Петин В. А.

СОЗДАНИЕ УМНОГО ДОМА НА БАЗЕ ARDUINO

ДМК ПРЕСС
ИЗДАТЕЛЬСТВО

В. А. Петин

СОЗДАНИЕ УМНОГО ДОМА НА БАЗЕ ARDUINO

ДМК Пресс
издательство
Москва, 2018

УДК 681.4:004.9Arduino
ББК 32.816с515+32.965с515
П29

Петин В. А.
П29 Создание умного дома на базе Arduino. – М.: ДМК Пресс, 2018. – 180 с.

ISBN 978-5-97060-620-9

С появлением интернета вещей отношения умного дома с владельцем переходят на новый уровень – теперь контроллер, управляющий жилищем, может в любой момент связаться с хозяином и получить от него новое задание. Специальное приложение для Android или iOS позволит вам управлять своим домом с экрана смартфона из соседней комнаты или с другого континента. Взаимодействовать с техникой будущего и разрабатывать новые способы применения интернета вещей научит вас эта книга – в ней есть всё, что нужно для творчества. Издание познакомит вас с основами создания и отладки проектов по автоматизации дома на основе контроллеров Arduino и NodeMCU.

УДК 681.4:004.9Arduino
ББК 32.816с515+32.965с515

ISBN 978-5-97060-620-9

СОДЕРЖАНИЕ

Умный дом – это жилой дом, организованный для удобства проживания людей при помощи различных высокотехнологичных устройств.

Умный дом понимает конкретные ситуации, происходящие в здании, и соответствующим образом на них реагирует по заранее выработанным алгоритмам.

При этом человек одной командой задает желаемую обстановку, а уже автоматика в соответствии с внешними и внутренними условиями задает и отслеживает режимы работы всех инженерных систем и электроприборов.

Умный дом сам настроит работу всех систем в соответствии с пожеланием человека, временем суток, его положением в доме, погодой, внешней освещенностью для обеспечения комфортного состояния внутри дома.

Создание умного дома предполагает наличие умных устройств.

Но как устройство может стать «умным»?

Первый вариант – за счет изменения своей конструкции: эта конструкция может быть таковой, что поведение системы может выглядеть разумным.

Второй вариант – за счет «интеллектуализации» (оснащения системы устройствами сбора информации, ее обработки и принятий решений). Такой подход позволяет обеспечить достаточно сложное и «разумное» поведение гораздо более простыми способами, чем за счет создания соответствующей конструкции.

Наконец, третий вариант – поведение системы становится «разумным» за счет того, что она взаимодействует с другими системами. Технология IoT (интернет вещей) как раз и предоставляет возможность каждому элементу умного дома (вещи) и всему умному дому выйти в пространство интернет-паутины и обмениваться информацией с другими вещами и системами.

Чем же привлекателен третий вариант?

Во-первых, предоставляет гораздо больше возможностей для организации умного дома (можно использовать данные со всего интернет-пространства), во-вторых, он более экономичен (провести Интернет стоит гораздо дешевле создания сложных интеллектуальных устройств).

Наш набор «Интернет вещей для умного дома» на основе контроллера Arduino MEGA или платы NodeMCU ESP8266 позволит вам создать элементы умного дома с использованием технологии интернет вещей.

В книге мы пошагово рассмотрим процесс подключения и мониторинга датчиков, удаленного управления исполнительными устройствами умного дома через Интернет. Мы познакомимся с мобильным приложением IoT Manager, которое поможет нам использовать телефон или планшет в качестве пульта управления из любой точки земного шара.

2 ОБЗОР НАБОРА «ИНТЕРНЕТ ВЕЩЕЙ ДЛЯ УМНОГО ДОМА»

Откроем наш набор и рассмотрим его содержимое. Самый главный компонент любой «умной» системы – его контроллер. Контроллер предназначен для получения информации и управления «умным» домом. В нашем наборе два контроллера! Это плата Arduino MEGA и модуль NodeMCU v3 Lua Wi-Fi ESP8266 CH340. Вы можете выбрать любой из них.

Arduino сейчас представляет собой удобный электронный конструктор, понятную среду для программирования и в целом удобный инструмент для создания собственных разработок как новичкам, так и профессионалам. Популярность платформе Arduino придает то, что она имеет 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++. На контроллер программы переносятся через USB (не нужен программатор, проще говоря, передатчик программы на нужное устройство). Arduino имеет открытый исходный код (та основа, на которой создается платформа, ее программное ядро, с помощью которого и создаются все нужные программы). Открытый код полезен для пользователей тем, что на основе него они могут создавать свои собственные самодельные программы, а не использовать только те, которые поставляются самим Arduino. Платформа Arduino постоянно развивается, и существует большое количество плат данной платформы.

Основные версии плат Arduino можно приобрести в интернет-магазине Arduino-Kit (www.arduino-kit.ru).

Ваш умный дом потребует большого количества устройств, подключенных к контроллеру, поэтому в наборе используется контроллер Arduino высокой производительности и с большим количеством контактов – плата Arduino MEGA (см. рис. 2.1).

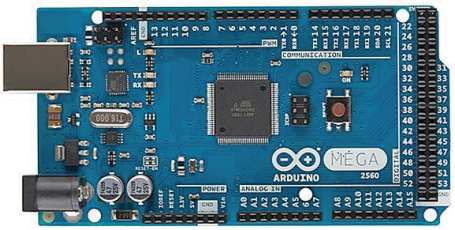


Рис. 2.1. Плата Arduino MEGA

Второй контроллер – модуль NodeMCU v3 Lua Wi-Fi ESP8266 CH340 (рис. 2.2).

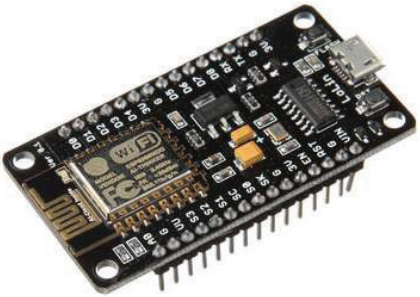


Рис. 2.2. Модуль NodeMCU ESP8266

Это полноценная платформа для создания устройств интернет-вещей на основе модуля ESP8266, который умеет принимать и посылать данные в локальную сеть или Интернет через Wi-Fi. Модуль ESP8266 представляет собой полноценный 32-битный микроконтроллер, который содержит выводы GPIO, в том числе SPI, UART, I2C, и на данный момент составляет серьезную конкуренцию плате Arduino. Программирование платы возможно в среде Arduino IDE.

Идем далее.

Во-первых, необходимо оперативно получать всю необходимую информацию о климатических параметрах в вашем доме: температура и влажность воздуха, увлажненность почвы для растений, нет ли пожара, потопа или утечки газа.

Какую проблему клиента решит функция мониторинга? Прежде всего устранит беспокойство насчет того, все ли в порядке в доме во время вашего отсутствия.

Для этого в набор включены следующие датчики:

- датчик температуры DS18B20;
- датчик влажности DHT11 (DHT22);
- датчик увлажненности почвы;
- датчик воды;
- датчик огня;
- датчик пропана;
- датчик движения.

Выводить данные мониторинга необходимо на дисплей, или с помощью светодиодов и звукового сигнала оповещать о критических значениях климатических параметров, чтобы видеть

показания датчиков в то время, когда вы будете дома. Поэтому в набор включены:

- цифровой дисплей Nokia 5110 (рис. 2.3);
- разноцветные светодиоды;
- маленький динамик 8 Ом.

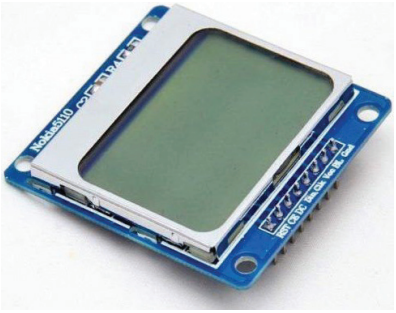


Рис. 2.3. Цифровой дисплей Nokia 5110

Следующая важнейшая функция – управление исполнительными электронными устройствами вашего умного дома. Это и освещение, и вентиляция, и полив растений, и обогрев жилища. Для подключения данных устройств к контроллеру в набор включен Relay Shield. Для управления данными устройствами в доме по инфракрасному каналу в набор включены ИК-пульт и ИК-приемник.

Еще одна полезная функция – доступ в дом с помощью RFID-карт. Для реализации в набор включены RFID-приемник RC522 и несколько RFID-карт и брелоков (см. рис. 2.4).

Рис. 2.4. RFID-приемник RC522

Умный дом невозможно представить без организации включения приборов по времени, для создания различных будильников необходим модуль часов реального времени RTC (см. рис. 2.5).

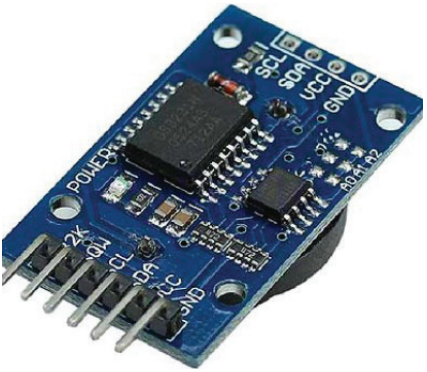


Рис. 2.5. Модуль часов реального времени DS3231

И конечно же, наш микроконтроллер должен иметь выход в Интернет. Здесь у нас будет несколько вариантов, в зависимости от вашего доступа к сети Интернет. С помощью модуля NodeMCU вы сможете подключиться к любой точке доступа Wi-Fi. Если у вас нет доступа к сети Интернет – это не проблема: бюджетный модуль GSM/GPRS SIM800L (рис. 2.6) предоставит возможность использовать сеть GSM для удаленного приема и передачи данных в Интернет.

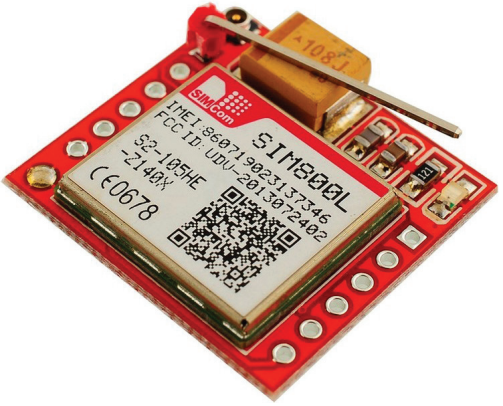


Рис. 2.6. Модуль GSM/GPRS SIM800L

В наборе также есть плата прототипирования, провода, резисторы, транзисторы. И этот учебник, который поможет вам создать элементы умного дома с использованием технологии интернет вещей. Мы начнем с чистого листа и в каждой главе будем добавлять функционала умному дому. В каждой главе представлен список необходимых деталей, приведена схема соединения деталей в формате интегрированной среды разработки Fritzing, скетч программы на встроенном языке Arduino с комментариями. В конце каждой главы содержатся ссылки для скачивания скетчей с сайта https://arduino-kit.ru/iotprog.

Готовы? Переворачивайте страницу и приступим!

3 УСТАНОВКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Разработка собственных приложений на базе плат, совместимых с архитектурой Arduino, осуществляется в официальной бесплатной среде программирования Arduino IDE. Среда предназначена для написания, компиляции и загрузки собственных программ в память микроконтроллера, установленного на плате Arduino-совместимого устройства. Основой среды разработки является язык Processing/Wiring – это фактически обычный C++, дополненный простыми и понятными функциями для управления вводом/выводом на контактах. Существуют версии среды для операционных систем Windows, Mac OS и Linux.

Я использую проверенные версии Arduino IDE, при написании этой книги использовались версии 1.6.1 и 1.6.5. Скачать Arduino IDE можно на официальном сайте www.arduino.cc.

3.1. Установка Arduino IDE в Windows

Отправляемся на страницу https://www.arduino.cc/en/Main/OldSoftwareReleases#previous (рис. 3.1), выбираем версию для операционной системы Windows и скачиваем архивный файл. Он занимает чуть более 80 Мбайт и содержит все необходимое, в том числе и драйверы. По окончании загрузки распаковываем скачанный файл в удобное для себя место.

Теперь необходимо установить драйверы. Подключаем Arduino к компьютеру. На контроллере должен загореться индикатор питания – зеленый светодиод. Windows начинает попытку установки драйвера, которая заканчивается сообщением **«Программное обеспечение драйвера не было установлено»**.

Открываем Диспетчер устройств. В составе устройств находим значок **Arduino MEGA** – устройство отмечено восклицательным знаком. Щелкаем правой кнопкой мыши на значке **Arduino MEGA** и в открывшемся окне выбираем пункт **Обновить драйверы** и далее пункт **Выполнить поиск драйверов на этом компьютере**. Указываем путь к драйверу – ту папку на компьютере, куда распаковывали скачанный архив. Пусть это будет папка drivers каталога установки Arduino – например, C:\arduino-1.6.5\drivers. Игнорируем все предупреждения Windows и получаем в результате сообщение **«Обновление программного обеспечения для данного устройства завершено успешно»**. В заголовке окна будет указан и COM-порт, на который установлено устройство.

Осталось запустить среду разработки Arduino IDE (рис. 3.2). В новой версии Arduino IDE в списке доступных портов отображается название платы Arduino.

https://www.arduino.cc/en/Main/OldSoftwareReleases#previous

HOME BUY SOFTWARE PRODUCTS LEARNING FORUM SUPPORT BLOG

1.6.11	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.10	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.9	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.8	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.7	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.6	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.5	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.4	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.3	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.2	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.1	Windows Windows Installer	MAC OS X MAC OS X Java 7+	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.0	Windows Windows Installer	MAC OS X MAC OS X Java 7	Linux 32 Bit Linux 64 Bit	Source code on Github

Рис. 3.1. Страница загрузки всех версий Arduino IDE с официального сайта Arduino

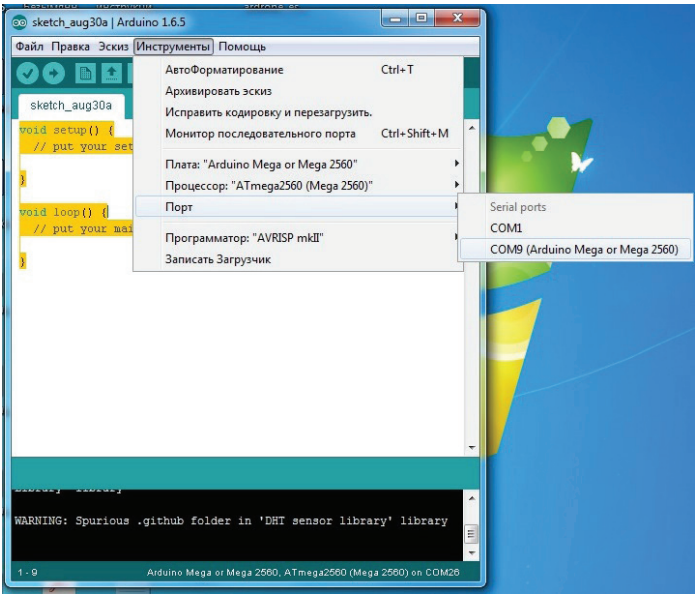


Рис. 3.2. Arduino IDE – среда разработки

3.2. Установка Arduino IDE в Linux

Рассмотрим установку Arduino IDE в операционной системе Linux. В Linux Ubuntu среда Arduino IDE устанавливается просто – она находится в репозитории стандартных приложений Linux. Выбираем Arduino IDE из списка доступных программ в меню Ubuntu **Приложения** ⇒ **Центр приложений Ubuntu** ⇒ **Загрузить приложение**. В списке разделов выбираем **Инструменты разработчика**, в списке следующего уровня – **Все приложения** и в следующем открывшемся списке – **Arduino IDE** (рис. 3.3). Щелкаем левой кнопкой мыши на значке этой программы, справа от нее появляется кнопка **Установить**, нажимаем на эту кнопку, и среда устанавливается автоматически. Для запуска Arduino IDE выбираем пункт **Приложения** ⇒ **Программирование** ⇒ **Arduino IDE.**

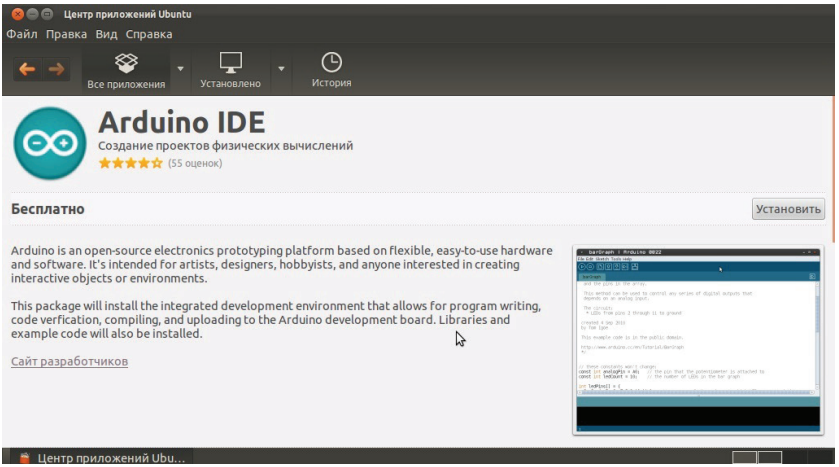


Рис. 3.3. Выбор программы из центра приложений Ubuntu

Для работы с необходимой версией программы скачиваем со страницы загрузки официального сайта проекта Arduino https://www.arduino.cc/en/Main/OldSoftwareReleases#previous (рис. 3.1) архив с версией программы для Linux. Затем распаковываем ее в желаемое место, например **/home/<user>/Arduino**. Для запуска программы из терминала

```
cd ~/Arduino
./arduino
```

3.3. Установка Arduino IDE в Mac OS X

Теперь рассмотрим установку Arduino IDE в операционной системе Mac OS X. Скачиваем со страницы загрузки официального сайта проекта Arduino https://www.arduino.cc/en/Main/OldSoftwareReleases#previous (рис. 3.1) архив с версией программы для OS X. Распаковываем архив и копируем его содержимое в папку **Программы**. Теперь Arduino IDE находится в списке программ Launchpad (рис. 3.4).

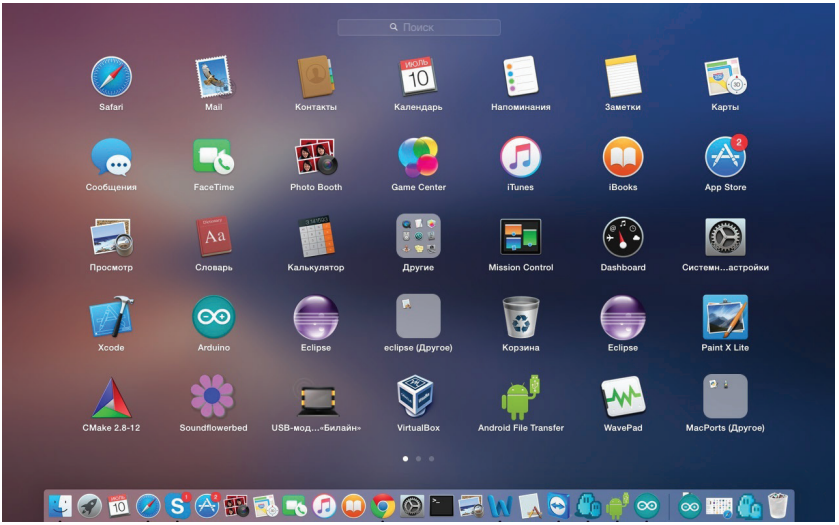


Рис. 3.4. Установка Arduino IDE в OS X

3.4. Настройка среды Arduino IDE

Среда разработки Arduino состоит из:

- редактора программного кода;
- области сообщений;
- окна вывода текста;
- панели инструментов с кнопками часто используемых команд;
- нескольких меню.

Программа, написанная в среде Arduino, носит название скетч. Скетч пишется в текстовом редакторе, который имеет цветовую подсветку создаваемого программного кода. Во время сохранения и экспорта проекта в области сообщений появляются пояс-

нения и информация об ошибках. Окно вывода текста показывает сообщения Arduino, включающие полные отчеты об ошибках и другую информацию. Кнопки панели инструментов позволяют проверить и записать программу, создать, открыть и сохранить скетч, открыть мониторинг последовательной шины.

Разрабатываемым скетчам дополнительная функциональность может быть добавлена с помощью библиотек, представляющих собой специальным образом оформленный программный код, реализующий некоторый функционал, который можно подключить к создаваемому проекту. Специализированных библиотек существует множество. Обычно библиотеки пишутся так, чтобы упростить решение той или иной задачи и скрыть от разработчика детали программно-аппаратной реализации. Среда Arduino IDE поставляется с набором стандартных библиотек: Serial, EEPROM, SPI, Wire и др. Они находятся в подкаталоге libraries каталога установки Arduino. Необходимые библиотеки могут быть также загружены с различных ресурсов. Папка библиотеки копируется в каталог стандартных библиотек (подкаталог libraries каталога установки Arduino). Внутри каталога с именем библиотеки находятся файлы *.cpp, *.h. Многие библиотеки снабжаются примерами, расположенными в папке examples. Если библиотека установлена правильно, то она появляется в меню **Эскиз** ⇒ **Импорт библиотек**. Выбор библиотеки в меню приведет к добавлению в исходный код строчки

```
#include <имя библиотеки.h>
```

Эта директива подключает заголовочный файл с описанием объектов, функций и констант библиотеки, которые теперь могут быть использованы в проекте. Среда Arduino будет компилировать создаваемый проект вместе с указанной библиотекой.

Перед загрузкой скетча требуется задать необходимые параметры в меню **Инструменты** ⇒ **Плата**, как показано на рис. 3.5, и **Инструменты** ⇒ **Последовательный порт**, показано на рис. 3.2.

Современные платформы Arduino перезагружаются автоматически перед загрузкой. На старых платформах необходимо нажать кнопку перезагрузки. На большинстве плат во время процесса загрузки будут мигать светодиоды RX и TX.

При загрузке скетча используется загрузчик (bootloader) Arduino – небольшая программа, загружаемая в микроконтроллер на плате. Она позволяет загружать программный код без исполь-

зования дополнительных аппаратных средств. Работа загрузчика распознается по миганию светодиода на цифровом выводе D13.

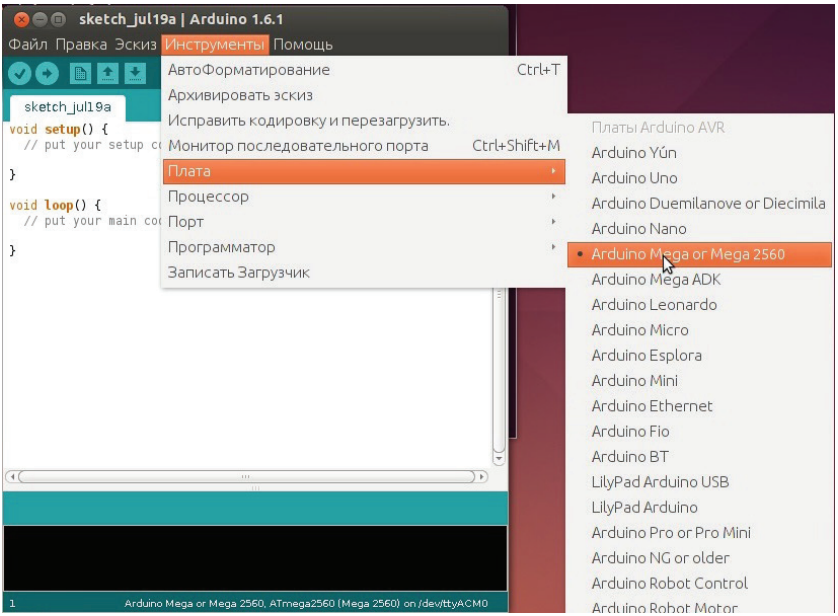


Рис. 3.5. Arduino IDE – выбор платы

Монитор последовательного порта (Serial Monitor) отображает данные, посылаемые в платформу Arduino (плату USB или плату последовательной шины). Для отправки данных необходимо ввести в соответствующее поле текст и нажать кнопку **Послать** (Send) или клавишу <**Enter**> (рис. 3.6). Затем следует из выпадающего списка выбрать скорость передачи, соответствующую значению Serial.begin в скетче. На ОС Mac или Linux при подключении мониторинга последовательной шины платформа Arduino будет перезагружена (скетч начнется сначала).

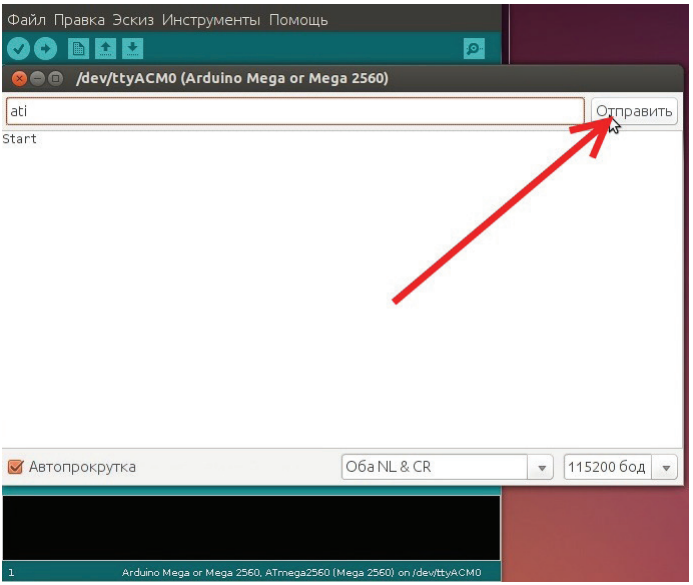


Рис. 3.6. Arduino IDE – монитор последовательного порта

3.5. Установка Arduino IDE для ESP8266

Arduino IDE для ESP8266 позволяет писать скетчи и загружать их одним кликом в ESP8266 в знакомой среде Arduino IDE. Рассмотрим установку Arduino IDE для ESP8266.

Сначала необходимо установить Arduino IDE с официального сайта версии не ниже 1.6.5. Запускаем Arduino IDE. Выбираем пункт **Файл** ⇒ **Настройки** и в поле **Additional Boards Manager URLs** вводим http://arduino.esp8266.com/stable/package_esp8266com_index.json или, если возникнут трудности, http://arduino.esp8266.com/versions/2.3.0/package_esp8266com_index.json. Нажимаем **OK** (см. рис. 3.7).

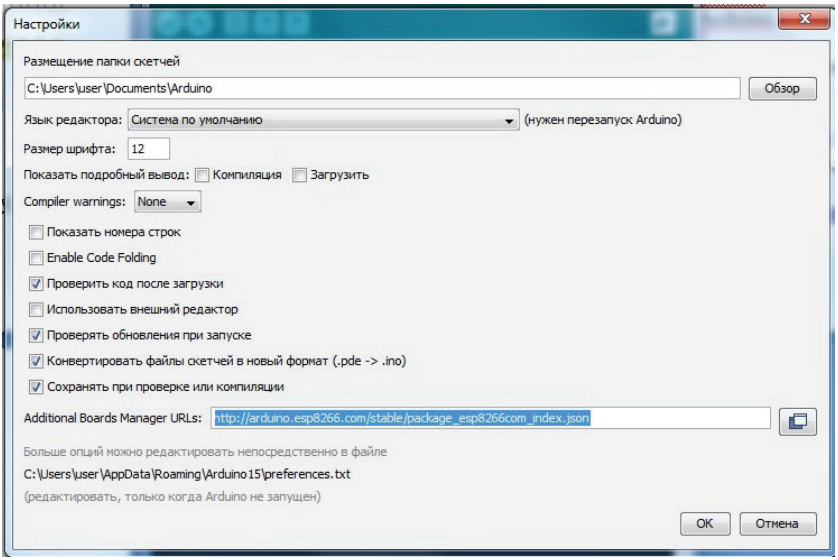


Рис. 3.7. Ввод адреса для скачивания Arduino IDE для ESP8266

Выбираем пункт **Инструменты** ⇒ **Плата** ⇒ **BoardsManager** и в списке ищем плату ESP8266. Выбираем этот пункт, версию и нажимаем на **Install** (см. рис. 3.8).

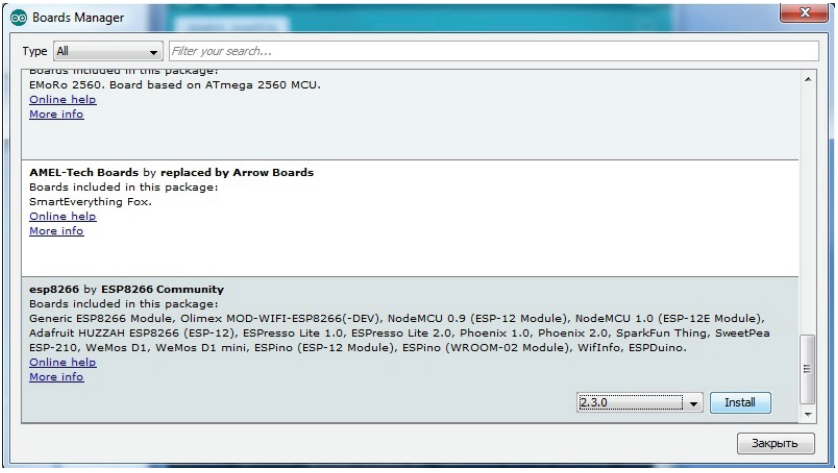
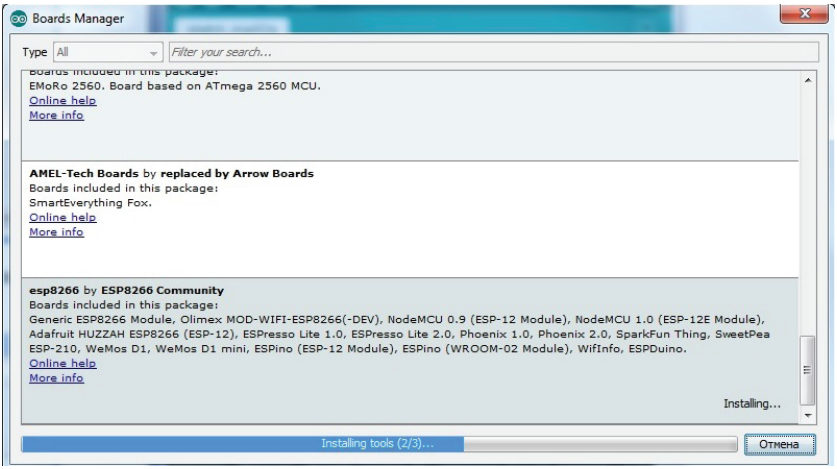


Рис. 3.8. Загрузка Arduino IDE для ESP8266 (начало)



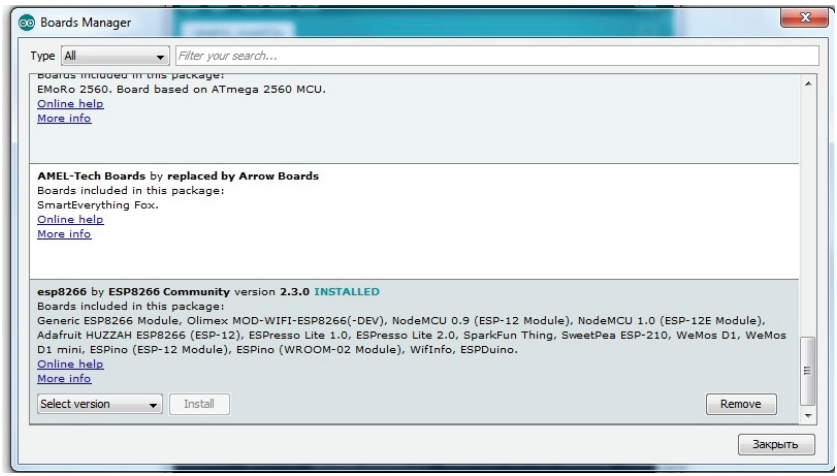


Рис. 3.8. Загрузка Arduino IDE для ESP8266 (окончание)

После загрузки программного обеспечения в списке плат (**Инструменты** ⇒ **Плата** ⇒) появятся платы ESP8266 (см. рис. 3.9).

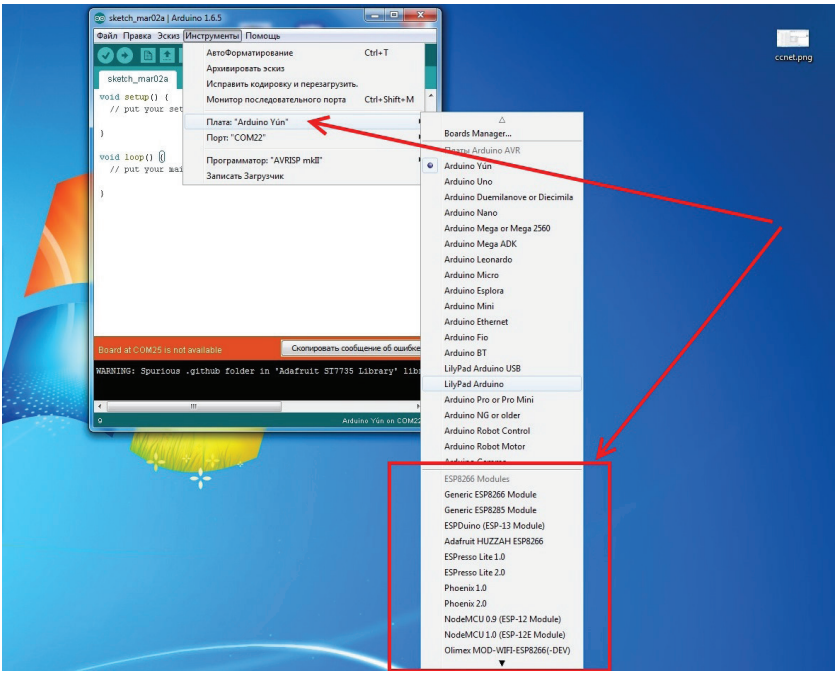


Рис. 3.9. Выбор платы ESP8266

4 ПОДКЛЮЧЕНИЕ ДАТЧИКОВ

Датчики являются основой любого «умного» дома. Независимо от индивидуальных требований и перечня задач, которые должна решать система в целом, именно датчики обеспечивают необходимую степень автоматизации и передают другим устройствам сигнал о необходимости включения или выключения в определенный момент. Правильный выбор данных приборов становится основой работоспособности и функциональности «умного» дома. В набор «Интернет вещей для умного дома» включены следующие датчики:

- датчик температуры DS18B20;
- датчик влажности DHT22;
- датчик увлажненности почвы;
- датчик воды;
- датчик огня;
- датчик пропана;
- датчик движения.

В следующих главах мы рассмотрим подключение данных датчиков к контроллерам Arduino MEGA и к модулю NodeMCU ESP8266, а также отправку данных с этих датчиков в сервис IoT Manager для возможности мониторинга данных датчиков на экране смартфона или телефона.

4.1. Датчик влажности и температуры DHT11 (DHT22)

В повседневной жизни влажность выступает немаловажным параметром, от степени влажности воздуха немало зависит наше самочувствие. Особенно чувствительными к влажности являются метеозависимые люди, а также люди, страдающие гипертонической болезнью, бронхиальной астмой, заболеваниями сердечно-сосудистой системы. При высокой сухости воздуха даже здоровые люди ощущают дискомфорт, сонливость, зуд и раздражение кожных покровов. Часто сухой воздух может спровоцировать заболевания дыхательной системы, начиная с ОРЗ и ОРВИ и заканчивая даже пневмонией.

Для измерения влажности воздуха умного дома в набор включен модуль датчика влажности и температуры DHT22 (рис. 4.1), который, в отличие от самого популярного датчика влажности DHT11, измеряет весь диапазон относительной влажности воздуха (0%..100%) и работает при отрицательных температурах (от –40 до 125 °C).

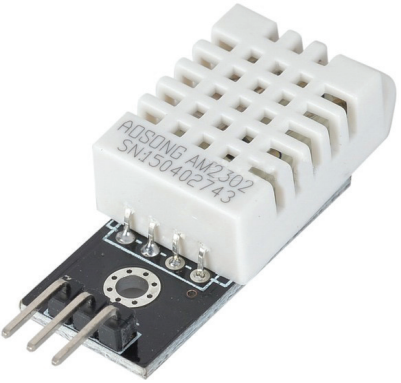
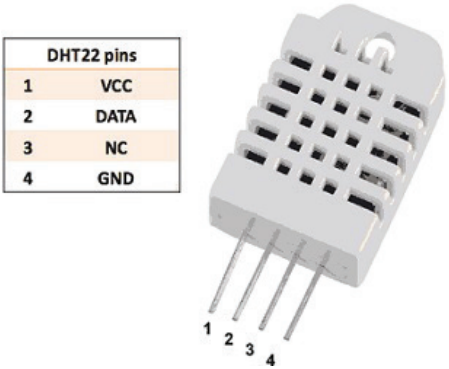


Рис. 4.1. Модуль датчика влажности и температуры DHT22

Плата модуля содержит основные компоненты: датчик температуры и относительной влажности DHT22 (рис. 4.2) в белом корпусе, светодиод индикации питания и вилка соединителя. Внутри DHT22 небольшая плата с компонентами: емкостным датчиком влажности, терморезистором, имеющим отрицательную характеристику, и микроконтроллером.



DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND

Рис. 4.2. Датчик DH22

Изготовитель вносит в память МК таблицу корректировки измерений каждого экземпляра для повышения точности работы. Данные модуля передаются в цифровом виде по однопроводному интерфейсу.

Рассмотрим подключение модуля датчика влажности и температуры DHT22 к плате Arduino MEGA и модулю NodeMCU ESP8266.

4.1.1. Подключение датчика DHT22 к плате Arduino MEGA

Для подключения модуля датчика влажности и температуры DHT22 к плате Arduino MEGA используется однопроводной интерфейс. Питание для датчика берем также с платы Arduino. Для удобного подключения к плате Arduino MEGA лучше всего использовать плату расширения MEGA Sensor Shield. Схема соединений представлена на рис. 4.3.

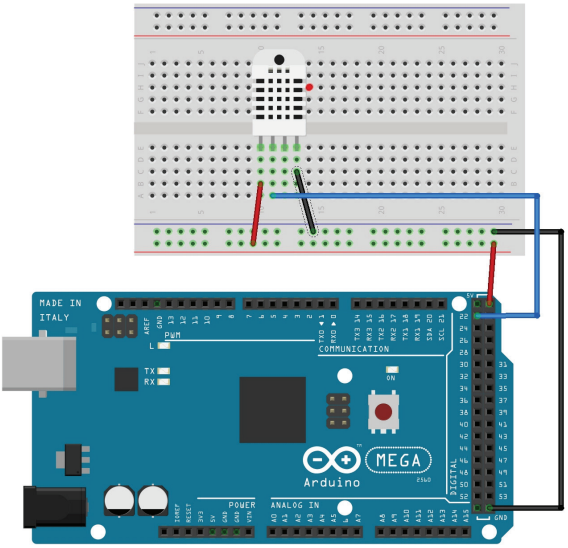


Рис. 4.3. Схема подключений DHT22 к Arduino MEGA

Для считывания данных датчика DTH22 с помощью Arduino существует готовая библиотека DHT. Для ее установки необходимо скопировать папку с файлами библиотеки в директорию libraries своего Arduino IDE.

Загрузим на плату Arduino MEGA скетч получения данных с датчика DHT11 и вывода в последовательный порт Arduino. Получение данных влажности оформим в виде отдельной процедуры get_data_humidity(). Содержимое скетча представлено в листинге 4.1.

Листинг 4.1

```
// подключение библиотеки DTH
#include "DHT.h"
// константы
```

```
#define DHTPIN 22               // пин подключения контакта DATA
#define DHTTYPE DHT22           // датчик DHT 22
#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
// создание экземпляра объекта DHT
DHT dht(DHTPIN, DHTTYPE);
// переменная для интервала измерений
unsigned long millis_int1=0;

void setup() {
   Serial.begin(9600);  // запуск последовательного порта
   dht.begin();         // запуск DHT
}

void loop() {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // получение данных с DHT11
      int humidity = get_data_humidity();
      // вывод в монитор последовательного порта
      Serial.print("humidity=");
      Serial.println(humidity);
      // запуск отсчета интервала
      millis_int1 = millis();
   }
}

int get_data_humidity() {
   int h = dht.readHumidity();
   return h;
}
```

Загрузим скетч на плату Arduino MEGA, откроем монитор последовательного порта и увидим вывод данных, получаемых с датчика влажности и температуры DHT22 (рис. 4.4).

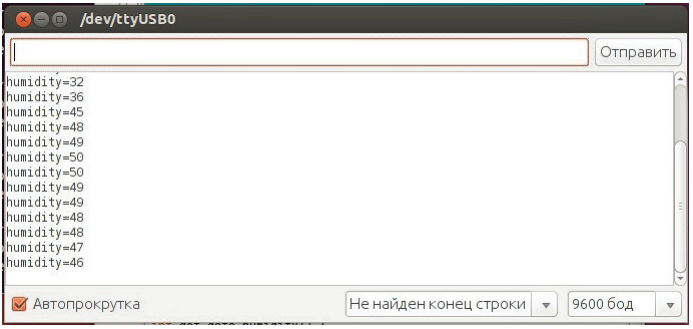


Рис. 4.4. Вывод данных DHT22 в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.1.2. Подключение датчика DHT22 к модулю NodeMCU ESP8266

Теперь рассмотрим подключение датчика DHT22 к модулю NodeMCU ESP8266. Схема соединений представлена на рис. 4.5.

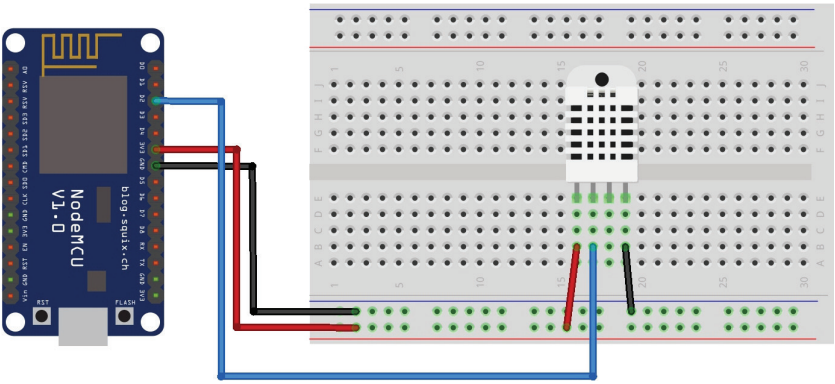


Рис. 4.5. Схема подключений DHT22 к NodeMCU ESP8266

Для считывания данных датчика DTH22 с помощью ESP8266 существует готовая библиотека DHT. Заметим, что эта библиотека именно для ESP8266, библиотека для Arduino не подходит. Содержимое скетча представлено в листинге 4.2.

Листинг 4.2

```
// подключение библиотеки DTH
#include "DHT.h"
// константы
#define DHTPIN 4                  // пин (D2) подключения контакта DATA
#define DHTTYPE DHT22             // датчик DHT 22
#define INTERVAL_GET_DATA 2000    // интервал измерений, мс
// создание экземпляра объекта DHT
DHT dht(DHTPIN, DHTTYPE);
// переменная для интервала измерений
unsigned long millis_int1 = 0;

void setup() {
   Serial.begin(9600);  // запуск последовательного порта
   dht.begin();         // запуск DHT
}

void loop() {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
```

```
      // получение данных с DHT11
      int humidity = get_data_humidity();
      // вывод в монитор последовательного порта
      Serial.print("humidity=");
      Serial.println(humidity);
      // запуск отсчета интервала
      millis_int1 = millis();
   }
}

int get_data_humidity() {
   int h = dht.readHumidity();
   return h;
}
```

Загрузим скетч на модуль NodeMcu ESP8266, откроем монитор последовательного порта и видим вывод данных, получаемых с датчика DHT22 (рис. 4.6)

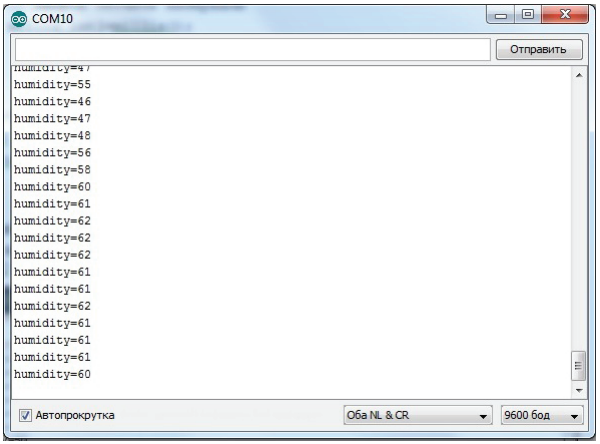


Рис. 4.6. Вывод данных DHT22 в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.2. Цифровой датчик температуры RI002

Мечта каждого человека – обеспечить максимальный комфорт и уют в своем доме. И первый шаг на пути к цели – создание оптимальной температуры в жилище – загородном доме, даче или квартире. «Умный дом» включает в себя полноценную кли-

матическую систему. И первый шаг на пути к этому – получение реальных данных значения температуры.

Для измерения температуры «умного» дома в набор включен датчик температуры RI002 (рис. 4.7). Это хорошо известный цифровой датчик температуры DS18B20 в водонепроницаемом корпусе из нержавейки. Преимущества водонепроницаемого корпуса – возможность измерить температуру в неблагоприятной для микросхем среде: в почве, на дожде или даже в аквариуме.

Рис. 4.7. Датчик температуры RI002

Этот датчик температуры основан на популярной микросхеме DS18B20. Он позволяет определить температуру окружающей среды в диапазоне от –55 °C до +125 °C и получать данные в виде цифрового сигнала с 12-битным разрешением по протоколу 1-Wire. Этот протокол позволит подключить огромное количество таких датчиков, используя всего 1 цифровой порт контроллера и всего 2 провода для всех датчиков: земли и сигнала. В этом случае применяется так называемое «паразитное питание», при котором датчик получает энергию прямо с линии сигнала. Каждый датчик имеет уникальный прошитый на производстве 64-битный код, который может использоваться микроконтроллером для общения с конкретным сенсором на общей шине.

Датчик температуры RI002 изготавливается с тремя выходными контактами (черный – GND, красный – Vdd и белый – Data).

Рассмотрим подключение модуля датчика температуры RI002 к плате Arduino MEGA и модулю NodeMCU ESP8266.

4.2.1. Подключение датчика RI002 к плате Arduino MEGA

Для подключения модуля датчика DS18B20 к плате Arduino MEGA используется однопроводной интерфейс 1-Wire. Питание для датчика берем также с платы Arduino. 1-Wire вывод датчика необходимо подтянуть к питанию резистором номиналом 4,7 кОм. Схема соединений представлена на рис. 4.8.

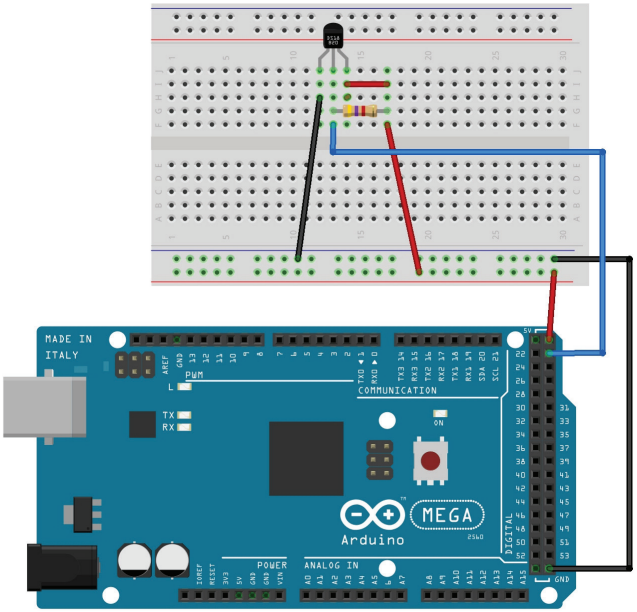


Рис. 4.8. Схема подключений DS18B20 к Arduino MEGA

Для считывания данных с 1-Wire датчика DS18B20 будем использовать библиотеку OneWire. Для ее установки необходимо скопировать папку с файлами библиотеки в директорию libraries своего Arduino IDE.

Загрузим на плату Arduino MEGA скетч получения данных с датчика температуры DS18B20 и вывода в последовательный порт Arduino. Получение данных влажности оформим в виде отдельной процедуры get_data_ds18b20(). Содержимое скетча представлено в листинге 4.3.

Листинг 4.3

```
#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
#define DS18B20PIN 23           // пин подключения контакта DATA
// подключение библиотеки OneWire
#include <OneWire.h>
// создание объекта OneWire
OneWire ds(DS18B20PIN);

// переменная для интервала измерений
unsigned long millis_int1 = 0;

void setup(void) {
   // запуск последовательного порта
   Serial.begin(9600);
}

void loop(void) {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // получение данных с DS18B20
      float temp = get_data_ds18b20();
      // вывод в монитор последовательного порта
      if(temp<100) {
        Serial.print("temp=");Serial.println(temp);
      }
      // старт интервала отсчета
      millis_int1 = millis();
   }
}
// получение данных с датчика DS18B20
float get_data_ds18b20()  {
  byte i;
  byte present = 0;
  byte data[12];
  byte addr[8];
  int Temp;
  float fTemp = 0.0;

  if ( !ds.search(addr)) {
    Serial.print("No more addresses.\n");
    ds.reset_search();
    return 999;
  }
  // вывод в монитор уникального адреса 1-Wire устройства
  for( i = 0; i < 8; i++)  {
    Serial.print(addr[i], HEX);
    Serial.print(" ");
  }
  if ( OneWire::crc8( addr, 7) != addr[7]) {
    Serial.print("CRC is not valid!\n");
    return 999;
```

```
  }
  if ( addr[0] != 0x28) {
    Serial.print("Device is not a DS18S20 family device.\n");
    return 999;
  }
  ds.reset();
  ds.select(addr);
  // запустить конвертацию температуры датчиком
  ds.write(0x44,1);
  delay(750); // ждем 750 мс
  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);
  // считываем ОЗУ датчика
  for ( i = 0; i < 9; i++) {
    data[i] = ds.read();
  }
  Temp=((data[1]<<8)+data[0]);
  // перевод в значение float
  fTemp=1.0*Temp/16+(float(Temp%16))*1.0/16;

  return fTemp;
}
```

Загрузим скетч на плату Arduino MEGA, откроем монитор последовательного порта и увидим вывод данных, получаемых с датчика температуры RI002 (рис. 4.9).

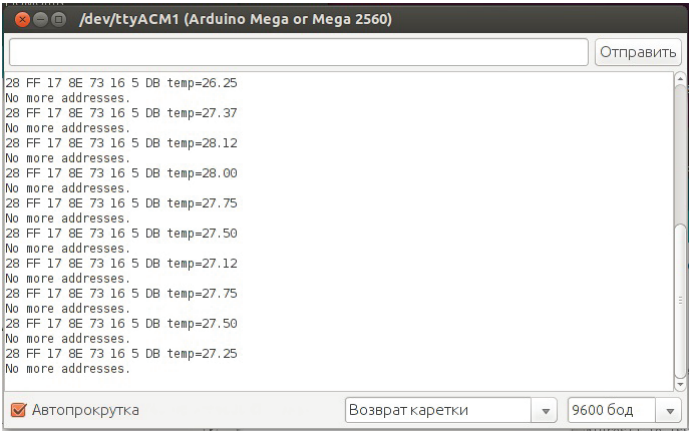


Рис. 4.9. Вывод данных RI002 в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.2.2. Подключение датчика DS18B20 к модулю NodeMCU ESP8266

Теперь рассмотрим подключение датчика DS18B20 к модулю NodeMCU ESP8266. 1-Wire вывод датчика необходимо подтянуть к питанию резистором номиналом 4,7 кОм. Схема соединений представлена на рис. 4.10.

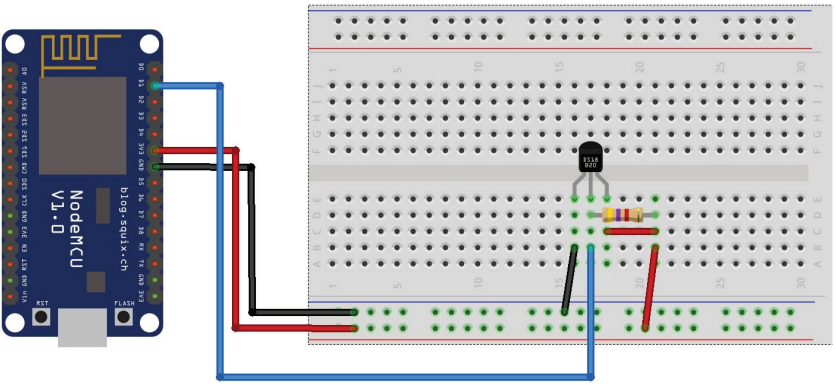


Рис. 4.10. Схема подключений DS18B20 к NodeMCU ESP8266

Для считывания данных датчика DS18B20 с помощью ESP8266 будем использовать библиотеку OneWire. Библиотека OneWire должна быть адаптирована для ESP8266 (внесены изменения в OneWire.h) . Содержимое скетча представлено в листинге 4.4.

Листинг 4.4

```
// константы
#define DSD18B20PIN 5      // пин подключения контакта DATA (D1, GPIO5)
#define INTERVAL_GET_DATA 2000  // интервал измерений, мс

// подключение библиотеки
// http://www.pjrc.com/teensy/td_libs_OneWire.html
#include <OneWire.h>
// создание экземпляра OneWire
OneWire  ds(DSD18B20PIN);
// переменная для интервала измерений
unsigned long millis_int1 = 0;

void setup(void) {
  // запуск последовательного порта
  Serial.begin(9600);
```

```
}

void loop() {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // получение данных с DS18B20
      float temp = get_data_ds18b20();
      // вывод в монитор последовательного порта
      if(temp<100) {
        Serial.print("temp=");Serial.println(temp);
      }
      // старт интервала отсчета
      millis_int1 = millis();
      // старт интервала отсчета
      millis_int1 = millis();
   }
}

float get_data_ds18b20(void) {
  byte i;
  byte present = 0;
  byte type_s;
  byte data[12];
  byte addr[8];
  float fTemp;

  if ( !ds.search(addr)) {
    Serial.println("No more addresses.");
    Serial.println();
    ds.reset_search();
    delay(250);
    return 999;
  }

  Serial.print("ROM =");
  for( i = 0; i < 8; i++) {
    Serial.write(' ');
    Serial.print(addr[i], HEX);
  }

  if (OneWire::crc8(addr, 7) != addr[7]) {
      Serial.println("CRC is not valid!");
      return 999;
  }
  Serial.println();

  // the first ROM byte indicates which chip
  switch (addr[0]) {
    case 0x10:
      Serial.println("  Chip = DS18S20");  // or old DS1820
      type_s = 1;
      break;
    case 0x28:
```

```
      Serial.println("  Chip = DS18B20");
      type_s = 0;
      break;
    case 0x22:
      Serial.println("  Chip = DS1822");
      type_s = 0;
      break;
    default:
      Serial.println("Device is not a DS18x20 family device.");
      return 999;
  }

  ds.reset();
  ds.select(addr);
  // запустить конвертацию температуры датчиком
  ds.write(0x44, 1);
  delay(1000);     // ждем 750 мс
  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);
  // считываем ОЗУ датчика
  for ( i = 0; i < 9; i++) {
    data[i] = ds.read();
    Serial.print(data[i], HEX);
    Serial.print(" ");
  }
  // перевод полученных данных в значение температуры
  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3;
    if (data[7] == 0x10) {
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    if (cfg == 0x00) raw = raw & ~7;
    else if (cfg == 0x20) raw = raw & ~3;
    else if (cfg == 0x40) raw = raw & ~1;
  }
  fTemp = (float)raw / 16.0;

  return fTemp;
}
```

Загрузим скетч на модуль NodeMCU ESP8266, откроем монитор последовательного порта и видим вывод данных, получаемых с датчика DS18B20 (рис. 4.11).

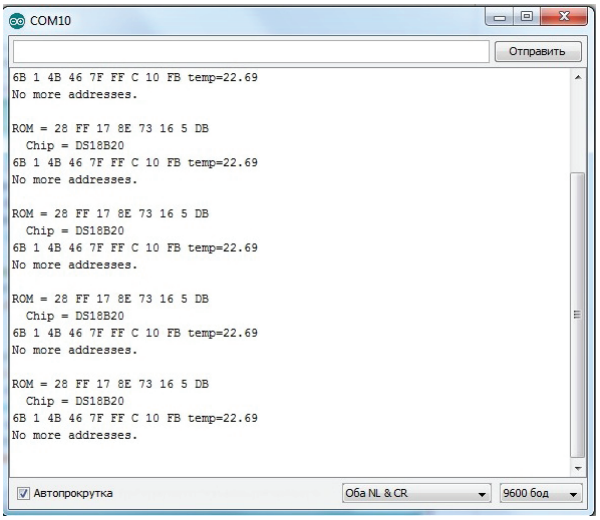


Рис. 4.11. Вывод данных DS18B20 в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.3. Датчик увлажненности почвы

Домашний уют – это атмосфера тепла в вашей квартире, желание возвращаться туда после трудного дня. Уют и комфорт в вашем доме оказывают непосредственное влияние на ваше самочувствие и настроение. Необходимое условие для создания уюта имеет использование комнатных цветов. Они доступны каждому из нас и при этом лучше любой мебели помогут создать уют и комфорт и как ничто другое просто вдохнуть в ваш дом чистую энергию.

Но чтобы домашние цветы радовали вас красотой, следует выполнять общие правила по уходу за комнатными растениями – необходимо создать благоприятный для них режим температуры воздуха, влажности и освещения.

Модуль влажности почвы (рис. 4.12) предназначен для определения влажности земли, в которую он погружен. Он позволяет узнать о недостаточном или избыточном поливе ваших домашних или садовых растений. Модуль состоит из двух частей:

контактного щупа YL-28 и датчика YL-38, щуп YL-28 соединен с датчиком YL-38 по двум проводам. Между двумя электродами щупа YL-28 создается небольшое напряжение. Если почва сухая, сопротивление велико, и ток будет меньше. Если земля влажная – сопротивление меньше, ток – чуть больше. По итоговому аналоговому сигналу можно судить о степени влажности.

Подключение данного модуля к контроллеру позволяет автоматизировать процесс полива ваших растений (своего рода «умный полив»).

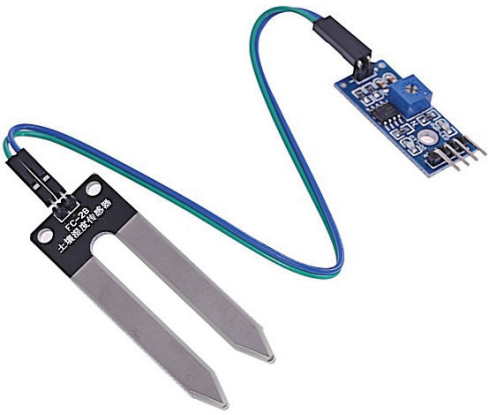


Рис. 4.12. Датчик увлажненности почвы Soil Moisture

Кроме контактов соединения со щупом, датчик YL-38 имеет четыре контакта для подключения к контроллеру.

- Vcc – питание датчика;
- GND – земля;
- A0 – аналоговое значение;
- D0 – цифровое значение уровня влажности.

Датчик YL-38 построен на основе компаратора LM393, который выдает напряжение на выход D0 по принципу: влажная почва – низкий логический уровень, сухая почва – высокий логический уровень. Уровень определяется пороговым значением, которое можно регулировать с помощью потенциометра. На вывод A0 подается аналоговое значение, которое можно передавать в контроллер для дальнейшей обработки, анализа и принятия решений.

Датчик YL-38 имеет два светодиода, сигнализирующих о наличии поступающего на датчик питания и уровня цифрового сиг-

нала на выходе D0. Наличие цифрового вывода D0 и светодиода уровня D0 позволяет использовать модуль автономно, без подключения к контроллеру.

Рассмотрим подключение датчика увлажненности почвы Soil Moisture к плате Arduino Mega и модулю NodeMcu ESP8266.

4.3.1. Подключение датчика Soil Moisture к плате Arduino MEGA

Подключение датчика Soil Moisture к плате Arduino MEGA мы будем производить по аналоговому входу. Питание для датчика берем также с платы Arduino. Схема соединений представлена на рис. 4.13.

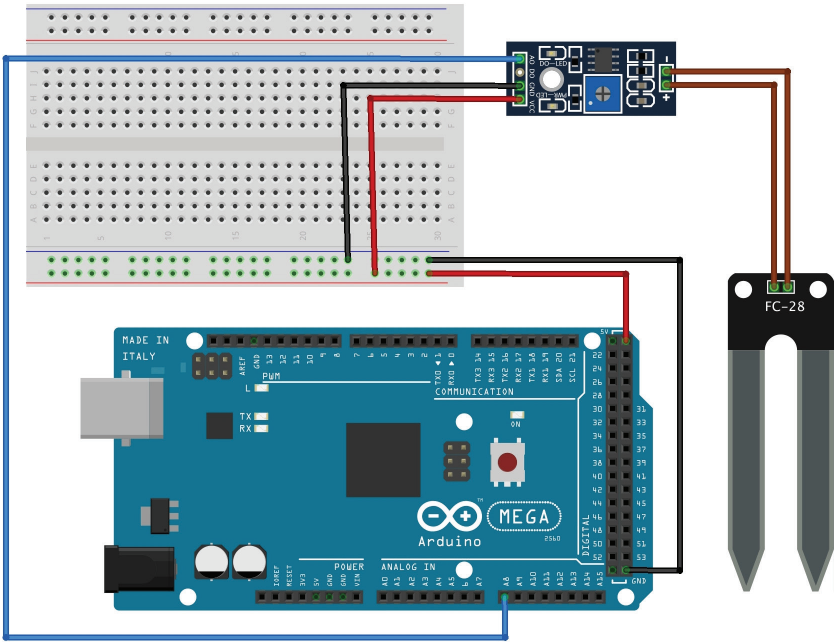


Рис. 4.13. Схема подключений датчика Soil Moisture к Arduino MEGA

Загрузим на плату Arduino MEGA скетч получения данных с датчика Soil Moisture и вывода в последовательный порт Arduino. Получение данных влажности оформим в виде отдельной процедуры get_data_soilmoisture(). Содержимое скетча представлено в листинге 4.5.

Листинг 4.5

```
#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
#define SOILMOISTUREPIN A8      // пин подключения контакта A0
// значение полного полива
#define MINVALUESOILMOISTURE 220
// значение критической сухости
#define MAXVALUESOILMOISTURE 900

// переменная для интервала измерений
unsigned long millis_int1=0;

void setup(void) {
   // запуск последовательного порта
   Serial.begin(9600);
}

void loop(void) {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // получение данных с датчика SoilMoisture
      float  moisture= get_data_soilmoisture();
      // вывод в монитор последовательного порта
      Serial.print("soilmoisture =");Serial.println(moisture);
      Serial.println(" %");
      // старт интервала отсчета
      millis_int1=millis();
   }
}
// получение данных с датчика SoilMoisture
float get_data_soilmoisture()  {
  // получение значения с аналогового вывода датчика
  int avalue=analogRead(SOILMOISTUREPIN);
  // масштабируем значение в проценты
  avalue=constrain(avalue, MINVALUESOILMOISTURE,MAXVALUESOILMOISTURE);
  int moisture=map(avalue, MINVALUESOILMOISTURE,
                   MAXVALUESOILMOISTURE,100,0);
  return (float)moisture;
}
```

Загрузим скетч на плату Arduino MEGA, откроем монитор последовательного порта и видим вывод данных, получаемых с датчика Soil Moisture (рис. 4.14). Подберите практическим путем аналоговые значения для констант MINVALUESOILMOISTURE (полный полив) и MINVALUESOILMOISTURE (критическая сухость).

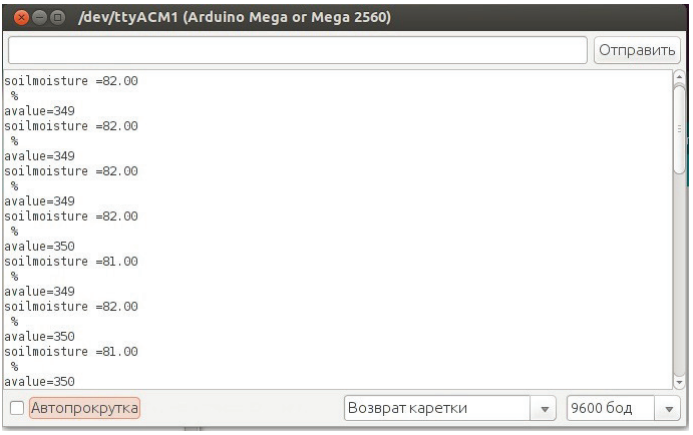


Рис. 4.14. Вывод данных с датчика Soil Moisture в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.3.2. Расширение аналоговых входов – мультиплексор CD4051

Модуль Node MCU имеет один канал АЦП, доступный для пользователей. Однако нам понадобится их гораздо больше. Как увеличить количество аналоговых входов? Для этого будем использовать мультиплексор CD4051 (см. рис. 4.15).

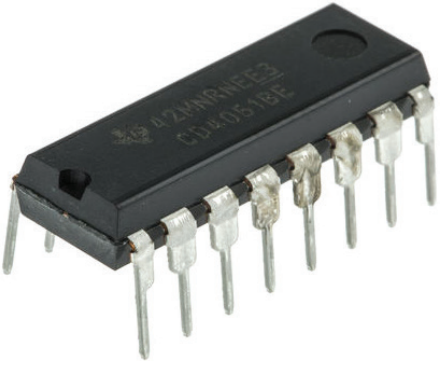


Рис. 4.15. Мультиплексор CD4051

Микросхема CD4051 является 8-канальным аналоговым мультиплексор/ демультиплексором, имеющим 8 входов (y0–y7) и 1 выход Z (см. рис. 4.16). Выбор считываемого входа осуществляется подачей цифровых сигналов на выходы s0–s2. То есть для подключения к модулю NodeMCU 8 аналоговых датчиков необходимо задействовать 3 цифровых выхода модуля и 1 аналоговый вход.

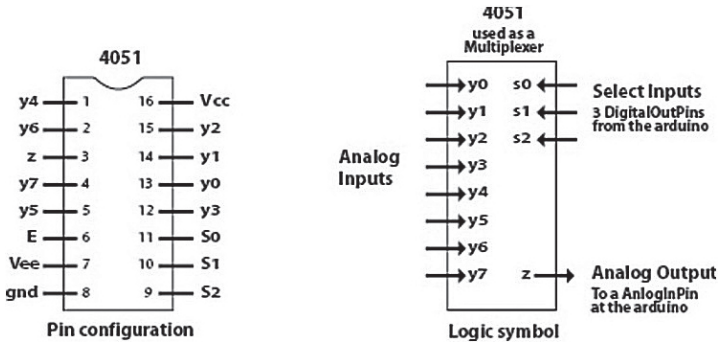


Рис. 4.16. Контакты мультиплексора CD4051

В листинге 4.6 представлен скетч циклического опроса 8 аналоговых датчиков, подключенных к 8 входам мультиплексора и через вход Z к аналоговому входу A0 модуля NodeMCU.

Листинг 4.6

```
// список пинов для подключения к s0, s1, s2 мультиплексора
// D5, D7, D8 (GPIO 14, 13, 15)
int pins[]={14, 13, 15};
// Массив двоичных чисел, определяющих номер выбранного входа/выхода
// микросхемы 4051, с 1 по 8.
int bin [] = { B000, B001, B010, B011, B100, B101, B110, B111 } ;
// служебные переменные
int row;
int r0 = 0;
int r1 = 0;
int r2 = 0;
int  avalue =0;

void setup(void) {
   // входы подключения к мультиплексору как OUTPUT
```

```
   for(int i=0;i<3;i++) {
      pinMode(pins[i],OUTPUT);
   }
   // запуск последовательного порта
   Serial.begin(9600);
}

void loop(void) {
   for(int i=0;i<8;i++) {
      // выбор входа мультиплексора

      row = bin [i] ;
      r0 = row & 0x01 ;
      r1 = (row >> 1) & 0x01 ; //
      r2 = (row >> 2) & 0x01 ; //
      digitalWrite (pins[i], r0) ;
      digitalWrite (pins[i], r1) ;
      digitalWrite (pins[i], r2) ;
      // получение данных с A0
      Avalue = analogRead(A0);
      // вывод в монитор последовательного порта
      Serial.print("analog input =");Serial.print(i);
      Serial.println(" = "); Serial.println(avalue);
   }
   // пауза
   delay(2000);
}
```

4.3.3. Подключение датчика Soil Moisture к модулю NodeMCU ESP8266

Теперь рассмотрим подключение датчика Soil Moisture к модулю NodeMCU ESP8266. Датчик Soil Moisture подключаем ко входу y0 мультиплексора. Для выбора аналогового входа мультиплексора используем контакты D5, D7, D8 модуля NodeMCU. Схема соединений представлена на рис. 4.17.

Загрузим на модуль NodeMCU скетч получения данных с датчика Soil Moisture и вывода в последовательный порт Arduino. Получение данных влажности оформим в виде отдельной процедуры get_data_soilmoisture(). Для выбора аналогового входа мультиплексора y0 подаем на контакты D5, D7, D8 сигнал низкого уровня LOW. Содержимое скетча представлено в листинге 4.7.

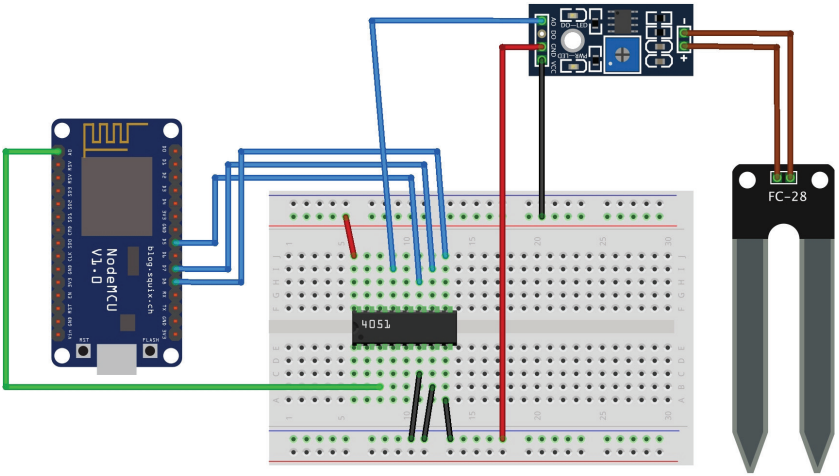


Рис. 4.17. Схема подключений датчика Soil Moisture к NodeMCU ESP8266

Листинг 4.7

```
#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
#define SOILMOISTUREPIN A0      // аналоговый вход
// значение полного полива
#define MINVALUESOILMOISTURE 220
// значение критической сухости
#define MAXVALUESOILMOISTURE 900

// переменная для интервала измерений
unsigned long millis_int1=0;

void setup(void) {
   // входы подключения к мультиплексору D5, D7, D8 (GPIO 14, 13, 15)
   // как OUTPUT
   pinMode(14,OUTPUT);
   pinMode(13,OUTPUT);
   pinMode(15,OUTPUT);
   // запуск последовательного порта
   Serial.begin(9600);
}

void loop(void) {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // получение данных с датчика SoilMoisture
      float  moisture = get_data_soilmoisture();
      // вывод в монитор последовательного порта
      Serial.print("soilmoisture =");Serial.print(moisture);
      Serial.println(" %");
```

```
      // старт интервала отсчета
      millis_int1=millis();
   }
}
// получение данных с датчика SoilMoisture
float get_data_soilmoisture()  {
  // выбор входа мультиплексора CD4051 - y0 (000)
  digitalWrite(14,LOW);
  digitalWrite(13,LOW);
  digitalWrite(15,LOW);
  // получение значения с аналогового вывода датчика
  int avalue = analogRead(SOILMOISTUREPIN);
  Serial.print("avalue =");Serial.println(avalue);
  // масштабируем значение в проценты
  Avalue=constrain(avalue, MINVALUESOILMOISTURE,MAXVALUESOILMOISTURE);
  int moisture = map(avalue, MINVALUESOILMOISTURE,
                   MAXVALUESOILMOISTURE,100,0);
  return (float)moisture;
}
```

Загрузим скетч на модуль NodeMCU, откроем монитор последовательного порта и видим вывод данных, получаемых с датчика Soil Moisture (рис. 4.18). Подберите практическим путем аналоговые значения для констант MINVALUESOILMOISTURE (полный полив) и MINVALUESOILMOISTURE (критическая сухость).

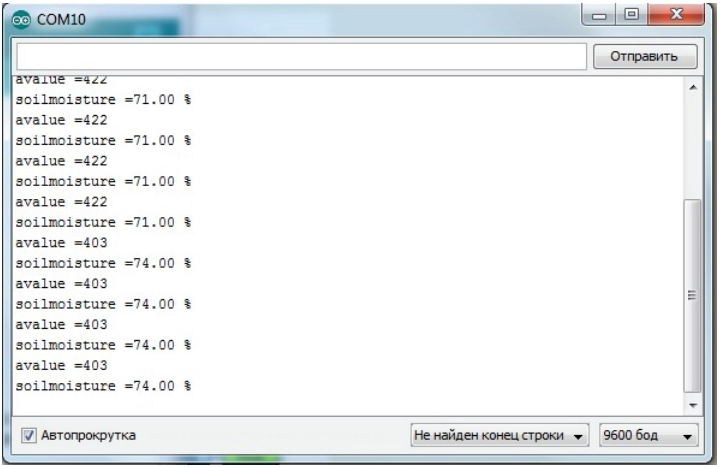


Рис. 4.18. Вывод данных Soil Moisture в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.4. Датчик уровня воды

Одна из главных задач умного дома – заботиться о своей сохранности, не допускать взломов, пожаров, затоплений и прочих повреждений. Вот о защите от протечек и затопления мы сегодня и поговорим. Точнее сказать, пока только об обнаружении протечек.

Для обнаружения протечек будем использовать датчик воды. Датчики воды предназначены для определения уровня воды в различных емкостях, где недоступен визуальный контроль, с целью предупреждения перенаполнения емкости водой через критическую отметку. Данный датчик воды (рис. 4.19) – погружной. Чем больше погружение датчика в воду, тем меньше сопротивление между двумя соседними проводами.

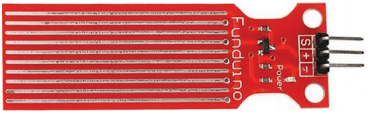


Рис. 4.19. Датчик уровня воды

Датчик имеет три контакта для подключения к контроллеру.

- + – питание датчика;
- - – земля;
- S – аналоговое значение.

На вывод S подается аналоговое значение, которое можно передавать в контроллер для дальнейшей обработки, анализа и принятия решений. Датчик имеет красный светодиод, сигнализирующий о наличии поступающего на датчик питания.

Рассмотрим подключение датчика уровня воды к плате Arduino Mega и модулю NodeMCU ESP8266.

4.4.1. Подключение датчика уровня воды к плате Arduino MEGA

Подключение датчика уровня воды к плате Arduino MEGA мы будем производить по аналоговому входу. Питание для датчика берем также с платы Arduino. Схема соединений представлена на рис. 4.20.

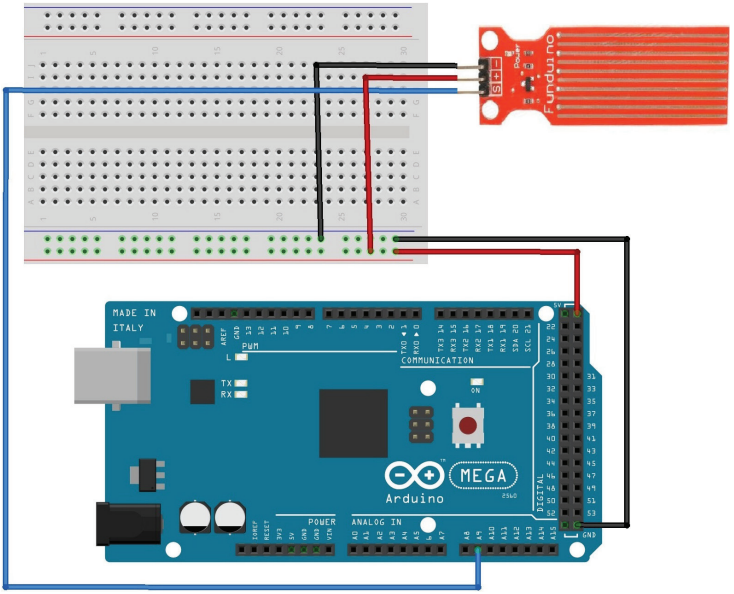


Рис. 4.20. Схема подключений датчика уровня воды к плате Arduino MEGA

Загрузим на плату Arduino MEGA скетч получения данных с датчика уровня воды, перевода аналогового значения в сантиметры (0–4) и вывода в последовательный порт Arduino. Получение данных влажности оформим в виде отдельной процедуры get_data_levelwater(). Содержимое скетча представлено в листинге 4.8.

Листинг 4.8

```
#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
#define LEVELWATERPIN A9        // пин подключения контакта S
// пороговое значение протечки
#define LEVELWATER 100

// переменная для интервала измерений
unsigned long millis_int1 = 0;

void setup(void) {
   // запуск последовательного порта
   Serial.begin(9600);
}
```

```
void loop(void) {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // получение данных с датчика уровня воды
      float  levelwater = get_data_levelwater();
      // вывод в монитор последовательного порта
      Serial.print("levelwater =");Serial.println(levelwater);
      if(levelwater>LEVELWATER)
         Serial.println(" flood !!!");
      // старт интервала отсчета
      millis_int1 = millis();
   }
}

// получение данных с датчика уровня воды
float get_data_levelwater()  {
  // получение значения с аналогового вывода датчика
  int avalue = analogRead(LEVELWATERPIN);
  return (float) avalue;
}
```

Загрузим скетч на плату Arduino MEGA, откроем монитор последовательного порта и видим вывод данных, получаемых с датчика уровня воды (рис. 4.21). При попадании воды на датчик выводим сообщение о протекании. Подберите практическим путем аналоговое значение для константы LEVELWATER (пороговое значение погружения).

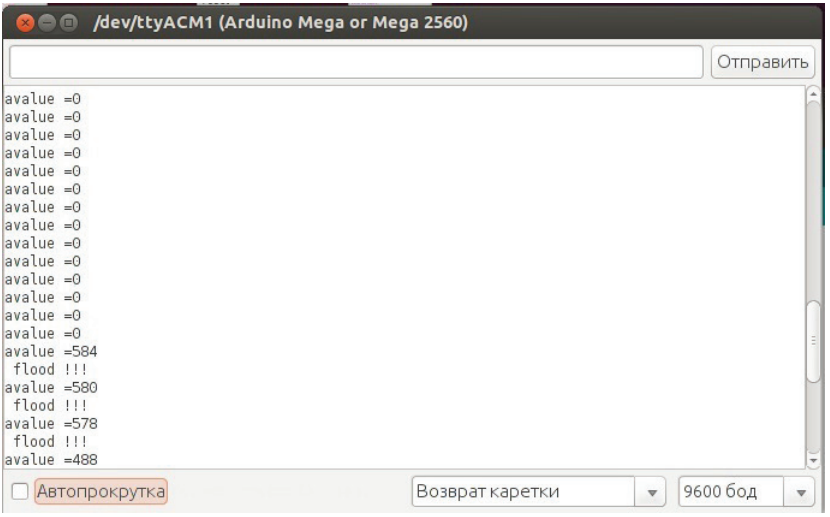


Рис. 4.21. Вывод данных уровня затопления в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.4.2. Подключение датчика уровня воды к модулю NodeMCU ESP8266

Теперь рассмотрим подключение датчика уровня воды к модулю NodeMCU ESP8266.

Схема соединений представлена на рис. 4.22.

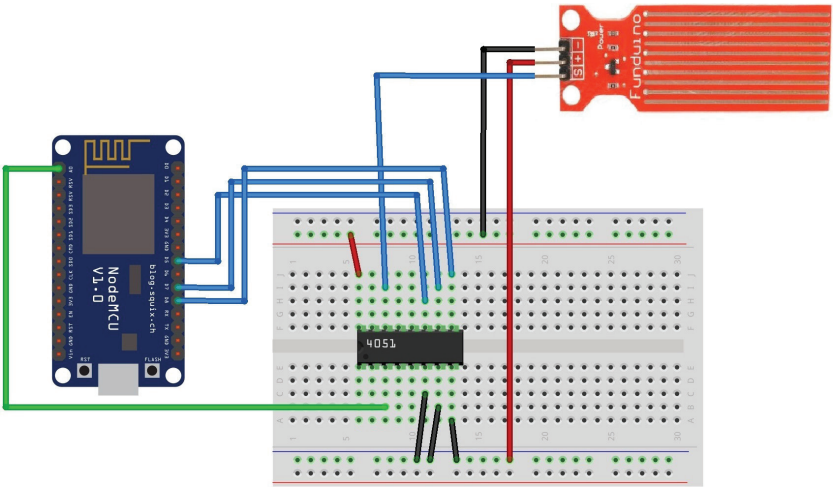


Рис. 4.22. Схема подключений датчика Soil Moisture к NodeMCU ESP8266

Загрузим на модуль NodeMCU скетч получения данных с датчика уровня воды и вывода в последовательный порт Arduino. Получение данных влажности оформим в виде отдельной процедуры get_data_soilmoisture(). Для выбора аналогового входа мультиплексора y1 подаем на контакты D7, D8 сигнал низкого уровня LOW, а на контакт D5 – высокого уровня HIGH. Содержимое скетча представлено в листинге 4.9.

Листинг 4.9

```
#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
#define LEVELWATERPIN A0        // аналоговый пин
// пороговое значение протечки
#define LEVELWATER 100

// переменная для интервала измерений
unsigned long millis_int1 = 0;
```

```
void setup(void) {
   // входы подключения к мультиплексору D5, D7, D8 (GPIO 14, 13, 15)
   // как OUTPUT
   pinMode(14,OUTPUT);
   pinMode(13,OUTPUT);
   pinMode(15,OUTPUT);
   // запуск последовательного порта
   Serial.begin(9600);
}

void loop(void) {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // получение данных с датчика уровня воды
      float  levelwater = get_data_levelwater();
      // вывод в монитор последовательного порта
      Serial.print("levelwater =");Serial.println(levelwater);
      if(levelwater>LEVELWATER)
         Serial.println(" flood !!!");
      // старт интервала отсчета
      millis_int1 = millis();
   }
}

// получение данных с датчика уровня воды
float get_data_levelwater()  {
  // выбор входа мультиплексора CD4051 - y1 (001)
  digitalWrite(14,HIGH);
  digitalWrite(13,LOW);
  digitalWrite(15,LOW);
  // получение значения с аналогового вывода датчика
  int avalue = analogRead(LEVELWATERPIN);
  return (float) avalue;
}
```

Загрузим скетч на модуль NodeMCU, откроем монитор последовательного порта и увидим вывод данных, получаемых с датчика уровня воды (рис. 4.23). При попадании воды на датчик выводим сообщение о протекании. Подберите практическим путем аналоговое значение для константы LEVELWATER (пороговое значение погружения).

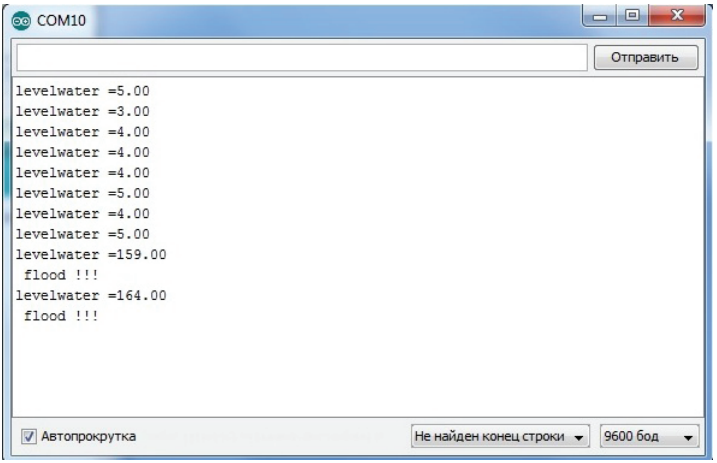


Рис. 4.23. Вывод данных уровня затопления в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.5. Датчик газов MQ-2

Одна из самых важных задач в вопросе безопасности умного дома – обнаружение утечки газа. Для того чтобы плата Arduino успешно решала задачи такого рода, нужно подключить к ней датчик газа MQ-2. Датчик MQ-2 (рис. 4.24) определит концентрацию углеводородных газов (пропан, метан, н-бутан), дыма (взвешенных частиц, являющихся результатом горения) и водорода в окружающей среде. Датчик можно использовать для обнаружения утечек газа и задымления. В газоанализатор встроен нагревательный элемент, который необходим для химической реакции. Поэтому во время работы сенсор будет горячим. Для получения стабильных показаний новый сенсор необходимо один раз прогреть (оставить включенным) в течение 24 часов. После этого стабилизация после включения будет занимать около минуты.

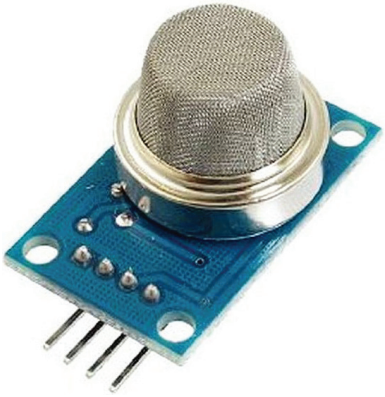


Рис. 4.24. Датчик газов MQ-2

В зависимости от уровня газа в атмосфере меняется внутреннее сопротивление датчика. MQ-2 имеет аналоговый выход, поэтому напряжение на этом выходе будет меняться пропорционально уровню газа в окружающей среде. Для определения по логическому уровню также имеется цифровой выход. На модуле датчика есть встроенный потенциометр, который позволяет настроить чувствительность этого датчика в зависимости от того, насколько точно вы хотите регистрировать уровень газа.

Теперь о единицах измерения. На территории бывшего Советского Союза показатели принято измерять в процентах (%) или же непосредственно в массе к объему (мг/м^3). В зарубежных странах применяет такой показатель, как ppm.

Сокращение ppm расшифровывается как parts per million (частей на миллион). Например, 1 ppm = 0,0001%.

Диапазон измерений датчика:

- пропан: 200–5000 ppm;
- бутан: 300–5000 ppm;
- метан: 500–20000 ppm;
- водород: 300–5000 ppm.

Рассмотрим подключение датчика MQ-2 к плате Arduino Mega и модулю NodeMcu ESP8266.

4.5.1. Подключение датчика MQ-2 к плате Arduino MEGA

Подключение датчика MQ-2 к плате Arduino MEGA мы будем производить по аналоговому входу. Питание для датчика берем также с платы Arduino. Схема соединений представлена на рис. 4.25.

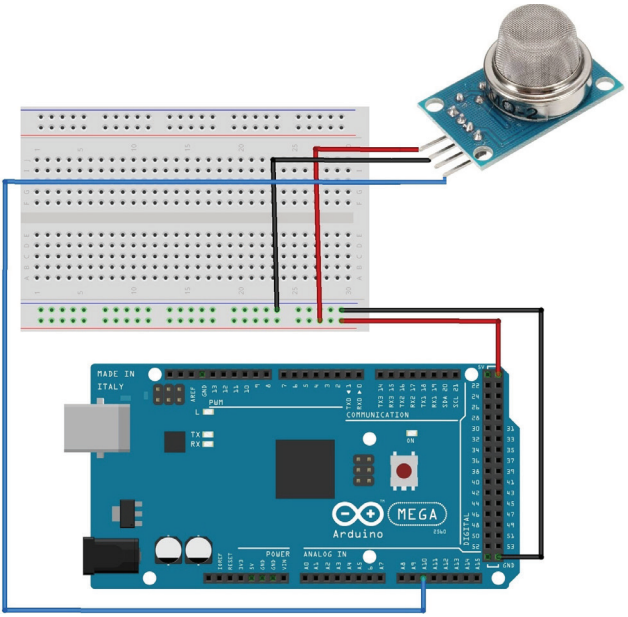


Рис. 4.25. Схема подключений датчика MQ-2 к плате Arduino MEGA

Загрузим на плату Arduino MEGA скетч получения данных с датчика MQ-2 и вывода в последовательный порт Arduino. Процедуры определения по данным, приходящим с аналогового входа:

- get_data_ppmpropan() – содержание пропана в ppm;
- get_data_ppmmethan() – содержание пропана в ppm;
- get_data_ppmsmoke() – содержание дыма.

Для получения данных с датчиков MQ будем использовать библиотеку TroykaMQ.

Содержимое скетча представлено в листинге 4.10.

Листинг 4.10

```
// библиотека для работы с датчиками MQ
#include <TroykaMQ.h>

#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
// пин, к которому подключен датчик
#define MQ2PIN         A10
// создаем объект для работы с датчиком
MQ2 mq2(MQ2PIN);

// переменная для интервала измерений
unsigned long millis_int1 = 0;
```

```
void setup() {
  // открываем последовательный порт
  Serial.begin(9600);
  // калибровка
  mq2.calibrate();
  mq2.getRo();
}

void loop() {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // получение данных с датчика mq2
      float  propan = get_data_ppmpropan();
      // выводим значения газа в ppm
      Serial.print("propan =");
      Serial.print(propan);
      Serial.println(" ppm ");
      float  methan= get_data_ppmmethan();
      // выводим значения газа в ppm
      Serial.print("methan =");
      Serial.print(methan);
      Serial.println(" ppm ");
      float  smoke= get_data_ppmsmoke();
      // выводим значения газа в ppm
      Serial.print("smoke =");
      Serial.print(smoke);
      Serial.println(" ppm ");
      // старт интервала отсчета
      millis_int1=millis();
   }
}

// получение данных содержания пропана с датчика MQ2
float get_data_ppmpropan()  {

  Serial.println(mq2.readRatio());
  // получение значения
  float value = mq2.readLPG();

  return value;
}
// получение данных содержания метана с датчика MQ2
float get_data_ppmmethan()  {

  Serial.println(mq2.readRatio());
  // получение значения
  float value = mq2.readMethane();

  return value;
}
// получение данных содержания дыма с датчика MQ2
```

```
float get_data_ppmsmoke()  {

  Serial.println(mq2.readRatio());
  // получение значения
  float value = mq2.readSmoke();

  return value;
}
```

Загрузим скетч на плату Arduino MEGA, откроем монитор последовательного порта и увидим вывод данных о содержании пропана, метана и дыма (рис. 4.26).

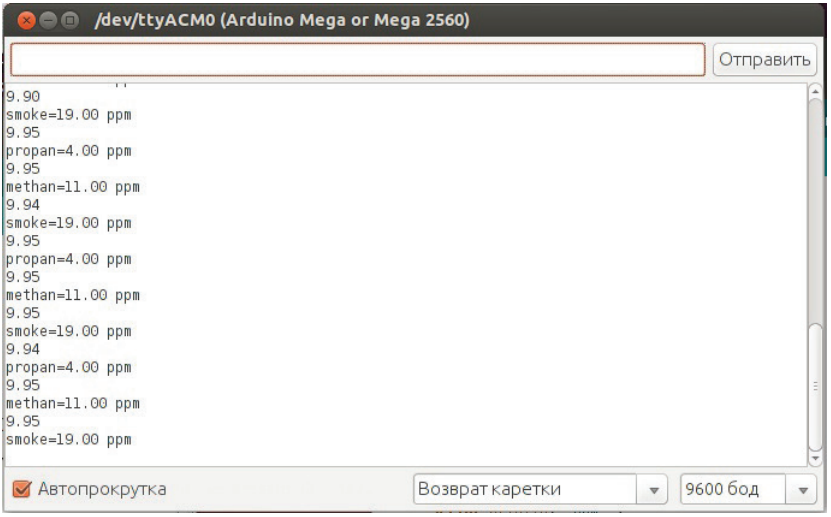


Рис. 4.26. Вывод данных датчика MQ-2 в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.5.2. Подключение датчика MQ-2 к модулю NodeMCU ESP8266

Теперь рассмотрим подключение датчика MQ-2 к модулю NodeMCU ESP8266. Датчик MQ-2 подключаем к входу y2 мультиплексора. Для выбора аналогового входа мультиплексора используем контакты D5, D7, D8 модуля NodeMCU. Схема соединений представлена на рис. 4.27.

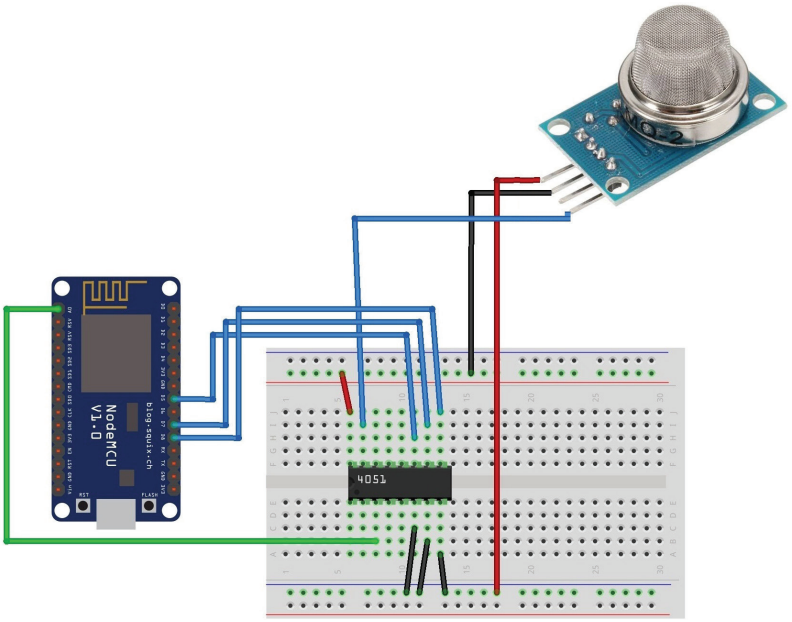


Рис. 4.27. Схема подключений датчика MQ-2 к NodeMCU ESP8266

Загрузим на модуль NodeMCU скетч получения данных с датчика MQ-2 и вывода в последовательный порт Arduino. Для выбора аналогового входа мультиплексора y2 подаем на контакты D5, D8 сигнал низкого уровня LOW, на контакт D7 – сигнал высокого уровня HIGH.

Процедуры определения по данным, приходящим с аналогового входа:

- get_data_ppmpropan() – содержание пропана в ppm;
- get_data_ppmmethan() – содержание пропана в ppm;
- get_data_ppmsmoke() – содержание дыма.

Для получения данных с датчиков MQ будем использовать библиотеку TroykaMQ.

Содержимое скетча представлено в листинге 4.11.

Листинг 4.11

```
// библиотека для работы с датчиками MQ
#include <TroykaMQ.h>

#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
```

```
// аналоговый пин
#define MQ2PIN         A0
// создаем объект для работы с датчиком
MQ2 mq2(MQ2PIN);

// переменная для интервала измерений
unsigned long millis_int1 = 0;

void setup() {
   // входы подключения к мультиплексору D5, D7, D8 (GPIO 14, 13, 15)
   // как OUTPUT
   pinMode(14,OUTPUT);
   pinMode(13,OUTPUT);
   pinMode(15,OUTPUT);
  // открываем последовательный порт
  Serial.begin(9600);
  // выбор входа мультиплексора CD4051 - y2 (010)
  digitalWrite(14,LOW);
  digitalWrite(13,HIGH);
  digitalWrite(15,LOW);
  // калибровка
  mq2.calibrate();
  mq2.getRo();
}

void loop() {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // выбор входа мультиплексора CD4051 - y2 (010)
      digitalWrite(14,LOW);
      digitalWrite(13,HIGH);
      digitalWrite(15,LOW);
      // получение данных с датчика mq2
      float  propan = get_data_ppmpropan();
      // выводим значения газа в ppm
      Serial.print("propan=");
      Serial.print(propan);
      Serial.println(" ppm ");
      float  methan = get_data_ppmmethan();
      // выводим значения газа в ppm
      Serial.print("methan=");
      Serial.print(methan);
      Serial.println(" ppm ");
      float  smoke = get_data_ppmsmoke();
      // выводим значения газа в ppm
      Serial.print("smoke=");
      Serial.print(smoke);
      Serial.println(" ppm ");
      // старт интервала отсчета
      millis_int1 = millis();
   }
}
```

```
// получение данных содержания пропана с датчика MQ2
float get_data_ppmpropan()  {
  Serial.println(mq2.readRatio());
  // получение значения
  float value = mq2.readLPG();
  return value;
}
// получение данных содержания метана с датчика MQ2
float get_data_ppmmethan()  {
  Serial.println(mq2.readRatio());
  // получение значения
  float value = mq2.readMethane();
  return value;
}
// получение данных содержания дыма с датчика MQ2
float get_data_ppmsmoke()  {
  Serial.println(mq2.readRatio());
  // получение значения
  float value = mq2.readSmoke();
  return value;
}
```

Загрузим скетч на модуль NodeMCU, откроем монитор последовательного порта и видим вывод данных, получаемых с датчика MQ-2 (рис. 4.28).

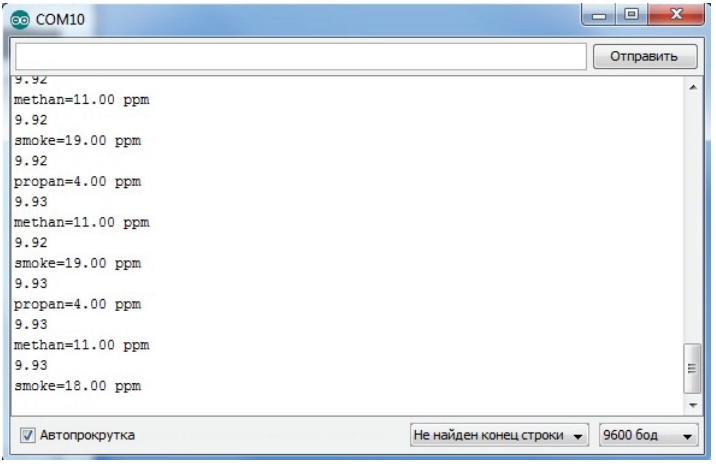


Рис. 4.28. Вывод данных датчика MQ-2 в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.6. Датчик угарного газа MQ-7

Рассмотрим еще один датчик для обнаружения утечки газа. Это датчик угарного газа MQ-7 (рис. 4.29).

Рис. 4.29. Датчик угарного газа MQ-7

Основным источником выделения угарного газа CO является сгорание углеродного топлива при недостаточном количестве кислорода. Углерод «не догорает», и вместо углекислого газа CO_2 в атмосферу выбрасывается угарный газ CO. Источником CO в доме, при неправильной эксплуатации, могут выступать дровяные печи, газовые конфорки, газовые котлы и прочая отопительная техника, работающая на углеродном топливе. В выхлопе бензинового двигателя автомобиля содержание CO может быть до 3%, а по гигиеническим нормам его должно быть не более 20 мг/м^3 (около 0,0017%).

Угарный газ (CO) чрезвычайно ядовит, но при этом не обладает ни цветом, ни запахом. Попав в помещение с угарным газом, вы только по косвенным симптомам поймете, что подвергаетесь воздействию яда. Сначала головная боль, головокружение, одышка, сердцебиение, потом посинение трупа. Угарный газ соединяется с гемоглобином крови, отчего последний перестает переносить кислород тканям вашего организма, и первыми страдают головной мозг и нервная система. А еще при определенных концентрациях он образует взрывоопасную смесь.

Поэтому датчик угарного газа – важный и необходимый компонент при построении «умного дома».

Рассмотрим подключение датчика MQ-7 к плате Arduino MEGA и модулю NodeMCU ESP8266.

4.6.1. Подключение датчика MQ-7 к плате Arduino MEGA

Подключение датчика MQ-7 к плате Arduino MEGA мы будем производить по аналоговому входу. Питание для датчика берем также с платы Arduino. Схема соединений представлена на рис. 4.30.

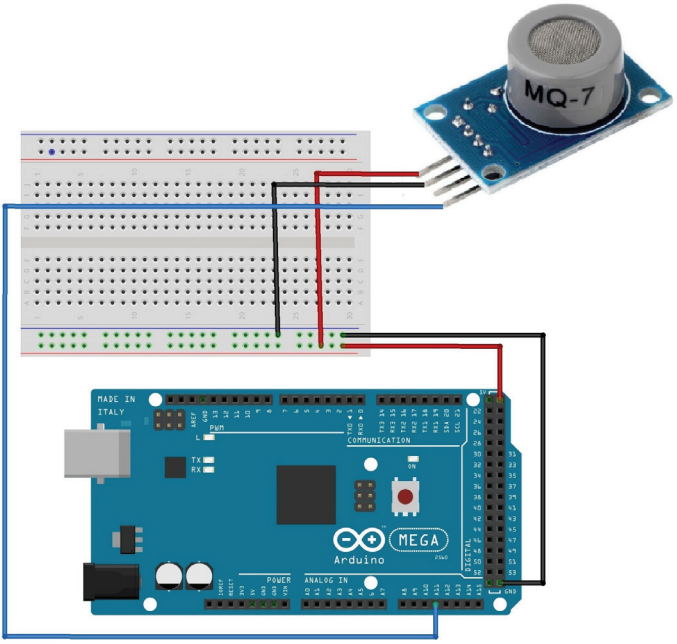


Рис. 4.30. Схема подключений датчика MQ-7 к плате Arduino MEGA

Загрузим на плату Arduino MEGA скетч получения данных с датчика MQ-7 и вывода в последовательный порт Arduino. Процедуры определения – по данным, приходящим с аналогового входа ppmcarbonmonoxide().

Содержимое скетча представлено в листинге 4.12.

Листинг 4.12

```
// библиотека для работы с датчиками MQ
#include <TroykaMQ.h>
#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
// пин, к которому подключен датчик
#define MQ7PIN         A11
// создаем объект для работы с датчиком
MQ7 mq7(MQ7PIN);

// переменная для интервала измерений
unsigned long millis_int1=0;

void setup() {
  // открываем последовательный порт
  Serial.begin(9600);
  // калибровка
  mq7.calibrate();
  mq7.getRo();
}

void loop() {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // получение данных с датчика mq7
      float  carbonmonoxide = get_data_ppmcarbonmonoxide();
      // выводим значения газа в ppm
      Serial.print("carbonmonoxide=");
      Serial.print(carbonmonoxide);
      Serial.println(" ppm ");
      // старт интервала отсчета
      millis_int1 = millis();
   }
}

// получение данных с датчика MQ7
float get_data_ppmcarbonmonoxide()  {

  Serial.println(mq7.readRatio());
  // получение значения
  float value = mq7.readCarbonMonoxide();

  return value;
}
```

Загрузим скетч на плату Arduino MEGA, откроем монитор последовательного порта и увидим вывод данных о содержании угарного газа CO.

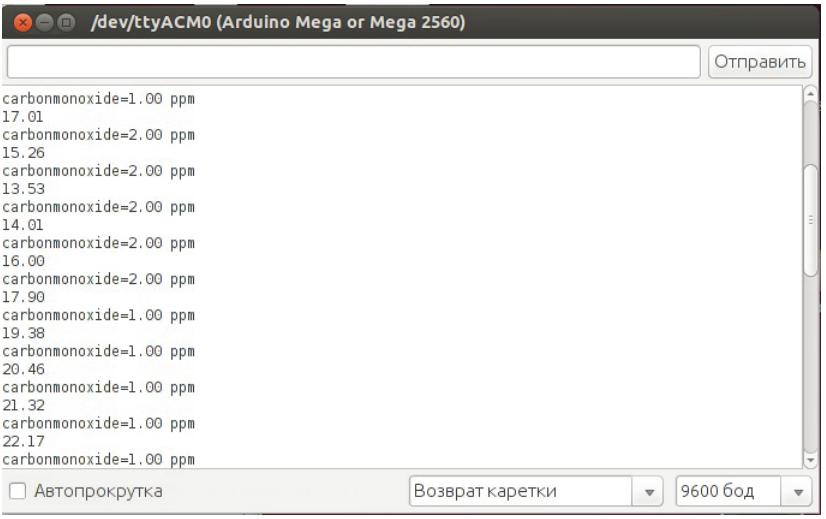


Рис. 4.31. Вывод данных датчика MQ-7 в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.6.2. Подключение датчика MQ-7 к модулю NodeMCU ESP8266

Теперь рассмотрим подключение датчика MQ-7 к модулю NodeMCU ESP8266. Датчик MQ-7 подключаем к входу y3 мультиплексора. Для выбора аналогового входа мультиплексора используем контакты D5, D7, D8 модуля NodeMCU. Схема соединений представлена на рис. 4.32.

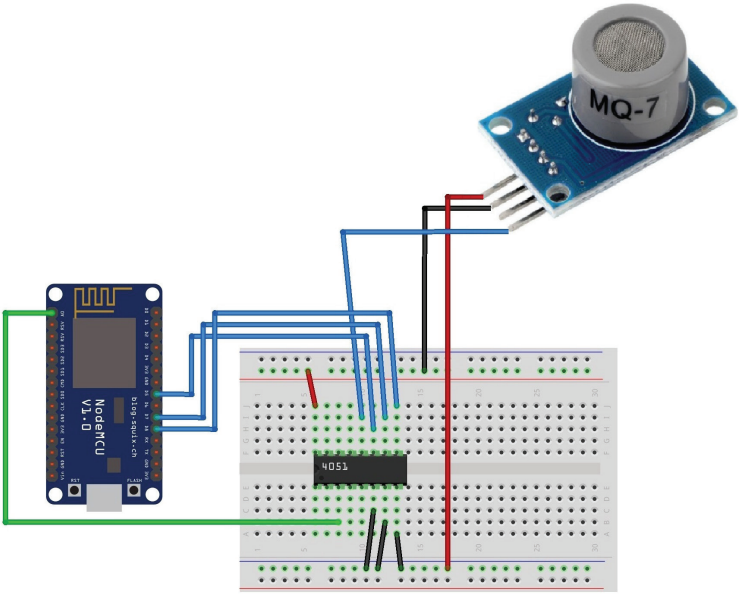


Рис. 4.32. Схема подключений датчика MQ-7 к NodeMCU ESP8266

Загрузим на модуль NodeMCU скетч получения данных с датчика MQ-7 и вывода в последовательный порт Arduino. Для выбора аналогового входа мультиплексора y3 подаем на контакты D5, D7 сигнал высокого уровня HIGH, на контакт D8 – сигнал низкого уровня LOW. Процедура определения содержания угарного газа CO в ppm – ppmcarbonmonoxide().

Содержимое скетча представлено в листинге 4.13.

Листинг 4.13

```
// библиотека для работы с датчиками MQ
#include <TroykaMQ.h>

#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
// аналоговый пин
#define MQ7PIN         A0
// создаем объект для работы с датчиком
MQ7 mq7(MQ7PIN);
```

```
// переменная для интервала измерений
unsigned long millis_int1=0;

void setup() {
  // входы подключения к мультиплексору D5, D7, D8 (GPIO 14, 13, 15)
  // как OUTPUT
  pinMode(14,OUTPUT);
  pinMode(13,OUTPUT);
  pinMode(15,OUTPUT);
  // открываем последовательный порт
  Serial.begin(9600);
  // выбор входа мультиплексора CD4051 - y3 (011)
  digitalWrite(14,HIGH);
  digitalWrite(13,HIGH);
  digitalWrite(15,LOW);
  // калибровка
  mq7.calibrate();
  mq7.getRo();
}

void loop() {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // выбор входа мультиплексора CD4051 - y3 (011)
      digitalWrite(14,HIGH);
      digitalWrite(13,HIGH);
      digitalWrite(15,LOW);
      // получение данных с датчика mq7
      float  carbonmonoxide = get_data_ppmcarbonmonoxide();
      // выводим значения газа в ppm
      Serial.print("carbonmonoxide=");
      Serial.print(carbonmonoxide);
      Serial.println(" ppm ");
      // старт интервала отсчета
      millis_int1 = millis();
   }
}

// получение данных с датчика MQ7
float get_data_ppmcarbonmonoxide()  {
  Serial.println(mq7.readRatio());
  // получение значения
  float value = mq7.readCarbonMonoxide();
  return value;
}
```

Загрузим скетч на модуль NodeMCU, откроем монитор последовательного порта и увидим вывод данных, получаемых с датчика MQ-7 (рис. 4.33).

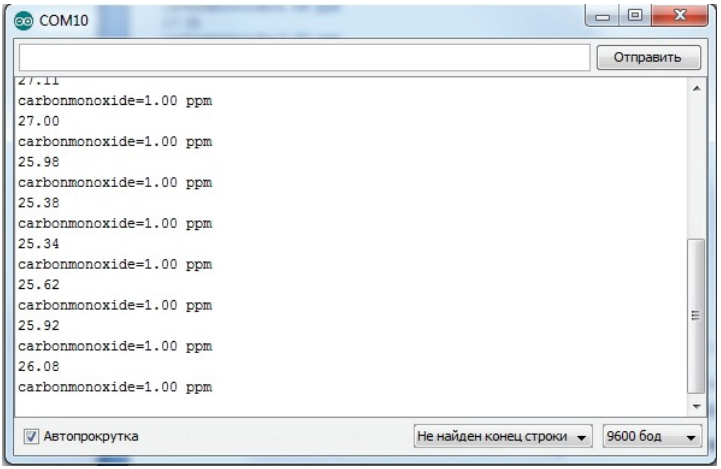


Рис. 4.33. Вывод данных датчика MQ-7 в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.7. Модуль датчика огня Flame Sensor

Модуль датчика огня Flame Sensor (рис. 4.34) позволяет фиксировать наличие пламени или другого источника огня в прямой видимости перед собой.

Датчик имеет 4 контакта (питание, земля, аналоговый вывод и цифровой вывод, срабатывание которого (выдачу сигнала HIGH) можно настроить с помощью потенциометра). Номинальное напряжение питания – 5 В. Сенсор определяет наличие огня в углу чувствительности 60°. Показания представляются в виде аналогового сигнала. Рабочая температура датчика пламени составляет от –25 до +85 °С.

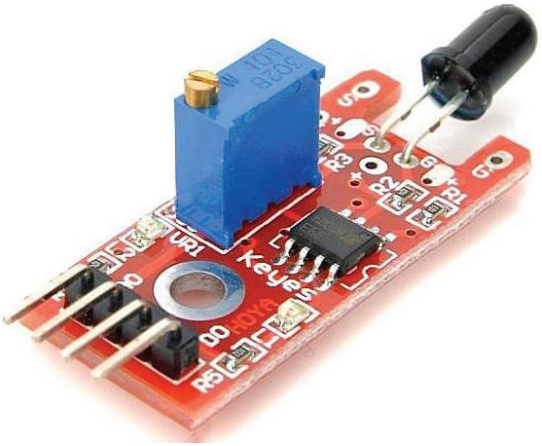


Рис. 4.34. Модуль датчика огня Flame Sensor

Рассмотрим подключение модуля Flame Sensor к плате Arduino Mega и модулю NodeMCU ESP8266.

4.7.1. Подключение модуля датчика Flame Sensor к плате Arduino MEGA

Подключение модуля датчика Flame Sensor к плате Arduino MEGA мы будем производить по аналоговому входу. Питание для датчика берем также с платы Arduino. Схема соединений представлена на рис. 4.35.

Загрузим на плату Arduino MEGA скетч получения данных с модуля датчика Flame Sensor и вывода в последовательный порт Arduino. Процедура определения – по данным, приходящим с аналогового входа get_data_flame().

Содержимое скетча представлено в листинге 4.14.

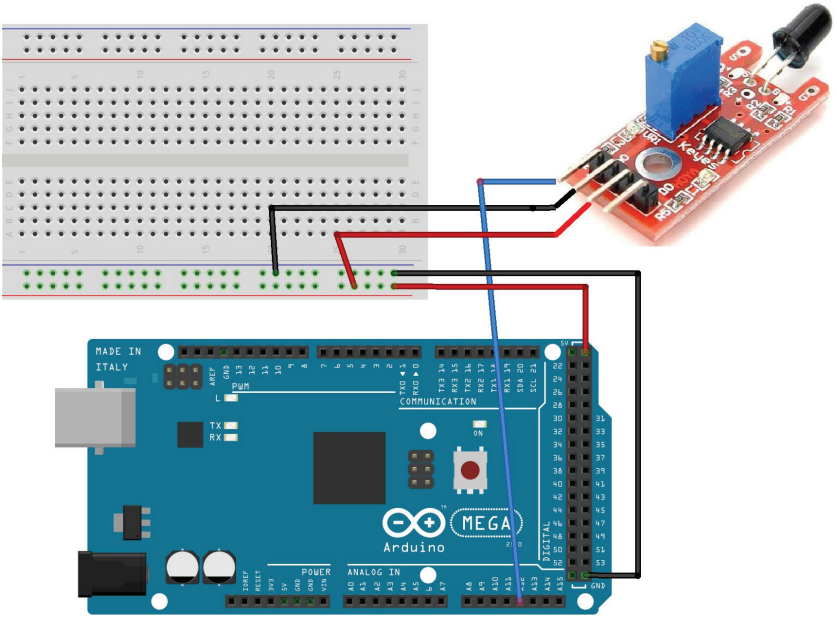


Рис. 4.35. Схема подключений модуля датчика огня Flame Sensor к плате Arduino MEGA

Листинг 4.14

```
#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
// пин, к которому подключен датчик огня flame sensor
#define FLAMEPIN         A12

// переменная для интервала измерений
unsigned long millis_int1 = 0;

void setup() {
  // открываем последовательный порт
  Serial.begin(9600);
}

void loop() {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
```

```
      // получение данных с датчика mq7
      float  flame = get_data_flame();
      // выводим значения flame sensor
      Serial.print("flame=");
      Serial.print(flame);
      // старт интервала отсчета
      millis_int1 = millis();
   }
}

// получение данных с датчика flame sensor
float get_data_flame()  {

  // получение значения
  float value = analogRead(FLAMEPIN);

  return (float)value;
}
```

Загрузим скетч на плату Arduino MEGA, откроем монитор последовательного порта и увидим вывод данных с модуля датчика огня Flame Sensor (рис. 4.36).

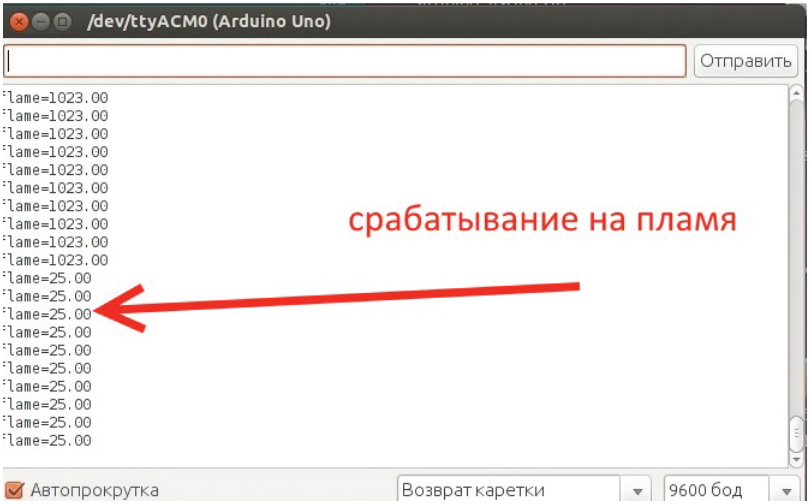


Рис. 4.36. Вывод данных модуля датчика огня Flame Sensor в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.7.2. Подключение модуля датчика Flame Sensor к модулю NodeMCU ESP8266

Теперь рассмотрим подключение модуля датчика Flame Sensor к модулю NodeMCU ESP8266. Датчик MQ-7 подключаем к входу y4 мультиплексора. Для выбора аналогового входа мультиплексора используем контакты D5, D7, D8 модуля NodeMCU. Схема соединений представлена на рис. 4.37.

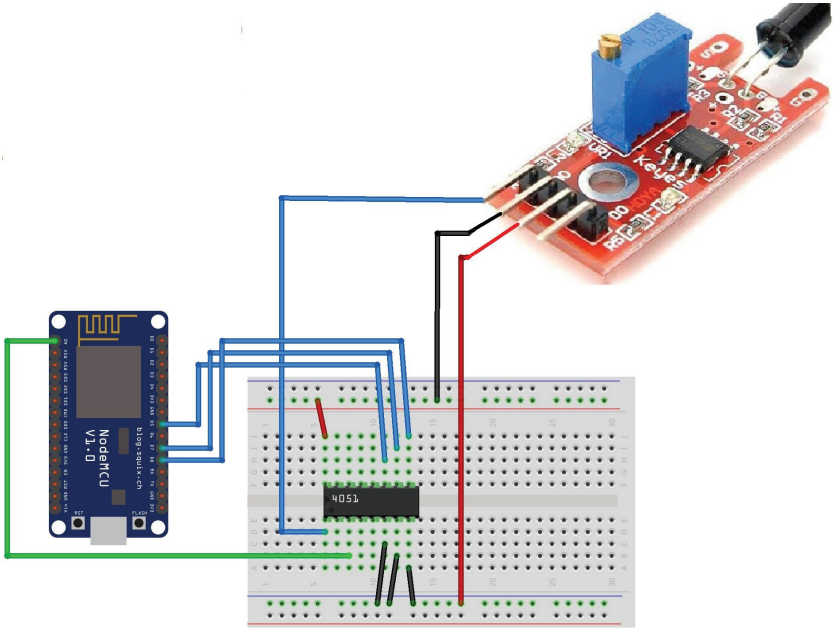


Рис. 4.37. Схема подключений модуля датчика огня Flame Sensor к NodeMCU ESP8266

Загрузим на модуль NodeMCU скетч получения данных с модуля датчика Flame Sensor и вывода в последовательный порт Arduino. Для выбора аналогового входа мультиплексора y4 подаем на контакты D5, D7 сигнал низкого уровня LOW, на контакт D8 – сигнал высокого уровня HIGH. Процедура определения – по данным, приходящим с аналогового входа get_data_flame().

Содержимое скетча представлено в листинге 4.15.

Листинг 4.15

```
#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
// аналоговый пин
#define FLAMEPIN         A0

// переменная для интервала измерений
unsigned long millis_int1=0;

void setup() {
  // входы подключения к мультиплексору D5, D7, D8 (GPIO 14, 13, 15)
  // как OUTPUT
  pinMode(14,OUTPUT);
  pinMode(13,OUTPUT);
  pinMode(15,OUTPUT);
  // открываем последовательный порт
  Serial.begin(9600);
}

void loop() {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // выбор входа мультиплексора CD4051 - y4 (100)
      digitalWrite(14,LOW);
      digitalWrite(13,LOW);
      digitalWrite(15,HIGH);
      // получение данных с датчика mq7
      float  flame = get_data_flame();
      // выводим значения flame sensor
      Serial.print("flame=");
      Serial.print(flame);
      // старт интервала отсчета
      millis_int1 = millis();
   }
}

// получение данных с датчика flame sensor
float get_data_flame()  {
  // получение значения
  float value = analogRead(FLAMEPIN);
  return (float)value;
}
```

Загрузим скетч на модуль NodeMCU, откроем монитор последовательного порта и увидим вывод данных с модуля датчика огня Flame Sensor (рис. 4.38).

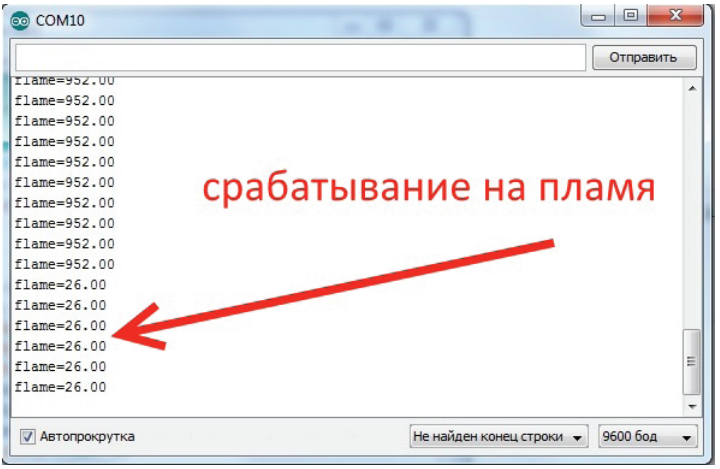


Рис. 4.38. Вывод данных с модуля датчика огня Flame Sensor в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.8. Модуль датчика присутствия HC-SR501

Рассмотрим еще один датчик, связанный с обеспечением безопасности для умного дома. Это модуль датчика присутствия HC-SR501 на основе пироэлектрического эффекта (рис. 4.39).

Рис. 4.39. Датчик присутствия HC-SR501

Состоит из самого PIR-датчика (Pyroelectric (Passive) InfraRed sensor) и схемы управления. Такие датчики часто используются в охранных системах и в быту для обнаружения движения в помещении.

Модуль имеет два переменных резистора и перемычку для настройки режима (рис. 4.40). Потенциометр Sx регулирует чувствительность прибора. Чувствительность влияет на размер детектируемого объекта и дистанцию обнаружения объекта.

Потенциометр Tx регулирует время срабатывания T. Если датчик обнаружил движение, он генерирует на выходе положительный импульс длиной T.

Перемычка переключает режим датчика. В режиме L на выходе при каждом срабатывании датчика появляется отдельный импульс. В режиме H при срабатывании датчика на выходе будет сигнал HIGH в течение некоторого периода времени T. По окончании периода сигнал на выходе вернется в исходное состояние, и датчик будет ждать следующего срабатывания.

Рассмотрим подключение модуля датчика присутствия HC-SR501 к плате Arduino MEGA и модулю NodeMCU ESP8266.

4.8.1. Подключение модуля датчика присутствия HC-SR501 к плате Arduino MEGA

Для подсчета срабатываний модуля датчика HC-SR501 будем использовать внешние прерывания на вход 18 платы Arduino MEGA. Это прерывание int5. Схема подключения модуля датчика присутствия HC-SR501 к плате Arduino MEGA показана на рис. 4.41.

Загрузим на плату Arduino MEGA скетч вывода счетчика срабатываний датчика HC-SR501 и вывода в последовательный порт Arduino. Счетчик инкрементируется при переходе уровня на входе 18 с LOW на HIGH – это процедура обработки внешнего прерывания incCounterHCSR501(). Каждые 2000 мсек (INTERVAL_GET_DATA=2000) мы выводим значение счетчика в последовательный порт.

Содержимое скетча представлено в листинге 4.16.

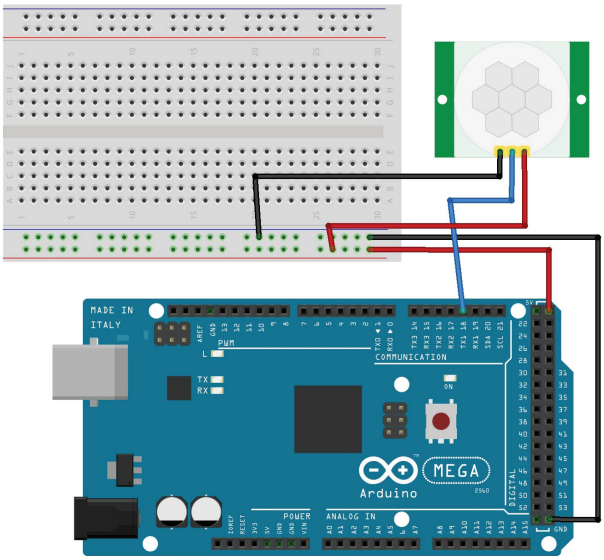


Рис. 4.41. Схема подключений датчика присутствия HC-SR501 к плате Arduino MEGA

Листинг 4.16

```
#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
// пин, к которому подключен датчик
#define HCSR501PIN         18
// счетчик срабатываний
int counterHCSR501 = 0;


// переменная для интервала измерений
unsigned long millis_int1 = 0;

void setup() {
  Serial.begin(9600);
  attachInterrupt(5,incCounterHCSR501,RISING);
}

void loop() {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
```

```
      // получение данных счетчика срабатываний HC-SR501
      int counter = get_data_counterHCSR501();
      // выводим значения газа в ppm
      Serial.print("counter=");
      Serial.println(counter);
      // старт интервала отсчета
      millis_int1 = millis();
   }
}

// получение данных счетчика срабатываний HC-SR501
int get_data_counterHCSR501()  {
  // вернуть значение счетчика срабатываний HC-SR501
  return counterHCSR501;
}

void incCounterHCSR501() {
   // инкремент счетчика
   counterHCSR501 = counterHCSR501+1;
}
```

Загрузим скетч на плату Arduino MEGA, откроем монитор последовательного порта и увидим вывод значений счетчика (рис. 4.42).

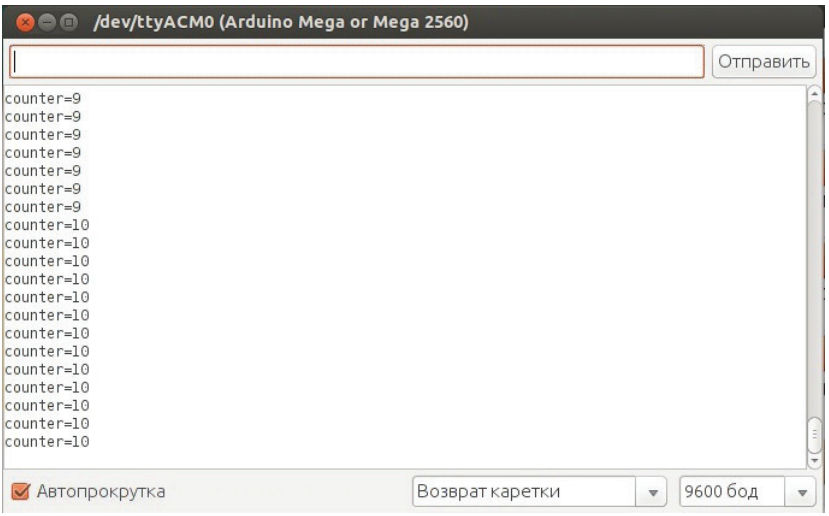


Рис. 4.42. Вывод счетчика срабатываний датчика присутствия HC-SR501 в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

4.8.2. Подключение модуля датчика присутствия HC-SR501 к модулю NodeMCU ESP8266

Теперь рассмотрим подключение модуля датчика присутствия HC-SR501 к модулю NodeMCU ESP8266. Для подсчета срабатываний модуля датчика HC-SR501 будем использовать внешние прерывания на вход 18 модуля NodeMCU ESP8266. Это прерывание int5. Схема подключения модуля датчика присутствия HC-SR501 к модулю NodeMCU показана на рис. 4.43.

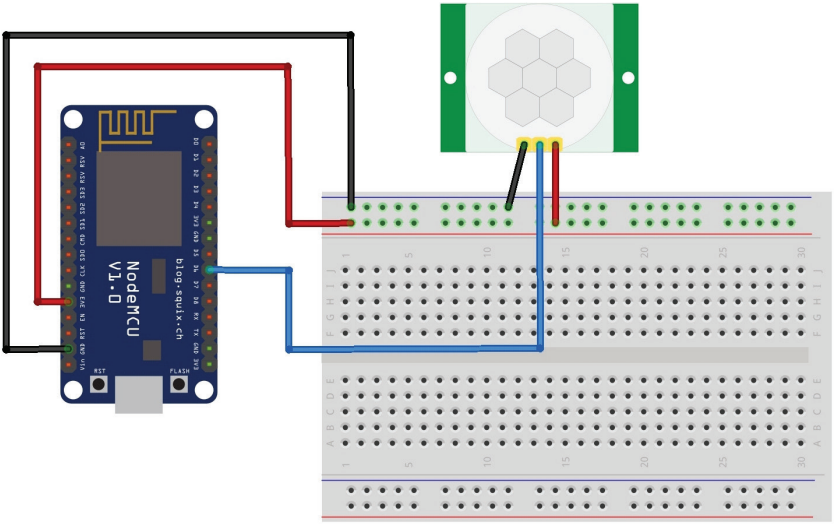


Рис. 4.43. Схема подключений датчика присутствия HC-SR501 к NodeMCU ESP8266

Загрузим на модуль NodeMCU скетч вывода счетчика срабатываний датчика HC-SR501 и вывода в последовательный порт Arduino. Счетчик инкрементируется при переходе уровня на входе D6 с LOW на HIGH – это процедура обработки внешнего прерывания incCounterHCSR501(). Каждые 2000 мсек (INTERVAL_GET_DATA = 2000) мы выводим значение счетчика в последовательный порт.

Содержимое скетча представлено в листинге 4.17.

Листинг 4.17

```
#define INTERVAL_GET_DATA 2000  // интервал измерений, мс
// пин, к которому подключен датчик
#define HCSR501PIN         D6
// счетчик срабатываний
int counterHCSR501 = 0;


// переменная для интервала измерений
unsigned long millis_int1 = 0;

void setup() {
  Serial.begin(9600);
  attachInterrupt(HCSR501PIN,incCounterHCSR501,RISING);
}

void loop() {
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      // получение данных счетчика срабатываний HC-SR501
      int counter = get_data_counterHCSR501();
      // выводим значения газа в ppm
      Serial.print("counter=");
      Serial.println(counter);
      // старт интервала отсчета
      millis_int1 = millis();
   }
}

// получение данных счетчика срабатываний HC-SR501
int get_data_counterHCSR501()  {
  // вернуть значение счетчика срабатываний HC-SR501
  return counterHCSR501;
}

void incCounterHCSR501() {
   // инкремент счетчика
   counterHCSR501 = counterHCSR501+1;
}
```

Загрузим скетч на модуль NodeMCU, откроем монитор последовательного порта и увидим вывод значений счетчика (рис. 4.44).

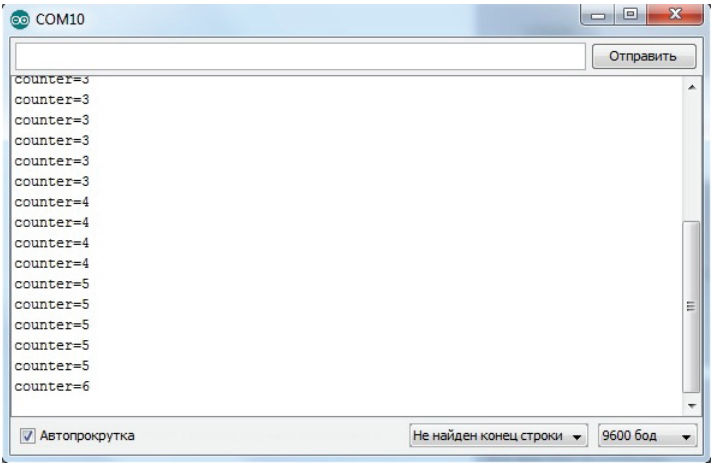


Рис. 4.44. Вывод счетчика срабатываний датчика присутствия HC-SR501 в монитор последовательного порта

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

5 ОТОБРАЖЕНИЕ ПОКАЗАНИЙ И ИНДИКАЦИЯ СОСТОЯНИЙ ДАТЧИКОВ

В предыдущей главе мы познакомились с датчиками, необходимыми для создания нашего «умного» дома, и реализовали подключение датчиков к контроллерам – плате Arduino MEGA и модулю NodeMCU ESP8266. Данные, получаемые с датчиков, мы выводили в монитор последовательного порта Arduino. Смотреть показания датчиков через последовательный порт не совсем удобно, нам необходимы более удобные устройства для отображения данных. Во-первых, мы реализуем вывод данных с датчиков на дисплей, во-вторых, мы подключим светодиоды, которые будут сигнализировать о наступлении неблагоприятных климатических условий, требующих нашего вмешательства (например, пониженная увлажненность почвы, слишком высокая температура, протекание воды), в-третьих, в дополнение к светодиодной индикации наступления неблагоприятных климатических условий добавим вывод звуковых сигналов на динамики.

5.1. Цифровой дисплей Nokia 5110

В качестве экрана для отображения показаний с датчиков мы будем использовать жидкокристаллический дисплей Nokia 5110 – монохромный дисплей с разрешением 84×48 на контроллере PCD8544, предназначен для вывода графической и текстовой информации. Питание дисплея должно лежать в пределах 2,7–3,3 В (максимум 3,3 В, при подаче 5 В на вывод VCC дисплей может выйти из строя). Но выводы контроллера толерантны к +5 В, поэтому их можно напрямую подключать к входам Arduino. Немаловажный момент – низкое потребление, что позволяет питать дисплей от платы Arduino без внешнего источника питания.

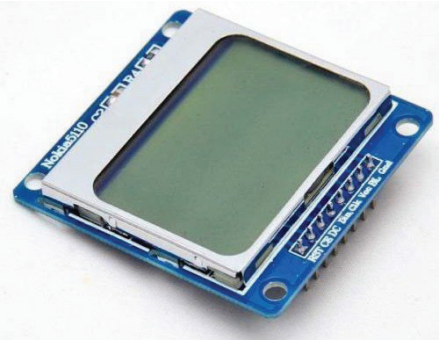


Рис. 5.1. Дисплей Nokia 5110

Основное применение дисплея – отображение простой графики и символьных данных с использованием 16 градаций яркости.

Для работы с дисплеем Nokia 5110 будем использовать библиотеку **Adafruit_GFX**, которая имеет богатые возможности для вывода графики и текста. Архив с библиотеками можно скачать по ссылке https://arduino-kit.ru/iotprog и распаковать в директорию **libraries** вашего Arduino IDE.

5.2. Вывод показаний датчиков на дисплей Nokia 5110 для Arduino MEGA

Подключим дисплей Nokia 5110 к плате Arduino MEGA. Схема соединений представлена на рис. 5.2.

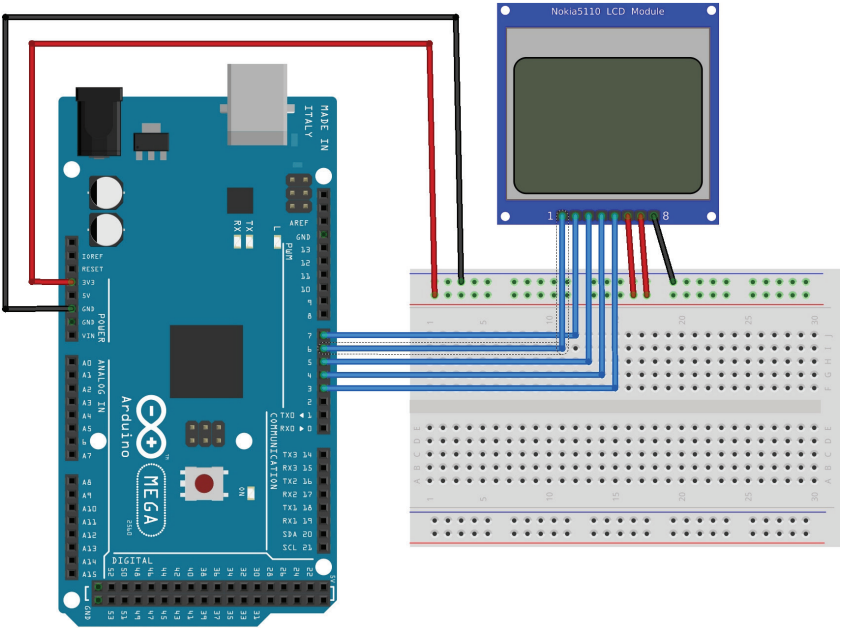


Рис. 5.2. Схема соединений для подключения Nokia 5110 к плате Arduino Mega

Теперь загрузим на плату Arduino MEGA скетч listing_05_01. Содержимое файла _05_01.ino показано в листинге 5.1. Все константы скетча вынесены в файл defines.h, процедуры получения

данных с датчиков – в файл get_data_sensors.ino, вывод данных на экран – в файл display.ino.

Листинг 5.1

```
// подключение библиотек для работы с Nokia 5110
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>// подключение библиотеки DHT
#include "DHT.h"
// подключение библиотеки OneWire
#include <OneWire.h>
// константы
//#define DHTPIN 22          // пин подключения контакта DHT22
//#define DS18B20PIN 23      // пин подключения контакта DS18B20
//#define DHTTYPE DHT11      // датчик DHT 11
//#define INTERVAL_GET_DATA 2000  // интервал измерений, мс

// создание экземпляра объекта
Adafruit_PCD8544 display = Adafruit_PCD8544(3, 4, 5, 7, 6);
// создание экземпляра объекта DHT
DHT dht(DHTPIN,DHTTYPE);
// создание экземпляра объекта OneWire
OneWire ds(DS18B20PIN);

// переменная для интервала измерений
unsigned long millis_int1 = 0;
int teksensor = 0;
// массив для хранения данных датчиков
float datasensors[] = {999,999,999,999,999,999,999,999,999,999};
int aktivesensors[] = {1,1,1,1,0,0,0,0,0,0};
int ysensors[] = {100,100,100,100,100,100,100,100,100,100};
int xsensors[] = {60,80,100,120,140,160,180,200,220,240};
int posdata = 0;


void setup(void) {
  Serial.begin(9600);
  ini_display();
  // вывод главного экрана
  //view_display_clock();
  // вывод экрана сенсоров
  view_display_sensors();
}

void loop(void) {
   // показания с датчиков
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      teksensor = (teksensor+1)%COUNT_SENSORS;
      if(aktivesensors[teksensor] == 1 )  {
         float fvar = 0;
```

```
switch(teksensor) {
   case 0:  // DHT22
       fvar = get_data_humidity();
       Serial.print("humidity=");Serial.println(fvar);
     break;
   case 1:  // DS18B20
       fvar = get_data_ds18b20();
       Serial.print("tds18b20=");Serial.println(fvar);
     break;
   case 2:  // Soil Moisture
       fvar = get_data_soilmoisture();
       Serial.print("soilmoisture=");Serial.println(fvar);
     break;
   case 3:  // level water
       fvar = get_data_levelwater();
       Serial.print("levelwater=");Serial.println(fvar);
    break;
   case 5:  // mq-2 propan
       fvar = get_data_ppmpropan();
       Serial.print("ppmpropan=");Serial.println(fvar);
     break;
   case 6:  // mq-2 methan
       fvar = get_data_ppmmethan();
       Serial.print("ppmmethan=");Serial.println(fvar);
       control_data_limit(5);
     break;
   case 7:  // mq-2 smoke
       fvar = get_data_ppmsmoke();
       Serial.print("ppmsmoke=");Serial.println(fvar);
       control_data_limit(5);
     break;
   case 8:  // mq-7 CO
       fvar = get_data_ppmcarbonmonoxide();
       Serial.print("ppmcarbonmonoxide=");
       Serial.println(fvar);
       control_data_limit(6);
     break;
   default:  // другие датчики
     break;
}
if(fvar<999) {
   set_display_data_sensor(teksensor,
        datasensors[teksensor],BLACK);
   datasensors[teksensor] = fvar;
   set_display_data_sensor(teksensor,
       datasensors[teksensor],YELLOW);
}
```

```
      }
      millis_int1 = millis();
   }
}
// Процедуры вывода данных
// на дисплей
// Nokia 5110

// ************************************************
// инициализация дисплея
void ini_display() {
  display.begin();
  display.setContrast(50);
  display.clearDisplay();   // очистить экран
  display.display();
  display.setTextSize(1);

  Serial.println(F("start!"));
}
// ************************************************
// вывод экрана с показаниями датчиков
void view_display_sensors() {
  // очистить экран
  display.clearDisplay();
  // вывод заголовка экрана
  // вывод списка датчиков
  for(int i=0;i<COUNT_SENSORS;i++) {
     display.setCursor(ysensors[i], xsensors[i]);
     display.print(strS1[i]);display.print("=");
  }
  // вывод списка значений датчиков

  display.display();
}
// ************************************************
// вывести текущее значение для датчика
void set_display_data_sensor(int pos,float var,unsigned int color) {
   display.setTextColor(color);
   display.setCursor(ysensors[pos]+15, xsensors[pos]);
   display.print(var);
   display.display();
}
```

Загружаем скетч на плату Arduino MEGA и получаем вывод данных с датчиков на экран Nokia 5110.

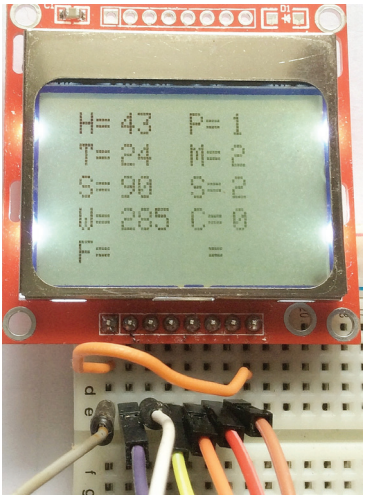


Рис. 5.3. Вывод показаний сенсоров на дисплей

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

5.3. Светодиодная индикация и звуковая сигнализация о критических параметрах датчиков для Arduino MEGA

Введем светодиодную индикацию и звуковую сигнализацию, чтобы информировать вас о наступлении неблагоприятных климатических условий или условий, представляющих опасность для дома (пожар, утечка газов). Для светодиодной индикации будем использовать обычные светодиоды, которые подсоединим к цифровым выводам (26–32) Arduino MEGA (используем ограничительные резисторы номиналом 220 Ом). Для звуковой индикации будем использовать небольшой динамик (рис. 5.4). При подключении динамика к выводу Arduino MEGA используем npn-транзистор и резистор номиналом 510 Ом. Схема соединений для подключения светодиодов и динамика к плате Arduino MEGA показана на рис. 5.5.

Рис. 5.4. Динамик для звуковой сигнализации

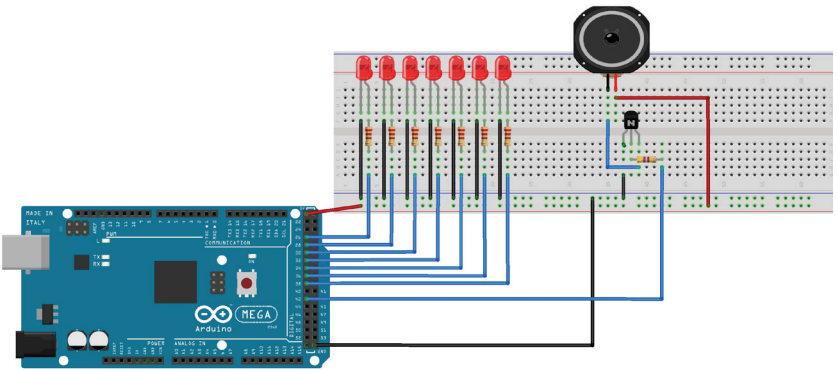


Рис. 5.5. Схема подключения светодиодов и динамика к плате Arduino MEGA

Внесем дополнительный функционал в наш скетч. Создадим константы граничных значений датчиков. В файл defines.h добавим:

```
// параметры датчиков для срабатывания
// светодиодной индикации и звуковой сигнализации
#define LIMIT_DHT22MIN 18           // влажность воздуха низкая
#define LIMIT_DHT22MAX 90           // влажность воздуха высокая
#define LIMIT_DS18B20MIN 15         // температура низкая
#define LIMIT_DS18B20MAX 30         // температура высокая
#define LIMIT_SOILMOISTUREMIN 300    // влажность почвы - чрезмерная влажность
#define LIMIT_SOILMOISTUREMAX 600    // влажность почвы - сухость
#define LIMIT_LEVELWATER 100        // вода
#define LIMIT_FAIR 100              // огонь
#define LIMIT_MQ2P 30               // пропан
#define LIMIT_MQ2M 30               // метан
#define LIMIT_MQ2S 30               // дым
#define LIMIT_MQ7 30                // CO
```

А также пин подключения динамика:

```
// пин, к которому подключен динамик
#define SOUNDPIN         42
```

Далее в файле 05_03.ino создадим массивы пинов для подключения светодиодов, а также частот для звуковых оповещений:

```
// массив для хранения данных пинов светодиодов
int pinleds[] = {26,28,30,32,34,36,38};
// массив для хранения частот и длительностей
// для звуковых оповещений
int freq[] = {293,329,349,370,392,440,494};
int durations[] = {3000,3000,3000,4000,4000,4000,4000};
```

В цикле loop() после получения данных с датчика делаем обращение к процедуре проверки данных датчиков на критические значения:

```
// проверка данных на критичность
void control_data_limit(int par) {
   float val;

   switch(par) {
      case 0:  // DHT22
           val = datasensors[0];
           if((val<LIMIT_DHT22MIN || val>LIMIT_DHT22MAX)
               && aktivesensors[0]==1) {
             set_status_leds(0,HIGH);
             set_status_sound(0);
           }
           else
              set_status_leds(0,LOW);
         break;
      case 1:  // DS18B20
           val = datasensors[1];
           if((val<LIMIT_DS18B20MIN || val>LIMIT_DS18B20MAX)
               && aktivesensors[1]==1) {
             set_status_leds(1,HIGH);
             set_status_sound(1);
           }
           else
              set_status_leds(1,LOW);
         break;
      case 2:  // SoilMoisture
           val = datasensors[2];
           if((val<LIMIT_SOILMOISTUREMIN
               || val>LIMIT_SOILMOISTUREMAX)
                && aktivesensors[2]==1) {
             set_status_leds(2,HIGH);
```

```
            set_status_sound(2);
          }
          else
             set_status_leds(2,LOW);
        break;
      case 3:  // waterlevel
          val = datasensors[3];
          if(val>LIMIT_LEVELWATER && aktivesensors[3]==1) {
            set_status_leds(3,HIGH);
            set_status_sound(3);
          }
          else
             set_status_leds(3,LOW);
        break;
      case 4:  // fire
          val = datasensors[4];
          if(val>LIMIT_FAIR && aktivesensors[4]==1) {
            set_status_leds(4,HIGH);
            set_status_sound(4);
          }
          else
             set_status_leds(4,LOW);
        break;
      case 5:  // mq2
          val = datasensors[5];
          if((datasensors[5]>LIMIT_MQ2P
              || datasensors[6]>LIMIT_MQ2M
              || datasensors[7]>LIMIT_MQ2S)
              && aktivesensors[5]==1) {
            set_status_leds(5,HIGH);
            set_status_sound(5);
          }
          else
             set_status_leds(5,LOW);
        break;
      case 6:  // mq7
          val = datasensors[8];
          if(val>LIMIT_MQ7 && aktivesensors[6]==1) {
            set_status_leds(6,HIGH);
            set_status_sound(6);
          }
          else
             set_status_leds(6,LOW);
        break;
      default:
        break;
   }
}
```

Функции set_status_leds() и set_status_sound() отвечают за свечение светодиода и вывод на динамик звука с помощью встроенной функции tone().

```
// индикация на светодиоды
void set_status_leds(int par,int val) {
     digitalWrite(pinleds[par],val);
}

// звуковая сигнализация
void set_status_sound(int par) {
   tone(SOUNDPIN,freq[par],durations[par]);
}
```

Для каждого датчика в случае критических значений выводится звук частотой и длительностью, определенной в массивах:

```
int freq[] = {293,329,349,370,392,440,494};
int durations[] = {3000,3000,3000,4000,4000,4000,4000};
```

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

5.4. Увеличение цифровых контактов для NodeMCU для ESP8266. Микросхема MCP23017

Введем светодиодную индикацию и звуковую сигнализацию и при использовании в качестве контроллера умного дома модуля NodeMCU. Количество выводов у модуля NodeMCU гораздо меньше, чем у Arduino MEGA, поэтому нам понадобится микросхема расширителя входов MCP23017. Микросхема MCP23017 добавляет 16 портов, которые можно настроить как на вход, так и на выход (рис. 5.6). Микросхема использует популярную двухпроводную шину I2C.

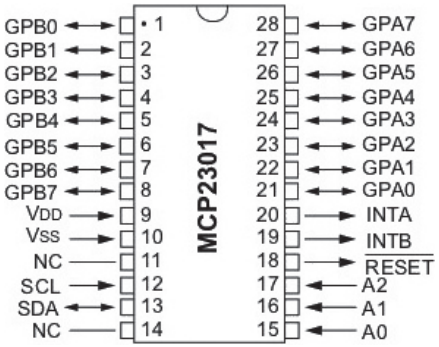


Рис. 5.6. Выводы микросхемы MCP23017

Адрес микросхемы MCP23017 для протокола I2C можно установить комбинацией сигналов на цифровых входах A0–A2 (рис. 5.7), что позволяет к микроконтроллеру подключить одновременно 8 микросхем MCP23017, соответственно 16 × 8 = 128 контактов.

i2c Address								
0	1	0	0	ADDR2	ADDR1	ADDR0	Hex Addr.	Dec. Addr.
0	1	0	0	**0**	**0**	**0**	0x20	32
0	1	0	0	**0**	**0**	1	0x21	33
0	1	0	0	**0**	1	**0**	0x22	34
0	1	0	0	**0**	1	1	0x23	35
0	1	0	0	1	**0**	**0**	0x24	36
0	1	0	0	1	**0**	1	0x25	37
0	1	0	0	1	1	**0**	0x26	38
0	1	0	0	1	1	1	0x27	39

Рис. 5.7. Установка адреса микросхемы MCP23017

Микросхема имеет 2 банка портов: A (GPA0–GPA7) и B (GPB0–GPAB), каждый из которых можно настроить на ввод или вывод.

В листинге 5.3 показан пример настройки банков выводов A и B.

Листинг 5.3

```
// подключение библиотеки Wire.h
#include <Wire.h>
byte input = 0;
void setup()
{
  Serial.begin(9600);
  Wire.begin(0,2);     // запуск I2C
  Wire.beginTransmission(0x20);  // i2c - адрес (A0-0,A1-0,A2-0)
  Wire.write(0x00); // IODIRA register
  Wire.write(0x00); // настроить PORT A как output
  Wire.endTransmission();
}

void loop()
{
  // чтение данных из PORT B
  Wire.beginTransmission(0x20);
  Wire.write(0x13);
  Wire.endTransmission();
  Wire.requestFrom(0x20, 1);
  input=Wire.read();

  // записать полученные данные в PORT A
```

```
  Wire.beginTransmission(0x20);
  Wire.write(0x12);     // address PORT A
  Wire.write(input);    // PORT A
  Wire.endTransmission();
  delay(100);          // пауза
}
```

Использование микросхемы MCP23017 позволит расширить количество цифровых контактов модуля NodeMCU на 16 и организовать светодиодную индикацию и звуковую сигнализацию о критических параметрах датчиков.

5.5. Светодиодная индикация и звуковая сигнализация о критических параметрах датчиков для NodeMCU

Для светодиодной индикации будем использовать обычные светодиоды, которые подсоединим к микросхеме расширителя входов MCP23017 (банку A, выводы GPA0–GPA7). При подключении светодиодов используем ограничительные резисторы номиналом 150 Ом. Для звуковой индикации будем использовать небольшой динамик (рис. 5.4). При подключении динамика к выводу NodeMcu используем npn-транзистор и резистор номиналом 510 Ом. Схема соединений для подключения светодиодов и динамика к модулю NodeMCU показана на рис. 5.8.

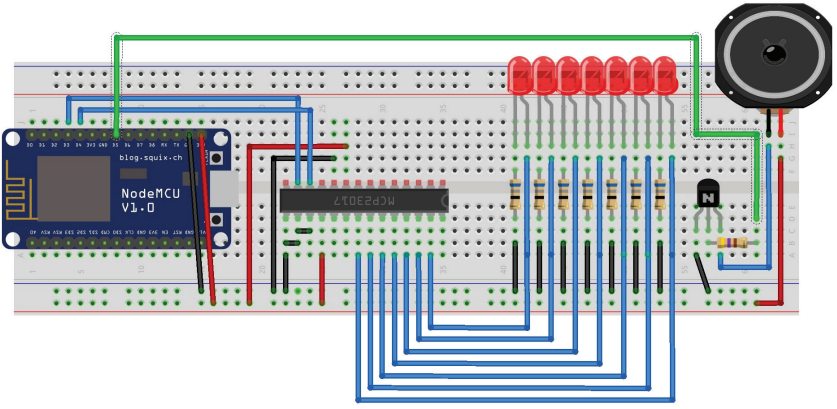


Рис. 5.8. Схема подключения светодиодов и динамика к модулю NodeMCU

Внесем дополнительный функционал в наш скетч. Создадим константы граничных значений датчиков. В файл defines.h добавим:

```
// параметры датчиков для срабатывания
// светодиодной индикации и звуковой сигнализации
#define LIMIT_DHT22MIN 18           // влажность воздуха низкая
#define LIMIT_DHT22MAX 90           // влажность воздуха высокая
#define LIMIT_DS18B20MIN 15         // температура низкая
#define LIMIT_DS18B20MAX 30         // температура высокая
#define LIMIT_SOILMOISTUREMIN 300   // влажность почвы -
                                    // чрезмерная влажность
#define LIMIT_SOILMOISTUREMAX 600   // влажность почвы -
                                    // сухость
#define LIMIT_LEVELWATER 100        // вода
#define LIMIT_FAIR 100              // огонь
#define LIMIT_MQ2P 30               // пропан
#define LIMIT_MQ2M 30               // метан
#define LIMIT_MQ2S 30               // дым
#define LIMIT_MQ7 30                // CO
```

А также пин подключения динамика:

```
// пин, к которому подключен динамик
#define SOUNDPIN         14
```

Далее в файле 05_04.ino создадим массивы состояний светодиодов, а также частот для звуковых оповещений:

```
// массив для хранения данных состояний светодиодов
int statusleds[]={0,0,0,0,0,0,0};
// массив для хранения частот и длительностей
// для звуковых оповещений
int freq[]={293,329,349,370,392,440,494};
int durations[]={3000,3000,3000,4000,4000,4000,4000};
```

Поскольку светодиоды подключены через I2C расширителя выводов MCP23017, в setup производим настройку портов микросхемы MCP23017 (процедура ini_mcp23017().

```
void ini_mcp23017() {
  Wire.beginTransmission(0x20);  // i2c - адрес (A0-0,A1-0,A2-0)
  Wire.write(0x00); // IODIRA register
  Wire.write(0x00); // настроить PORT A как output
  Wire.endTransmission();
  Wire.beginTransmission(0x20);
  Wire.write(0x01); // IODIRB register
  Wire.write(0x00); // настроить PORT B как output
  Wire.endTransmission();
}
```

Далее в цикле loop() после получения данных с датчика делаем обращение к процедуре проверки данных датчиков на критические значения:

```
// проверка данных на критичность
void control_data_limit(int par) {
   float val;

   switch(par) {
      case 0:  // DHT22
           val=datasensors[0];
           if((val<LIMIT_DHT22MIN || val>LIMIT_DHT22MAX)
               && aktivesensors[0]==1) {
             set_status_sound(0);
             statusleds[0]=1;
           }
           else
             statusleds[0]=0;
           set_status_leds();
         break;
      case 1:  // DS18B20
           val=datasensors[1];
           if((val<LIMIT_DS18B20MIN || val>LIMIT_DS18B20MAX)
               && aktivesensors[1]==1) {
             set_status_sound(1);
             statusleds[1]=1;
           }
           else
             statusleds[1]=0;
           set_status_leds();
         break;
      case 2:  // SoilMoisture
           val=datasensors[2];
           if((val<LIMIT_SOILMOISTUREMIN
               || val>LIMIT_SOILMOISTUREMAX)
                && aktivesensors[2]==1) {
             set_status_sound(2);
             statusleds[2]=1;
           }
           else
             statusleds[2]=0;
           set_status_leds();
         break;
      case 3:  // waterlevel
           val=datasensors[3];
           if(val>LIMIT_LEVELWATER && aktivesensors[3]==1) {
             set_status_sound(3);
             statusleds[3]=1;
           }
           else
             statusleds[3]=0;
           set_status_leds();
```

```
        break;
      case 4:  // fire
          val=datasensors[4];
          if(val>LIMIT_FAIR && aktivesensors[4]==1) {
            set_status_sound(4);
            statusleds[4]=1;
          }
          else
            statusleds[4]=0;
          set_status_leds();
        break;
      case 5:  // mq2
          val=datasensors[5];
          if((datasensors[5]>LIMIT_MQ2P
              || datasensors[6]>LIMIT_MQ2M
              || datasensors[7]>LIMIT_MQ2S)
              && aktivesensors[5]==1) {
            set_status_sound(5);
            statusleds[5]=1;
          }
          else
            statusleds[5]=0;
          set_status_leds();
        break;
      case 6:  // mq7
          val=datasensors[8];
          if(val>LIMIT_MQ7 && aktivesensors[6]==1) {
            set_status_sound(6);
            statusleds[6]=1;
          }
          else
            statusleds[6]=0;
          set_status_leds();
        break;
      default:
        break;
   }
}
```

Функции set_status_leds() и set_status_sound() отвечают за свечение светодиода и вывод на динамик звука с помощью встроенной функции tone().

```
// индикация на светодиоды
void set_status_leds() {
  int leds=0;
  for(int i=0;i<7;i++)
     leds=leds+( statusleds[5]<<i);
  // записать данные в PORT A
  Wire.beginTransmission(0x20);
  Wire.write(0x12); // address PORT A
  Wire.write(leds);    // PORT A
```

```
  Wire.endTransmission();
  delay(100); // пауза
}
```

Встроенной функции tone() для ESP8266 нет, поэтому вывод звука происходит иначе.

```
// звуковая сигнализация
void set_status_sound(int par) {
  analogWriteFreq(freq[par]);
  analogWrite(SOUNDPIN,512);
  delay(durations[par]);
  analogWrite(SOUNDPIN,0);
}
```

Для каждого датчика в случае критических значений выводится звук частотой и длительностью, определенной в массивах:

```
int freq[]={293,329,349,370,392,440,494};
int durations[]={3000,3000,3000,4000,4000,4000,4000};
```

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

5.6. TFT 2.4" Shield 240x320

Для вывода показаний на экран в наборе используется дисплей Nokia 5110, которого, в принципе, хватает для простого вывода параметров, но прогресс не стоит на месте, и появляются новые, более удобные дисплеи, более того, еще и с сенсорной панелью. Возможно, вы захотите выводить информацию на один из таких дисплеев. Мы рассмотрим в качестве примера TFT Shield (см. рис. 5.9) с цветным ЖКИ QVGA, имеющим диагональ 2,4 дюйма и разрешение экрана 240×320. ЖКИ-индикатор имеет постоянно включенную подсветку. Есть одна кнопка. На плате расположен контейнер для SD-карты. На шилде встроен резистивный тачскрин, что позволяет определять положение при нажатии на экран. На плате есть кнопка. Нажатие кнопки вызывает сброс Arduino.

Рис. 5.9. TFT Shield 2.4" 240×320

Основное применение дисплея – отображение простой графики и символьных данных с использованием 16 цветов.

TFT Shield может устанавливаться в соединители Arduino MEGA (см. рис. 5.10).

Рис. 5.10. Установка TFT Shield 2.4" 240×320 на плату Arduino MEGA

Дисплеи такого типа построены на различных чипах, для каждого из которых нужна своя библиотека. Для нашего дисплея подошли библиотеки **Adafruit_GFX** и **SWTFT**. Для работы с тачскрином будем использовать библиотеку **Adafruit Touch-Screen-Library**. Архив с библиотеками можно скачать по ссылке https://arduino-kit.ru/iotprog и распаковать в директорию **libraries** вашего Arduino IDE.

5.7. Вывод показаний датчиков на TFT 2.4" Shield 240×320 для Arduino MEGA

Подключим TFT Shield к Arduino MEGA (см. рис. 5.10). Для использования библиотеки SWTFT с платой Arduino MEGA необходимо внести изменения в файл SWTFT.cpp (см. рис. 5.11).

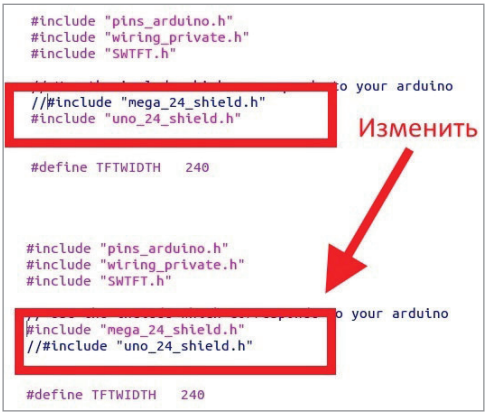


Рис. 5.11. Изменения в файле SWTFT.cpp

Теперь загрузим на плату Arduino MEGA скетч listing_05_05. Содержимое файла _05_05.ino показано в листинге 5.5. Все константы скетча вынесены в файл defines.h, процедуры получения данных с датчиков – в файл get_data_sensors.ino, вывод данных на экран – в файл display.ino.

Листинг 5.5

```
// подключение библиотеки DHT
#include "DHT.h"
// подключение библиотеки OneWire
#include <OneWire.h>
// константы
//#define DHTPIN 22          // пин подключения контакта DHT22
//#define DS18B20PIN 23      // пин подключения контакта DS18B20
//#define DHTTYPE DHT11      // датчик DHT 11
//#define INTERVAL_GET_DATA 2000  // интервал измерений, мс

// создание экземпляра объекта SWTFT
SWTFT tft;
// создание экземпляра объекта DHT
DHT dht(DHTPIN,DHTTYPE);
```

```
// создание экземпляра объекта OneWire
OneWire ds(DS18B20PIN);

// переменная для интервала измерений
unsigned long millis_int1 = 0;
int teksensor = 0;
// массив для хранения данных датчиков
float datasensors[] = {999,999,999,999,999,999,999,999,999,999};
int aktivesensors[] = {1,1,1,1,0,0,0,0,0,0};
int ysensors[] = {100,100,100,100,100,100,100,100,100,100};
int xsensors[] = {60,80,100,120,140,160,180,200,220,240};
int posdata = 0;


void setup(void) {
  Serial.begin(9600);
  Serial.println(F("TFT LCD test"));
  ini_display();
  // вывод главного экрана
  //view_display_clock();
  // вывод экрана сенсоров
  view_display_sensors();
}

void loop(void) {
   // показания с датчиков
   if(millis()-millis_int1 >= INTERVAL_GET_DATA) {
      teksensor = (teksensor+1)%COUNT_SENSORS;
      if(aktivesensors[teksensor] == 1 )  {
         float fvar = 0;
         switch(teksensor) {
            case 0:  // DHT22
                fvar = get_data_humidity();
                Serial.print("humidity=");Serial.println(fvar);
              break;
            case 1:  // DS18B20
                fvar = get_data_ds18b20();
                Serial.print("tds18b20=");Serial.println(fvar);
              break;
            case 2:  // Soil Moisture
                fvar = get_data_soilmoisture();
                Serial.print("soilmoisture=");Serial.println(fvar);
              break;
            case 3:  // level water
                fvar = get_data_levelwater();
                Serial.print("levelwater=");Serial.println(fvar);
             break;
            case 5:  // mq-2 propan
                fvar = get_data_ppmpropan();
                Serial.print("ppmpropan=");Serial.println(fvar);
              break;
            case 6:  // mq-2 methan
                fvar = get_data_ppmmethan();
                Serial.print("ppmmethan=");Serial.println(fvar);
```

```
                control_data_limit(5);
              break;
            case 7:  // mq-2 smoke
                fvar = get_data_ppmsmoke();
                Serial.print("ppmsmoke=");Serial.println(fvar);
                control_data_limit(5);
              break;
            case 8:  // mq-7 CO
                fvar = get_data_ppmcarbonmonoxide();
                Serial.print("ppmcarbonmonoxide=");
                Serial.println(fvar);
                control_data_limit(6);
              break;
            default:  // другие датчики
              break;
         }
         if(fvar<999) {
            set_display_data_sensor(teksensor,
                 datasensors[teksensor],BLACK);
            datasensors[teksensor] = fvar;
            set_display_data_sensor(teksensor,
                datasensors[teksensor],YELLOW);
         }
      }
      millis_int1 = millis();
   }
}
// Процедуры вывода данных
// на дисплей
// shield TFT 2.4 240x320

// ************************************************
// инициализация дисплея
void ini_display() {
  tft.reset();
  uint16_t identifier = tft.readID();
  Serial.print(F("LCD driver chip: "));
  Serial.println(identifier, HEX);
  tft.begin(identifier);
  tft.fillScreen(BLACK);
  tft.setRotation(0);
  Serial.println(F("start!"));
}
// ************************************************
// вывод экрана с показаниями датчиков
void view_display_sensors() {
  // очистить экран
  tft.fillScreen(BLACK);
  // вывод заголовка экрана
  tft.setCursor(10, 15);
  tft.setTextColor(MAGENTA);
  tft.setTextSize(3);
  tft.println("Sensors:");
  // вывод списка датчиков
```

```
  tft.setTextColor(YELLOW);
  tft.setTextSize(2);
  for(int i = 0; i < COUNT_SENSORS; i++ ) {
     tft.setCursor(10, xsensors[i]);
     tft.print(strS1[i]);
  }
  // вывод списка значений датчиков

  // вывод единиц измерения
  tft.setTextColor(YELLOW);
  tft.setTextSize(2);
  for(int i = 0; i < COUNT_SENSORS; i++) {
     tft.setCursor(200, xsensors[i]);
     tft.print(strS2[i]);
  }
  // вывод для возврата в главное меню
}
// ************************************************
// вывести текущее значение для датчика
void set_display_data_sensor(int pos,float var,unsigned int color) {
   tft.setTextColor(color);
   tft.setCursor(100, xsensors[pos]);
   tft.print(var);
}
```

Загружаем скетч на плату Arduino MEGA и получаем вывод данных с датчиков на экран TFT 2.4" Shield.

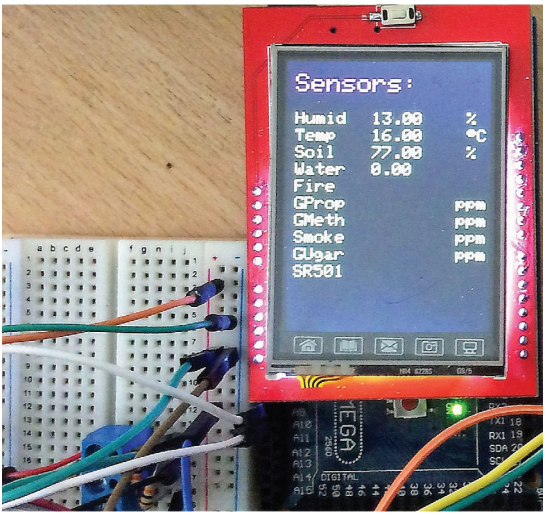


Рис. 5.12. Вывод показаний сенсоров на дисплей

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

6 УПРАВЛЕНИЕ ИСПОЛНИТЕЛЬНЫМИ УСТРОЙСТВАМИ

В предыдущих главах мы рассмотрели подключение датчиков – «органов чувств» нашего умного дома, которые позволяют контроллеру получать требуемые данные для создания необходимых условий комфорта и безопасности. Теперь мы рассмотрим следующую задачу – управление исполнительными устройствами.

Исполнительные устройства – это элементы автоматики, создающие управляющее воздействие на объект управления. Они изменяют положение или состояние регулирующего органа объекта управления таким образом, чтобы управляемый параметр соответствовал заданному значению. Исполнительное устройство или механизм (actuator) преобразует электрическую энергию в механическую или в физическую величину для воздействия на управляемый процесс. Это могут быть световые и звуковые устройства, электромагнитные клапаны, DC- и AC-моторы, сервоприводы, релейные системы и многое другое.

В нашем умном доме нам потребуются исполнительные устройства для управления совещением умного дома, вентилятором для создания прохлады, увлажнителем для управления влажностью воздуха, помпой для полива растений, возможно, для автоматического открытия/закрытия входных и гаражных дверей.

Рассмотрим организацию управления исполнительными устройствами нашего умного дома с контроллеров Arduino MEGA и модуля NodeMCU.

6.1. Подключение блока реле для управления исполнительными устройствами

Для управления электроприборами пользуются различными клавишными выключателями и тумблерами. Чтобы управлять электроприборами с помощью микроконтроллера, существует специальный тип выключателей – электромеханические реле. В набор «Интернет вещей для умного дома» включен Relay Shield (см. рис. 6.1), который содержит четыре реле с необходимой обвязкой и позволяет контроллерам управлять четырьмя электроприборами.

По своей сути реле – это просто механический рубильник, которым можно управлять при помощи микроконтроллера. При этом электрическая связь между управляющей электроникой и коммутируемой нагрузкой отсутствует. А это значит, что помехи не попадают на микроконтроллер и не сбивают управляющую программу.

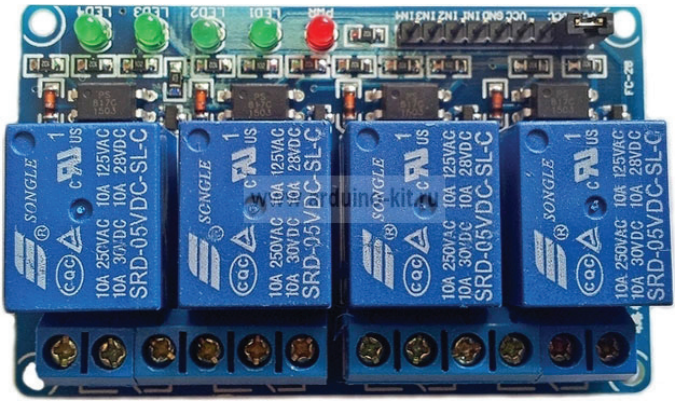


Рис. 6.1. Relay Shield

В модулях relay shield обычно используется n-канальное управление. При таком управлении реле включается подачей на вывод Arduino низкого уровня LOW, а выключается подачей высокого уровня HIGH.

6.2. Подключение блока реле к плате Arduino MEGA

Рассмотрим подключение Relay Shield к плате Arduino MEGA. Relay Shield мы будем использовать для включения/выключения света для освещения растений, вентилятора, насоса для полива растений. Включение/выключение вентилятора и помпы будет осуществляться в зависимости от значений температуры воздуха (вентилятор) и влажности почвы (мембранный вакуумный насос (см. рис. 6.2)). Мембранный вакуумный насос будем использовать для полива почвы. Он предназначен для всасывания воды из емкости. Рабочее напряжение 12 В, потребляемый рабочий ток 0,5–0,7 А, расход 1,5 л/мин.

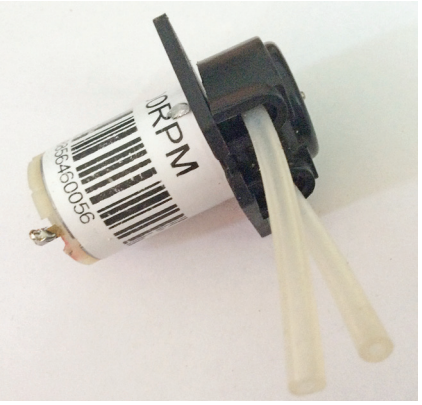


Рис. 6.2. Мембранный вакуумный насос

Управление светом будем осуществлять с сенсонного экрана TFT 2.4" Shield (рассмотрим это в разделе 6.6).

Схема соединений представлена на рис. 6.3.

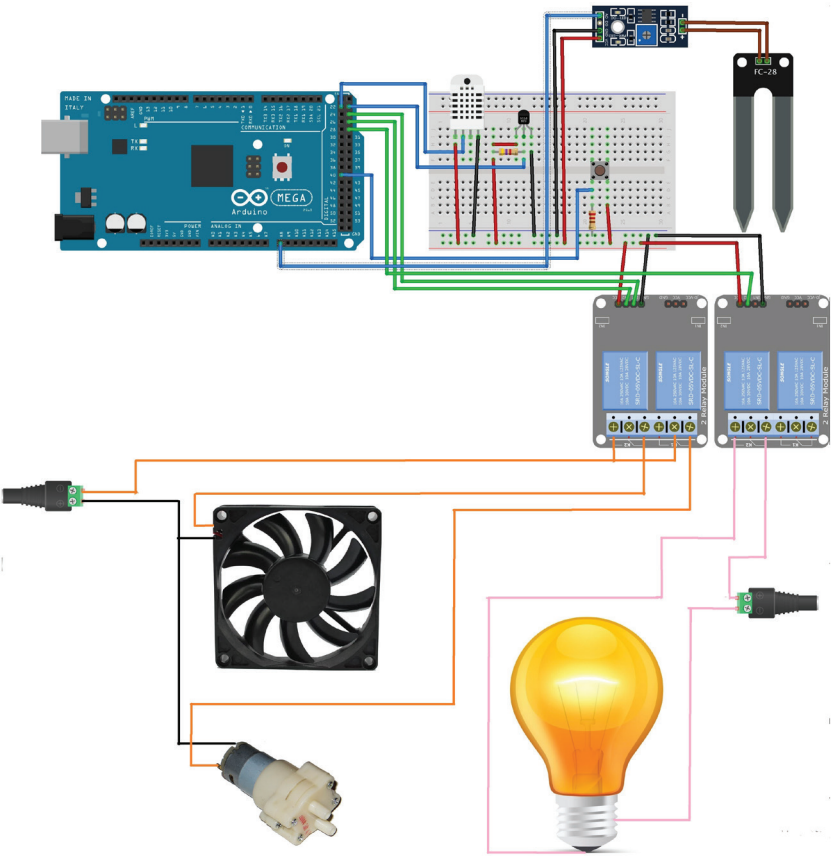


Рис. 6.3. Схема подключений Relay Shield к плате Arduino MEGA

Внимание!

Работа с высоким напряжением опасна для вашего здоровья и жизни. На плате существуют области, прикосновение к которым приведет к поражению электрическим током. Это винты контактных колодок и места пайки выводов контактных колодок и реле. Не работайте с платой, если она подключена к бытовой сети. Для готового устройства используйте изолированный корпус.

Если вы сомневаетесь, как подключить к реле электроприбор, работающий от общей сети 220 В, и у вас есть сомнения, вопросы на тему того, как это делает-

ся, остановитесь: вы можете устроить пожар или убить себя. Убедитесь, что у вас в голове – кристальное понимание принципа работы устройства и опасностей, которые связаны с высоким напряжением.

Загрузим на плату Arduino MEGA скетч включения/выключения вентилятора и насоса в зависимости от данных температуры воздуха и влажности почвы.

Необходимо внести изменения в существующий скетч. Во-первых, добавляем массив пинов для подключения реле (пусть их будет 8, с запасом под расширения):

```
int pinrelays[] = {27,29,31,33,35,37,39};
```

А также массив для хранения данных реле datarelays[] (0 – выключено, 1 –включено) и массив, в котором будем хранить фактически подключенные устройства aktiverelays[] (1 – подлючено, 0 – выключено):

```
int aktiverelays[] = {1,1,1,1,1,0,1,0};
int datarelays[] = {0,0,0,0,0,0,0,0};
```

Необходимо добавить и константы в файл defines.h.

Теперь в основной файл скетча в цикл loop() добавим код включения/выключения реле, управляющих вентилятором и помпой в зависимости от показаний датчиков температуры DS18B20 и увлажненности почвы Soil Moisture.

Содержимое данного фрагмента скетча представлено в листинге 6.1.

Листинг 6.1

```
void loop() {
   ............
   // установка значений реле
   // вентилятор
   if(aktivesensors[1] == 1 && datasensors[1] < 999
         && datasensors[1] > LIMIT_DS18B20MAX) {
      digitalWrite(pinrelays[2],RELAY_ON);
      datarelays[2] = 1;
   }
   else {
      digitalWrite(pinrelays[2],RELAY_OFF);
      datarelays[2] = 0;
   }
   // помпа
   if(aktivesensors[2] == 1 && datasensors[2] < 999
```

```
        && datasensors[1]>MAXVALUESOILMOISTURE 900) {
      digitalWrite(pinrelays[1],RELAY_ON);
      datarelays[1] = 1;
   }
   else {
      digitalWrite(pinrelays[1],RELAY_OFF);
      datarelays[1] = 0;
   }
……..
}
```

После загрузки скетча на плату Arduino у вас будут обеспечены автоматическое включение/выключение вентилятора и автоматический полив растений.

6.3. Подключение блока реле к модулю NodeMCU

Теперь рассмотрим подключение Relay Shield к модулю NodeMCU. Relay Shield мы будем использовать также – для включения/выключения света для освещения растений, вентилятора, насоса для полива растений. Светом будем управлять с помощью кнопки включения/выключения. Управление вентилятором и помпой будет осуществляться в зависимости от значений температуры воздуха (вентилятор) и влажности почвы (мембранный вакуумный насос.

Схема соединений представлена на рис. 6.4.

Загрузим на плату Arduino MEGA скетч включения/выключения вентилятора и насоса в зависимости от данных температуры воздуха и влажности почвы, а также включения/выключения света с помощью кнопки.

Необходимо внести изменения в существующий скетч. Во-первых добавляем массив пинов для подключения реле (пусть их будет 8, с запасом под расширения):

```
// массив для хранения данных состояний реле
int datarelays[] = {0,0,0,0,0,0,0,0};
```

Поскольку реле так же, как и светодиоды, подключены через I2C-расширитель выводов MCP23017, в setup производим настройку портов микросхемы MCP23017 (процедура ini_mcp23017().

```
void ini_mcp23017() {
  Wire.beginTransmission(0x20);  // i2c - адрес (A0-0,A1-0,A2-0)
  Wire.write(0x00); // IODIRA register
  Wire.write(0x00); // настроить PORT A как output (для светодиодов)
```

```
  Wire.endTransmission();
  Wire.beginTransmission(0x20);
  Wire.write(0x01); // IODIRB register
  Wire.write(0x00); // настроить PORT B как output (для реле)
  Wire.endTransmission();
}
```

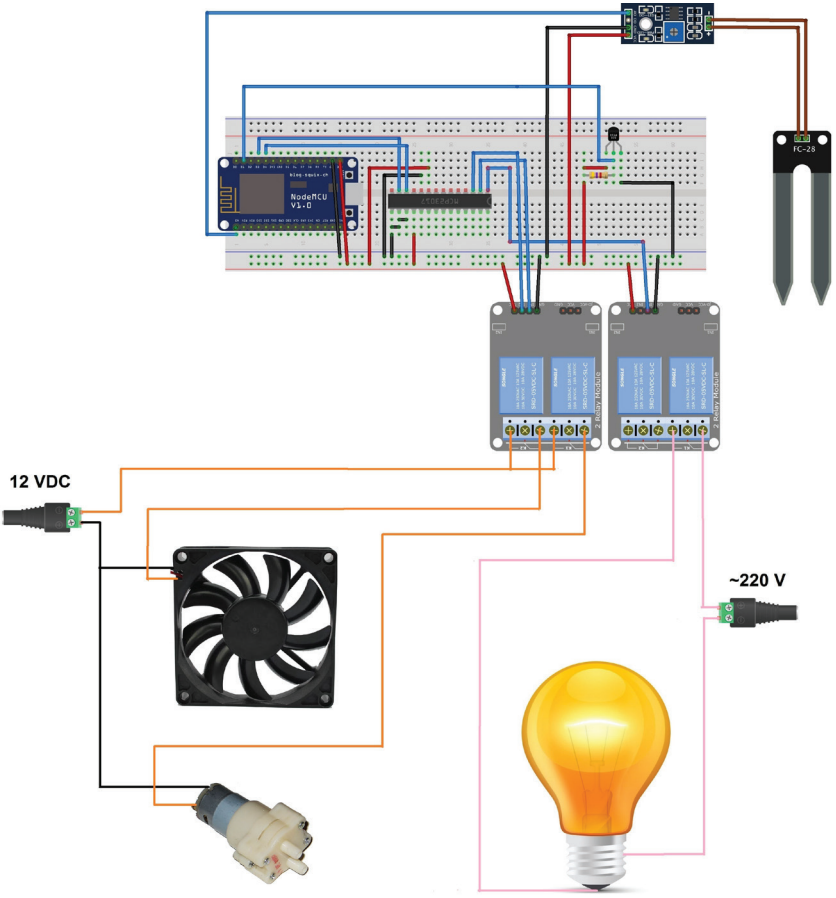


Рис. 6.4. Схема подключений Relay Shield к модулю NodeMCU

Необходимо добавить и константу в файл defines.h (пин подключения кнопки включения света).

```
#define BUTTONLIGHTPIN1          10
```

И переменные состояния кнопки:

```
int lastButtons1 = 0;
int currentButtons1 = 0;
```

Теперь в основной файл скетча в цикл loop() добавим код включения/выключения реле, управляющих вентилятором и помпой в зависимости от показаний датчиков температуры DS18B20 и увлажненности почвы SoilMoisture и включения/выключения света по нажатии кнопки.

Содержимое данного фрагмента скетча представлено в листинге 6.2.

Листинг 6.2

```
void loop() {
   ............
   // установка значений реле
   // вентилятор
   if(aktivesensors[1]==1 && datasensors[1]<999
         && datasensors[1]>LIMIT_DS18B20MAX) {
      datarelays[2] = 1;
      set_status_relays();
   }
   else {
      datarelays[2] = 0;
      set_status_relays();
   }
   // помпа
   if(aktivesensors[2]==1 && datasensors[2]<999
         && datasensors[1]>MAXVALUESOILMOISTURE 900) {
      datarelays[1] = 1;
      set_status_relays();
   }
   else {
      datarelays[1] = 0;
      set_status_relays();
   }
   // нажатие кнопки включения/выключения света
   currentButtons1 = debounce(lastButtons1, BUTTONLIGHTPIN1);
   if (lastButtons1 == 0 && currentButtons1 == 1) // если нажатие...
     {
     // изменить состояние реле
     datarelays[0] = 1 - datarelays[0];
     // вывести в порты MCP23017
     set_status_relays();
     }
   lastButtons1 = currentButtons1;
.......
}
```

Функция set_status_relays() осуществляет включение/выключение реле установкой необходимых уровней на выходах порта B микросхемы MCP23017.

```
// управление реле
void set_status_relays() {
  int relays = 0;
  for(int i=0;i<7;i++)
     relays=relays +( datarelays[i]<<i);
  // записать данные в PORT B
  Wire.beginTransmission(0x20);
  Wire.write(0x13); // address PORT B
  Wire.write(relays);    // PORT B
  Wire.endTransmission();
  delay(100); // пауза
}
```

Функция сглаживания дребезга debounce() принимает в качестве аргумента предыдущее состояние кнопки и выдает фактическое.

```
int debounce(int last,int pin1)
  {
  int current = digitalRead(pin1);// Считать состояние кнопки
  if (last != current)            // если изменилось...
    {
    delay(5);                     // ждем 5мс
    current = digitalRead(pin1);  // считываем состояние кнопки
    return current;               // возвращаем состояние кнопки
    }
  }
```

После загрузки скетча на плату Arduino у вас будут обеспечены автоматическое включение/выключение вентилятора и автоматический полив растений, а также включение/выключение света по нажатии клавиши.

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

6.4. Управление блока реле по ИК-каналу (для NodeMCU)

В набор включен инфракрасный пульт дистанционного управления с платой инфракрасного приемника (рис. 6.5).

Рис. 6.5. Пульт ДУ с ИК-приемником в комплекте

Это позволяет нам организовать управление исполнительными устройствами, подключенными к Relay Shield с помощью ИК-пульта. Схема подключения ИК-приемника к модулю NodeMCU приведена на рис. 6.6.

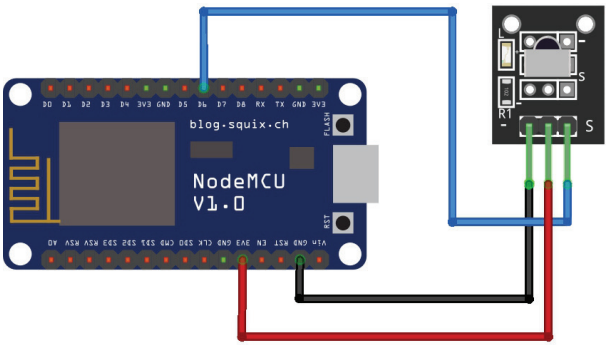


Рис. 6.6. Схема подключения ИК-приемника к модулю NodeMCU

После подключения ИК-приемника необходимо узнать коды клавиш пульта, которые мы будем использовать для управления исполнительными устройствами. Скачаем и установим в Arduino IDE библиотеку IRremoteESP8266 и загрузим на модуль NodeMCU пример из данной библиотеки IRrecvDemo. Нажимая кнопки пульта,

узнаем коды клавиш, которые будем использовать (например, цифровые «1», «2», «3»).

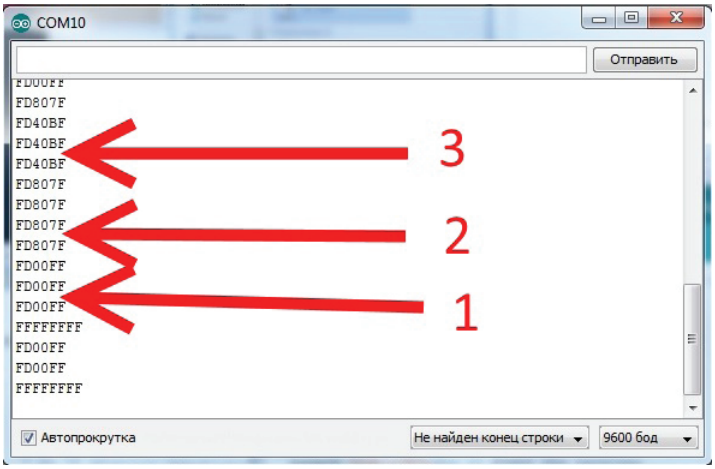


Рис. 6.7. Получение кодов ИК-пульта (клавиши «1», «2», «3»)

Теперь мы будем использовать эти коды в скетче управления «умным домом». Вносим изменения в скетч. Подключаем библиотеку

```
#include <IRremoteESP8266.h>
```

Пин подключения ИК-приемника D6 (GPIO12):

```
int RECV_PIN = 12;
```

Создаем необходимые переменные:

```
IRrecv irrecv(RECV_PIN);
decode_results results;
```

В процедуре setup() запускаем ИК-приемник:

```
irrecv.enableIRIn();
```

Добавляем в цикл loop() код для обработки команд с ИК-пульта (см. листинг 6.3).

Листинг 6.3

```
void loop() {
   ............
   // обработка кодов с пульта
  if (irrecv.decode(&results)) {
    switch(results.value, HEX){
       case 0xFD00FF:   // "1"
            datarelays[0] = 1 - datarelays[0];
            set_status_relays();
         break;
       case 0xFD807F:   // "2"
            datarelays[1] = 1 - datarelays[1];
            set_status_relays();
         break;
       case 0xFD40BF:   // "3"
            datarelays[2] = 1 - datarelays[2];
            set_status_relays();
         break;
      default:
         break;
    }
    irrecv.resume();
  }
.......
}
```

Функция set_status_relays() осуществляет включение/выключение реле установкой необходимых уровней на выходах порта B микросхемы MCP23017.

```
// управление реле
void set_status_leds() {
  int relays = 0;
  for(int i = 0; i<7; i++)
     relays=relays +( statusrelays[i]<<i);
  // записать данные в PORT B
  Wire.beginTransmission(0x20);
  Wire.write(0x13); // address PORT B
  Wire.write(leds); // PORT B
  Wire.endTransmission();
  delay(100);       // пауза
}
```

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

6.5. Организация доступа в дом с помощью RFID-модуля для Arduino MEGA

Один из важнейших вопросов при создании умного дома – безопасность. Рассмотрим организацию доступа по технологии RFID с помощью карт доступа.

Радиочастотная RFID – это технология автоматической бесконтактной идентификации объектов при помощи радиочастотного канала связи. Любая RFID-система состоит из:

- RFID-метки;
- считывателя информации (RFID-ридера);
- микроконтроллера или компьютера для дальнейшей обработки информации.

Идентификация объектов производится по уникальному цифровому коду, который считывается из памяти электронной метки, прикрепляемой к объекту идентификации.

Считыватель содержит в своем составе передатчик и антенну и посылает в эфир электромагнитные сигналы определенной частоты. RFID-метки «отвечают» собственным сигналом, который содержит информацию об идентификационном номере данной метки и данные об объекте, оснащенном данной меткой. Этот сигнал улавливается антенной считывателя, информация расшифровывается и передается для дальнейшей обработки.

В состав набора входит модуль RC522 – RFID-модуль 13,56 МГц с SPI-интерфейсом (см. рис. 6.8). В комплекте к модулю идут 2 RFID-метки – в виде карты и брелока. В RFID-метках, работающих на данной частоте, реализована криптографическая защита, что обеспечивает защиту от копирования и подделки.

Основные характеристики:

- модуль основан на микросхеме MFRC522;
- напряжение питания: 3.3 V;
- потребляемый ток: 13–26 mA;
- рабочая частота: 13.56 MHz;
- дальность считывания: 0 ~ 60 мм;
- интерфейс: SPI, максимальная скорость передачи 10 МБит/с;
- размер: 40 мм × 60 мм;
- чтение и запись RFID-меток.

Схема подключения модуля к плате Arduino показана на рис. 6.9. При определении RFID-считывателем карты из списка мы можем отправить сигнал на открытие дверей (например, на электромеханический замок, подключенный через реле).

Рис. 6.8. RFID-модуль RC522 и RFID-метки

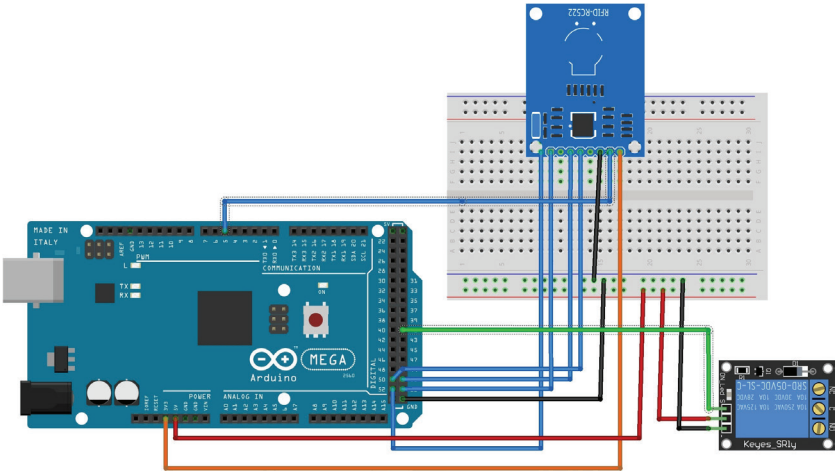


Рис. 6.9. Схема подключения RFID-модуля к плате Arduino MEGA

Приступим к написанию скетча. Для работы с RFID-модулем RC522 будем использовать библиотеки SPI (встроенная в Arduino IDE) и MFRC522.

```
#include <SPI.h>
#include <MFRC522.h>
```

Уникальные идентификаторы карт или брелоков, по которым доступен вход, запишем в массиве uid_ok:

```
byte uidok[][4] = {
                  {0x11,0x22,0x33,0x44},
                  {0x11,0x22,0x33,0x44}
};
```

Пины для подключения реле:

```
#define RELAY_RC522_PIN 41
```

В цикле будем считывать данные с поднесенной карты или брелока, затем проверять uid карты со списком разрешенных uid.

Код скетча представлен в листинге 6.4.

Листинг 6.4

```
#include <SPI.h>
#include <MFRC522.h>
// пин для подключения реле
#define RELAY_RC522_PIN 41
// уровни для включения/выключения реле
#define RELAY_ON 0
#define RELAY_OFF 1
// Инициализация MFRC522
// константы подключения контактов SS и RST
#define RST_PIN = 5;
#define SS_PIN = 53;
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.
// массив разрешенных uid
byte uidok[][4] = {
                  {0xE0, 0x2A, 0x87, 0x1B},
                  {0xD9, 0xFA, 0x90, 0x55 }
};

void setup() {
   Serial.begin(9600); // инициализация последовательного порта
   SPI.begin();        // инициализация SPI
   mfrc522.PCD_Init(); // инициализация MFRC522
   // сконфигурировать вывод реле как OUTPUT
   pinMode(RELAY_RC522_PIN,OUTPUT);
   digitalWrite(RELAY_RC522_PIN, RELAY_OFF);
}
```

```
void loop() {
  if (mfrc522.PICC_IsNewCardPresent()) {
   // чтение карты
   if ( mfrc522.PICC_ReadCardSerial()) {
      // показать результат чтения UID и тип метки
      Serial.print(F("Card UID:"));
      if(compare_uid(mfrc522.uid.uidByte, mfrc522.uid.size)) {
          Serial.println();
          Serial.println(" - ok !!");
          // включить реле
          digitalWrite(RELAY_RC522_PIN, RELAY_ON);
          delay(4000);
          digitalWrite(RELAY_RC522_PIN, RELAY_OFF);
      } else {
          Serial.println();
          Serial.println(" - false !!");
          delay(1000);
      }
   }
  }
}
// поиск считанного uid в списке разрешенных
boolean compare_uid(byte *buffer, byte bufferSize) {
int bytes_ok = 0;
   for(int i1 = 0; i1<2; i1++) {
      bytes_ok = 0;
      for (byte i2 = 0; i2 < 4; i2++) {
         Serial.print(buffer[i2] < 0x10 ? " 0" : " ");
         Serial.print(buffer[i2], HEX);
         if(buffer[i2]== uidok[i1][i2]) {
            bytes_ok= bytes_ok + 1;
         }
      }
      if(bytes_ok == 4) {
         return true;
      }
   }
   return false;
}
```

Загрузим данный скетч на плату Arduino MEGA и проверим ограничение доступа к работе с помощью RFID (см. рис. 6.10).

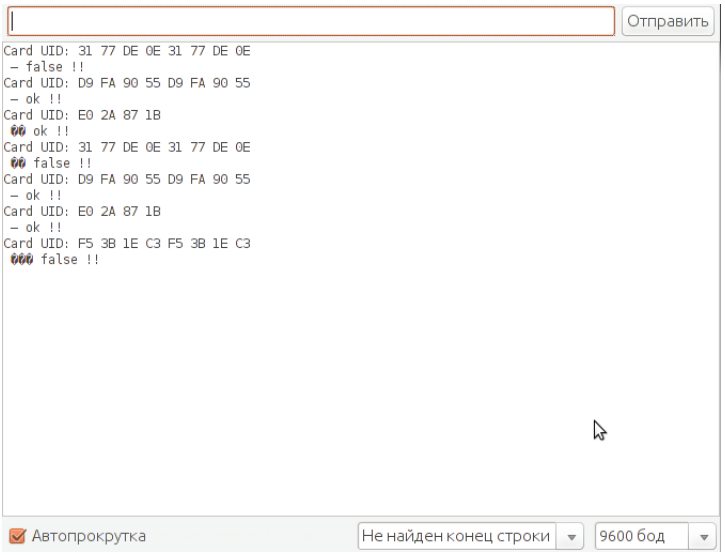


Рис. 6.10. Доступ по RFID-картам

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

6.6. Отображение данных о статусе исполнительных устройств на экране дисплея и управление с помощью сенсора

Теперь посмотрим, что можно сделать, используя TFT 2.4" Shield. В разделе 5.7 мы рассматривали вывод данных, получаемых с датчиков на экран TFT 2.4" Shield. Теперь необходимо на экран дисплея выводить и данные о состоянии исполнительных устройств, подключенных к реле. Для этого нам необходимо формировать другие экраны, а также главный экран. Как мы будем делать переходы между экранами? Будем использовать сенсорный экран TFT 2.4" Shield.

Сенсорная часть дисплея использует 4 контакта совместно с ЖКИ:

- A1 – TOUCH_YP;
- A0 – TOUCH_XM;
- D6 – TOUCH_XP;
- D7 – TOUCH_YM.

Для работы с тачскрином будем использовать библиотеку **Touch Screen**. Рассмотрим, какие изменения необходимо внести в код. Во-первых, подключим библиотеку Touch Screen:

```
#include "TouchScreen.h"
```

Создадим объект TouchScreen ts:

```
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);
```

Создадим новую переменную screen – текущий экран для отображения:

```
int screen = 0;    // 0 - главное меню
                   // 1 - датчики
                   // 2 - реле
                   // 3 - будильники
```

Сформируем код для формирования экранов – главного меню (view_display_main()) и состояния реле (view_display_relays()) – см. листинг 6.5.

Листинг 6.5

```
// вывод главного экрана
void view_display_main() {
  // очистить экран
  tft.fillScreen(BLACK);
  // вывод заголовка экрана
  tft.setTextColor(MAGENTA);
  tft.setTextSize(3);
  for(int i=0;i<6;i++) {
     tft.setCursor(15, xmenu[i]);
     tft.print(strM1[i]);
  }
}
// вывод экрана с показаниями реле
void view_display_relays() {
  // очистить экран
  tft.fillScreen(BLACK);
  // вывод заголовка экрана
  tft.setCursor(10, 15);
  tft.setTextColor(MAGENTA);
  tft.setTextSize(3);
  tft.println("Relays:");
  // вывод списка реле
  tft.setTextColor(YELLOW);
  tft.setTextSize(2);
```

```
  for(int i=0;i<COUNT_RELAYS;i++) {
     tft.setCursor(10, xrelays[i]);
     tft.print(strR1[i]);
  }
  // вывод списка значений
  for(int i=0;i<COUNT_RELAYS;i++) {
     if(aktiverelays[i]==1) {
        Serial.print("COUNT_RELAYS=");Serial.println(i);
        if(datarelays[i]==1) {
            set_display_data_relay(i,datarelays[i],RED);
        }
        else {
            set_display_data_relay(i,datarelays[i],YELLOW);
        }
     }
  }


  // ardyino-kit
  tft.setTextSize(3);
  tft.setTextColor(MAGENTA);
  tft.setCursor(10, 270);
  tft.print("arduino-kit");
  // вывод для возврата в главное меню
}
```

А также процедуры для вывода меняющихся значений позиции реле (set_display_data_relay()).

```
void set_display_data_relay(int pos,float var,unsigned int color) {
   tft.setTextSize(2);
   tft.setTextColor(color);
   tft.setCursor(100, xrelays[pos]);
   if(var==0)
      tft.print("OFF");
   else
      tft.print("ON");
   }
```

Теперь нам необходимо в цикле loop() добавить проверку текущего экрана перед выводом показаний на дисплей TFT 2.4" Shield-a – листинг 6.6.

Листинг 6.6

```
void loop() {
   ……..
     // для датчиков
         if(fvar<999 && screen==1) {
            set_display_data_sensor(teksensor,
```

```
                    datasensors[teksensor],BLACK);
            datasensors[teksensor]=fvar;
            set_display_data_sensor(teksensor,
                    datasensors[teksensor],YELLOW);
          }
    ……..
    // для реле
    // вентилятор
    if(aktivesensors[1]==1 && datasensors[1]<999
         && datasensors[1]>LIMIT_DS18B20MAX) {
      digitalWrite(pinrelays[2],RELAY_ON);
      if( aktiverelays[2]<1 && screen==2)
        set_display_data_relay(2,datarelays[2],RED);
      aktiverelays[2] = 1;
    }
    else {
      digitalWrite(pinrelays[2],RELAY_OFF);
      if( aktiverelays[2]>0 && screen==2)
        set_display_data_relay(2,datarelays[2],YELLOW);
      aktiverelays[2] = 0;
    }
    // помпа
    if(aktivesensors[2]==1 && datasensors[2]<999
         && datasensors[1]>MAXVALUESOILMOISTURE 900) {
      digitalWrite(pinrelays[1],RELAY_ON);
      if( aktiverelays[1]<1 && screen==2)
        set_display_data_relay(1,datarelays[1],RED);
      aktiverelays[1] = 1;
    }
    else {
      digitalWrite(pinrelays[1],RELAY_OFF);
      if( aktiverelays[1]>0 && screen==2)
        set_display_data_relay(1,datarelays[1],YELLOW);
      aktiverelays[1] = 0;
    }
    ……..
}
```

И добавляем код для работы с сенсорным экраном. В основном цикле loop() проверяем, было ли нажатие на экран:

```
void loop() {
   ……..
  // определяем нажатие на экран
  int tt = ts.pressure();
  if(tt > MINPRESSURE && tt < MAXPRESSURE) {
     Serial.println("touch");
     int x=ts.readTouchX();int y = ts.readTouchY();
     Serial.print("X = "); Serial.print(x);
     Serial.print("\tY = "); Serial.print(y);
     pinMode(XM, OUTPUT);pinMode(YP, OUTPUT);
     pinMode(XP, OUTPUT);pinMode(YM, OUTPUT);
     do_for_touch(x,y,screen) ;
```

```
    delay(1000);
  }
  // возврат к назначениям пинов для вывода изображения
  pinMode(XM, OUTPUT);pinMode(YP, OUTPUT);
  pinMode(XP, OUTPUT);pinMode(YM, OUTPUT);
    ……..
}
```

В случае нажатия – проверка попадания в нужную область (в зависимости от текущего экрана отображения) – процедура do_for_touch():

```
// по нажатии по экрану
void do_for_touch(int x,int y,int res) {
    // главный экран
    if(res==0) {
        // на sensors
        if(x<370 && x>320 && y<680 && y>280 ) {
           screen = 1;
           view_display_sensors();
        }
        // на relays
        if(x<470 && x>420 && y<680 && y>280 ) {
           screen = 2;
           view_display_relays();
        }
    }
    // экран sensors
    else if(res==1) {
        // на main
        if(x<820 && x>780 && y<800 && y>160 ) {
           screen = 0;
           view_display_main();
           Serial.println("00000000000");
       }
    }
    // экран relays
    else if(res==2) {
        // на main
        if(x<820 && x>780 && y<800 && y>160 ) {
           screen = 0;
           view_display_main();
           Serial.println("00000000000");
       }
    }
    // экран alarms
    else if(res==3) {
       ;
    }
    //
    else;
}
```

Загружаем скетч на плату и переходим по экранам по нажатии на нужные области (рис. 6.11, 6.12, 6.13). Вывод на экран главного меню – по нажатии по надписи **arduino-kit**.

Рис. 6.11. Экран главного меню

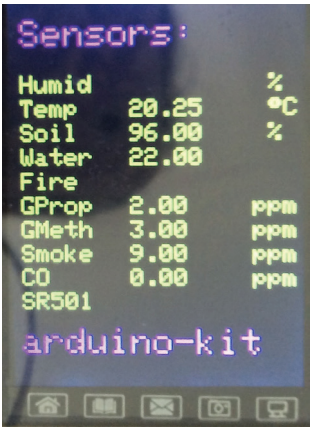


Рис. 6.12. Экран показаний сенсоров

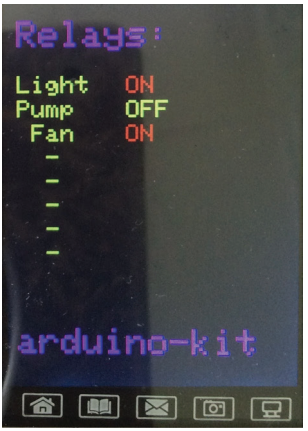


Рис. 6.13. Экран состояний реле

Далее нам необходимо добавить код включения лампы по нажатии на сенсорный экран. Для этого вносим изменения в процедуру do_for_touch(). Добавляем следующий код:

```
// экран relays
else if(res==2) {
    // на main
    if(x<820 && x>780 && y<800 && y>160 ) {
       screen = 0;
       view_display_main();
    }
    // ON / OFF лампы
    else if(x<290 && x>270 && y<520 && y>440 ) {
       set_display_data_relay(0,datarelays[0],BLACK);
       datarelays[0] = 1-datarelays[0];
       if(datarelays[0]==0) {
          digitalWrite(pinrelays[1],RELAY_OFF);
          set_display_data_relay(0,datarelays[0],YELLOW);
       }
       else {
          digitalWrite(pinrelays[1],RELAY_ON);
          set_display_data_relay(0,datarelays[0],RED);
       }
   }
}
// экран alarms
```

Теперь мы можем управлять включением/выключением лампы нажатием по сенсорному экрану.

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

7 СОЗДАНИЕ БУДИЛЬНИКОВ ДЛЯ ЗАПУСКА ИСПОЛНИТЕЛЬНЫХ УСТРОЙСТВ ПО РАСПИСАНИЮ

В предыдущей главе мы рассмотрели управление исполнительными устройствами «умного дома» либо с помощью команд, отправляемых по нажатии кнопки или нажатии по кнопке на сенсорном дисплее, либо при наступлении определенных климатических параметров, данные о которых мы получаем с датчиков.

Но очень часто исполнительные устройства требуется включать/выключать по расписанию: включение освещения перед домом при наступлении сумерек, полив растений по расписанию, выключение наружного освещения днем и т.д.

Для этого наши контроллеры должны знать реальное время. В наборе для этого есть модуль часов реального времени (RTC) на микросхеме DS3231 (рис. 7.1).

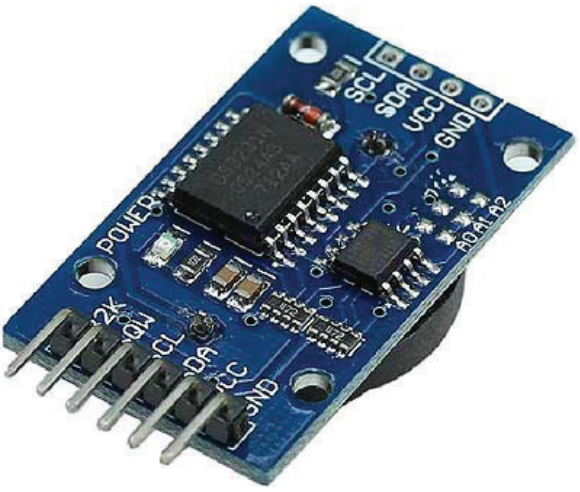


Рис. 7.1. Модуль часов реального времени (RTC) на микросхеме DS3231

Микросхема DS3231 – часы реального времени с температурной компенсацией кварцевого генератора и кристалла. В чипе используется не внешний кварцевый генератор 32 кГц, а внутренний и датчик температуры, который компенсирует изменения температуры, так что время остается точным. Погрешность составляет ±2 минуты за год при температуре окружающей среды от –40 °C до +85 °C. Низкая потребляемая мощность, полный календарь с учетом високосных лет до 2100 года, часы плюс 56 байтов энергонезависимого статического ОЗУ. В микросхеме DS3231 имеется встроенная схема, определяющая аварийное отключение питания и автоматически подключающая резервную

батарейку. Также микросхема содержит два будильника, которые могут генерировать прерывания и выводить данное событие на один из выводов микросхемы.

Модуль подключается к микроконтроллеру при помощи шины I2C. На модуле уже предусмотрена подтяжка выводов SCL и SDA к шине питания с помощью резисторов 2 кОм.

Рассмотрим подключение модуля DS3231 к контроллерам Arduino MEGA и модулю NodeMCU для получения точного времени и организации управления исполнительными устройствами по будильникам.

7.1. Подключение модуля DS3231 к плате Arduino MEGA. Добавление срабатывания устройств умного дома по будильнику (для Arduino MEGA)

Рассмотрим подключение модуля DS3231 к плате Arduino MEGA. Используем выводы Arduino MEGA 20 (SDA) и 21(SCL).

Схема соединений представлена на рис. 7.2.

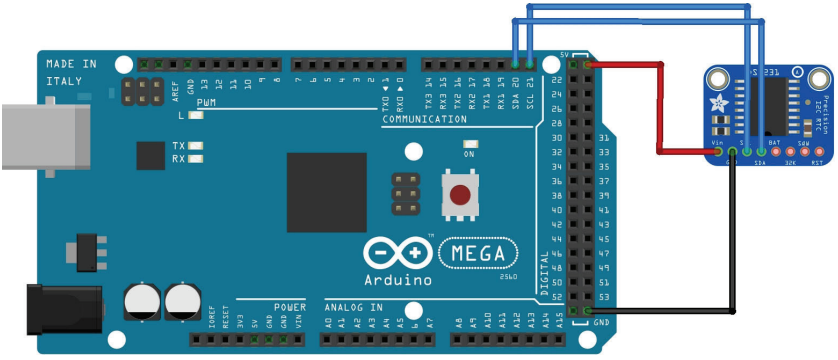


Рис. 7.2. Схема подключения модуля DS3231 к плате Arduino MEGA

Для работы с модулем будем использовать Arduino-библиотеки Wire (встроенная в Arduino IDE), Time и DS1307RTC. Мы можем организовать запуск исполнительных устройств «умного дома» по расписанию. Для этого создадим объект, описывающий будильник:

```
struct ALARM{
  int hours;       // час срабатывания будильника
  int minutes;     // минута срабатывания будильника
```

```
  int relay;      // номер реле исполнительного устройства (0-N)
  int value;      // устанавливаемое значение (0 или 1)
};
```

И создадим список необходимых будильников:

```
ALARM alarms[] = {{7,30,2,1},
                  {7,35,2,0},
                  {8,30,1,1},
                  {8,50,1,0}
                  };
```

Подключаем библиотеки:

```
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
```

Создаем переменную:

```
tmElements_t tm;
```

Добавляем в скетч переменную для периодичности опроса модуля RTC:

```
unsigned long millis3231;
```

В цикле loop() нашего скетча добавим проверку наступления события по расписанию и необходимых действий при наступлении события (см. листинг 7.1).

Листинг 7.1

```
void loop() {
   ............
// проверяем наступление события для будильников
  if(millis() - millisalarms >= 20000) {
     for(int i=0; i < sizeof(alarms)/8; i++) {
        if(alarms[i].hours == tm.Hour && alarms[i].minutes == tm.Minute) {
           // установить реле
           digitalWrite(pinrelays[alarms[i].relay],alarms[i].value);
        }
     }
     Millisalarms = millis();
  }
.......
}
```

Скачать полностью данный скетч можно на сайте https://arduino-kit.ru/iotprog.

7.2. Использование TFT 2.4" Shield 240×320. Вывод времени на экран дисплея

Теперь рассмотрим вариант с использованием TFT 2.4" Shield 240×320. Сначала добавим вывод времени на экран дисплея (главное меню). Вносим дополнительный код в скетч. Подключаем библиотеки:

```
#include <Wire.h>
#include <Time.h>
#include <DS1307RTC.h>
```

Создаем переменную:

```
tmElements_t tm;
```

Добавляем в скетч переменную для периодичности опроса модуля RTC:

```
unsigned long millis3231;
```

И в основной цикл loop() добавляем код получения данных времени и вывода на экран дисплея (см. листинг 7.2).

Листинг 7.2

```
void loop() {
   ............
  // показания RTC
  if(millis()-millisds3231 >= 5000 && screen == 0) {
     set_display_data_RTC(BLACK);
     RTC.read(tm);
     set_display_data_RTC(MAGENTA);  // вывод на дисплей
     millisds3231 = millis();
  }
.......
}
```

Процедура set_display_data_RTC() расположена в файле display.ino.

```
// вывести текущее значение для RTC
void set_display_data_RTC(unsigned int color) {
```

```
  tft.setTextSize(2);
  tft.setTextColor(color);
  tft.setCursor(80, 50);
  if(tm.Hour < 10)
     tft.print("0");
  tft.print(tm.Hour);
  tft.print(":");
  if(tm.Minute < 10)
     tft.print("0");
  tft.print(tm.Minute);
}
```

После загрузки скетча на плату Arduino на экране отображается реальное время, получаемое с модуля DS3231 (см. рис. 7.3).

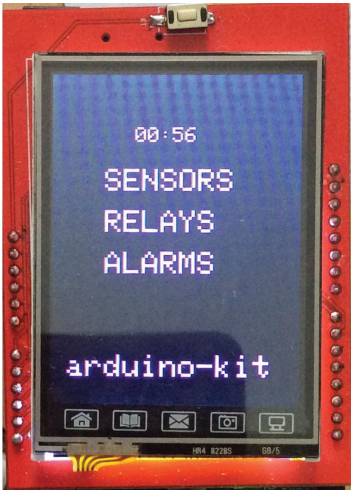


Рис. 7.3. Вывод текущего времени на экран дисплея

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

7.3. Вывод списка будильников на TFT 2.4 Shield 240×320

Добавим процедуру вывода списка будильников на экран дисплея (см. рис. 7.4):

```
// вывод  экрана будильников
void view_display_alarms() {
  // очистить экран
```

```
  tft.fillScreen(BLACK);
  // вывод заголовка экрана
  tft.setCursor(10, 15);
  tft.setTextColor(MAGENTA);
  tft.setTextSize(3);
  tft.println("Alarms:");
  // вывод списка будильников
  tft.setTextColor(YELLOW);
  tft.setTextSize(2);
  for(int i = 0; i < 4; i++) {
     tft.setCursor(10, xsensors[i]);
     if(alarms[i].hours < 10)
       tft.print("0");
     tft.print(alarms[i].hours);
     tft.print(":");
     if(alarms[i].minutes < 10)
        tft.print("0");
     tft.print(alarms[i].minutes);
     tft.setCursor(90, xsensors[i]);
     tft.print(strR1[alarms[i].relay]);
     tft.setCursor(170, xsensors[i]);
     if(alarms[i].value == 0)
        tft.print("OFF");
     else
        tft.print("ON");
  }
  // ardyino-kit
  tft.setTextSize(3);
  tft.setTextColor(MAGENTA);
  tft.setCursor(10, 270);
  tft.print("arduino-kit");
  // вывод для возврата в главное меню
}
```

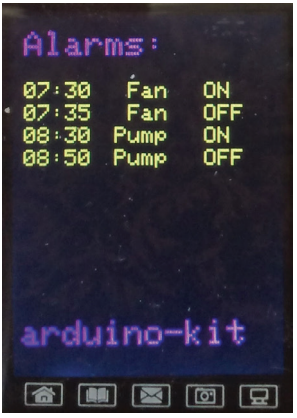


Рис. 7.4. Экран будильников

А также переход на данный экран с экрана главного меню и обратно на экран главного меню (добавляем в процедуру do_for_touch()):

```
// по нажатии по экрану
void do_for_touch(int x,int y,int res) {
…….
    // экран alarms
    else if(res == 3) {
        // на main
        if(x < 820 && x > 780 && y < 800 && y > 160 ) {
           screen = 0;
           view_display_main();
        }
    }
     ……

    // на relays
    if(x < 570 && x > 520 && y < 680 && y > 280 ) {
       screen = 2;
       view_display_alarms();
    }
……
}
```

Скачать полностью данный скетч можно на сайте https://arduino-kit.ru/iotprog.

7.3. Подключение модуля DS3231 к модулю NodeMCU

Рассмотрим подключение модуля DS3231 к модулю NodeMCU. У нас в проекте есть устройство, подключенное к модулю NodeMCU по протоколу I2C, – это микросхема расширителя входов MCP23017 (см. раздел 5.5). Подсоединяем к контактам NodeMCU D3 (GPIO0) – SCL и D4(GPIO2). Схема соединений представлена на рис. 7.5.

Загрузим на модуль NodeMCU скетч установки и получения текущего времени с RTC модуля DS3231 (см. листинг 7.4).

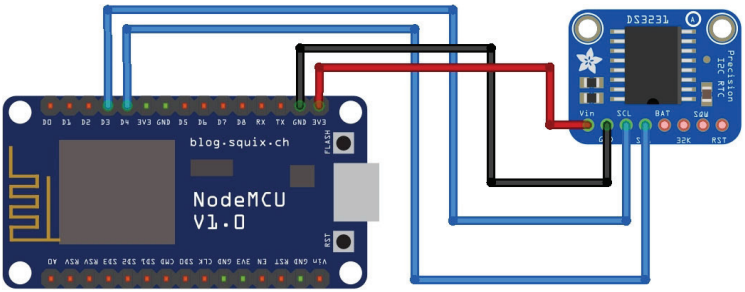


Рис. 7.5. Схема подключения модуля DS3231 к модулю NodeMCU

Листинг 7.4

```
#include <Wire.h>

int clockAddress = 0x68;  // I2C-адрес микросхемы
byte second, minute, hour, dayOfWeek, day, dayOfMonth, month, year;

#define DS3231_SCL 0
#define DS3231_SDA 2

void setup() {
  // запустить Wire
  Wire.begin(DS3231_SCL, DS3231_SDA);
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    String str = Serial.readStringUntil('\n');
    if (str == "show time") {
      getDateDs3231(); // показать время
      String s = getTimeStr();
      Serial.println(s);
    }
    if (str == "set time") {
      time_set();
    }
  }
}

// получить дату и время
void getDateDs3231() {
```

```
  Wire.beginTransmission(clockAddress);
  Wire.write(byte(0x00));
  Wire.endTransmission();
  Wire.requestFrom(clockAddress, 7);

  second = bcdToDec(Wire.read() & 0x7f);
  minute = bcdToDec(Wire.read());
  hour   = bcdToDec(Wire.read() & 0x3f);
  dayOfWeek  = bcdToDec(Wire.read());
  dayOfMonth = bcdToDec(Wire.read());
  day = dayOfMonth;
  month  = bcdToDec(Wire.read());
  year   = bcdToDec(Wire.read());
}
void setDateDs1307()
{
  Wire.beginTransmission(clockAddress);
  Wire.write(byte(0x00));
  Wire.write(decToBcd(second));
  Wire.write(decToBcd(minute));
  Wire.write(decToBcd(hour));
  Wire.write(decToBcd(dayOfWeek));
  Wire.write(decToBcd(dayOfMonth));
  Wire.write(decToBcd(month));
  Wire.write(decToBcd(year));
  Wire.endTransmission();
}

byte decToBcd(byte val) {
  return ( (val / 10 * 16) + (val % 10) );
}
byte bcdToDec(byte val) {
  return ( (val / 16 * 10) + (val % 16) );
}

String getTimeStr() {
  String str = String(day) + "." + formatDigit(month, 2) + "." +
      formatDigit(year, 2) + " " + formatDigit(hour, 2) + ":" +
      formatDigit(minute, 2) + ":" + formatDigit(second, 2);
  return str;
}
String formatDigit(int i, int len) {
  String s = String(i);
  while (s.length() < len) {
    s = "0" + s;
```

```
  }
  return (s);
}

void time_set() {
  Serial.println("Enter Year 2 last digits");
  while (Serial.available() <= 0);
  year = Serial.parseInt();
  Serial.println(year);

  Serial.println("Month");
  while (Serial.available() <= 0);
  month = Serial.parseInt();
  Serial.println(month);

  Serial.println("Day");
  while (Serial.available() <= 0);
  day = Serial.parseInt();
  dayOfMonth = day;
  Serial.println(day);

  Serial.println("Day of Week");
  while (Serial.available() <= 0);
  dayOfWeek = Serial.parseInt();
  Serial.println(dayOfWeek);

  Serial.println("Hour");
  while (Serial.available() <= 0);
  hour = Serial.parseInt();
  Serial.println(hour);

  Serial.println("Minute");
  while (Serial.available() <= 0);
  minute = Serial.parseInt();
  Serial.println(minute);

  setDateDs1307();

  Serial.println("Time recvd OK");
}
```

И результат работы скетча, позволяющего установить время отправкой команд по последовательному порту (рис. 7.6).

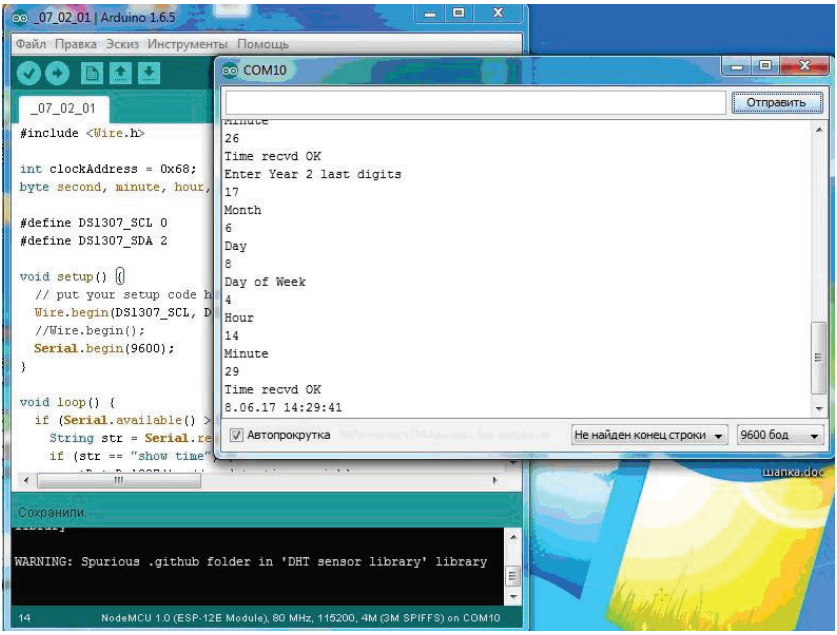


Рис. 7.6. Установка времени через последовательный порт

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

7.4. Добавление срабатывания устройств умного дома по будильнику (для NodeMCU)

После подключения модуля RTC мы можем организовать запуск исполнительных устройств «умного дома» по расписанию. Для этого создадим объект, описывающий будильник:

```
struct ALARM{
  int hours;       // час срабатывания будильника
  int minutes;     // минута срабатывания будильника
  int relay;       // номер реле исполнительного устройства (0-N)
  int value;       // устанавливаемое значение (0 или 1)
};
```

И создадим список необходимых будильников:

```
ALARM alarms[] = {{7,30,2,1},
                  {7,35,2,0},
                  {8,30,1,1},
                  {8,50,1,0}
                  };
```

В цикле loop() нашего скетча добавим проверку наступления события по расписанию и необходимых действий при наступлении события (см. листинг 7.5).

Листинг 7.5

```
void loop() {
   ............
// проверяем наступление события для будильников
  if(millis() - millisalarms >= 20000) {
     for(int i = 0; i < sizeof(alarms)/8; i++) {
        if(alarms[i].hours == tm.Hour && alarms[i].minutes == tm.Minute) {
           // установить реле
           digitalWrite(pinrelays[alarms[i].relay],alarms[i].value);
        }
     }
     Millisalarms = millis();
  }
.......
}
```

Скачать полностью данный скетч можно на сайте https://arduino-kit.ru/iotprog.

8 ОРГАНИЗАЦИЯ ПОДКЛЮЧЕНИЯ К СЕТИ ИНТЕРНЕТ

В предыдущих главах мы сделали большие шаги построения «умного дома» – оснастили его датчиками и исполнительными устройствами, создали и обеспечили определенную степень автоматизации для создания комфорта и безопасности. Теперь пришло время сделать наш «умный дом» устройством IoT (интернета вещей), чтобы получить доступ к нему для мониторинга и управления из любой точки мира по сети Интернет. Организуем доступ контроллеров нашего дома к сети Интернет.

NodeMCU – это плата на основе Wi-Fi-модуля ESP8266, что позволяет подключиться ей к сети Интернет по Wi-Fi-соединению. С Arduino MEGA немного сложнее – необходимы внешние модули для организации доступа к Интернету. В качестве внешних модулей часто используется плата Ethenet Shield W5100 или W5500. Можно использовать и модули ESP8266, управляя ими по UART. Мы рассмотрим организацию доступа в Интернет с использованием GPRS Shield – платы расширения, позволяющей Arduino работать в сетях сотовой связи по технологиям GSM/GPRS для приема и передачи данных, SMS и голосовой связи.

8.1. Модуль GSM/GPRS SIM800L

В качестве GPRS-Shield мы будем использовать модуль GSM/GPRS SIM800L на плате с антенной (см. рис. 8.1).

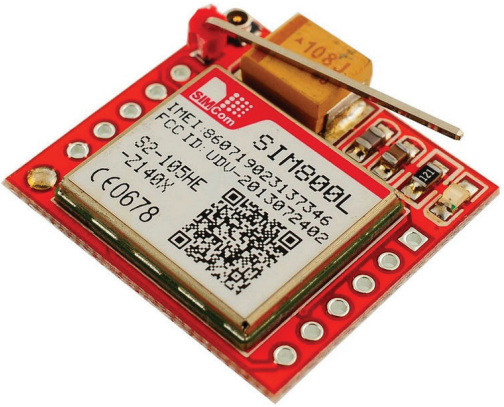


Рис. 8.1. Модуль GSM/GPRS SIM800L на плате с антенной

Стандартный интерфейс управления компонента SIM800L предоставляет доступ к сервисам сетей GSM/GPRS 850/900/1800/1900

МГц для отправки звонков, СМС-сообщений и обмена цифровыми данными GPRS. Поставляется со встроенной антенной, также можно подключить дополнительные антенны для улучшения качества сигнала. Данный модуль работает с российскими операторами Билайн, МТС, Мегафон. Имеет 1 слот под micro-SIM-карту. Скорость модуля по умолчанию 9600. Модуль поддерживает AT-команды. Работу с компьютером можно производить через преобразователь UART.

Характеристики модуля:

- напряжение питания номинальное, 4 В;
- напряжение питания, диапазон 3,4–4,4 В;
- потребление тока в режиме ожидания 0,7 мА;
- потребление тока, предельное значение 500 мА;
- максимальное напряжение высокого уровня интерфейса UART 2,8 В;
- скорость UART 1200–115200 бод;
- четыре диапазона EGSM900, DCS1800, GSM850, PCS1900;
- мощность передачи в диапазонах DCS1800, PCS1900 – 1 Вт;
- мощность передачи в диапазонах GSM850, EGSM900 – 2 Вт;
- отправка и получение GPRS-данных (TCP/IP, HTTP и т. д.);
- макс. скорость передачи GPRS данных 85,6 Кбод;
- поддержка часов реального времени RTC;
- размеры 25×25 мм.

Превышение входного напряжения интерфейса UART приведет к порче модуля SIM800. Не существует преобразователя интерфейса USB–UART с выходным напряжением 2,8 В. Существующие преобразователи имеют более высокое напряжение на выходе UART. Поэтому между выходом преобразователя и входом модуля GSM/GPRS SIM800L устанавливается резисторный делитель напряжения (см. рис. 8.2).

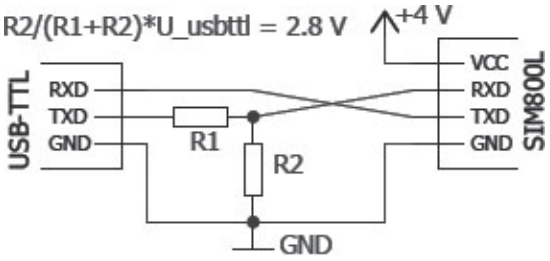


Рис. 8.2. Резисторный делитель при подключении модуля GSM/GPRS SIM800L

В наборе имеются сопротивления номиналом R2 = 1,8 кОм и R1 = 1,5 кОм, что вполне приемлемо:

1,8/(1,8 + 1,5) × 5 = 2,7 В.

8.2. Управление модулем GSM/GPRS SIM800L с помощью АТ-команд

Рассмотрим управление модулем GSM/GPRS SIM800L с помощью АТ-команд. Для этого подключим модуль к компьютеру, используя переходник FTDI и делители напряжения. Для питания модуля будем использовать преобразователь напряжения на микросхеме LM2596, на котором установим напряжения питания модуля 4 В. Схема соединений представлена на рис. 8.3.

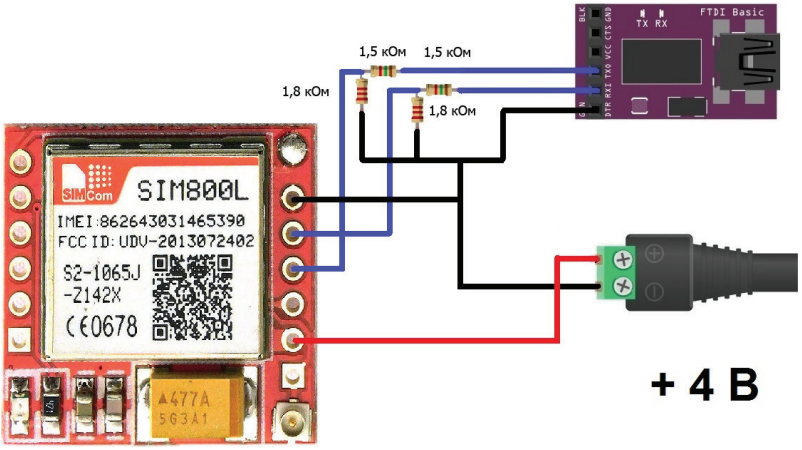


Рис. 8.3. Подключение модуля GSM/GPRS SIM800L к компьютеру

Подключаем модуль к компьютеру и запускаем программу для общения с последовательным портом. Можно использовать монитор последовательного порта Arduino IDE. Будем отправлять на модуль GSM/GPRS SIM800L АТ-команды. Загружаем на плату Arduino MEGA скетч из листинга 8.1.

Листинг 8.1

```
#define SIM800board Serial1
#define SIM800baud 9600
```

```
String hh;
char buffer[100];
char end_c[2];

void setup() {
  Serial.begin(9600);
  SIM800board.begin(SIM800baud);
  Serial.println("Start");
  end_c[0] = 0x1a;
  end_c[1] = '\0';
  delay(2000);
}

void loop() {
  if (Serial.available()) {
    hh = Serial.readStringUntil('\n');
    hh.toCharArray(buffer, hh.length() + 1);
    if (hh.indexOf("end") == 0) {
      SIM800board.write(end_c);
      Serial.println("end");
    } else{
      SIM800board.write(buffer);
      SIM800board.write('\n');
    }
  }
  if (SIM800board.available()) {
    Serial.write(SIM800board.read());
  }
}
```

После загрузки скетча набираем в мониторе последовательного порта следующие команды:

```
AT + CREG?
```

Если ответ +CREG:0,5 или +CREG:0,5, установим 1:

```
AT + CREG = 1
```

Далее проверка подключения модуля к GPRS-сети:

```
AT + CGATT?
```

Если ответ +CGATT: 0, установим 1:

```
AT + CGATT = 1
```

Это может занять много времени и написать COMMAND NO RESPONSE, необходимо повторять и потом поверить командой

```
AT + CGATT?
```

Подключаемся к точке доступа оператора связи. Для МТС это будет так:

```
AT + CGDCONT = 1, "IP", "internet.mts.ru"
AT + CSTT = "internet.mts.ru","mts", "mts"
```

Далее устанавливаем интернет-соединение:

```
AT + CGACT = 1,1
```

В случае ответа OK можем посмотреть наш динамический IP-адрес:

```
AT + CIFSR
```

И затем обращаемся к какому-нибудь серверу, например google.ru:

```
AT + CIPSTART = "TCP", www.google.com ", 80
```

Весь процесс подключения представлен на рис. 8.4.

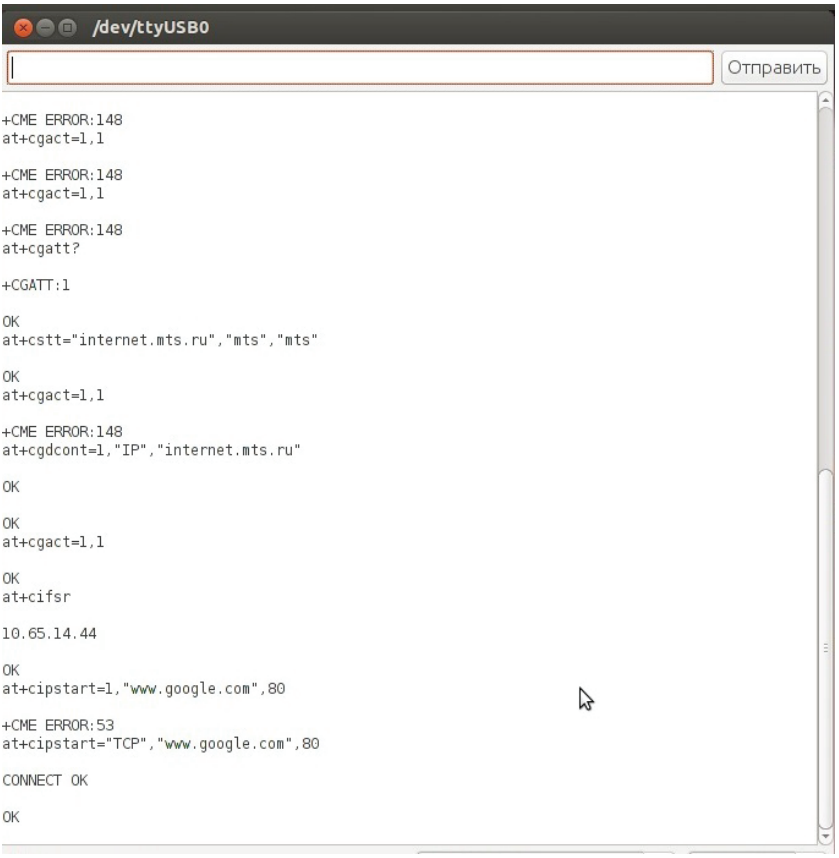


Рис. 8.4. Процесс работы с модулем GSM/GPRS SIM800L в мониторе последовательного порта

Теперь попробуем подключить модуль GSM/GPRS SIM800L к плате Arduino и отправить сообщение с данными на сервер, например http://sparkfun.com.

8.3. Подключение модуля GSM/GPRS SIM800L к плате Arduino MEGA

Подключим модуль GSM/GPRS SIM800L к плате Arduino MEGA и сделаем тестовую отправку данных по GPRS на удаленный сервер. Схема подключения приведена на рис. 8.5.

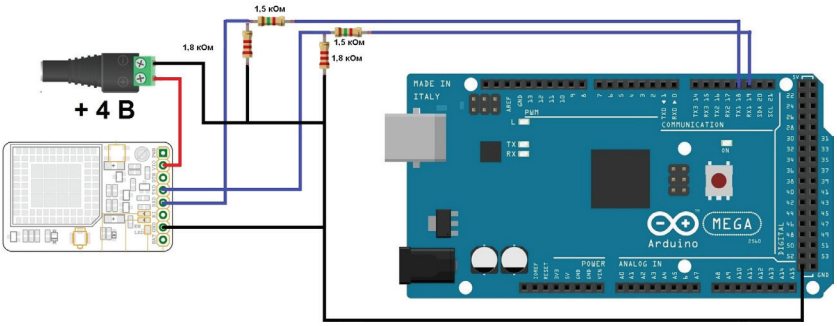


Рис. 8.5. Подключение модуля GSM/GPRS SIM800L к плате Arduino MEGA

В качестве удаленного сервера для отправки данных удобно использовать сервер data.sparkfun.com – здесь не требуется регистрации и можно быстро проверить правильность отправки данных. На главной странице переходим по ссылке создания бесплатного потока данных (рис. 8.6).

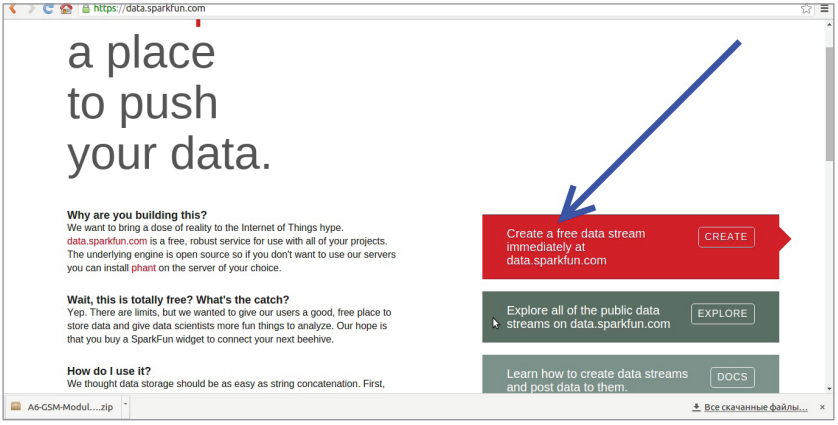


Рис. 8.6. Ссылка для создания бесплатного потока данных

На странице создания потока заполняем необходимые данные (рис. 8.7):

- **Название** (Title);
- **Описание** (Description);
- **Приватность** (Visible/Hidden);
- **Список полей** (Fields), –

и подтверждаем.

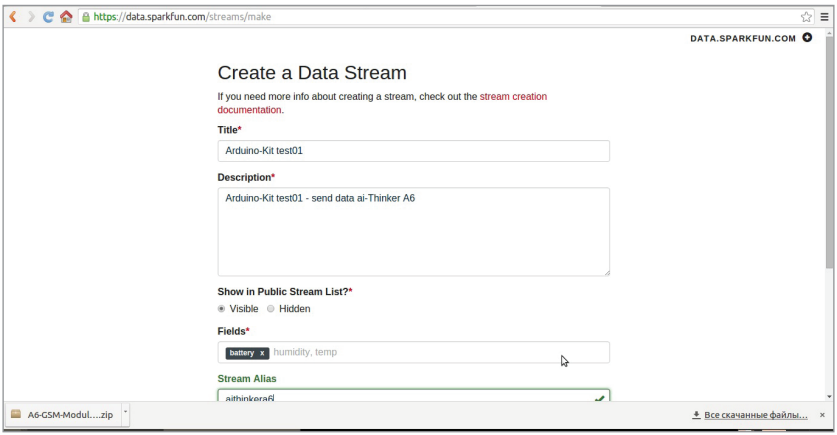


Рис. 8.7. Форма создания потока данных

При создании потока генерируются публичный ключ (Public key), приватный ключ (Private key), которые нам понадобятся при создании скетча отправки данных, и страница для просмотра отправленных данных (Public URL) (рис. 8.8).

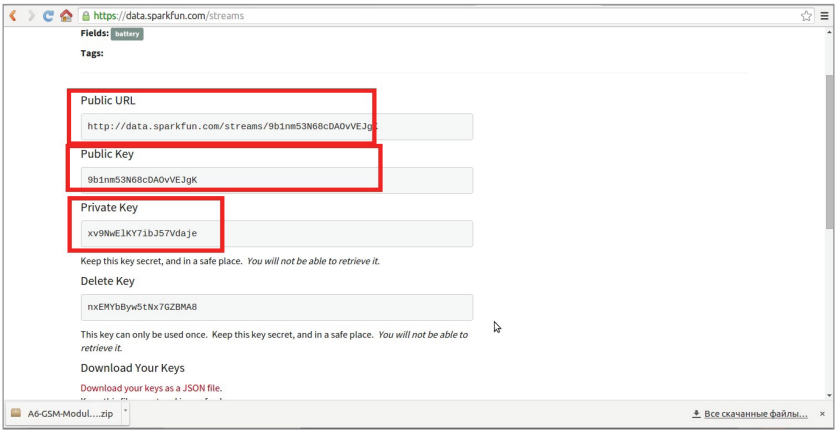


Рис. 8.8. Данные, необходимые для создания программы отправки

Теперь загрузим на плату Arduino Mega скетч из листинга 8.2, внеся в него изменения – свои данные Public key и Private key для сервиса data.sparkfun.com и данные к точке доступа своего сотового оператора (в скетче 8.2 данные для сотового оператора МТС).

Листинг 8.2

```
#define OK 1
#define NOTOK 2
#define TIMEOUT 3
#define RST 2

#define SIM800board Serial1
#define SIM800baud 9600
#define SERIALTIMEOUT 3000
char end_c[2];

void setup() {
  SIM800board.begin(SIM800baud);   // порт Serial1
  Serial.begin(9600);    // порт Serial
  // ctrlZ
  end_c[0] = 0x1a;
  end_c[1] = '\0';
  Serial.println("Start");
}

void loop()
{
  Serial.println("Waiting for command");
  if (Serial.available())
    switch (Serial.read())
    { // проверка команды на отправку данных
      case 's':
        Serial.println("-Post value to Sparkfun-");
        // получение аналоговых данных A0
        float batt = (float)analogRead(A0)*5.165/594.0;
        // вызов процедуры отправки данных
        sendSparkfunGSM(1, batt);
        break;
    }
  if (SIM800.available())
    Serial.write(SIM800.read());

  delay(2000);
}

///отправка данных на sparkfun()///
bool sendSparkfunGSM(byte sparkfunType, float value1) {
  String host = "data.sparkfun.com";
  String publicKey   = "9b1nm53N68cDAOvVEJgK";
  String privateKey = "xv9NwElKY7ibJ57Vdaje";
  SIM800command("AT+CIPSTATUS", "OK", "yy", 10000, 2);
  SIM800command("AT+CGATT?", "OK", "yy", 20000, 2);
  SIM800command("AT+CGATT=1", "OK", "yy", 20000, 2);
  SIM800command("AT+CIPSTATUS", "OK", "yy", 10000, 2);
  SIM800command("AT+CSTT=\"internet.mts.ru\",\"mts\",\"mts\"", "OK", "yy", 20000,
2); //bring up wireless connection
```

```
  SIM800command("AT+CGDCONT=1,\"IP\",\"internet.mts.ru\"", "OK", "yy", 20000, 2);
  SIM800command("AT+CIPSTATUS", "OK", "yy", 10000, 2);
  SIM800command("AT+CGACT=1,1", "OK", "yy", 10000, 2);
  SIM800command("AT+CIPSTATUS", "OK", "yy", 10000, 2);
  SIM800command("AT+CIFSR", "OK", "yy", 20000, 2); //get  IP adress
  SIM800command("AT+CIPSTATUS", "OK", "yy", 10000, 2);
  SIM800command("AT+CIPSTART=\"TCP\",\"" + host + "\",80", "CONNECT OK", "yy",
25000, 2); //start up the connection
  SIM800command("AT+CIPSTATUS", "OK", "yy", 10000, 2);
  // отправка данных на сервер
  SIM800command("AT+CIPSEND", ">", "yy", 10000, 1);
  delay(500);
  SIM800board.print("GET /input/");
  SIM800board.print(publicKey);
  SIM800board.print("?private_key=");
  SIM800board.print(privateKey);
  SIM800board.print("&battery=");
  SIM800board.print(value1, 2);
  SIM800board.print(" HTTP/1.1");
  SIM800board.print("\r\n");
  SIM800board.print("HOST: ");
  SIM800board.print(host);
  SIM800board.print("\r\n");
  SIM800board.print("\r\n");

  Serial.print("GET /input/");
  Serial.print(publicKey);
  Serial.print("?private_key=");
  Serial.print(privateKey);
  Serial.print("&battery=");
  Serial.print(value1, 2);
  Serial.print(" HTTP/1.1");
  Serial.print("\r\n");
  Serial.print("HOST: ");
  Serial.print(host);
  Serial.print("\r\n");
  Serial.print("\r\n");

  SIM800command(end_c, "HTTP/1.1", "yy", 30000, 1);
  SIM800board.println(end_c); //sending ctrlZ
  unsigned long   entry = millis();
  SIM800command("AT+CIPSTATUS", "OK", "yy", 10000, 2);
  SIM800command("AT+CIPCLOSE", "OK", "yy", 15000, 1); //sending
  SIM800command("AT+CIPSTATUS", "OK", "yy", 10000, 2);
  delay(100);
  Serial.println("-End-");
}


byte SIM800waitFor(String response1, String response2, int timeOut) {
  unsigned long entry = millis();
  int count = 0;
  String reply = SIM800read();
  byte retVal = 99;
```

```
  do {
    reply = SIM800read();
    if (reply != "") {
      Serial.print((millis() - entry));
      Serial.print(" ms ");
      Serial.println(reply);
    }
  } while ((reply.indexOf(response1) + reply.indexOf(response2) == -2) && mil-
lis() - entry < timeOut );
  if ((millis() - entry) >= timeOut) {
    retVal = TIMEOUT;
  } else {
    if (reply.indexOf(response1) + reply.indexOf(response2) > -2) retVal = OK;
    else retVal = NOTOK;
  }
  return retVal;
}

byte SIM800command(String command, String response1, String response2, int time-
Out, int repetitions) {
  byte returnValue = NOTOK;
  byte count = 0;
  while (count < repetitions && returnValue != OK) {
    SIM800board.println(command);
    Serial.print("Command: ");
    Serial.println(command);
    if (SIM800waitFor(response1, response2, timeOut) == OK) {
      returnValue = OK;
    } else returnValue = NOTOK;
    count++;
  }
  return returnValue;
}

bool SIM800begin() {
  SIM800board.println("AT+CREG?");
  byte hi = SIM800waitFor("1,", "5,", 1500);
  while ( hi != OK) {
    SIM800board.println("AT+CREG?");
    hi = SIM800waitFor("1,", "5,", 1500);
  }

  if (SIM800command("AT&F0", "OK", "yy", 5000, 2) == OK) {
    if (SIM800command("ATE0", "OK", "yy", 5000, 2) == OK) {
      if (SIM800command("AT+CMEE=2", "OK", "yy", 5000, 2) == OK)
         return OK;
      else return NOTOK;
    }
  }
}

void ShowSerialData()
{
  unsigned long entry = millis();
```

```
  while (SIM800board.available() != 0 && millis() - entry < SERIALTIMEOUT)
    Serial.println(SIM800board.readStringUntil('\n'));
}

String SIM800read() {
  String reply = "";
  if (SIM800board.available())  {
    reply = SIM800board.readString();
  }
  return reply;
}
```

После загрузки скетча на плату Arduino запускаем монитор последовательного порта, и при отправке символа **s** происходит процесс инициализации отправки данных с аналогового входа A0 в сервис data.sparkfun.com (см. рис. 8.9).

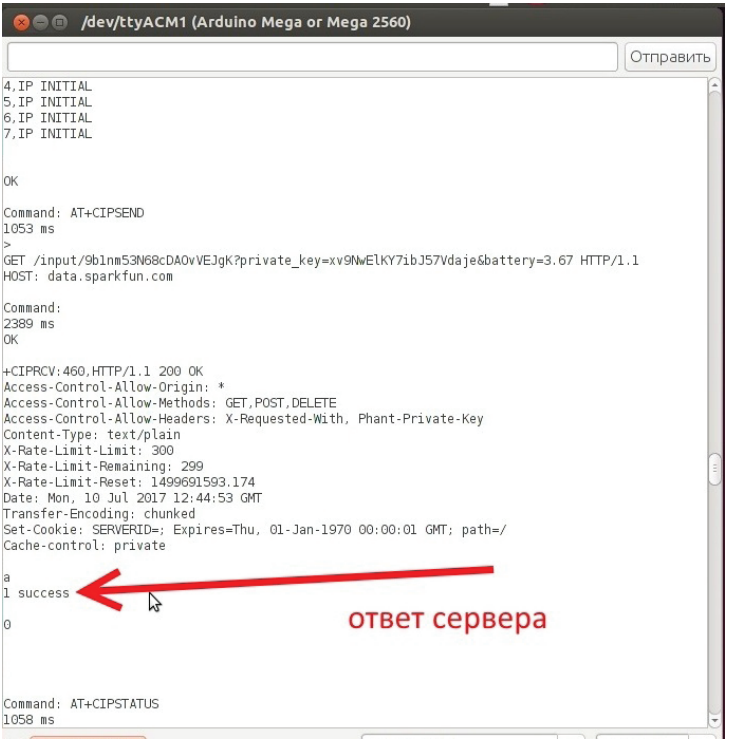


Рис. 8.9. Отображение в мониторе процесса отправки данных в сервис data.sparkfun.com

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

8.4. Подключение модуля NodeMCU к сети Интернет по Wi-Fi

Рассмотрим подключение платы NodeMCU к сети Интернет. Для этого необходимо подключиться по Wi-Fi к точке доступа, имеющей выход в Интернет. Загрузим в плату NodeMCU скетч подключения к Wi-Fi точке доступа для отправки данных в сервис data.sparkfun.com (листинг 8.3), изменим данные на свои для подключения к точке доступа и в сервисе data.sparkfun.com:

```
const char* ssid     = "my_point";
const char* password = " my_point pass";

const char* host = "data.sparkfun.com";
const char* streamId   = "----------------------";
const char* privateKey = "----------------------";
```

Листинг 8.3

```
#include <ESP8266WiFi.h>

const char* ssid     = "my_point ";
const char* password = "my_point pass ";

const char* host = "data.sparkfun.com";
const char* streamId   = "----------------------";
const char* privateKey = "----------------------";

void setup() {
  Serial.begin(115200);
  delay(10000);

  // подсоединение к Wi-Fi точке доступа
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
```

```
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

float value = 0;

void loop() {
  delay(10000);

  Serial.print("connecting to ");
  Serial.println(host);

  // создаем TCP-соединение
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }

  // создаем строку URL
  String url = "/input/";
  url += streamId;
  url += "?private_key=";
  url += privateKey;
  url += "&battery=";
  value=3.3*analogRead(A0)/1024;
  url += value;

  Serial.print("Requesting URL: ");
  Serial.println(url);

  // отправить данные на сервер
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  delay(10);

  // ответ сервера
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }

  Serial.println();
  Serial.println("closing connection");
}
```

После загрузки скетча на плату запускаем монитор последовательного порта и наблюдаем процесс отправки данных с аналогового входа A0 в сервис data.sparkfun.com (см. рис. 8.10) каждые 10 секунд.

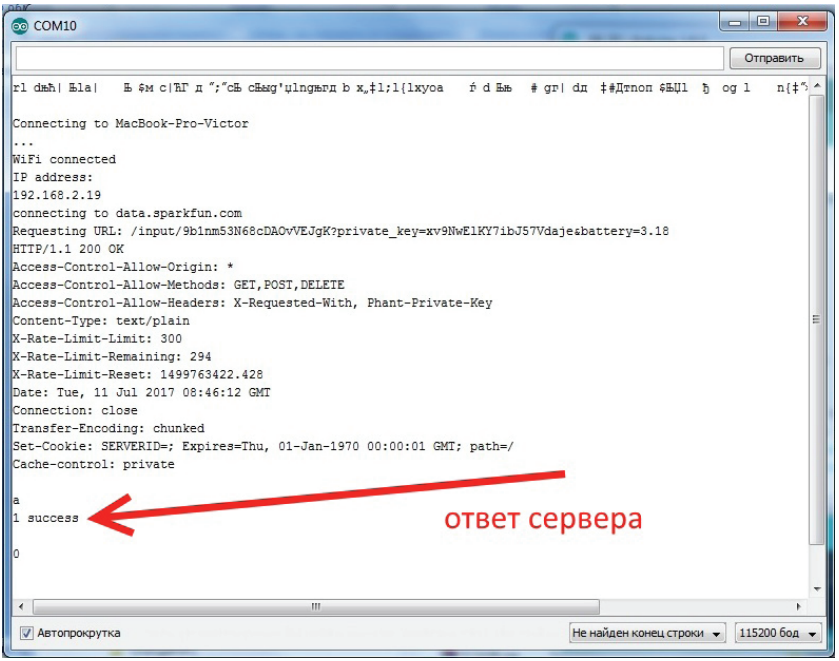


Рис. 8.10. Отображение в мониторе процесса отправки данных в сервис data.sparkfun.com

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

9 ПРОТОКОЛ MQTT – ПРОСТОЙ ПРОТОКОЛ ДЛЯ ИНТЕРНЕТА ВЕЩЕЙ

Наконец, мы готовы к тому, чтобы устройства нашего «умного» дома стали устройствами интернета вещей, что позволит получать данные с датчиков и управлять исполнительными устройствами нашего «умного дома» через Интернет из любой точки мира. В качестве устройства управления удобнее всего использовать мобильный телефон. Нас интересуют получение данных на телефон и управление исполнительными устройствами с телефона. То есть мобильный телефон выступает в качестве еще одного устройства IoT, совмещающего в себе табло для отображения данных с датчиков и пульт для управления исполнительными устройствами.

Устройства в сетях IoT взаимодействуют друг с другом по средствам различных интерфейсов и протоколов передачи данных. Рассмотрим наиболее популярный и одновременно простой протокол обмена для устройств интернета вещей – MQTT (Message Queue Telemetry Transport).

Основные особенности протокола MQTT:

- асинхронный протокол;
- компактные сообщения;
- работа в условиях нестабильной связи на линии передачи данных;
- поддержка нескольких уровней качества обслуживания (QoS);
- легкая интеграция новых устройств.

Обмен сообщениями в протоколе MQTT осуществляется между клиентом (client), который может быть издателем или подписчиком (publisher/subscriber) сообщений и брокером (broker) сообщений (например, Mosquitto MQTT). Издатель и подписчик не передают друг другу сообщений напрямую, не устанавливают прямого контакта, могут не знать о существовании друг друга. Издатель отправляет данные на MQTT брокера, указывая в сообщении определенную тему, топик (topic). Подписчики могут получать разные данные от множества издателей в зависимости от подписки на соответствующие топики.

Топики представляют собой символы с кодировкой UTF-8. Иерархическая структура топиков имеет формат «дерева», что упрощает их организацию и доступ к данным. Топики состоят из одного или нескольких уровней, которые разделены между собой символом «/».

```
/home/living /living-room1/temperature
```

Устройства MQTT используют определенные типы сообщений для взаимодействия с брокером, ниже представлены основные:

- Connect – установить соединение с брокером;
- Disconnect – разорвать соединение с брокером;
- Publish – опубликовать данные в топик на брокере;
- Subscribe – подписаться на топик на брокере;
- Unsubscribe – отписаться от топика.

Схема простого взаимодействия между подписчиком, издателем и брокером показана на рис. 9.1.

Рис. 9.1. Схема простого взаимодействия по протоколу MQTT

9.1. IoT Manager

IoT Manager – это мобильное приложение для телефонов и планшетов, совмещающее в себе табло для отображения данных с датчиков и пульт для управления исполнительными устройствами. Существуют версии для Android и iOS, которые можно скачать в GooglePlay и AppStore. Но прежде, чем скачивать приложение, определимся с брокером. В качестве брокеров выбираем сервис CloudMQTT.com (https://www.cloudmqtt.com/), в котором можно создать бесплатный аккаунт (по ссылке Control Panel). Для регистрации необходимо ввести адрес электронной почты (в качестве логина) и пароль (см. рис. 9.2).

Сразу попадаем в панель управления и создаем брокер (нажатие по кнопке **+Create**). Вводим название, выбираем датацентр (Европа или США), тарифный план – бесплатный **Cute Cat** и сохраняем (рис. 9.3). Можно создать несколько брокеров.

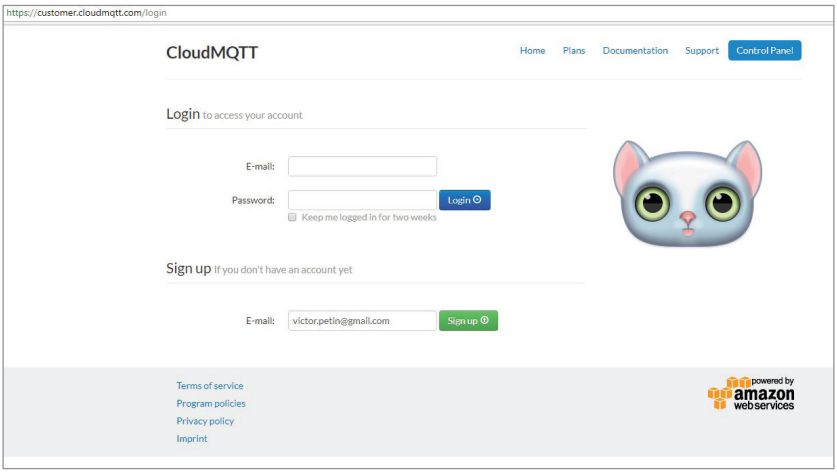


Рис. 9.2. Регистрация в сервисе CloudMQTT.com

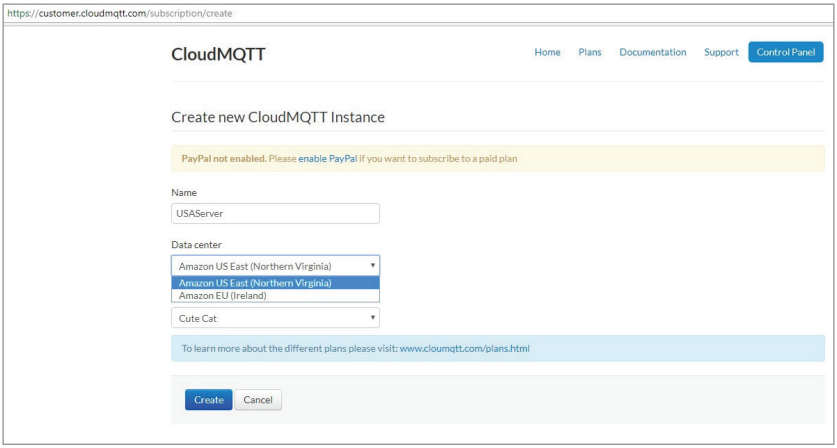


Рис. 9.3. Регистрация брокера в сервисе CloudMQTT.com

Теперь нажимаем на кнопку **Details** (см. рис. 9.4).

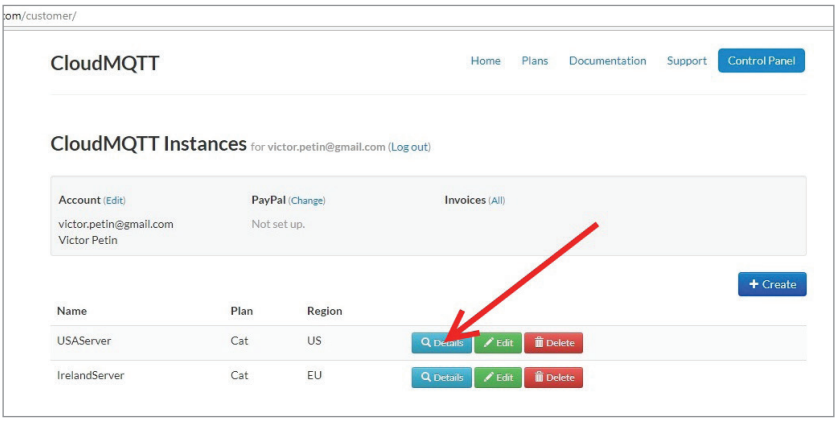


Рис. 9.4. Переход к настройкам выбранного брокера

Попадаем в настройки (рис. 9.5). Нам необходимы следующие настройки:

- имя хоста **m13.cloudmqtt.com**;
- порт **18274** (для скетча Arduino IDE);
- WebSockets порт **38274** (для мобильного приложения).

Здесь же находится менеджер пользователей, где можно создать пользователей для доступа к данным брокера и назначить им права (Read, Write). В поле **Topic** вводим # (ко всем топикам) (см. рис. 9.6).

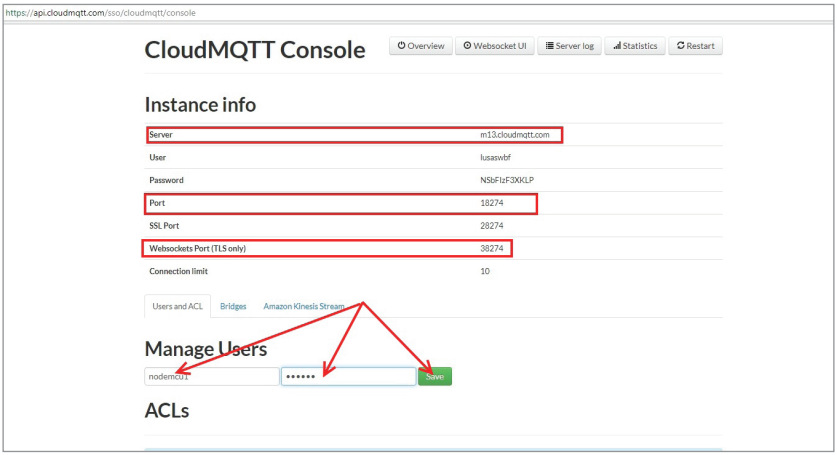


Рис. 9.5. Настройки выбранного брокера

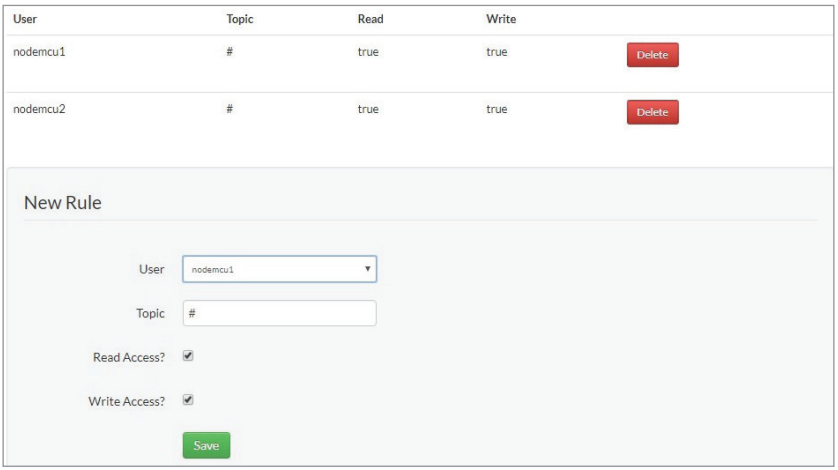


Рис. 9.6. Создание пользователей для доступа к брокеру

Теперь можно скачать и установить мобильное приложение IoT Manager. Запускаем. Необходимо произвести настройку. Нажимаем на **Settings** (рис. 9.7) и в появившейся форме вносим данные своего брокера (рис. 9.8):

- MQTT hostname – **m13.cloudmqtt.com**;
- MQTT Websocket port – **38274**;
- MQTT username – **nodemcu1**;
- MQTT password.

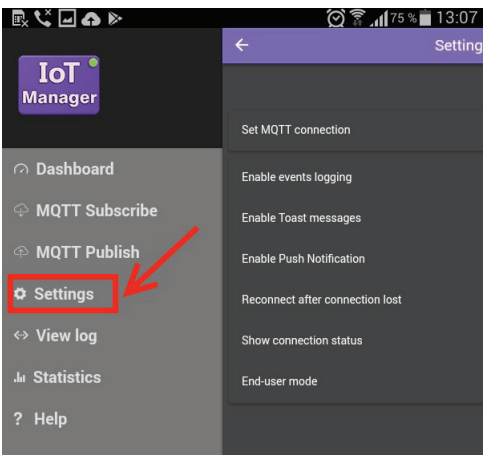


Рис. 9.7. Основное меню приложения IoT Manager

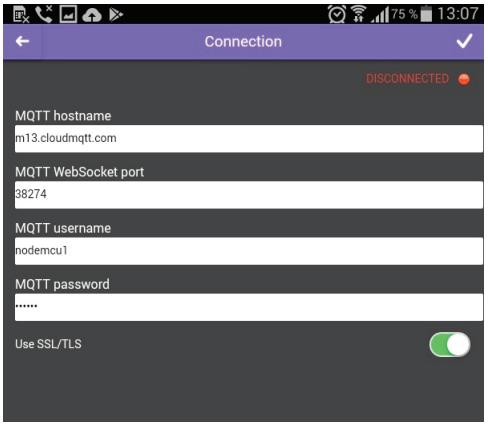


Рис. 9.8. Окно настроек брокера

Теперь выходим на страницу **Dashboard**, тут мы должны увидеть установленное соединение (рис. 9.9). Надпись **No data** не должна вас смущать – данные в топики еще не передавались.

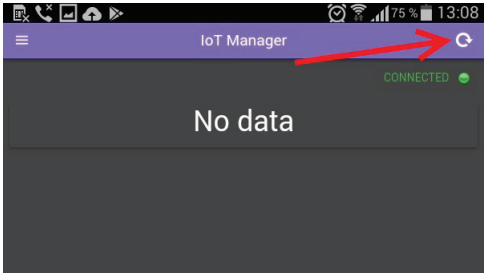


Рис. 9.9. Успешное подключение к брокеру

9.2. Передача данных брокеру (тестовый пример)

Для проверки работы брокера загрузим на плату NodeMCU тестовый пример, который можно скачать со страницы https://gist.github.com/4refr0nt/7d0ac08a5e530957b311. В скетч необходимо внести изменения – свои данные для точки доступа Wi-Fi, а также данные своего брокера:

```
const char *ssid =  "MacBook-Pro-Victor";
const char *pass =  "*******";
…..
String mqttServerName = "m13.cloudmqtt.com";
int    mqttport = 18274;
String mqttuser =  "nodemcu1";
String mqttpass =  "******";
```

Также необходимо установить в Arduino IDE библиотеку PubSubClient, скачать которую можно на сайте https://arduino-kit.ru/iotprog. Загружаем скетч на плату NodeMCU и открываем монитор последовательного порта, где видим отправку данных брокеру с платы (рис. 9.10).

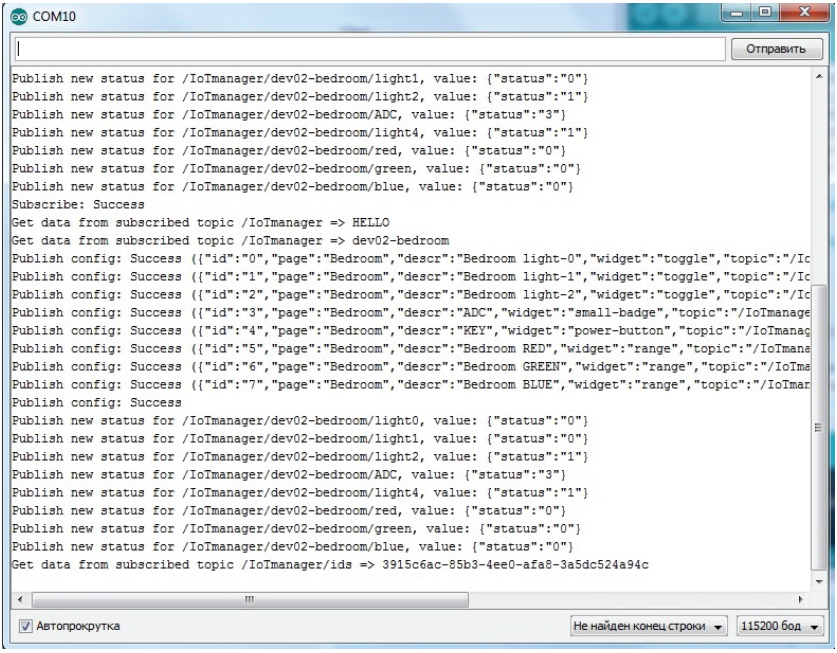


Рис. 9.10. Отправка и получение данных с платы NodeMCU

Как только началась отправка данных с платы NodeMCU брокеру, в мобильном приложении появятся эти данные (рис. 9.11).

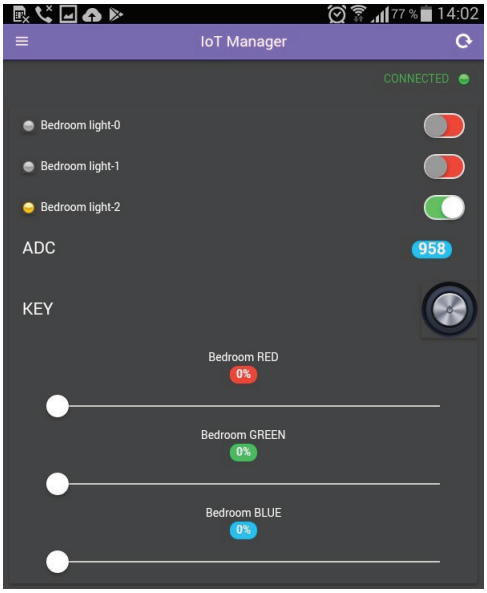


Рис. 9.11. Отправка и получение данных с платы NodeMCU

Мы можем также посмотреть и список тем, на которые подписано мобильное устройство (рис. 9.12).

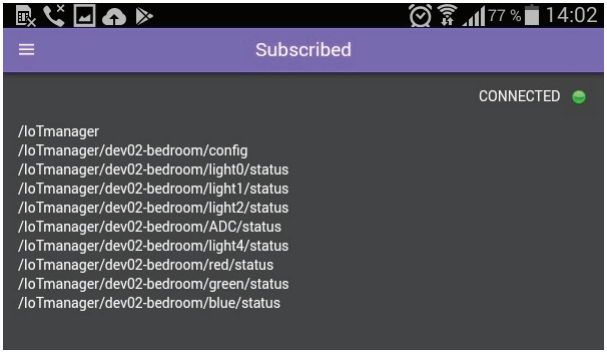


Рис. 9.12. Список тем (subscribe),
на которые подписан IoT Manager

Однако IoT Manager не только подписан на темы, но и выступает в роли publisher – публикует данные в темы. Это значения слайдеров и статус кнопки. Эти данные плата NodeMCU, подпи-

санная в качестве subscriber на эти темы, может использовать для управления подключенными к плате устройствами.

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

9.3. Публикация данных датчиков в темы брокера (для NodeMCU)

Рассмотрим подоробнее отправку данных с датчиков нашего умного дома брокеру. Будем отправлять брокеру данные с двух датчиков – DHT22 и DS18B20. Правки осуществляем в скетче из предыдущего раздела. Устанавливаем количество виджетов для отображения по количеству датчиков:

```
const int nWidgets = 2;
String stat        [nWidgets];
String sTopic      [nWidgets];
String color       [nWidgets];
String style       [nWidgets];
String badge       [nWidgets];
String widget      [nWidgets];
String descr       [nWidgets];
String page        [nWidgets];
String thing_config[nWidgets];
String id          [nWidgets];
int    pin         [nWidgets];
float  defaultVal  [nWidgets];
bool   inverted    [nWidgets];
```

В процедуре initVar() прописываем настройки для виджетов, параметр page[] (вкладка (страница) в Iot Manager для отображения виджетов) устанавливаем Sensors NodeMCU, параметр pin[] нам не нужен, т. к. данные датчиков мы будем получать программно, а не с пинов, тип для defaultVal[] назначаем float.

```
 id    [0] = «0»;
page  [0] = "SensorsNodemcu";
descr [0] = "DHT22";
widget[0] = "small-badge";
//pin [0] = A0;
sTopic[0] = prefix + "/" + deviceID + "/DHT22";
badge [0] = "\"badge\":\"badge-calm\"";
style [0]   = "\"style\":\"font-size:150%;\"";

id    [1] = "1";
page  [1] = " SensorsNodemcu ";
descr [1] = "DS18b20";
widget[1] = "small-badge";
```

```
  //pin [1] = A0;
  sTopic[1] = prefix + "/" + deviceID + "/DS18b20";
  badge [1] = "\"badge\":\"badge-calm\"";
  style [1]   = "\"style\":\"font-size:150%;\"";
```

Параметры конфигурации отображения датчиков, которые необходимо направить брокеру:

```
  thing_config[0] = "{\"id\":\"" + id[0] + "\",\"page\":\"" + page[0]+"\",\"-
descr\":\"" + descr[0] + "\",\"widget\":\"" + widget[0] + "\",\"topic\":\"" +
sTopic[0] + "\"," + badge[0] + "," + style[0] + "}";  // DHT11
  thing_config[1] = "{\"id\":\"" + id[1] + "\",\"page\":\"" + page[1]+"\",\"-
descr\":\"" + descr[1] + "\",\"widget\":\"" + widget[1] + "\",\"topic\":\"" +
sTopic[1] + "\"," + badge[1] + "," + style[1] + "}";  // ds18b20
```

Вносим изменения в функцию callback:

```
void callback(const MQTT::Publish& sub) {
  Serial.print("Get data from subscribed topic ");
  Serial.print(sub.topic());
  Serial.print(" => ");
  Serial.println(sub.payload_string());

   if (sub.topic() == sTopic[0] + "/control") {
   // DHT22
    } else if (sub.topic() == sTopic[1] + "/control") {
   // DS18B20 display only
    } else if (sub.topic() == prefix) {
      if (sub.payload_string() == "HELLO") {
        pubConfig();
      }
    }
}
```

Данные отправляем каждые 10 сек:

```
    if (client.connected()) {
      newtime = millis();
      if (newtime - oldtime > 10000) { // 10 sec
        float x = get_data_humidity();
        val = "{\"status\":\"" + String(x)+ "\"}";
        client.publish(sTopic[0] + "/status", val ); // widget 0
        x = get_data_ds18b20();
        val = "{\"status\":\"" + String(x)+ "\"}";
        client.publish(sTopic[1] + "/status", val ); // widget 1
        oldtime = newtime;
      }
      client.loop();
    }
```

Функции получения данных с датчиков DHT22 – get_data_humidity():

```
float get_data_humidity() {
   float h = dht.readHumidity();
   return h;
}
```

И датчика температуры DS18B20 – get_data_ds18b20():

```
float get_data_ds18b20(void) {
  byte i;
  byte present = 0;
  byte type_s;
  byte data[12];
  byte addr[8];
  float fTemp;

  if ( !ds.search(addr)) {
    Serial.println("No more addresses.");
    Serial.println();
    ds.reset_search();
    delay(250);
    return 999;
  }

  Serial.print("ROM =");
  for( i = 0; i < 8; i++) {
    Serial.write(‘ ‘);
    Serial.print(addr[i], HEX);
  }

  if (OneWire::crc8(addr, 7) != addr[7]) {
      Serial.println("CRC is not valid!");
      return 999;
  }
  Serial.println();

  switch (addr[0]) {
    case 0x10:
      Serial.println("  Chip = DS18S20");  // or old DS1820
      type_s = 1;
      break;
    case 0x28:
      Serial.println("  Chip = DS18B20");
      type_s = 0;
      break;
    case 0x22:
      Serial.println("  Chip = DS1822");
      type_s = 0;
      break;
```

```
    default:
      Serial.println("Device is not a DS18x20 family device.");
      return 999;
  }

  ds.reset();
  ds.select(addr);
  // запустить конвертацию температуры датчиком
  ds.write(0x44, 1);
  delay(1000);     // ждем 750 мс
  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);
  // считываем ОЗУ датчика
  for ( i = 0; i < 9; i++) {
    data[i] = ds.read();
    Serial.print(data[i], HEX);
    Serial.print(" ");
  }
  // перевод полученных данных в значение температуры
  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3;
    if (data[7] == 0x10) {
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    if (cfg == 0x00) raw = raw & ~7;
    else if (cfg == 0x20) raw = raw & ~3;
    else if (cfg == 0x40) raw = raw & ~1;
  }
  fTemp = (float)raw / 16.0;

  return fTemp;
}
```

Скачиваем полный скетч на сайте https://arduino-kit.ru/iotprog. Загружаем его на плату NodeMCU, открываем монитор последовательного порта и видим отправку данных с датчиков (рис. 9.13).

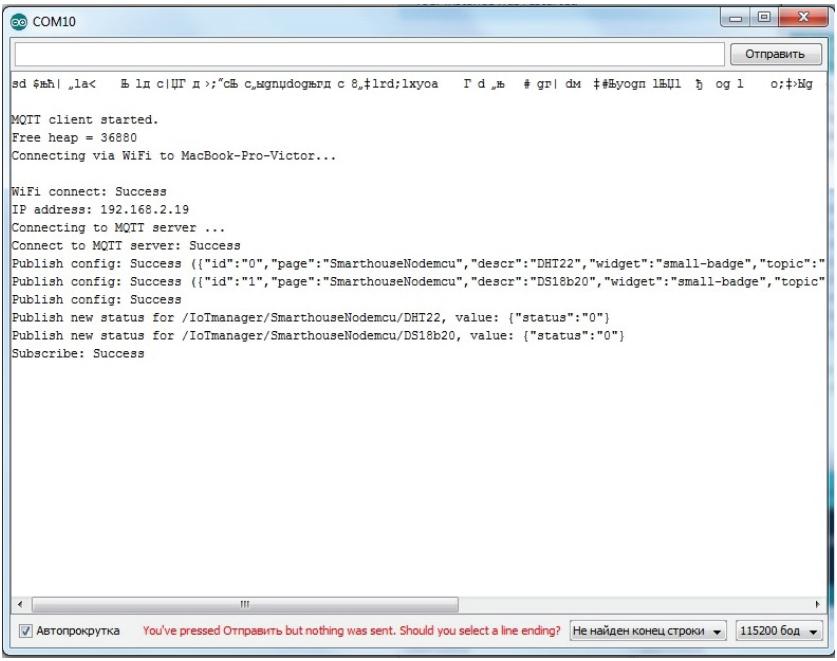


Рис. 9.13. Лог отправки в мониторе последовательного порта Arduino IDE

И смотрим данные на смартфоне в приложении IoT Manager (рис. 9.14).

Рис. 9.14. Отображение данных на смартфоне в приложении IoT Manager

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog. Вы можете самостоятельно добавить отправку данных с датчиков увлажненности почвы, уровня воды, огня, с датчиков газа.

9.4. Управление из IoT Manager исполнительными устройствами на плате NodeMCU

В данном разделе рассмотрим управление исполнительными устройствами, подключенными к NodeMCU, из мобильного приложения IoT Manager. В скетч для NodeMCU необходимо внести следующие изменения. Изменяем количество виджетов для отображения (увеличение на количество исполнительных устройств):

```
const int nWidgets = 5;
```

В процедуре initVar() прописываем настройки для виджетов управления исполнительными устройствами. Нам необходим виджет toggle. Параметру page[] присваиваем значение RelaysNodemcu, что позволит на экране мобильного приложения IoT Manager разделять датчики и исполнительные устройства по разным вкладкам.

```
  id    [2] = "2";
  page  [2] = "RelaysNodeMcu";
  descr [2] = "Light";
  widget[2] = "toggle";
  //pin[0] = 4;
  defaultVal[2] = 0;
  inverted[2] = false;
  sTopic[2]   = prefix + "/" + deviceID + "/light";
  color[2]   = "\"color\":\"red\"";
  id    [3] = "3";
  page  [3] = "RelaysNodeMcu";
  descr [3] = "Pump";
  widget[3] = "toggle";
  //pin[0] = 4;
  defaultVal[3] = 0;
  inverted[3] = false;
  sTopic[3]   = prefix + "/" + deviceID + "/pump";
  color[3]   = "\"color\":\"red\"";

  id    [4] = "4";
  page  [4] = "RelaysNodeMcu";
  descr [4] = "Fun";
  widget[4] = "toggle";
  //pin[0] = 4;
  defaultVal[4] = 0;
  inverted[4] = false;
  sTopic[4]   = prefix + "/" + deviceID + "/fun";
  color[4]   = "\"color\":\"red
```

Добавляем параметры конфигурации отображения виджетов управления исполнительными устройствами, которые необходимо направить брокеру:

```
thing_config[2] = "{\"id\":\"" + id[2] + "\",\"page\":\"" + page[2]+"\",\"de-
scr\":\"" + descr[2] + "\",\"widget\":\"" + widget[2] + "\",\"topic\":\"" +
sTopic[2] + "\"," + color[2] + "}";
  thing_config[3] = "{\"id\":\"" + id[3] + "\",\"page\":\"" + page[3]+"\",\"-
descr\":\"" + descr[3] + "\",\"widget\":\"" + widget[3] + "\",\"topic\":\"" +
sTopic[3] + "\"," + color[3] + "}";
  thing_config[4] = "{\"id\":\"" + id[4] + "\",\"page\":\"" + page[4]+"\",\"-
descr\":\"" + descr[4] + "\",\"widget\":\"" + widget[4] + "\",\"topic\":\"" +
sTopic[4] + "\"," + color[4] + "}";
```

Необходимо обрабатывать получаемые из брокера сообщения об изменении статуса исполнительных устройств. Вносим изменения в функцию callback:

```
void callback(const MQTT::Publish& sub) {
  Serial.print("Get data from subscribed topic ");
  Serial.print(sub.topic());
  Serial.print(" => ");
  Serial.println(sub.payload_string());

  if (sub.topic() == sTopic[2] + "/control") {
    if (sub.payload_string() == "0") {
       newValue = 1; // inverted
       stat[2] = stat0;
    } else {
       newValue = 0; // inverted
       stat[2] = stat1;
    }
    digitalWrite(pin[2],newValue);
    pubStatus(sTopic[2], stat[2]);
    } else if (sub.topic() == sTopic[0] + "/control") {
   // ADC : nothing, display only
    } else if (sub.topic() == sTopic[1] + "/control") {
   // ADC : nothing, display only
    } else if (sub.topic() == sTopic[2] + "/control") {
    if (sub.payload_string() == "0") {
       newValue = 1; // inverted
       stat[2] = stat0;
    } else {
       newValue = 0; // inverted
       stat[2] = stat1;
    }
    datarelays[0]= newValue;
    set_status_relays();
    pubStatus(sTopic[2], stat[2]);
    } else if (sub.topic() == sTopic[3] + "/control") {
    if (sub.payload_string() == "0") {
```

```
        newValue = 1; // inverted
        stat[3] = stat0;
     } else {
        newValue = 0; // inverted
        stat[3] = stat1;
     }
     datarelays[1]= newValue;
     set_status_relays();
     pubStatus(sTopic[3], stat[3]);
     } else if (sub.topic() == sTopic[4] + "/control") {
     if (sub.payload_string() == "0") {
        newValue = 1; // inverted
        stat[4] = stat0;
     } else {
        newValue = 0; // inverted
        stat[4] = stat1;
     }
     datarelays[2]= newValue;
     set_status_relays();
     pubStatus(sTopic[4], stat[4]);
     } else if (sub.topic() == prefix) {
     if (sub.payload_string() == "HELLO") {
       pubConfig();
     }
  }
}
```

Функция set_status_relays() включает/выключает исполнительные устройства:

```
void set_status_relays() {
  int relays=0;
  for(int i=0;i<7;i++)
     relays=relays +( datarelays[i]<<i);
  // записать данные в PORT B
  Wire.beginTransmission(0x20);
  Wire.write(0x13); // address PORT B
  Wire.write(leds);    // PORT B
  Wire.endTransmission();
  delay(100); // пауза
}
```

Скачиваем полный скетч на сайте https://arduino-kit.ru/iotprog. Загружаем его на плату NodeMCU, открываем монитор последовательного порта и видим отправку данных с датчиков (рис. 9.15).

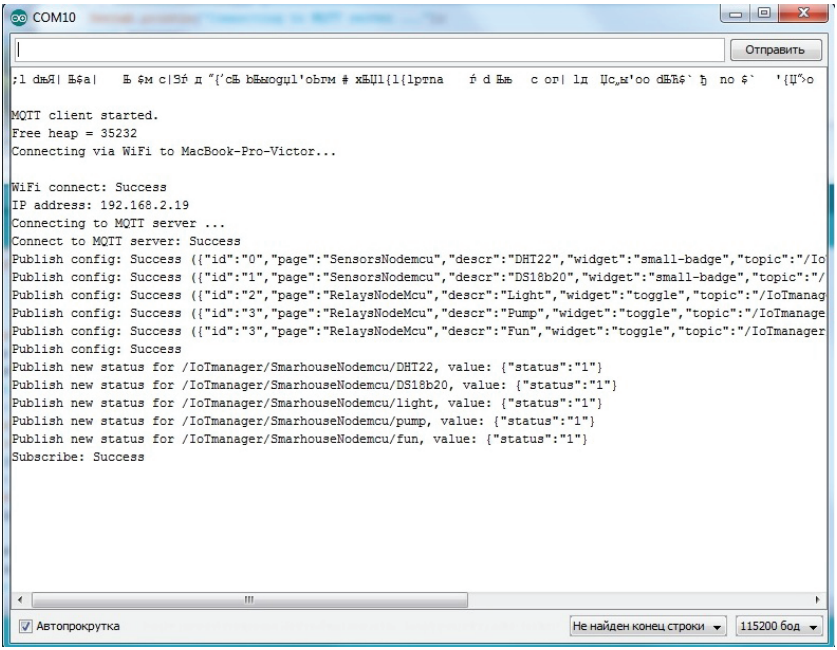


Рис. 9.15. Лог отправки в мониторе последовательного порта Arduino IDE

И смотрим данные на смартфоне в приложении IoT Manager (рис. 9.16). Виджеты получения данных и управления исполнительными устройствами расположены на разных вкладках (рис. 9.17).

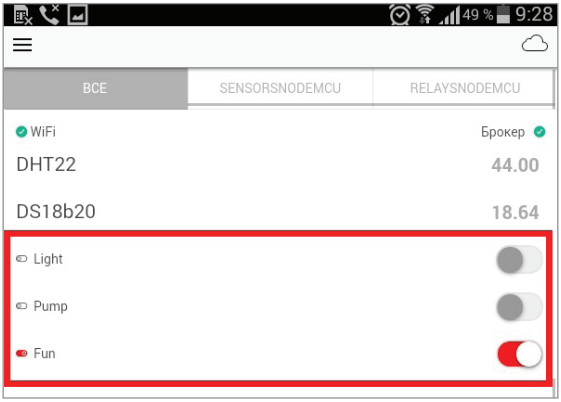


*Рис. 9.16. Отображение виджетов на смартфоне в приложении IoT Manager (вкладка **Все**)*

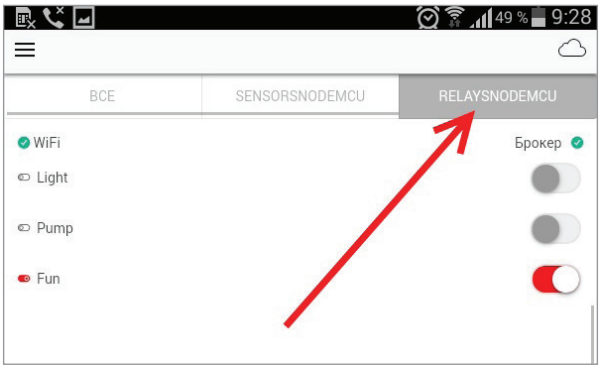


Рис. 9.17. Вкладка виджетов управления исполнительными устройствами

Изменяется состояние toggle к публикации измененных данных в тему /IoTmanager/SmarhouseNodemcu/xxx, и плата NodeMCU, получая эти данные (см. рис. 9.18), изменяет состояние соответствующего исполнительного устройства.

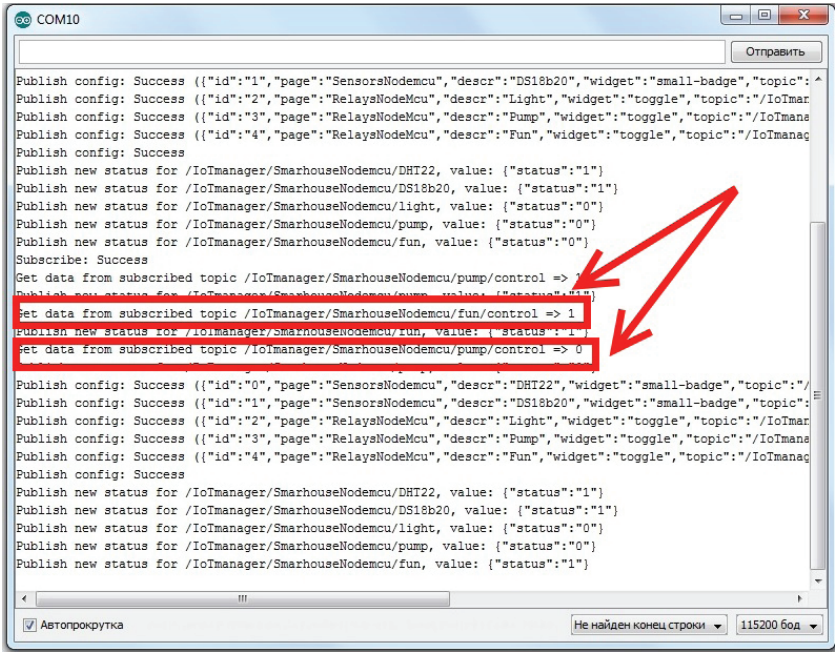


Рис. 9.18. Получение данных платой NodeMCU

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog.

9.5. Публикация данных датчиков в темы брокера (для Arduino MEGA)

Рассмотрим отправку данных брокеру Cloudmqtt с датчика DHT22, подключенного к контроллеру умного дома на Arduino MEGA. При написании скетча отправки будем использовать дополнительные библиотеки:

```
#include <Time.h>
#include <sim800Client.h>
#include <PubSubClientHotlog.h>
#include <TimeAlarms.h>
```

В константах поменяйте данные подключения сети для вашего сотового оператора, например для МТС:

```
#define GSMAPN "internet.mts.ru" //APN
#define GSMUSER "mts"            //APN USER
#define GSMPASSWORD "mts"        //APN PASSWORD
```

Устанавливаем количество виджетов, равное 1, для отображения данных с одного датчика:

```
const int nWidgets = 1;
String stat        [nWidgets];
String sTopic      [nWidgets];
String color       [nWidgets];
String style       [nWidgets];
String badge       [nWidgets];
String widget      [nWidgets];
String descr       [nWidgets];
String page        [nWidgets];
String thing_config[nWidgets];
String id          [nWidgets];
int    pin         [nWidgets];
float    defaultVal  [nWidgets];
bool   inverted    [nWidgets];
```

В процедуре initVar() прописываем настройки для виджета, параметр page[] (вкладка (страница) в Iot Manager для отображения виджетов) устанавливаем SensorsArduinoMega, тип для defaultVal[] назначаем float.

```
  id    [0] = "0";
  page  [0] = « SensorsArduinoMega «;
```

```
  descr [0] = "DHT22";
  widget[0] = "small-badge";
  sTopic[0] = prefix + "/" + deviceID + "/DHT22";
  badge [0] = "\"badge\":\"badge-calm\"";
  style [0] = "\"style\":\"font-size:150%;\"";
```

Параметры конфигурации отображения датчиков, которые необходимо направить брокеру:

```
  thing_config[0] = "{\"id\":\"" + id[0] + "\",\"page\":\"" + page[0]+"\",\"-
descr\":\"" + descr[0] + "\",\"widget\":\"" + widget[0] + "\",\"topic\":\"" +
sTopic[0] + "\"," + badge[0] + "," + style[0] + "}";  // DHT22
```

Вносим изменения в функцию callback:

```
void callback(const MQTT::Publish& sub) {
  Serial.print("Get data from subscribed topic ");
  Serial.print(sub.topic());
  Serial.print(" => ");
  Serial.println(sub.payload_string());

   if (sub.topic() == sTopic[0] + "/control") {
   // DHT22
    } else if (sub.topic() == prefix) {
      if (sub.payload_string() == "HELLO") {
        pubConfig();
      }
    }
}
```

Данные отправляем каждые 10 сек:

```
Alarm.timerRepeat(5, pub);              // timer
void loop() {
    client.loop();
    Alarm.delay(10000);
}
```

Каждый раз отправляем данные конфигурации в тему и данные значения в тему:

```
void pub()
{
   char charBufVar1[50];
   char charBufVar2[20];
   sTopic[0].toCharArray(charBufVar1, 50);
   Serial.println("Publish data");
   pubConfig();
   float x = get_data_humidity();
   String val = "{\"status\":\"" + String(x)+ "\"}";
```

```
   val.toCharArray(charBufVar2, 50);
   client.publish(charBufVar1,charBufVar2);  // widget 0
 }
void pubConfig() {
   char charBufVar1[50];
   char charBufVar2[100];
   Serial.println("Publish config");
   String s1=prefix + "/" + deviceID + "/config";
   s1.toCharArray(charBufVar1, 50);
   thing_config[0].toCharArray(charBufVar2, 50);
   client.publish(charBufVar1,charBufVar2);  // config
}
```

Функции получения данных с датчиков DHT22 – get_data_humidity():

```
float get_data_humidity() {
   float h = dht.readHumidity();
   return h;
}
```

Скачиваем полный скетч на сайте https://arduino-kit.ru/iotprog. Загружаем его на плату Arduino MEGA, открываем монитор последовательного порта и видим отправку данных с датчиков (рис. 9.19).

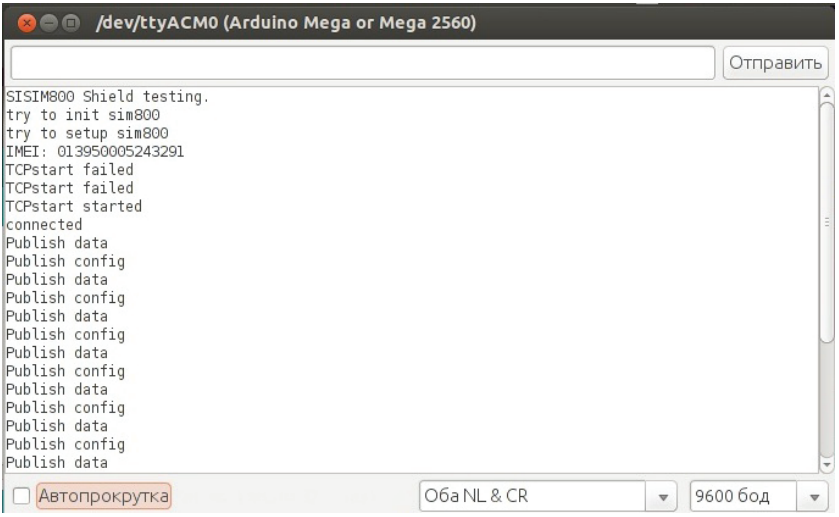


Рис. 9.19. Лог отправки в мониторе последовательного порта Arduino IDE

И смотрим данные на смартфоне в приложении IoT Manager (рис. 9.20).

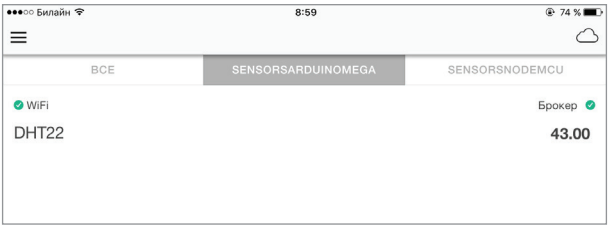


Рис. 9.20. Отображение данных на смартфоне в приложении IoT Manager

Скачать данный скетч можно на сайте https://arduino-kit.ru/iotprog. Вы можете самостоятельно добавить отправку данных с других датчиков, а также получение данных из IoT Manager для управления исполнительными устройствами.

ЗАКЛЮЧЕНИЕ

С этой книгой мы полностью прошли путь создания из нашего набора полноценного «умного» дома, а также превращения «умного» дома в устройство IoT (интернета вещей). На сайте https://arduino-kit.ru/iotprog вы можете скачать все представленные в книге скетчи и необходимые библиотеки. Материалы данной книги не являются догмой, а скорее руководством к действию, и вы можете самостоятельно изменять конфигурацию вашего умного дома, подстраивая его под свое видение и потребности. Желаем вам успехов в этом пусть и нелегком, но интересном деле.

Петин Виктор Александрович

Создание умного дома на базе Arduino

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Корректор *Синяева Г. И.*
Верстка *Чаннова А. А.*
Дизайн обложки *Мовчан А. Г.*

Формат 60×90 1/16.
Гарнитура «PT Serif». Печать офсетная.
Усл. печ. л. ***. Тираж 300 экз.

Веб-сайт издательства: www.дмк.рф

Петин В. А.

СОЗДАНИЕ УМНОГО ДОМА НА БАЗЕ ARDUINO

С появлением интернета вещей отношения умного дома с владельцем переходят на новый уровень – теперь контроллер, управляющий жилищем, может в любой момент связаться с хозяином и получить от него новое задание. Специальное приложение для Android или iOS позволит вам управлять своим домом с экрана смартфона из соседней комнаты или с другого континента. Взаимодействовать с техникой будущего и разрабатывать новые способы применения интернета вещей научит вас эта книга – в ней есть всё, что нужно для творчества. Издание познакомит вас с основами создания и отладки проектов по автоматизации дома на основе контроллеров Arduino и NodeMCU.

Вас ждет новый мир – мир разумных устройств и бесконечных возможностей новой глобальной сети!

ISBN 978-5-97060-620-9

9 785970 606209 >

ДМК ПРЕСС ИЗДАТЕЛЬСТВО
www.дмк.рф

mbitech
www.mbitech.ru