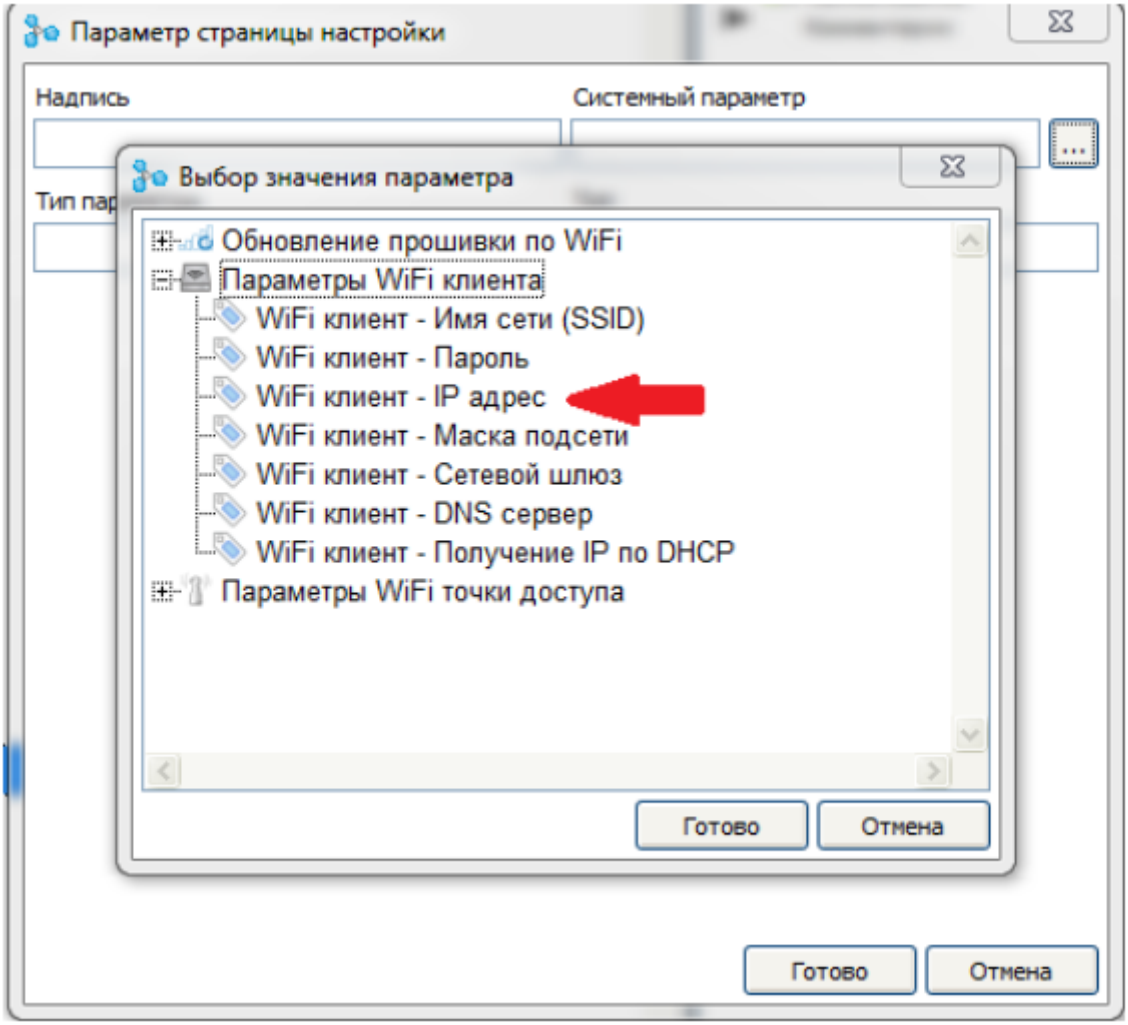
В узле “**Параметры**” задаются параметры, отображаемые на странице. На главной странице мы зададим отображение IP адреса полученного от роутера по DHCP в виде простого текста. Параметр добавляется с помощью двойного клика по ветке «Добавить параметр».



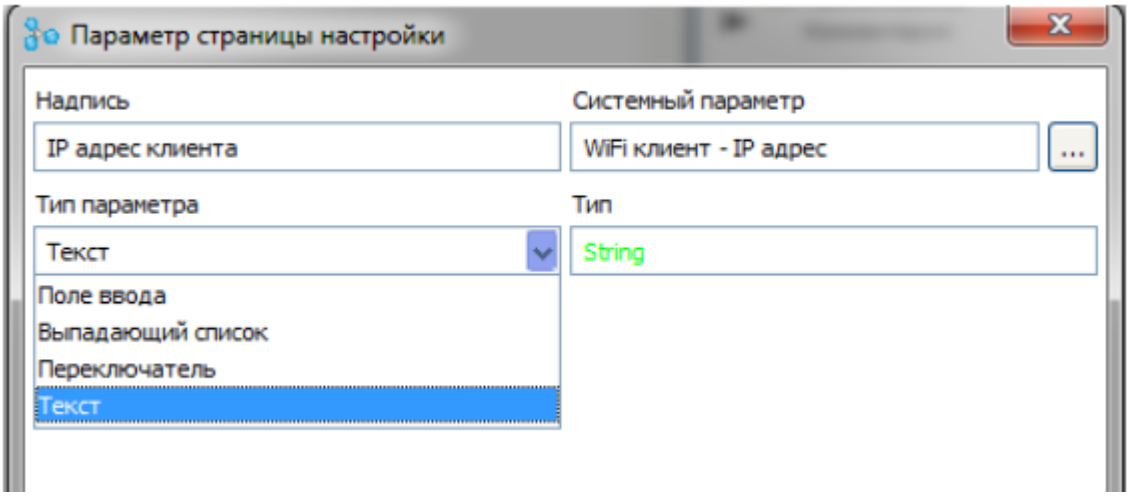
Открывается диалог создания нового параметра. В нём для начала нажимаем кнопку выбора системного параметра.



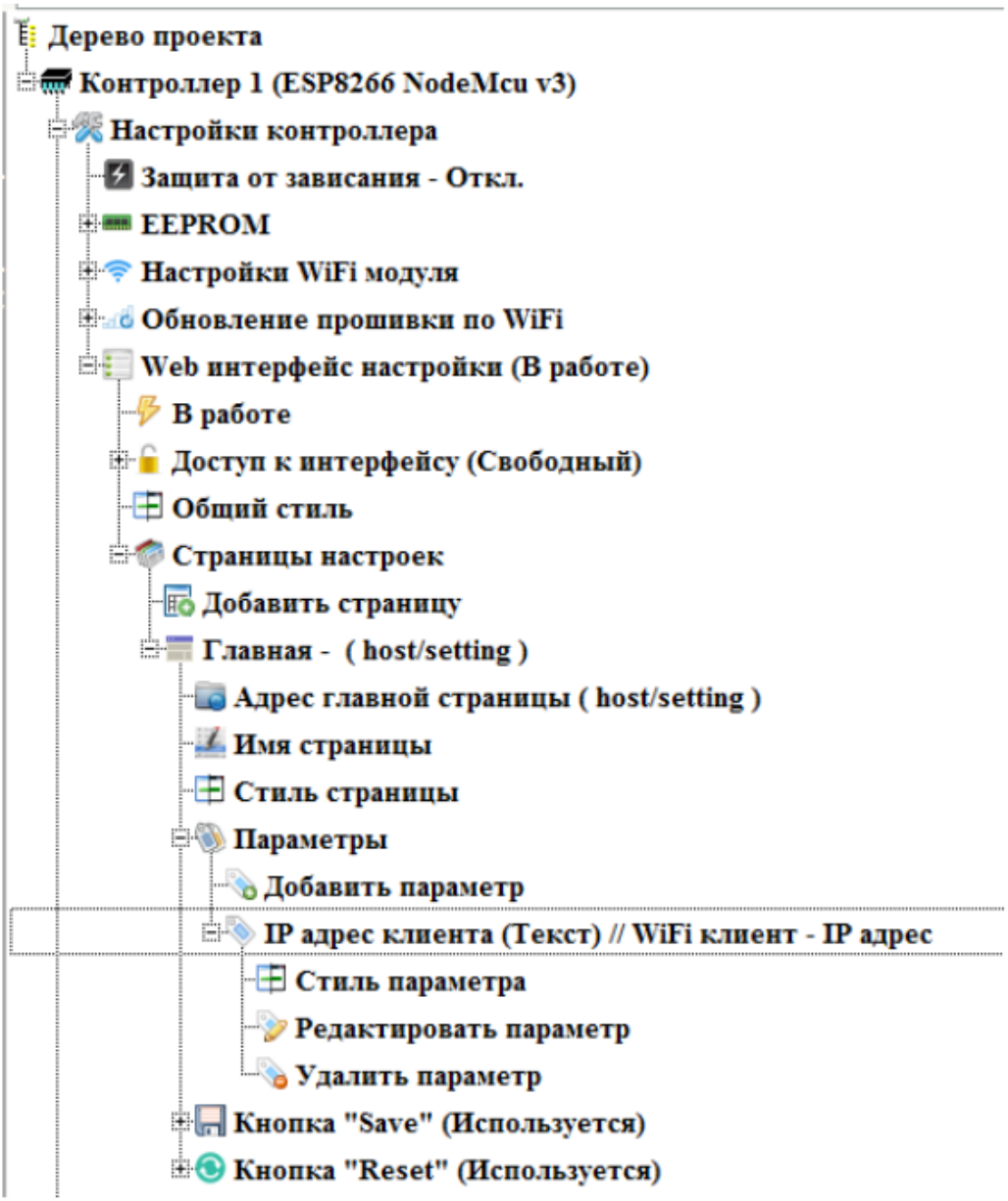
Откроется список доступных системных параметров. Выбираем параметр “Wi-Fi клиент – IP адрес”.



В поле надписи вводим текст лейблы для этого параметра, а в поле тип параметра выберем значение “Текст”.

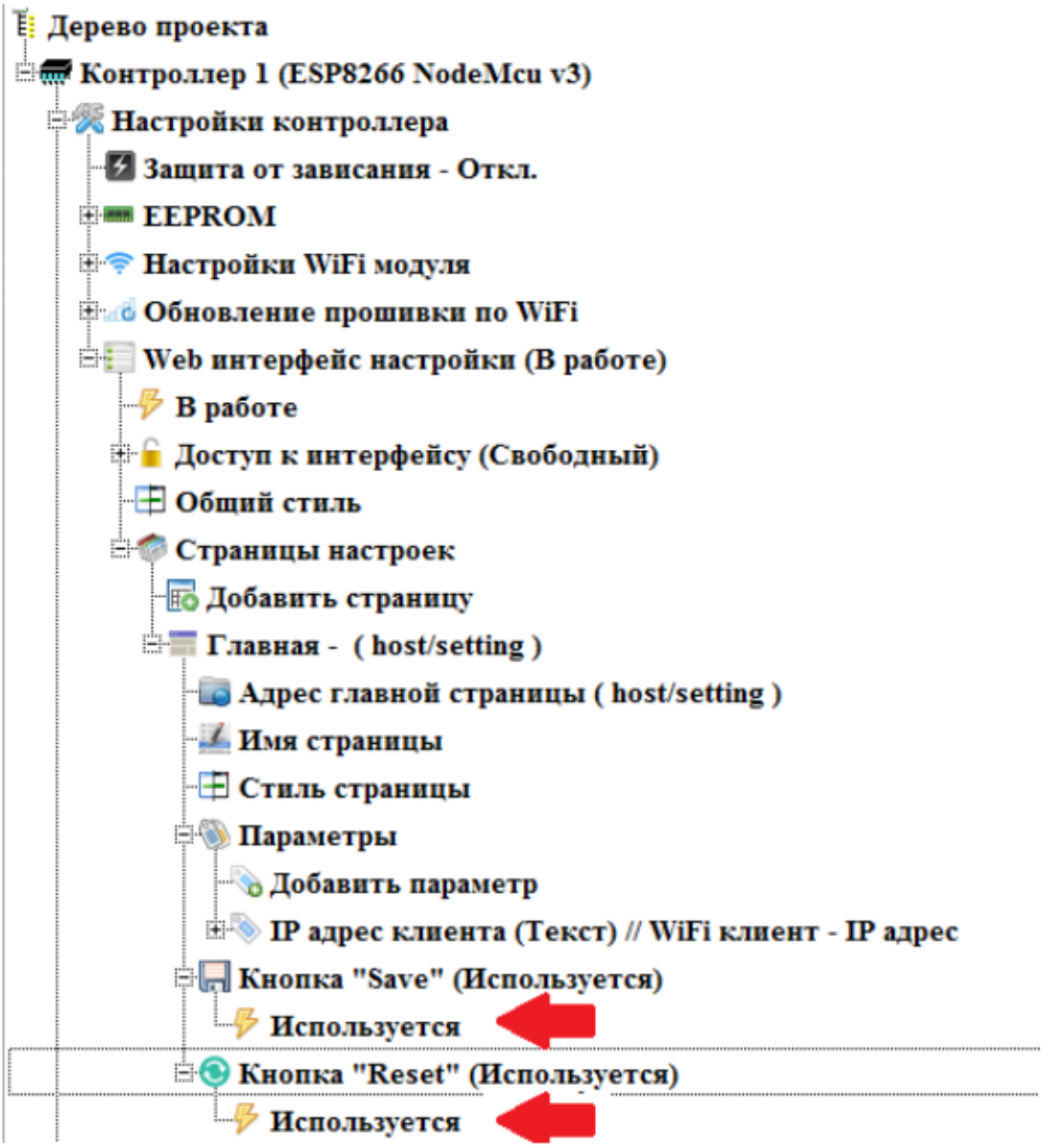


После создания параметра появится новый узел дерева проекта, в котором можно настроить стиль для данного параметра, изменить настройки параметра или удалить его.

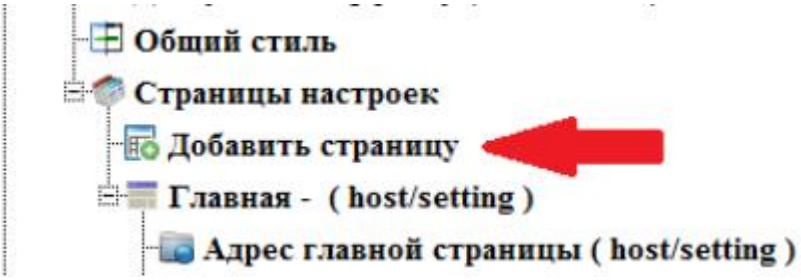


Узлы «Кнопка “Save”» и «Кнопка “Reset”» задают наличие кнопок сохранения данных и перезагрузки контроллера на странице.

Поскольку никаких изменяемых данных на главной странице у нас нет, отключим эти кнопки на странице двойным кликом на ветке «Используется» (по умолчанию кнопки присутствуют на странице).

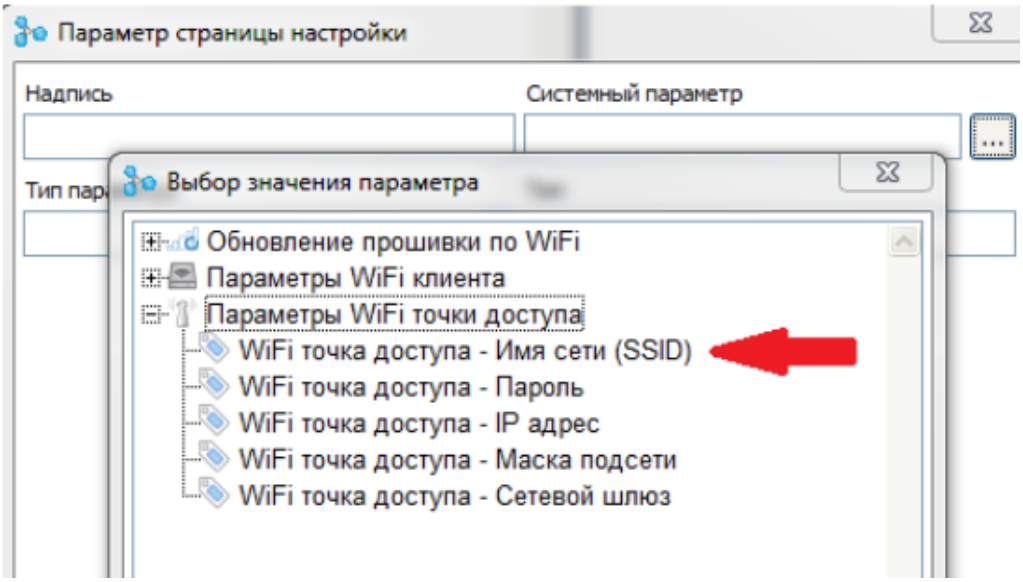


Добавим новую страницу двойным кликом по ветке «Добавить страницу»:

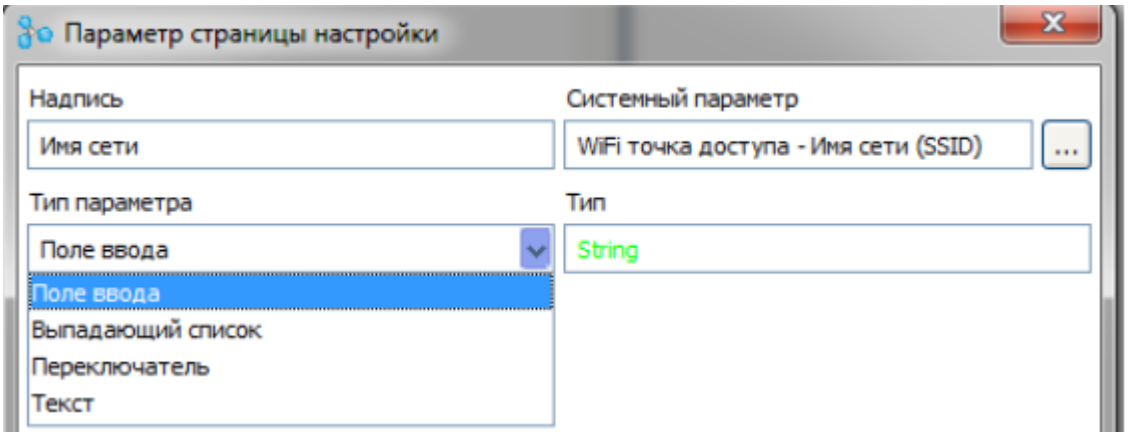


Откроется диалог создания новой страницы. В нём заполним имя страницы (как она будет называться в меню интерфейса настройки) и адрес. Адреса вспомогательных страниц всегда продолжают адрес главной страницы. На данной странице мы будем настраивать параметры точки доступа.

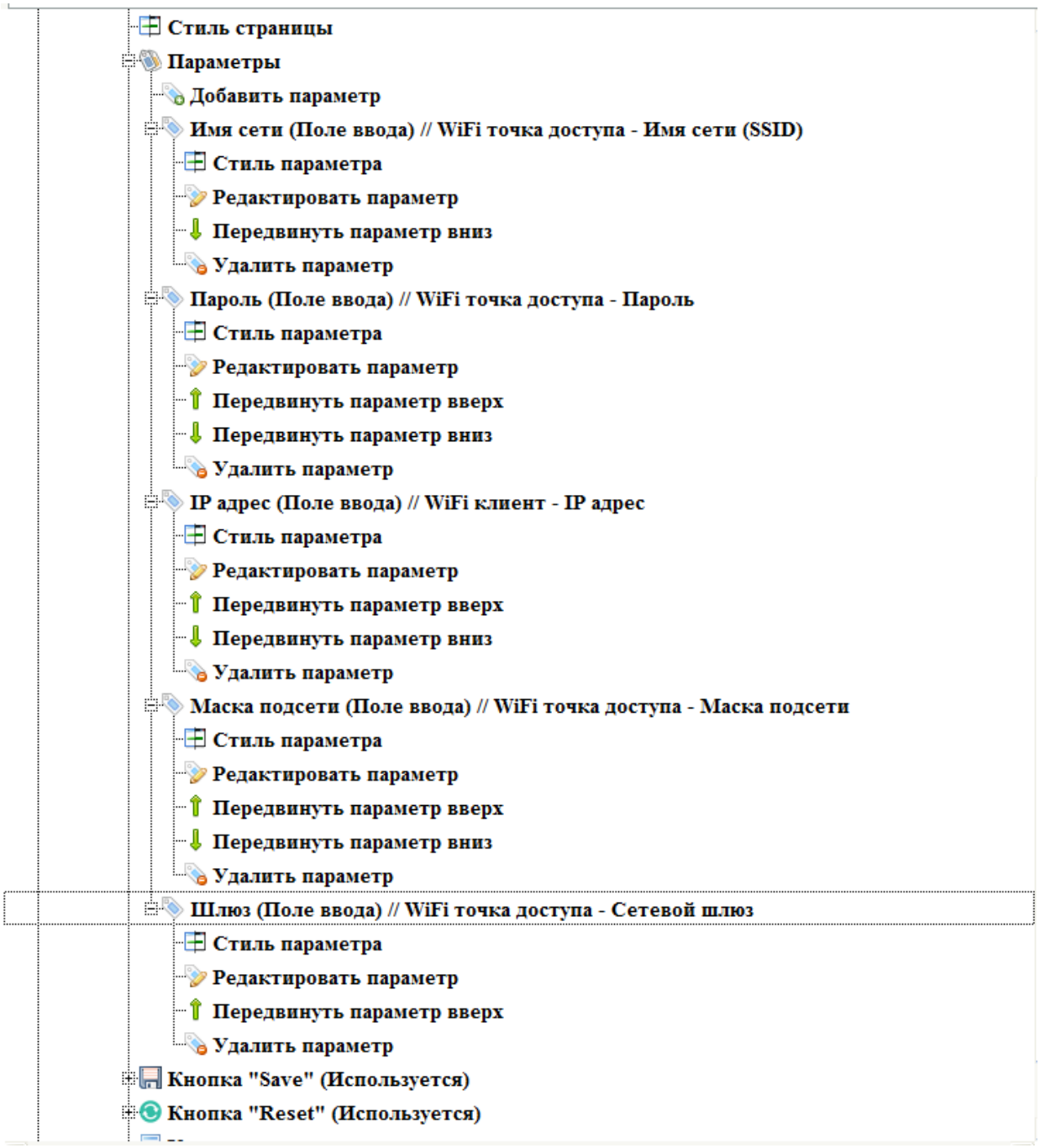
Добавим на страницу параметр “Wi-Fi точка доступа – имя сети (SSID)”.



И выберем для него тип «Поле ввода» и лейблу «Имя сети»

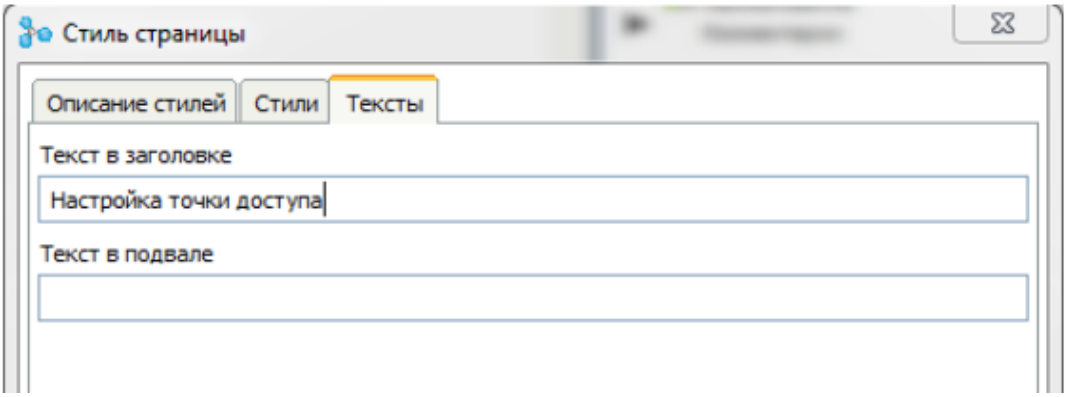


Таким же образом добавим остальные параметры точки доступа:

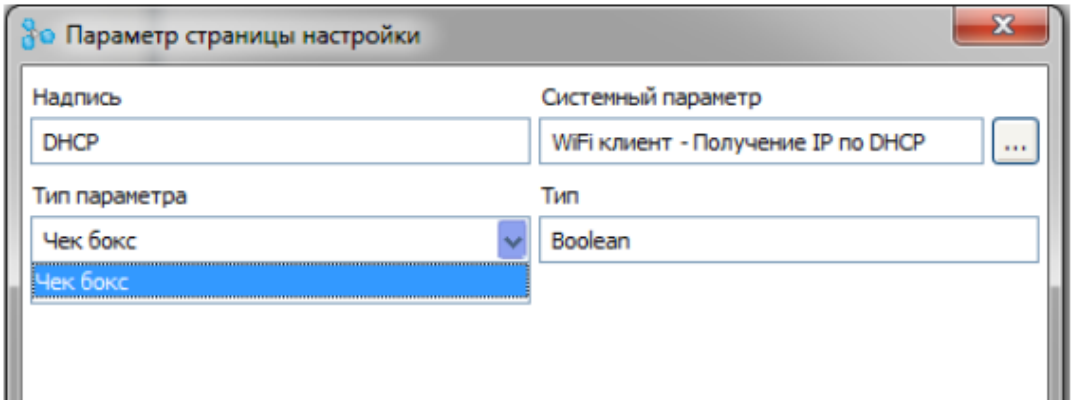


Если параметров на странице более одного в узлах параметров появляются ветки, позволяющие изменить порядок вывода параметров на странице. Поскольку на странице есть изменяемые параметры, оставим на ней кнопки сохранения и перезагрузки контроллера.

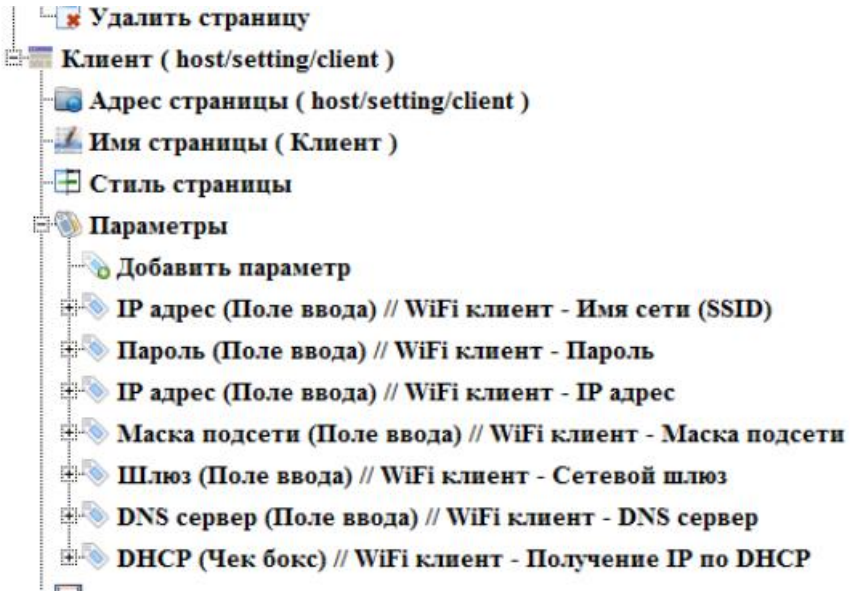
Для данной страницы в диалоге стилей зададим заголовок для данной страницы.



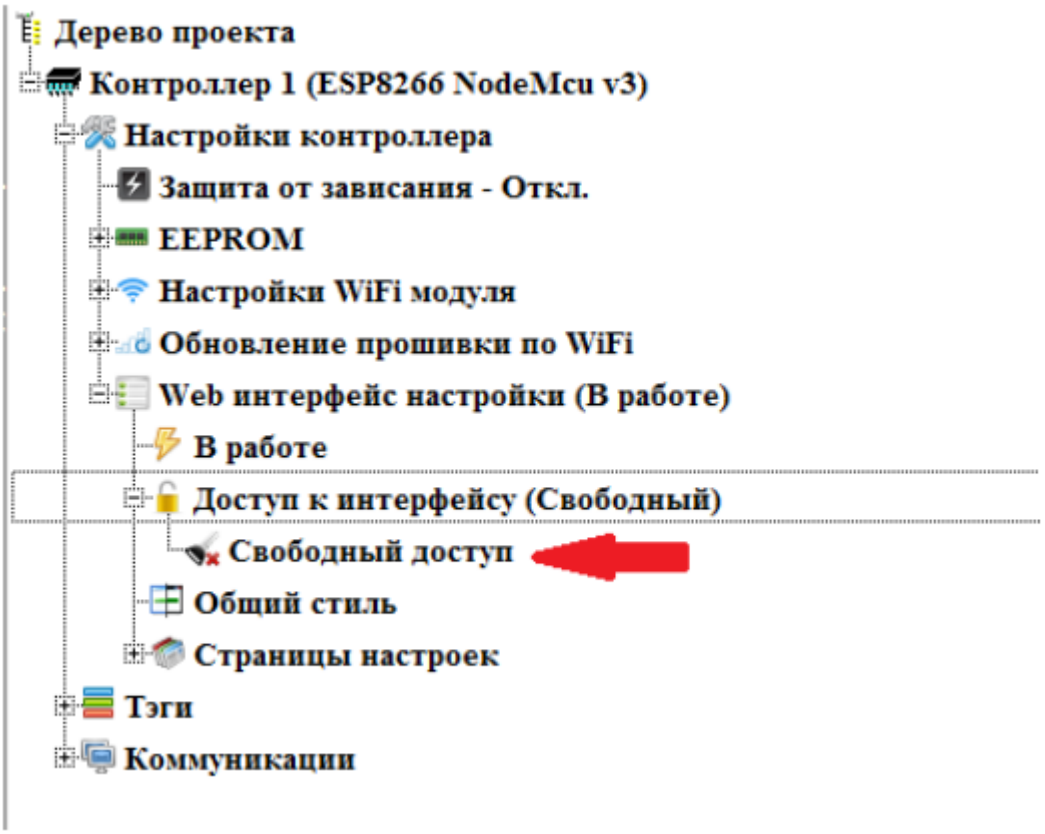
По такому же сценарию создадим страницу с настройками клиента. Для параметра «Wi-Fi клиент – получение IP по DHCP» зададим тип параметра «Чек бокс».



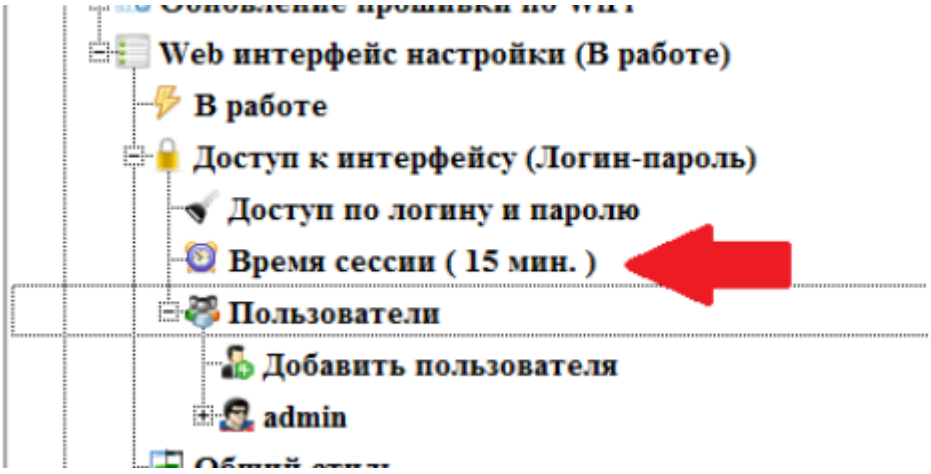
При наличии в интерфейсе более двух страниц в узлах страниц так же появляются ветки, управляющие положением страниц в меню интерфейса. Но главная страница всегда остаётся первой.



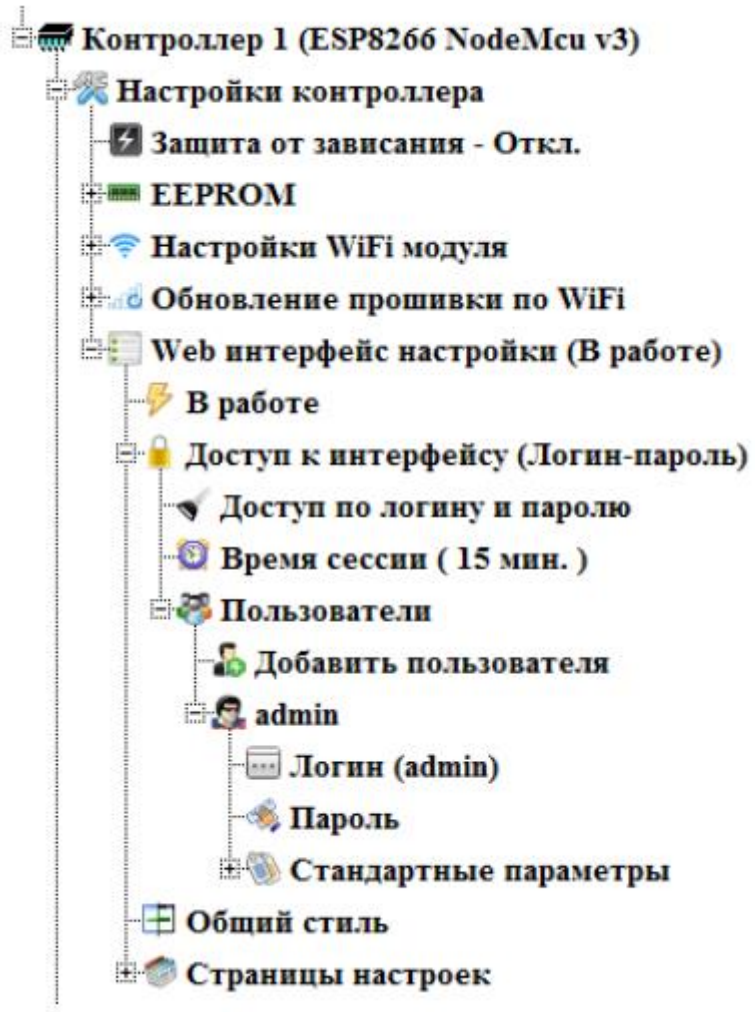
В данный момент доступ к интерфейсу и всем страницам свободный. Введём ограничение доступа. Для этого сделаем двойной клик по ветке «Свободный доступ» узла «Доступ к интерфейсу»:



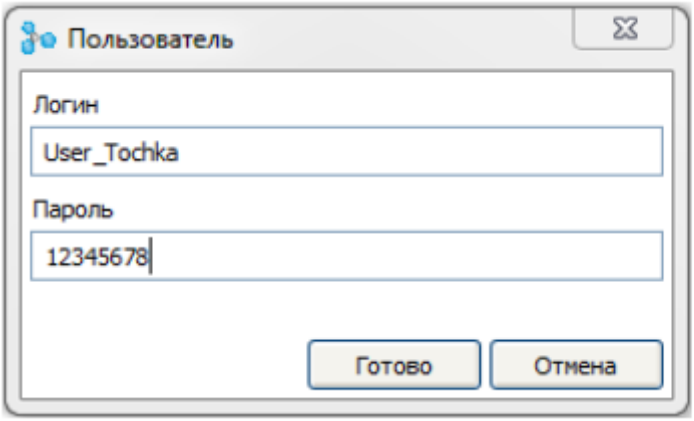
Теперь для доступа к интерфейсу необходимо будет авторизоваться. Время активной авторизации можно задать. По умолчанию оно составляет 15 минут. Через это время после последней активности пользователя будет произведён автоматический сброс текущего пользователя. Это время можно изменить путём двойного клика по соответствующей ветке.



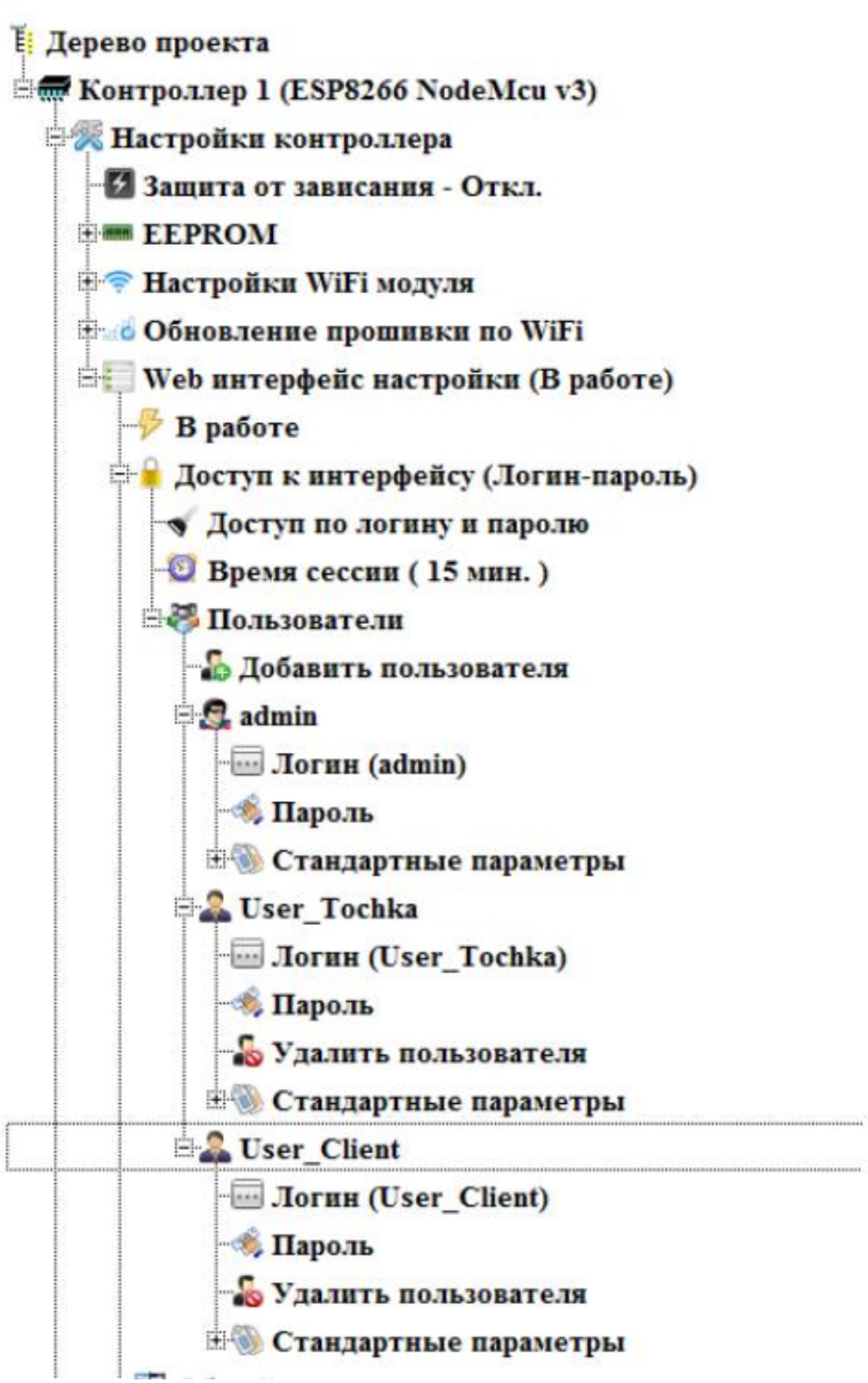
В узле «Пользователи» можно задать необходимое количество пользователей. Там всегда присутствует супер-пользователь (по умолчанию логин — admin). Ему всегда доступны все страницы и параметры. Как и для любого пользователя в его узле можно задать логин и пароль.



Создадим нового пользователя, который будет иметь право настраивать точку доступа. Для этого произведём двойной клик по ветке «Добавить пользователя». Откроется диалог добавления пользователя. В нем мы зададим логин и пароль нового пользователя.

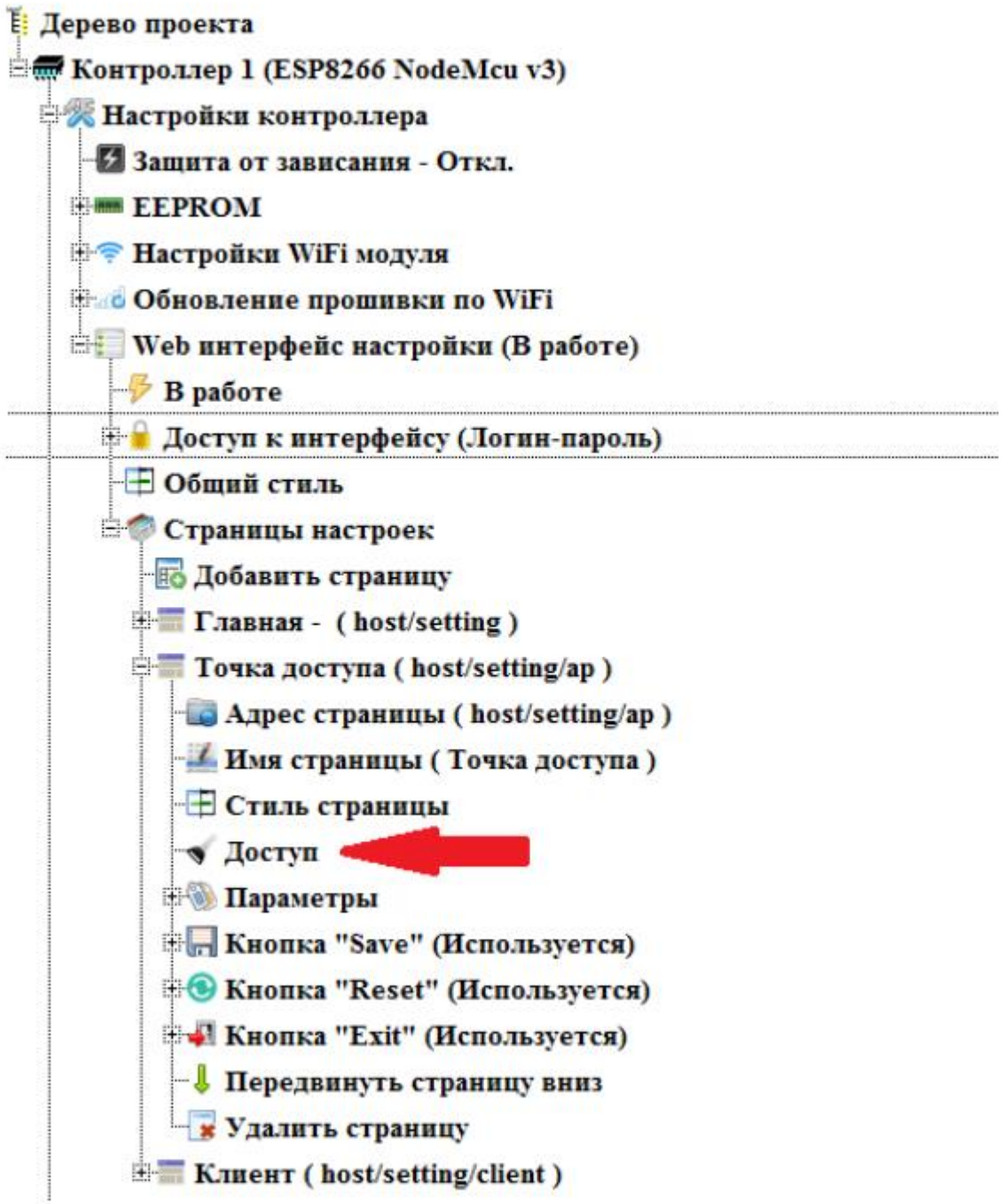


И создадим ещё одного пользователя, который сможет настраивать клиента.

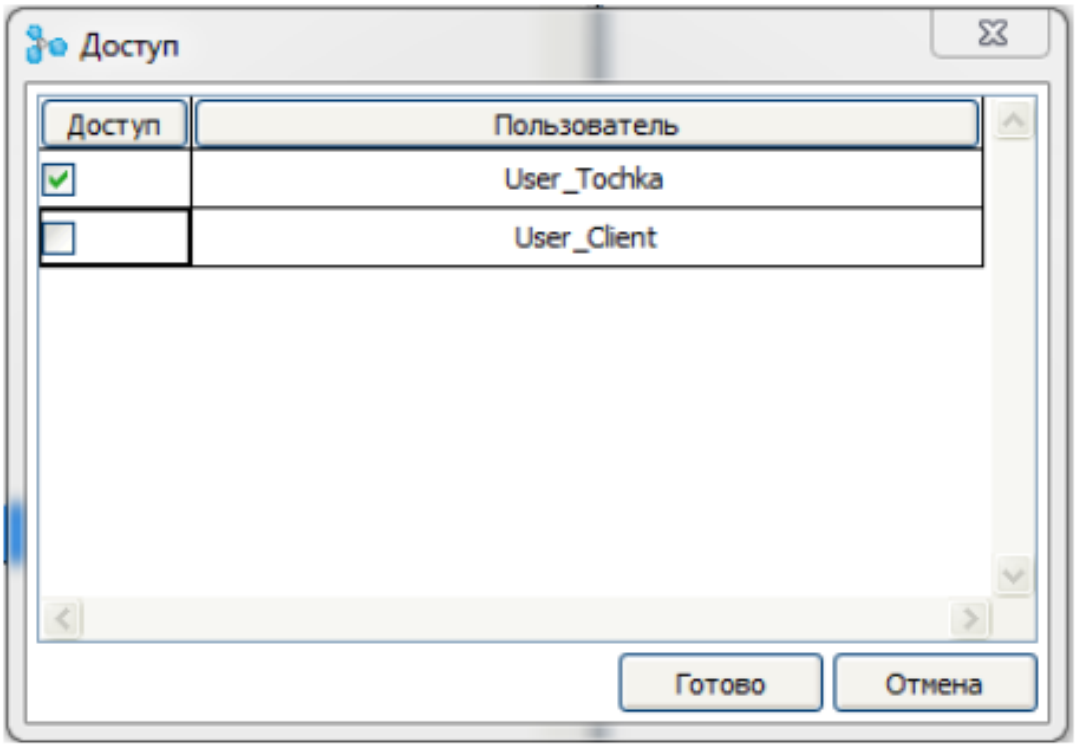


После включения доступа по логину и паролю на всех страницах появилась кнопка «Exit», которая служит для выхода текущего пользователя. Оставим ее на всех страницах.

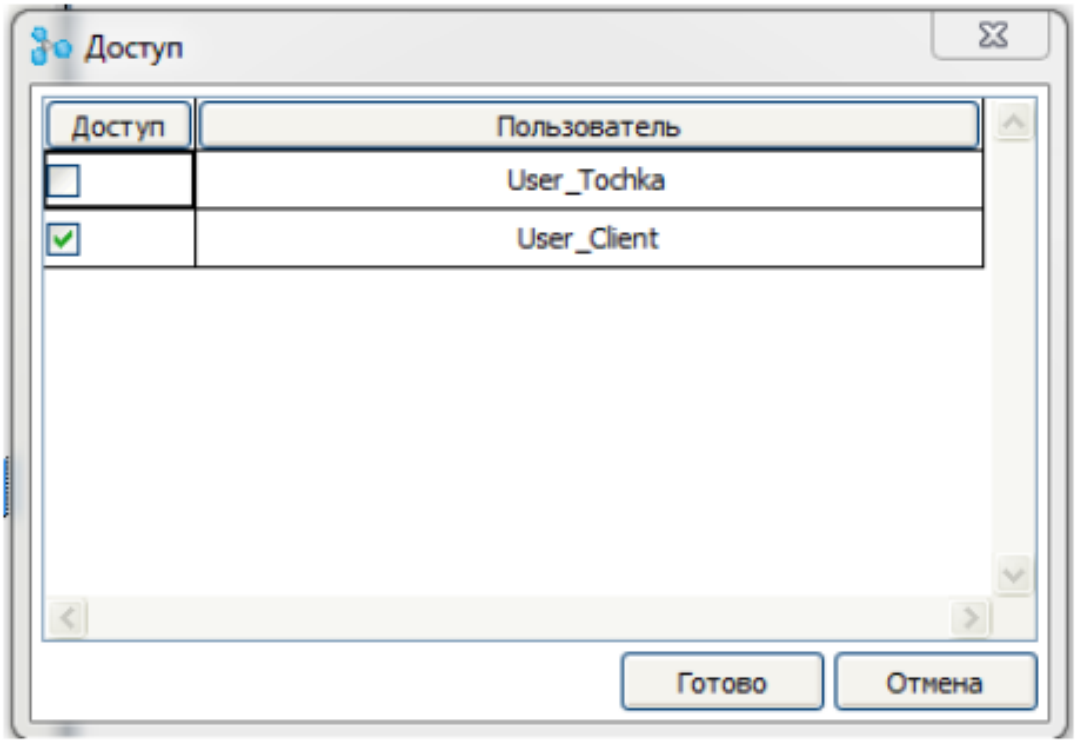
Теперь настроим доступ на страницах. Главная страница всегда доступна всем пользователям. А в узлах остальных страниц появилась новая ветка «Доступ». Дважды кликнем на этой ветке в узле страницы «Точка доступа».



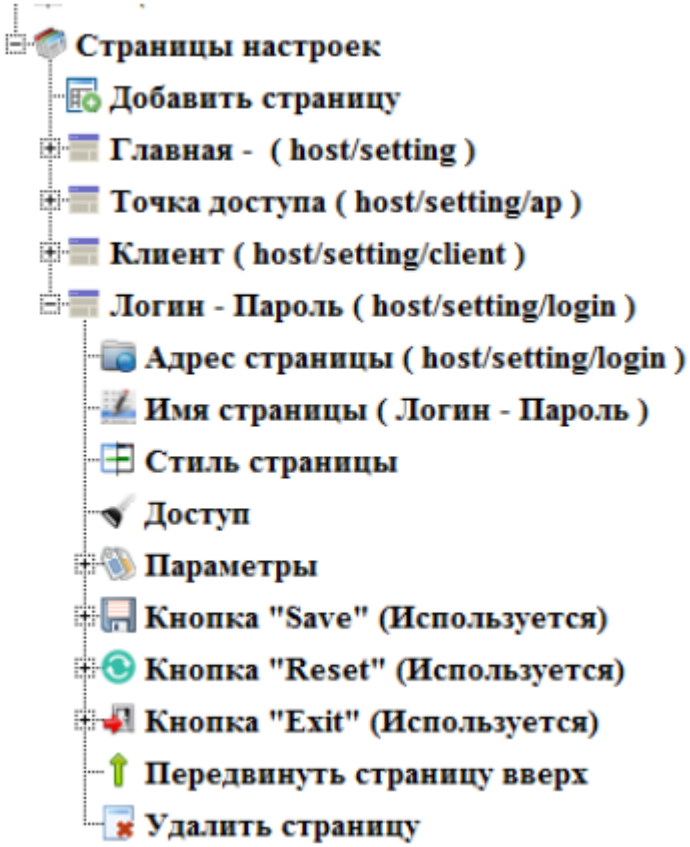
Откроется диалог выбора пользователей для доступа к странице. По умолчанию доступ открыт всем пользователям. Снимем галочку с пользователя «User_Client» тем самым ограничив его доступ к данной странице.



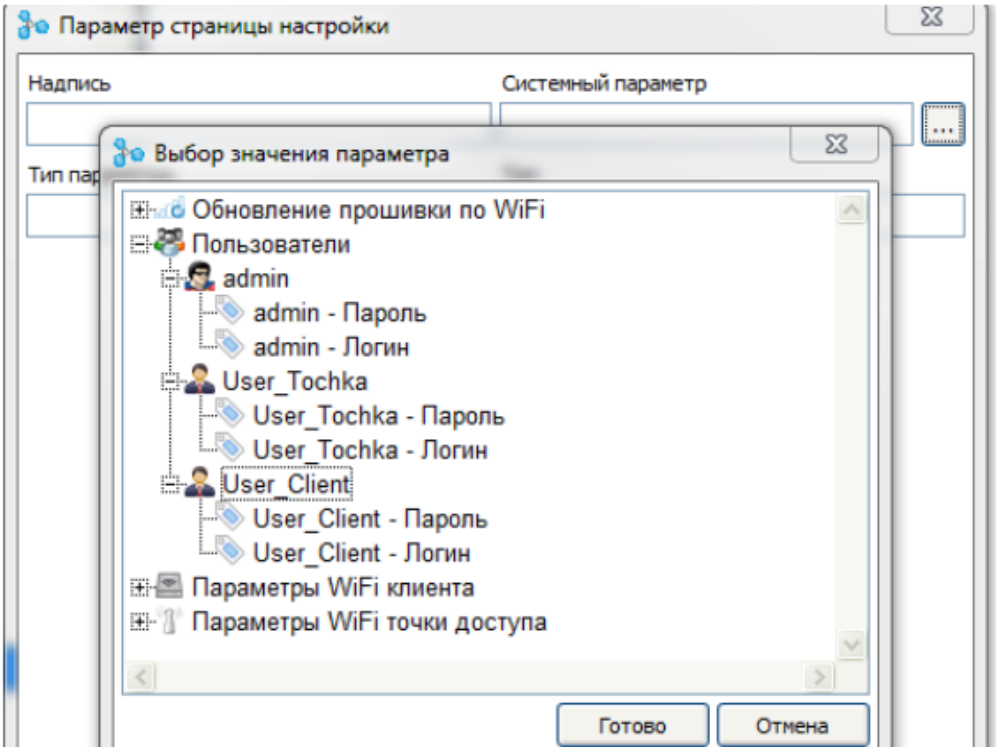
Таким же способом ограничим доступ к странице настроек клиента пользователю «User_Tochka».



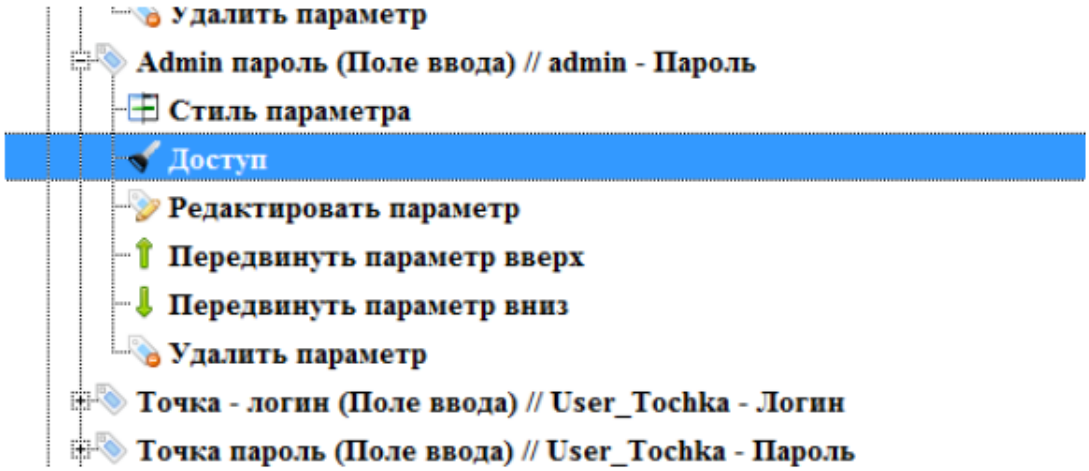
Создадим ещё одну страницу для настроек логинов и паролей:



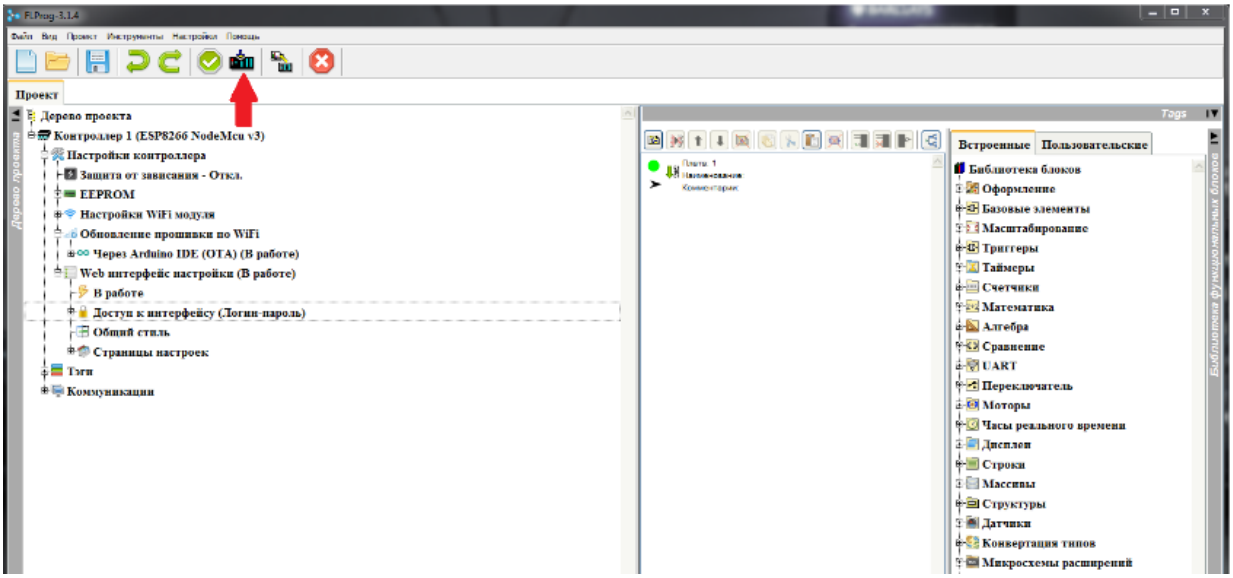
Добавим на эту страницу параметры логинов и паролей для всех пользователей. Эти параметры появились в списке системных параметров после включения доступа по логину и паролю.



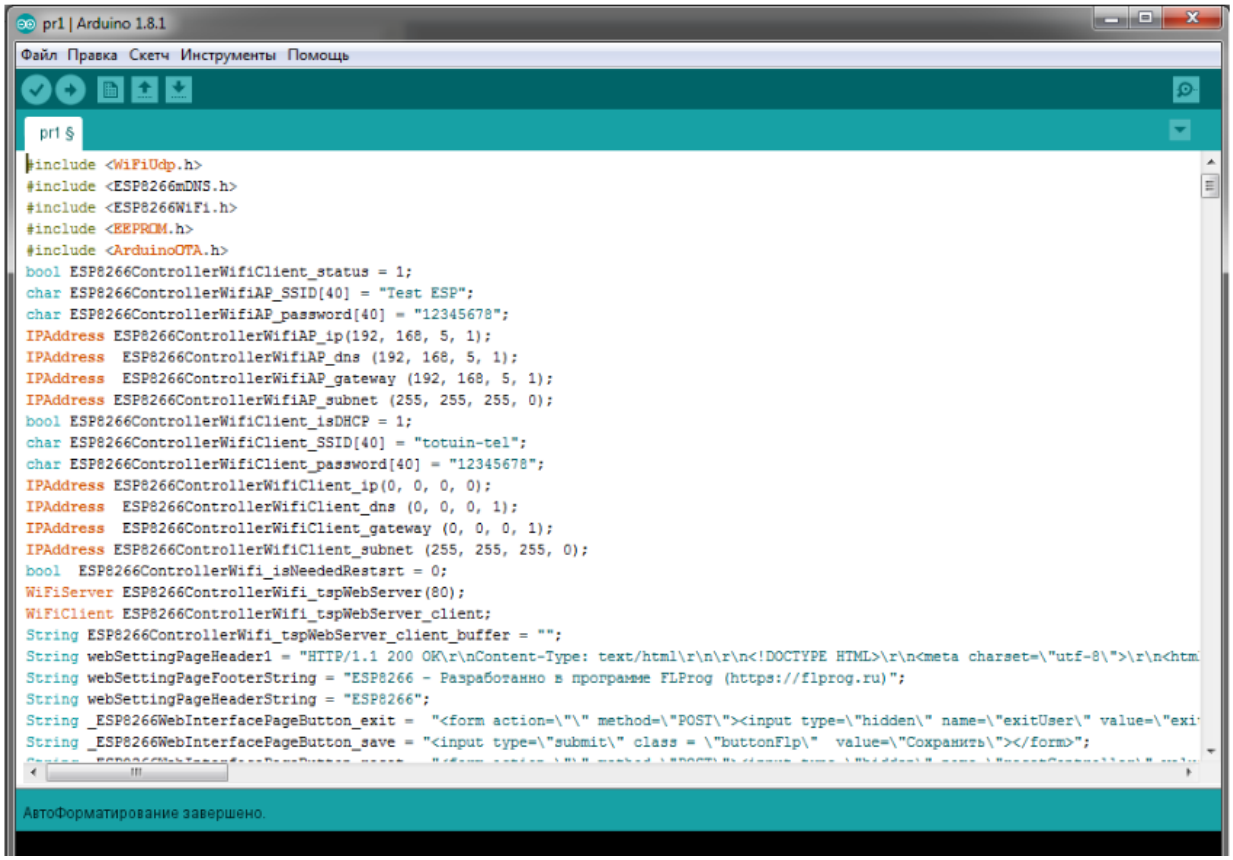
Для данной страницы не будем настраивать ограничение доступа, а настроим ограничение непосредственно на параметры. В узлах параметров появились ветки настройки доступа аналогичные настройкам доступа к страницам.

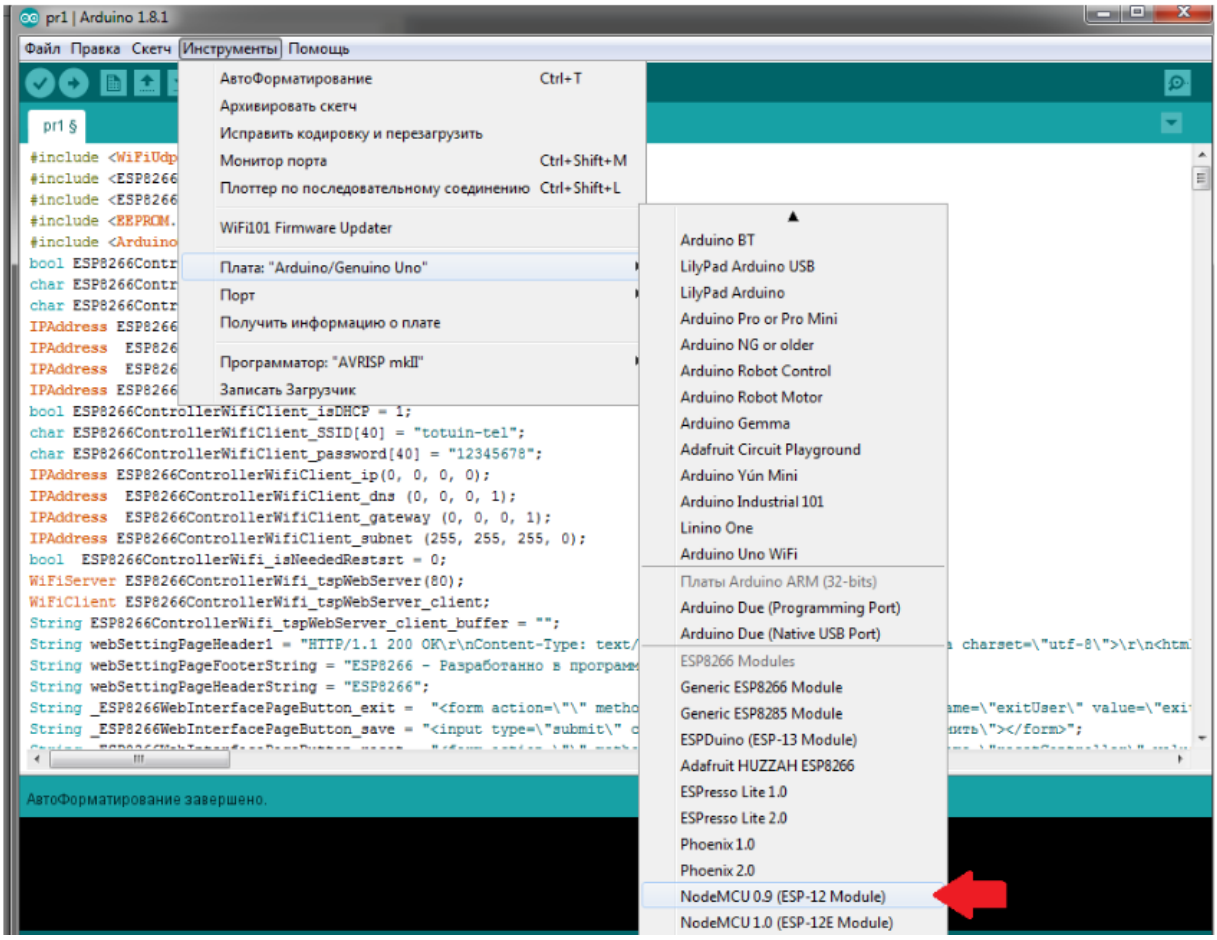


К параметрам админа запретим доступ обеим пользователям, а к параметрам пользователей оставим доступ только тому пользователю, которому эти параметры принадлежат. На этом настройку web интерфейса считаем законченной. Нажимаем кнопку «Компилировать проект» в главном интерфейсе программы и получаем готовый скетч в Arduino IDE:

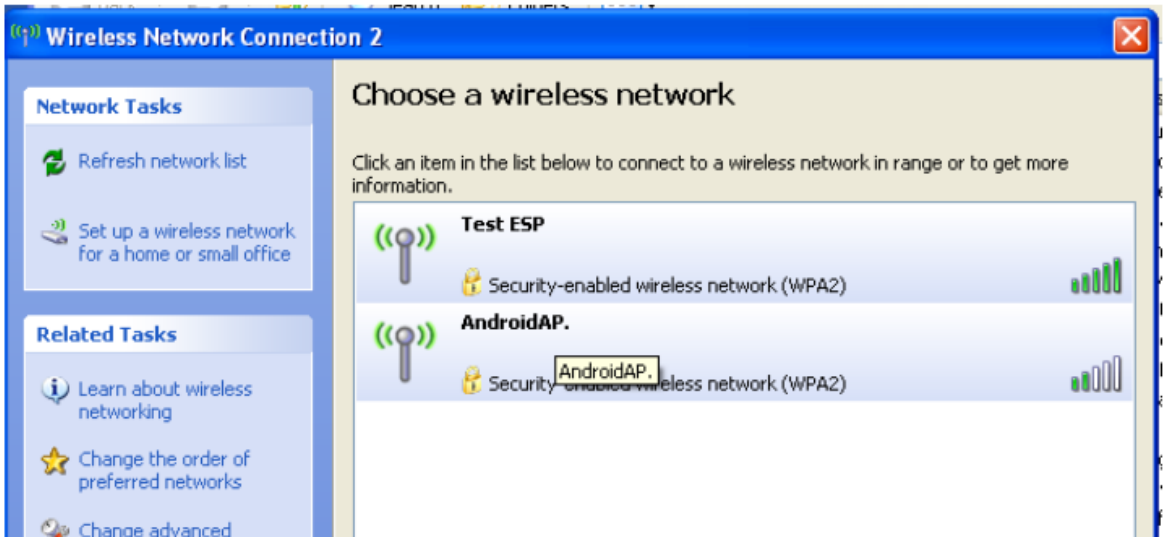


В Arduino IDE выбираем нашу плату. И порт, к которому подключен контроллер. После чего заливаем прошивку в контроллер:

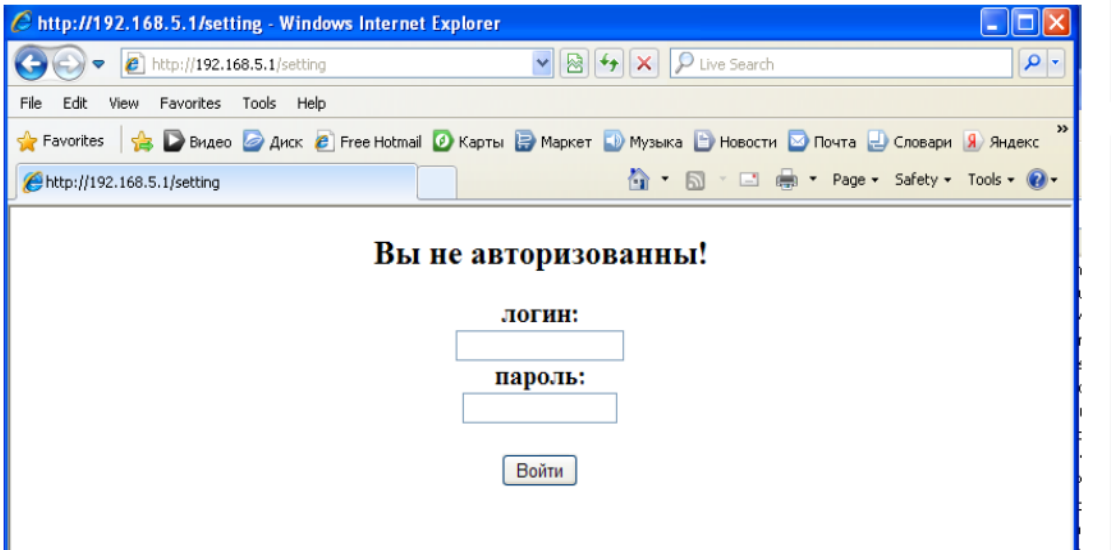

pr1 | Arduino 1.8.1
Файл Правка Скетч Инструменты Помощь
pr1 §
#include <WiFiUdp.h>
#include <ESP8266mDNS.h>
#include <ESP8266WiFi.h>
#include <EEPROM.h>
#include <ArduinoOTA.h>
bool ESP8266ControllerWifiClient_status = 1;
char ESP8266ControllerWifiAP_SSID[40] = "Test ESP";
char ESP8266ControllerWifiAP_password[40] = "12345678";
IPAddress ESP8266ControllerWifiAP_ip(192, 168, 5, 1);
IPAddress ESP8266ControllerWifiAP_dns (192, 168, 5, 1);
IPAddress ESP8266ControllerWifiAP_gateway (192, 168, 5, 1);
IPAddress ESP8266ControllerWifiAP_subnet (255, 255, 255, 0);
bool ESP8266ControllerWifiClient_isDHCP = 1;
char ESP8266ControllerWifiClient_SSID[40] = "totuin-tel";
char ESP8266ControllerWifiClient_password[40] = "12345678";
IPAddress ESP8266ControllerWifiClient_ip(0, 0, 0, 0);
IPAddress ESP8266ControllerWifiClient_dns (0, 0, 0, 1);
IPAddress ESP8266ControllerWifiClient_gateway (0, 0, 0, 1);
IPAddress ESP8266ControllerWifiClient_subnet (255, 255, 255, 0);
bool ESP8266ControllerWifi_isNeededRestart = 0;
WiFiServer ESP8266ControllerWifi_tspWebServer(80);
WiFiClient ESP8266ControllerWifi_tspWebServer_client;
String ESP8266ControllerWifi_tspWebServer_client_buffer = "";
String webSettingPageHeader1 = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<meta charset=\"utf-8\">\r\n<htm
String webSettingPageFooterString = "ESP8266 - Разработанно в программе FLProg (https://flprog.ru)";
String webSettingPageHeaderString = "ESP8266";
String _ESP8266WebInterfacePageButton_exit = "<form action=\"\" method=\"POST\"><input type=\"hidden\" name=\"exitUser\" value=\"exi
String _ESP8266WebInterfacePageButton_save = "<input type=\"submit\" class = \"buttonFlp\" value=\"Сохранить\"></form>";
АвтоФорматирование завершено.

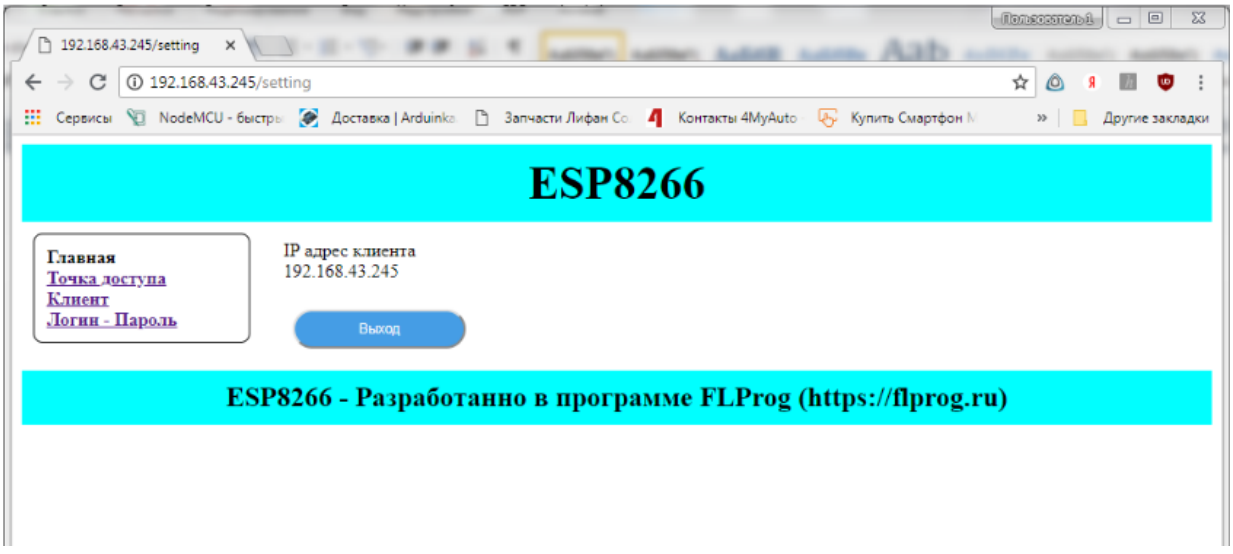

pr1 | Arduino 1.8.1
Файл Правка Скетч Инструменты Помощь
АвтоФорматирование Ctrl+T
Архивировать скетч
Исправить кодировку и перезагрузить
Монитор порта Ctrl+Shift+M
Плоттер по последовательному соединению Ctrl+Shift+L
WiFi101 Firmware Updater
Плата: "Arduino/Genuino Uno"
Порт
Получить информацию о плате
Программатор: "AVRISP mkII"
Записать Загрузчик
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini
Arduino NG or older
Arduino Robot Control
Arduino Robot Motor
Arduino Gemma
Adafruit Circuit Playground
Arduino Yún Mini
Arduino Industrial 101
Linino One
Arduino Uno WiFi
Платы Arduino ARM (32-bits)
Arduino Due (Programming Port)
Arduino Due (Native USB Port)
ESP8266 Modules
Generic ESP8266 Module
Generic ESP8285 Module
ESPDuino (ESP-13 Module)
Adafruit HUZZAH ESP8266
ESPresso Lite 1.0
ESPresso Lite 2.0
Phoenix 1.0
Phoenix 2.0
NodeMCU 0.9 (ESP-12 Module)
NodeMCU 1.0 (ESP-12E Module)
АвтоФорматирование завершено.

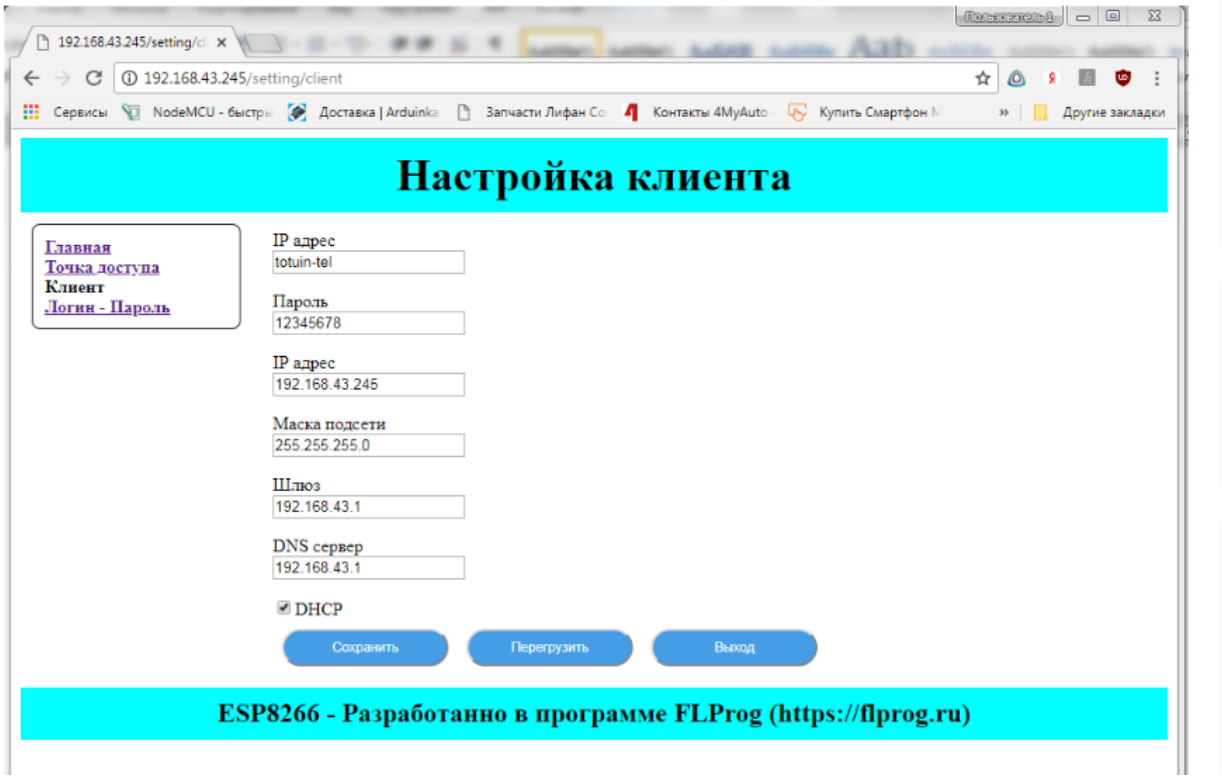
После загрузки появляется новая точка доступа:



Подключаемся к ней и заходим по заданному нами в проекте адресу. Появляется страница авторизации:

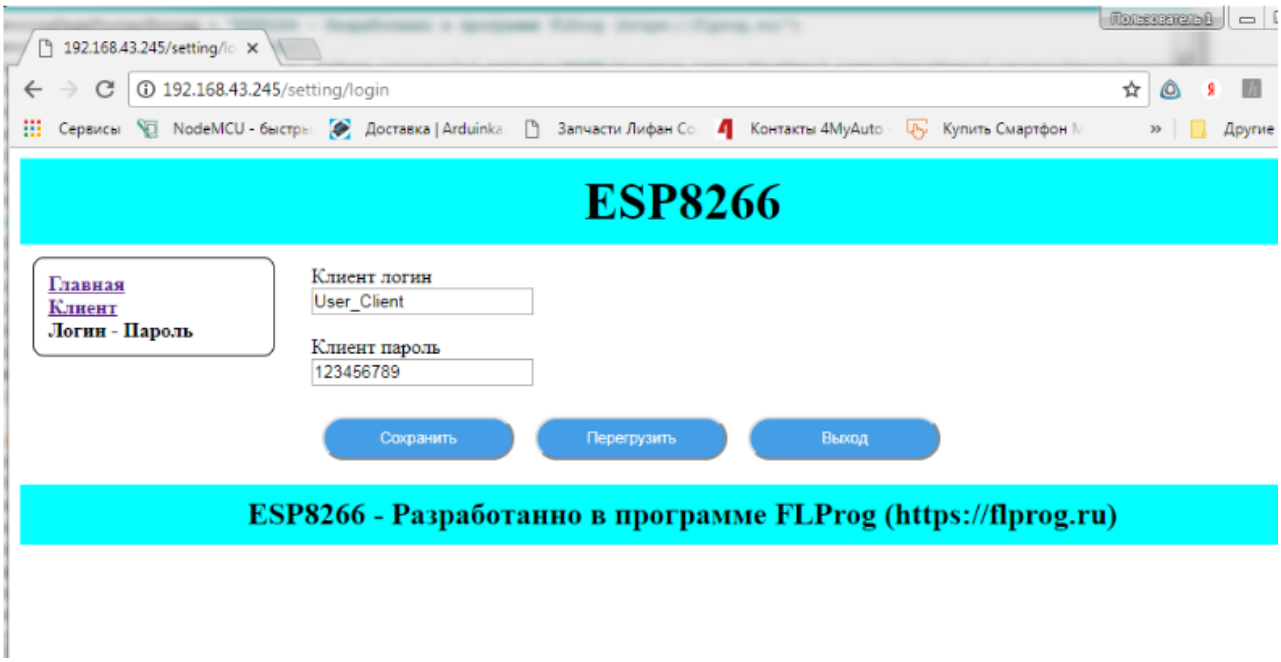




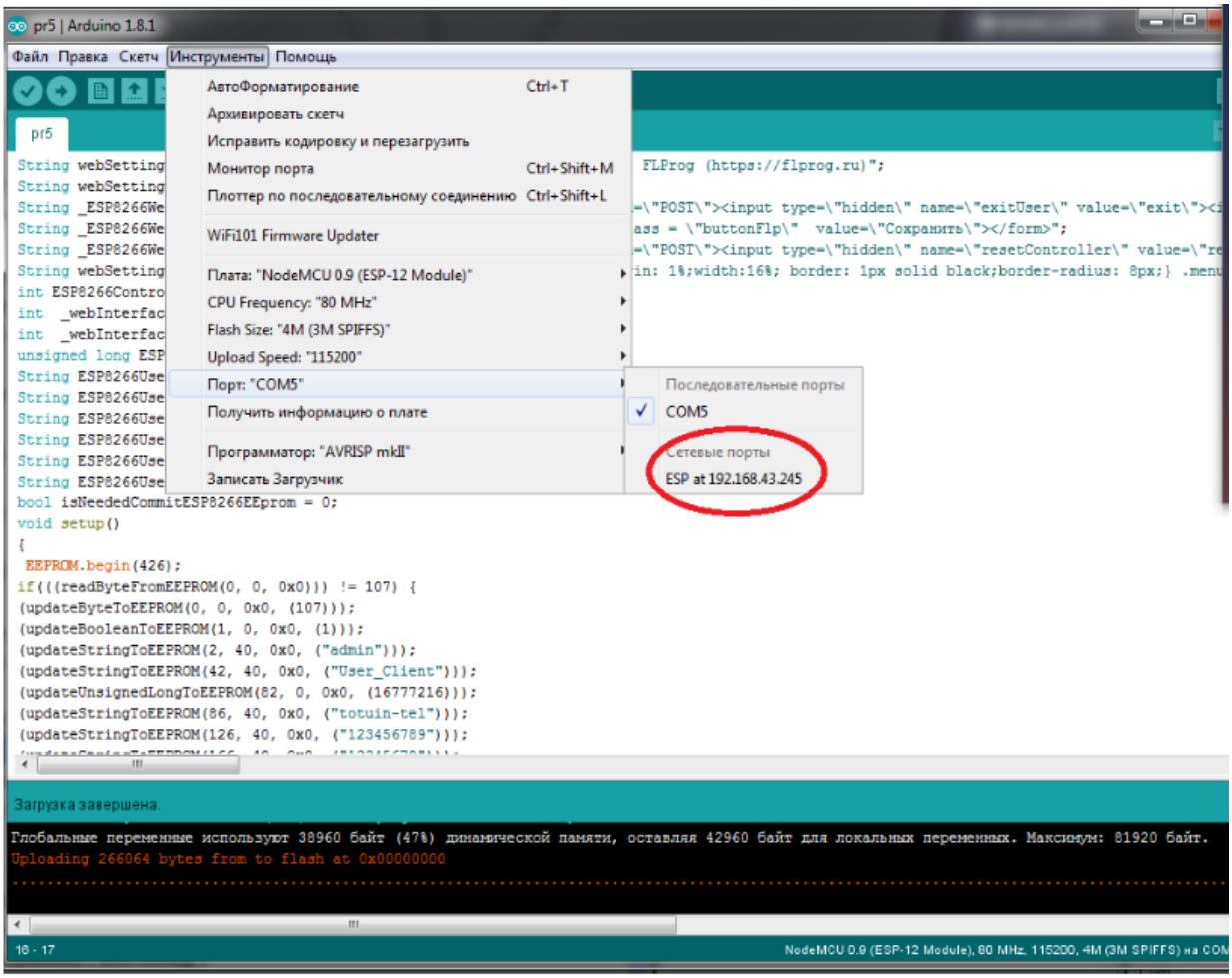


А на странице логина и пароля все параметры. Выходим и логинимся под **User_Tochka**. Видим в меню только доступные страницы, и только свой логин и пароль.

Перелогиниваемся под **User_Client** и картина соответствующая – видим только то, что положено.



Открываем Arduino IDE и в настройках порта видим, что контроллер готов к обновлению прошивки «по воздуху».



Требования к отчету:

Индивидуальное задание и требование к отчету будут реализованы вместе МДК «Организация и принципы построения компьютерных систем» или МДК «Защита информации в компьютерных системах».

Форма отчетности – ответы на вопросы по реализации проекта (устная форма), код программы на языке C/C++.

HMI (Human-Machine-Interface)

Человеко-машинный интерфейс (ЧМИ) (англ. Human-machine interface, HMI) — широкое понятие, охватывающее инженерные решения, обеспечивающие взаимодействие человека-оператора с управляемыми им машинами.

Создание систем человеко-машинного интерфейса тесно увязано с понятиями эргономика и юзабилити.

Проектирование ЧМИ включает в себя:

создание рабочего места: кресла, стола, или пульта управления, размещение приборов и органов управления (устройства ввода данных) (соответствием всего этого физиологии человека занимается эргономика), освещение рабочего места и, возможно, микроклимат.

далее рассматриваются взаимодействие оператора со всеми органами управления: их доступность и необходимые усилия, эффективность и скорость доступа, согласованность (непротиворечивость) управляющих воздействий (в том числе т. н. «защита от дурака»), расположение дисплеев и размеры надписей на них (всё это входит в сферу юзабилити).

Юзабилити (от англ. usability — «удобство и простота использования, степень удобства использования»), также удобство использования, пригодность использования, эргономичность — способность продукта быть понимаемым, изучаемым, используемым и привлекательным для пользователя в заданных условиях (**ISO/IEC 25010**); свойство системы, продукта или услуги, при наличии которого конкретный пользователь может эксплуатировать систему в определенных условиях для достижения установленных целей с необходимой результативностью, эффективностью и удовлетворённостью (**ISO 9241-210**).

Удобство (пригодность) использования системы не сводится только к тому, насколько её легко эксплуатировать. В соответствии со стандартами серии ISO 9241 эту характеристику следует понимать более широко, учитывая личные цели пользователя, его эмоции и ощущения, связанные с восприятием системы, а также удовлетворённость работой. Свойства, необходимые для обеспечения пригодности использования, зависят также от задачи и окружающей среды.

Пригодность использования — не абсолютное понятие, оно может различным образом проявляться в определённых условиях эксплуатации

Одной из наиболее сложных задач является создание эффективного ЧМИ рабочих мест сложных машин с множеством органов управления, например, пилотов самолёта и космических кораблей.

В промышленных условиях ЧМИ чаще всего реализуется с использованием типовых средств: операторских панелей, компьютеров и типового программного обеспечения.

Вашему вниманию предлагается изучение научной статьи **«Разработка человеко-машинного интерфейса и его применение в системах управления»** Сверчкова Дениса Сергеевича.

В статье (ссылка на нее расположена посредством интер-отклика) описаны различные типы человеко-машинных интерфейсов, области применения каждого интерфейса, этапы разработки человеко-машинного интерфейса и приведен пример реализации человеко-машинного интерфейса (ЧМИ) для системы управления (СУ).

Рекомендуется также ознакомиться с материалом, адаптированным под разработку и изучение в рамках **IDE FLProg** на портале «Хабрахабр»:

- FLProg + Nextion HMI. Урок 1
- FLProg + Nextion HMI. Урок 2
- FLProg + Nextion HMI. Урок 3
- (https://habr.com/ru/company/flprog/blog/392561/)

Дисплеи **Nextion** — это модули с цветными сенсорными экранами и контроллерами, в которые Вы можете записывать свои программы. На модулях дисплеев Nextion имеется разъём UART и выводы GPIO, что позволяет использовать дисплеи Nextion как совместно с Arduino/ESP (подключая дисплей к микропроцессорной системе по шине UART), так и отдельно (подключая кнопки, светодиоды, реле и т.д. напрямую к выводам GPIO дисплеев).

В завершении данного учебно-практического издания вашему вниманию предлагается для изучения **презентации «Обзор современных программируемых логических контроллеров. SCADA системы в автоматизированных производствах» состоящая из 38 слайдов.**

Программа презентации подготовлена таким образом, что в ней раскрываются технические особенности устройств (ПЛК), понятные студентам, осваивающих образовательные программы по укрупненной группе специальностей 09.00.00 «Информатика и вычислительная техника», а не только осваивающим учебные программы по укрупненной группе "Автоматизация технологических процессов и производств (по отраслям)".

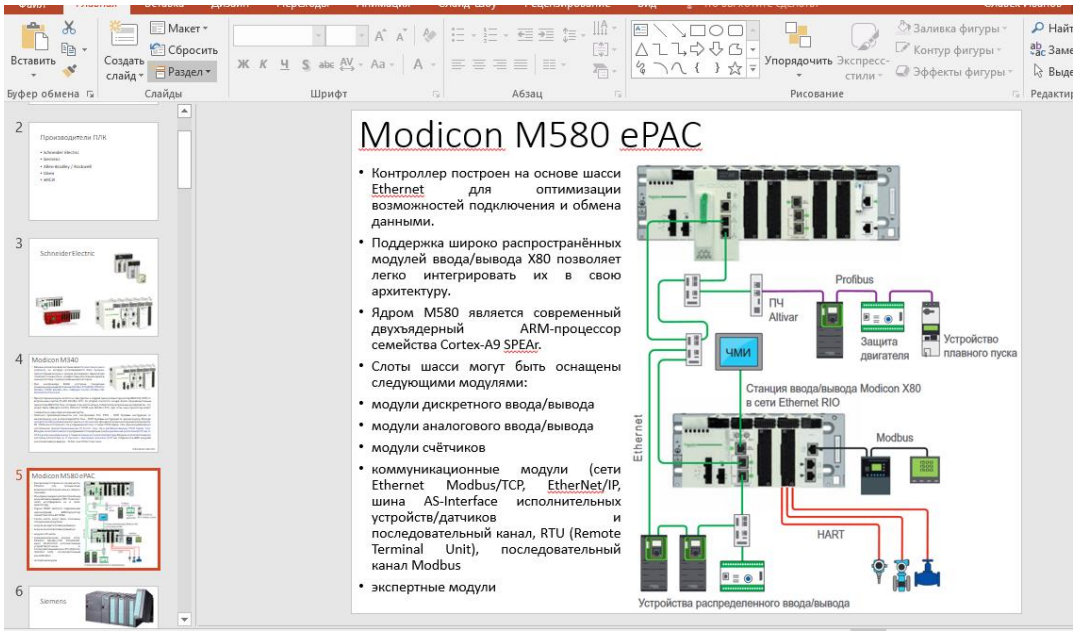


Рис. 40 – Скриншоты из предлагаемой для изучения презентации.

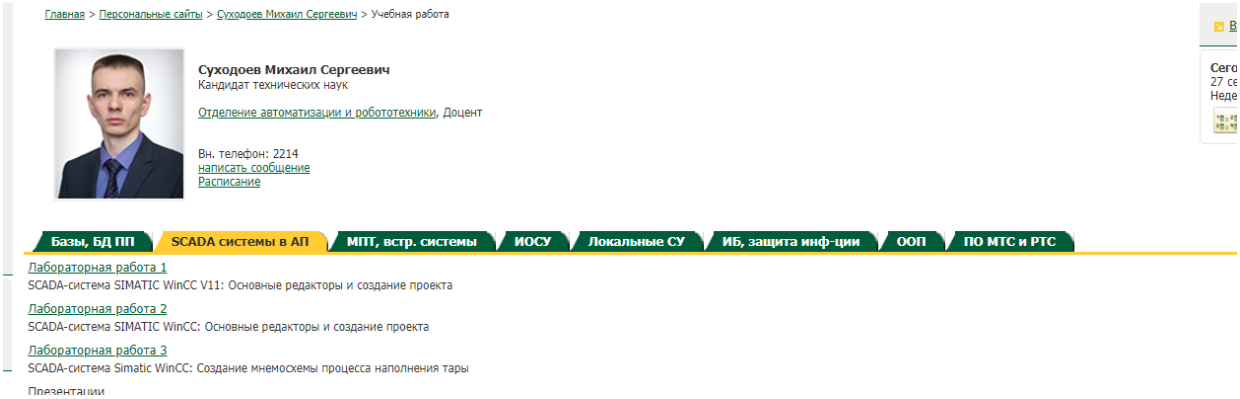


Рис. 41 – Составитель презентации – доцент Томского политехнического университета Суходоев Михаил Сергеевич

Таким образом вся программа данного издания построена на вовлечение будущих специалистов укрупненной группы 09.00.00 в смежные направления, для приобретения тех или иных профессиональных компетенций и первичных навыков, расширяющих область работы будущих специалистов.

Подводя итог, еще раз акцентируем на область применения данного практикума: ознакомление с принципами визуального программирования с несколькими видами микропроцессорных систем, получение теоретических навыков работы с специализированной документацией, приобретением умения работать с техническими заданиями в нескольких интегрированных средах разработки, ознакомлением с понятием человеко-машинного интерфейса, навыков дифференцирования микропроцессорных систем, программируемых логических контроллеров и иной управляющей и исполнительной цифровой техники в рамках концепции Internet of Things/Internet of Everything.

Презентация «Обзор современных программируемых логических контроллеров. SCADA системы в автоматизированных производствах)
https://yadi.sk/i/AtlkjUUVeV4cXA