Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

Многопрофильный колледж

УТВЕРЖДАЮ
Директор
_____ / С.А. Махновский
«09» февраля 2022 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

по учебной дисциплине
ОП.16 Интернет вещей

для обучающихся специальности

09.02.01 Компьютерные системы и комплексы
(базовой подготовки)

Магнитогорск, 2022

ОДОБРЕНО
Предметно-цикловой комиссией
«Информатики и вычислительной техники»
Председатель И.Г. Зорина
Протокол № 5 от 19 января 2022 г.

Методической комиссией МпК
Протокол №4 от «09» февраля 2022г

Разработчик:
преподаватель ФГБОУ ВО «МГТУ им. Г.И.Носова» МпК
Татьяна Борисовна Ремез

Методические указания по выполнению практических работ разработаны на основе рабочей программы учебной дисциплины «Интернет вещей».

Содержание практических работ ориентировано на подготовку студентов к освоению программы подготовки специалистов среднего звена по специальности 09.02.01 Компьютерные системы и комплексы и овладению профессиональными компетенциями.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

Состав и содержание практических занятий направлены на реализацию Федерального государственного образовательного стандарта среднего профессионального образования

Ведущей дидактической целью практических занятий является формирование профессиональных практических умений (умений выполнять определенные действия, операции, необходимые в последующем в профессиональной деятельности) или учебных практических умений, необходимых в последующей учебной деятельности.

В соответствии с рабочей программой учебной дисциплины «Интернет вещей» предусмотрено проведение практических работ.

В результате их выполнения, обучающийся должен:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем
- производить тестирование и отладку микропроцессорных систем (МПС)
- выбирать микроконтроллер/микропроцессор для конкретной системы управления

Содержание лабораторных работ ориентировано на подготовку студентов к освоению программы подготовки специалистов среднего звена по специальности и овладению ***профессиональными компетенциями***:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем

ПК 2.2. Производить тестирование и отладку микропроцессорных систем.

А также формированию ***общих компетенций***:

ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

ОК 3. Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.

ОК 6. Работать в коллективе и команде, эффективно общаться с коллегами, руководством, потребителями.

ОК 7. Брать на себя ответственность за работу членов команды (подчиненных), результат выполнения заданий.

Выполнение студентами практических работ по учебной дисциплине «Прикладная электроника» направлено на:

- обобщение, систематизацию, углубление, закрепление, развитие и детализацию полученных теоретических знаний по конкретным темам учебной дисциплины;
- формирование умений применять полученные знания на практике,
- реализацию единства интеллектуальной и практической деятельности;
- развитие интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;
- выработку при решении поставленных задач профессионально значимых качеств, таких как самостоятельность, ответственность, точность, творческая инициатива.

Практические занятия проводятся после соответствующей темы, которая обеспечивает наличие знаний, необходимых для ее выполнения.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Тема 2.1. Средства виртуальной идентификации объектов внешней среды

Практическая работа № 1
Знакомство с Ардуино

Цель работы: научиться подключать и программировать устройства ввода /вывода к микроконтроллеру Ардуино

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

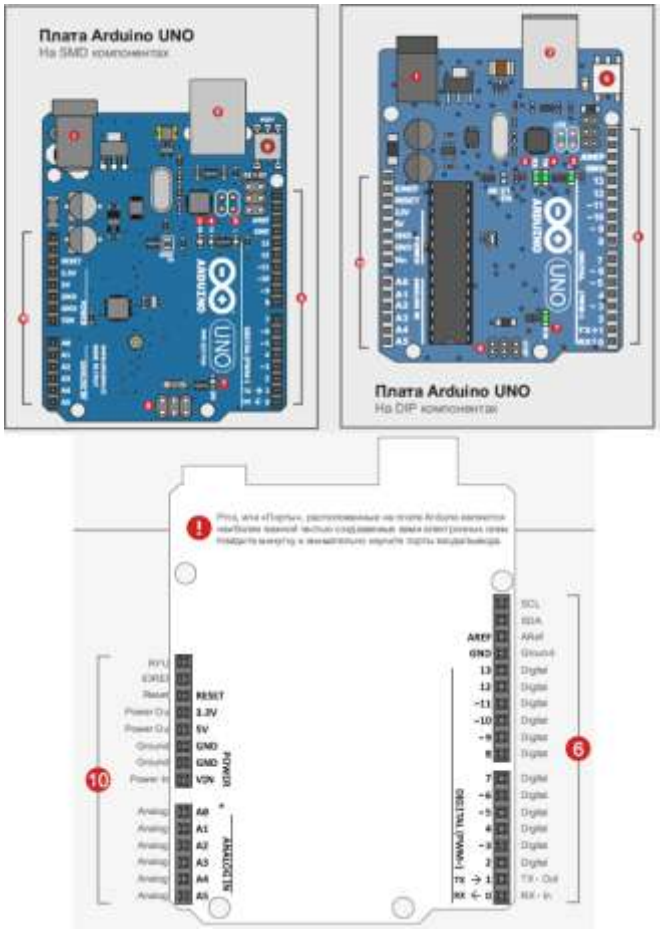
Теоретические сведения

Arduino — это электронный конструктор, который позволяет любому человеку создавать разнообразные электро-механические устройства. Ардуино состоит из программной и аппаратной части. Программная часть включает в себя среду разработки (программа для написания и отладки прошивок), множество готовых и удобных библиотек, упрощенный язык программирования. Аппаратная часть включает в себя большую линейку микроконтроллеров и готовых модулей для них. На основе Ардуино можно создавать различные устройства: от простых мигалок, метеостанций, систем автоматизации и до системы умного дома, ЧПУ станками и беспилотными летательными аппаратами.

Аппаратная часть

Arduino UNO - плата разработчиков, одна из нескольких (Mega, Nano и пр.), основана на микроконтроллере ATmega328 (рис. 1). Он популярен, главным образом, из-за обширной сети поддержки и универсальности.

У Arduino UNO есть 14 цифровых портов ввода - вывода, шесть из которых умеют выдавать ШИМ. Еще есть 6 входящих аналоговых портов. Есть генератор 16 Мгц, USB порт, разъем питания, кнопка сброса, и разъем ICSP. Программируется без использования программаторов, через USB.



1 – Разъем питания,
2 – Разъем USB,
3 – Индикатор передачи данных,
4 –Индикатор приема данных,
5 – Индикатор 13 пина,
6 – Порты,
7 – Индикатор питания,
8 – Сброс,
9- Разъем ICSP,
10 – Порты

Рисунок 1 – Внешний вид платы Ардуино

Дополнительными компонентами для конфигурирования МПС на основе МК являются: макетная плата, провода, светодиоды, резисторы, диоды, потенциометры, фоторезисторы, пьезоэлементы., датчики температуры, кнопки, ЖК-индикаторы (см. рис. 2 и 3)

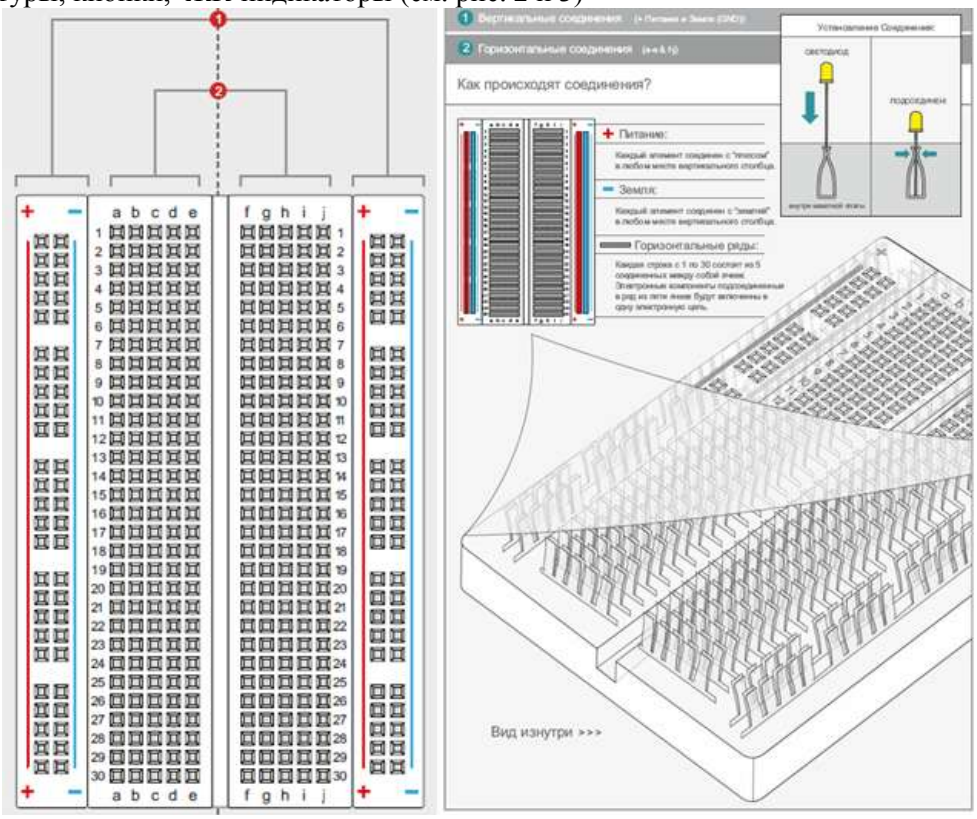


Рисунок 2 – Макетная плата

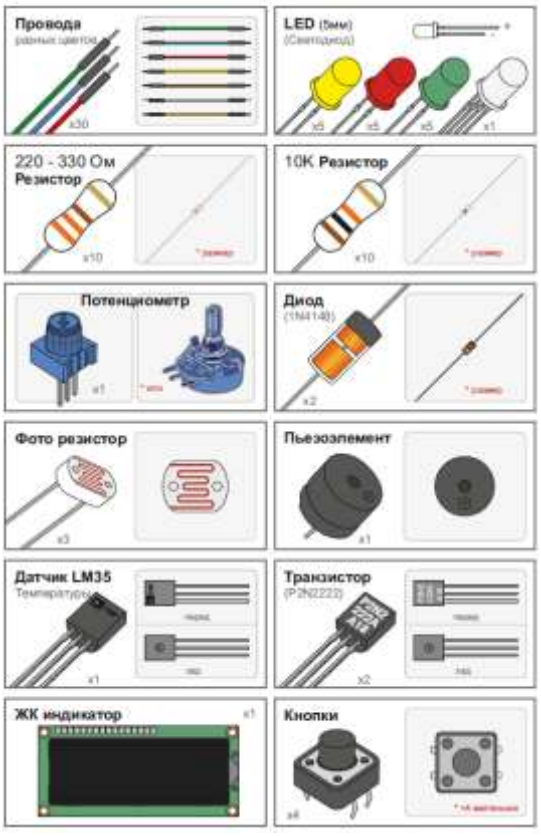


Рисунок 3 – Дополнительные компоненты

Программная часть

Arduino IDE – программное обеспечение для пользователей, позволяющее писать свои программы (скетчи) для платформы Arduino. Эта платформа в первую очередь ориентируется на конструкторов-любителей, которые применяют Arduino для построения простых систем автоматики и робототехники.

Язык программирования Arduino является стандартным C++ (используется компилятор AVR-GCC) с некоторыми особенностями, облегчающие написание программ новичкам в этом деле.

Преимущества Arduino IDE:

- доступность;
- удобный для использования и понимания интерфейс;
- программа совместима со всеми версиями операционных систем Windows;
- наличие необходимых для работы инструментов;
- несколько вариантов языков программирования;
- возможность углубить знания языка C++;
- встроенный набор примеров программ;
- функции сохранения, экспорта, проверки, поиска, замены скетчей.

Окно и меню Arduino IDE представлено на рис.4.

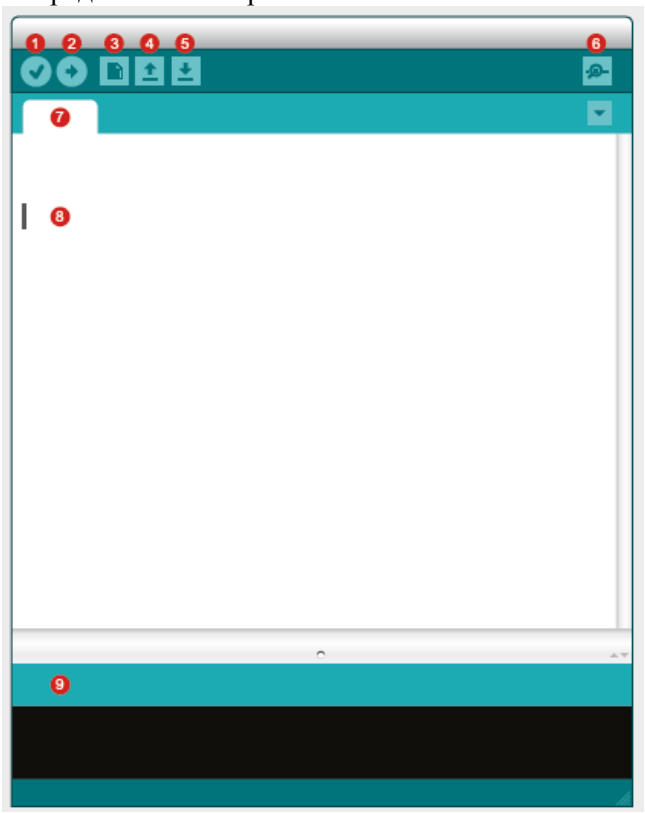


Рисунок 4 – Окно Arduino IDE

Рисунок 5 – Меню Arduino IDE

Для того, чтобы начать работу с Arduino необходимо зайти на сайт arduino.cc и скачать программу Arduino IDE из раздела Download, бесплатно. Нужно выбрать программу, подходящую под операционную систему (Windows Installer (.exe), Windows (ZIP file)).

Далее необходимо подключить плату Ардуино в ПК (рис. 6).

Рисунок 6 – Подключение Ардуино к ПК

Затем необходимо установить драйвера, соответствующие операционной системе и настроить порт (рис.7).

Рисунок 7 – Настройка порта

Практическое задание 1

Обрабатываем нажатие кнопки на примере зажигания светодиода

Это эксперимент по работе с кнопкой. Мы будем включать светодиод по нажатии кнопки и выключать по отпускании кнопки. Рассмотрим понятие дребезга и программные методы его устранения.

Необходимые компоненты:

- контроллер Arduino UNO R3;
- плата для прототипирования;
- кнопка;
- светодиод;
- резистор 220 Ом;
- резистор 10 кОм;
- провода папа-папа.

В данном эксперименте мы будем использовать контакт D2 Arduino в качестве входа. Это позволяет подключить к нему кнопку для взаимодействия с проектом в режиме реального времени. При использовании Arduino в качестве входов используют pull-up- и pulldown-резисторы, чтобы вход Arduino не находился в «подвешенном» состоянии (в этом состоянии он будет собирать внешние наводки и принимать произвольные значения), а имел заранее известное состояние (0 или 1). Резисторы pull-up подтягивают вход к питанию +5 В, pull-down-резисторы подтягивают вход к GND. Кроме этого, pull-up- и pull-down-резисторы гарантируют, что кнопка не создаст короткого замыкания между +5 В и землей при нажатии. В нашем эксперименте для подключения кнопки мы будем использовать pulldown-резистор. Схема подключения представлена на рис. 8.

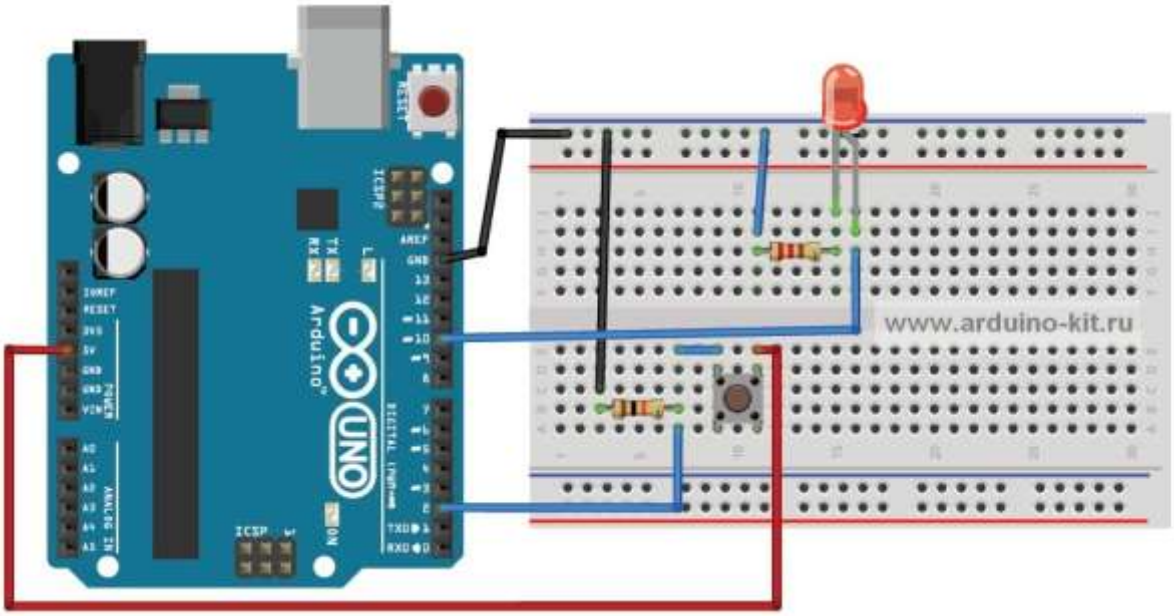


Рисунок 8.

Когда кнопка отключена, вход D2 будет подтянут к «земле» через резистор номиналом 10 кОм, который будет ограничивать поток тока, и на входном контакте будет установлено значение напряжения LOW. При нажатии на кнопку входной контакт напрямую связан с 5 В. Большая часть тока будет протекать по пути наименьшего сопротивления через замкнутую кнопку, и на входе генерируется уровень HIGH. При нажатии на кнопку включаем светодиод, при отпускании – гасим.

Код данного скетча приведен в листинге 1.1.

Порядок подключения:

1. Длинную ножку светодиода (анод) подключаем к цифровому выводу D10 Arduino, другую (катод) – через резистор 220 Ом к выводу GND (см. рис. 1).

2. Один вход кнопки подключаем к +5 В, другой – через резистор 10 кОм к GND, выход кнопки подключаем к входу D2 Arduino (см. рис. 1).

3. Загружаем в плату Arduino скетч из листинга 1.1.

4. При нажатии на кнопку светодиод должен гореть, при отпускании – затухнуть.

Практическое задание 2

Усложним задачу – будем переключать состояние светодиода (включен/выключен) при каждом нажатии кнопки. Загрузим на плату Arduino скетч из листинга 1.2.

При нажатии кнопки светодиод должен изменять свое состояние. Но это будет происходить не всегда. Виной тому – дребезг кнопок.

Кнопки представляют из себя механические устройства с системой пружинного контакта. Когда вы нажимаете на кнопку вниз, сигнал не просто меняется от низкого до высокого, он в течение нескольких миллисекунд меняет значение от одного до другого, прежде чем контакты плотно соприкоснутся и установится значение HIGH.

Микроконтроллер зафиксирует все эти нажатия, потому что дребезг неотличим от настоящего нажатия на кнопку. Устранить влияние дребезга можно программно. Алгоритм следующий:

1. Сохраняем предыдущее состояние кнопки и текущее состояние кнопки (при инициализации LOW).

2. Считываем текущее состояние кнопки.

3. Если текущее состояние кнопки отличается от предыдущего состояния кнопки, ждем 5 мс, потому что кнопка, возможно, изменила состояние.

4. После 5 мс считываем состояние кнопки и используем его в качестве текущего состояния кнопки.

5. Если предыдущее состояние кнопки было LOW, а текущее состояние кнопки HIGH, переключаем состояние светодиода.

6. Устанавливаем предыдущее состояние кнопки для текущего состояния кнопки.

7. Возврат к шагу 2. Добавляем к нашему скетчу подпрограмму устранения дребезга.

Получаем код, показанный в листинге 1.3.

Практическое задание 3

Жидкокристаллический индикатор применяется практически повсюду. Это электронные часы, табло на вокзале, в микроволновке, телевизор – есть не что иное, как жидкокристаллический дисплей.

Схема, представленная на рисунке 9, необходима для вывода на жидко кристаллический индикатор (ЖКИ) какой–нибудь информации. Потенциометр в схема служит для регулирования яркости экрана.

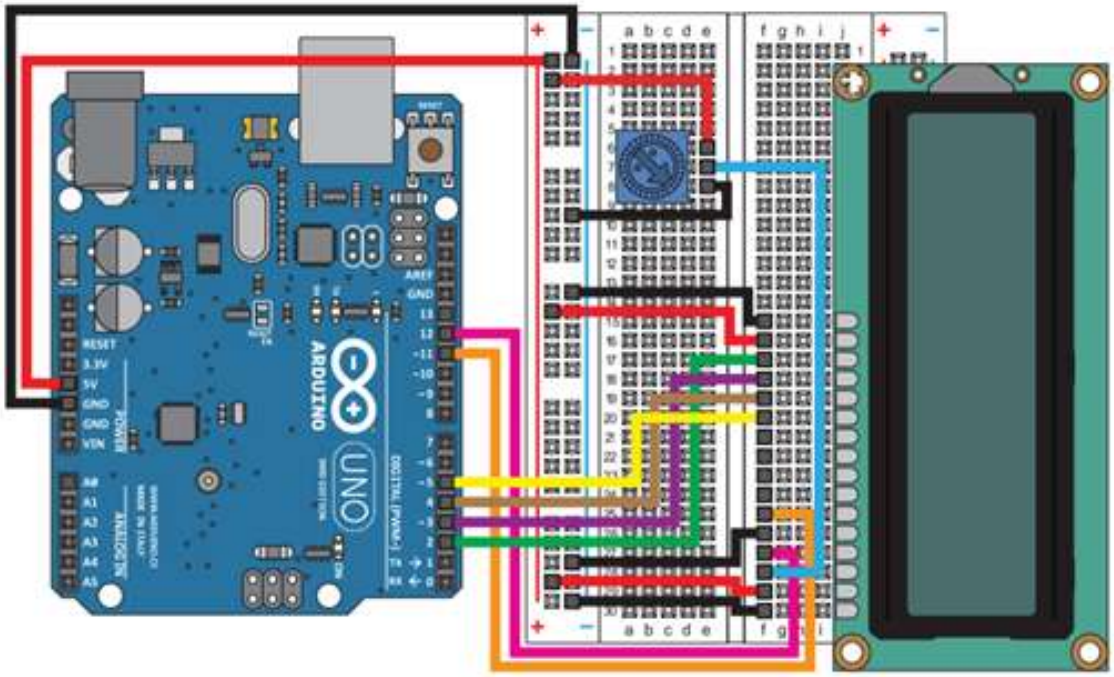


Рисунок 9 – Схема для вывода информации на ЖКИ

Первоначально на дисплее нужно отрегулировать яркость подсветки экрана с помощью потенциометра. Исходный текст программы выводит надпись «hello, world!». Изменив программу, можно вывести на экран любой текст.

Запустите среду программирования Arduino IDE. В данном задании к МПС подключается ЖКИ, поэтому перед тем как начать работу с программой, необходимо подключить библиотеку LiquidCrystal.h. Она упрощает управление различными типами жидкокристаллических индикаторов.

Откройте скетч, приведенный в листинге 1.4 из меню Файл, после того, как программа написана, необходимо дать компьютеру ее проверить, и, если всё правильно, можно переходить к следующему шагу. Кнопка «Загрузка» посылает скомпилированную программу в плату Arduino через USB шнур, после полной загрузки она сразу начнет свою работу.

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

Практическая работа № 2
Подключение и программирование считывателя RFID

Цель работы: научиться подключать и программировать устройство считывания RFID меток к микроконтроллеру Ардуино

Выполнив работу, Вы будете:

уметь:

уметь:

– составлять программы на языке ассемблера для микропроцессорных систем
– производить тестирование и отладку микропроцессорных систем (МПС)
– выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

Теоретические сведения

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, посредством которых излучается электромагнитное поле определенной частоты. Попавшие в зону действия считывающего поля радиочастотные метки «отвечают» собст-

венным сигналом, содержащим информацию (идентификационный номер товара, пользовательские данные и т. д.). Сигнал улавливается антенной считывателя, информация расшифровывается и передается в компьютер для обработки. Подавляющее большинство современных систем контроля доступа (СКД) использует в качестве средств доступа идентификаторы, работающие на частоте 125 кГц. Это проксимити-карты доступа (только чтение), самыми распространенными являются карты EM-Marin, а также HID, Indala. Карты этого стандарта являются удобным средством открывания дверей и турникетов. Но не более. Эти карты не обладают никакой защищенностью, легко копируются и подделываются и, соответственно, ничего не дают для защиты объекта от несанкционированного проникновения.

Настоящую защиту от копирования и подделки обеспечивают такие идентификаторы, в чипах которых реализована криптографическая защита. Это бесконтактные смарт-карты, работающие на частоте 13,56 МГц, наиболее распространенными из них являются карты Mifare®. В картах этих стандартов криптозащита организована на высоком уровне, и подделка таких карт практически невозможна.

Практическое задание

В этом эксперименте мы покажем, как плата Arduino получает доступ к данным RFID-карт и брелоков Mifare с помощью RFID-считывателя RC522C.

Необходимые компоненты:

• контроллер Arduino UNO R3;
• плата для прототипирования;
• RFID-считыватель RC522;
• брелок;
• карта;
• провода папа-папа.

Радиочастотная идентификация (RFID) – это технология автоматической бесконтактной идентификации объектов при помощи радиочастотного канала связи. Базовая система RFID состоит из:

• радиочастотной метки;
• считывателя информации (ридера);
• компьютера для обработки информации.

Модуль RC522 – RFID-модуль 13,56 МГц с SPI-интерфейсом (рис.10). В комплекте к модулю идут 2 RFID-метки – в виде карты и брелока.

Основные характеристики:

• основан на микросхеме MFRC522;
• напряжение питания: 3,3 В;
• потребляемый ток: 13–26 мА;
• рабочая частота: 13,56 МГц;
• дальность считывания: 0~60 мм;
• интерфейс: SPI, максимальная скорость передачи 10 МБит/с;
• размер: 40×60 мм;
• чтение и запись RFID-меток.

Рисунок 10

Схема подключения модуля к плате Arduino показана на рис. 11.

Напишем скетч считывания с карты и вывода в последовательный порт Arduino UID (уникальный идентификационный номер) RFID-метки (карты или брелока). При написании скетча будем использовать библиотеку MFRC522 (https://github.com/miguelbalboa/rfid). Содержимое скетча показано в листинге 2.1.

Порядок подключения:

1. Подключаем модули RFID-считывателя RC522 к плате Arduino по схеме на рис. 11.

2. Загружаем в плату Arduino скетч из листинга 28.1. Открываем монитор последовательного порта.

3. Подносим метку (карту или брелок) к считывателю и видим вывод в последовательный порт данных метки UID и тип (рис. 12).

Метки Mirafe позволяют записывать на них информацию. В следующем скетче мы организуем на карте счетчик, который будет инкрементироваться при поднесении карты к считывателю. В последовательный порт будем выводить показания счетчика (Рис.13). Содержимое скетча показано в листинге 2.2.

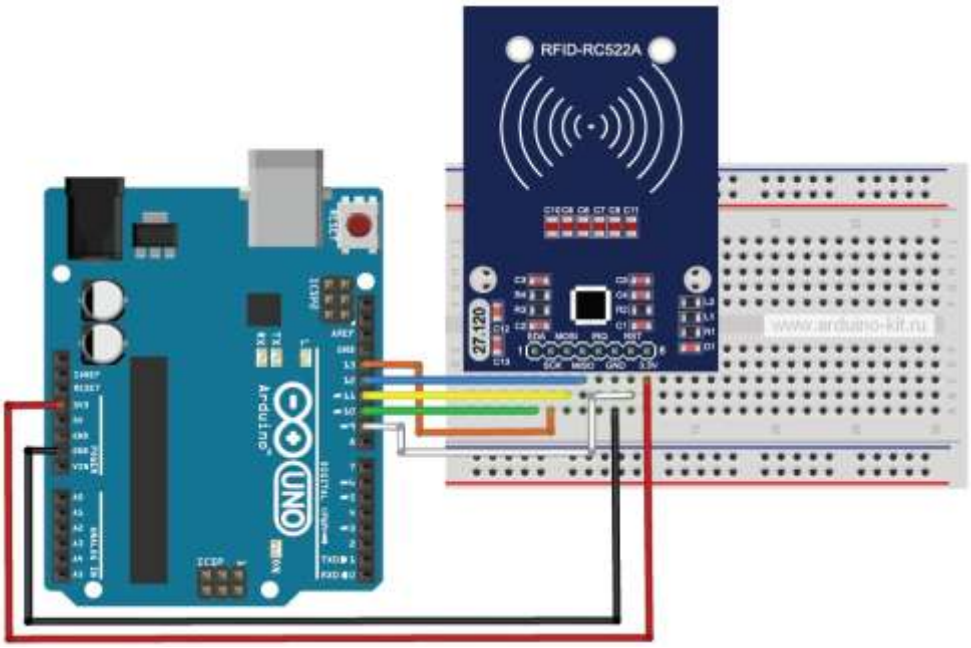


Рисунок 11.

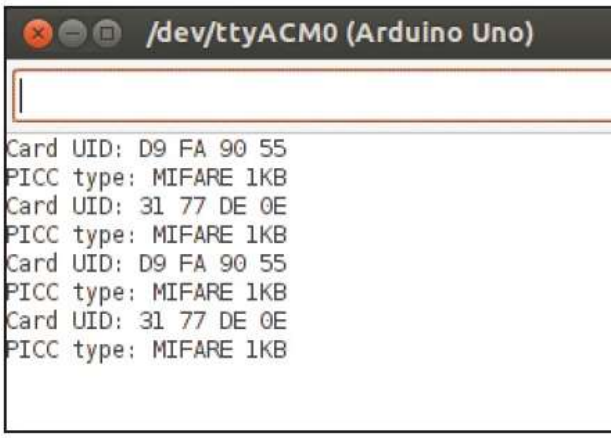


Рисунок 12

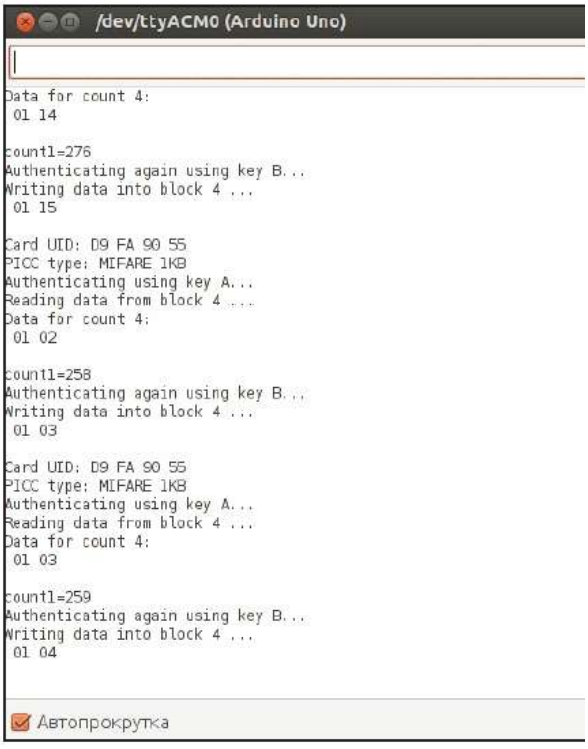


Рисунок 13

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

Тема 2.2. Датчики и сенсоры Интернета вещей
Практическая работа № 3
Подключение и программирование цифрового датчика движения

Цель работы: научиться подключать и программировать цифровой датчик движения к микроконтроллеру Ардуино

Выполнив работу, Вы будете:

уметь:

уметь:

– составлять программы на языке ассемблера для микропроцессорных систем
– производить тестирование и отладку микропроцессорных систем (МПС)
– выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

Теоретические сведения

В данной работе будет использован модуль, позволяющий отслеживать движение – пироэлектрический инфракрасный (PIR) датчик движения HC-SR501 (Рис.14). Чаще всего он используется в устройствах, предназначенных для управления освещением, и для этого может использоваться вкупе с датчиком освещённости. Этот модуль небольшой по размерам, потребляет малый ток и очень простой в использовании, благодаря чему его можно использовать и в устройствах с автономным питанием.

Рисунок 14

Характеристики датчика:

Широкий диапазон рабочего напряжения: 4,5 – 20 В постоянного тока;
Потребляемый ток покоя: ≈50 мкА;
Напряжение на выходе: 3.3 В;
Рабочая температура: от -15° С до 70° С;
Размеры: 32*24 мм;
Два режима работы;
Максимальный угол обнаружения – 110°;
Максимальная дистанция срабатывания – от 3 до 7 м (регулируется); При температуре более 30° С это расстояние может уменьшаться.

На модуль установлена линза Френеля, которая фокусирует инфракрасные сигналы на пироэлектрический датчик под названием 500BP. Датчик называется PIR (Passive Infra-Red). Пассивный он по-

тому, что для обнаружения движения не используется какая-либо дополнительная энергия, кроме той, что испускается самими объектами.

500BP состоит из двух чувствительных элементов. Управляющая микросхема модуля регистрирует изменения сигналов от обоих элементов и по характеру их изменения обнаруживает движение объектов, испускающих инфракрасные сигналы (живых организмов).

Модуль HC-SR501 имеет 3 вывода:

Питание (VCC);

Земля (GND);

Выход 3v3 (OUT).

Сразу после подачи питания несколько секунд модуль будет калиброваться, в это время возможны ложные срабатывания. Примерно через минуту он перейдёт в режим ожидания. При срабатывании датчика на выходе появляется логическая единица, напряжение – 3.3 вольта.

Изменения этого сигнала зависят от выбранного режима работы. Он меняется перемычкой (отмечена на фото с подписями какой режим будет выбран). Если выбран H– при нескольких срабатываниях подряд на выходе датчика остаётся высокий уровень, при L– для каждого срабатывания будет подан свой импульс.

Также на самом модуле можно найти два переменных резистора, регулирующих дистанцию обнаружения движения (Distance Adjust) и время, в течение которого на выходе будет логическая единица (Delay Time Adjust). Дистанция регулируется в пределах 3 – 7 метров, задержка от 5 до 300 секунд.

И ещё немного о его особенностях. При работе с датчиком следует избегать источников света и тепла, закрывающих поверхность объектива модуля. Ветер также может создавать помехи. На большем расстоянии датчик более чувствителен.

Практическое задание

Необходимые компоненты:

• контроллер Arduino UNO R3;
• плата для прототипирования;
• Модуль HC-SR501;
• 3 светодиода;
• 3 резистора (оранжевый-оранжевый-коричневый);
• провода папа-папа.

Соберите схему, представленную на рис. 15.

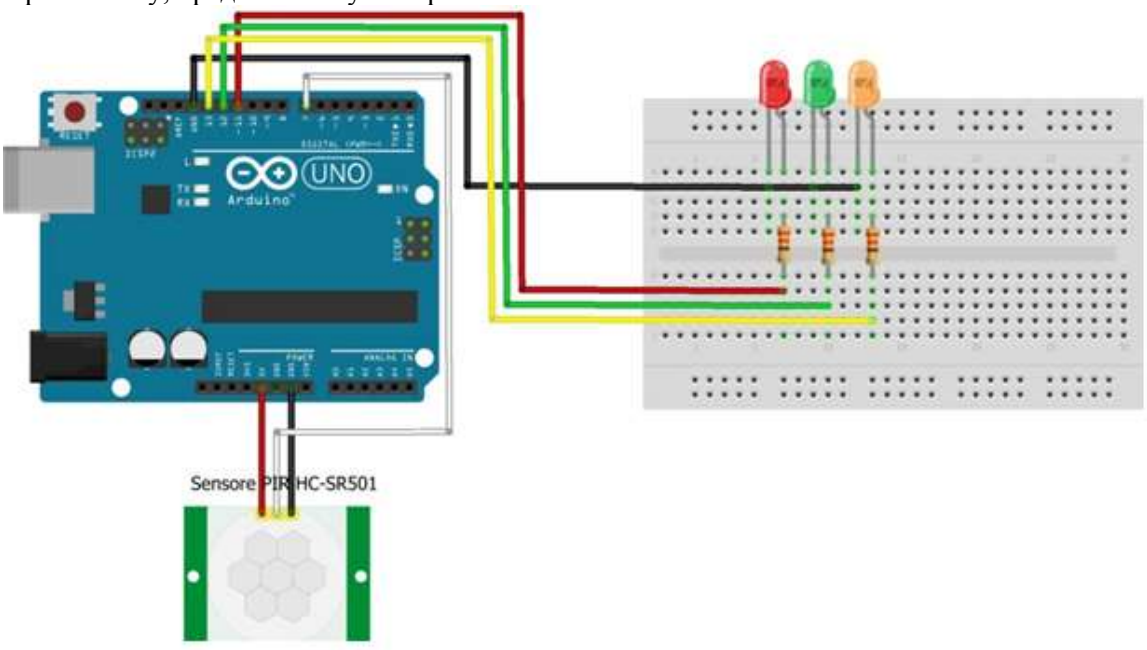


Рисунок 15.

Загружаем скетч из листинга 3.1 в контроллер Arduino. При включении загорится красный светодиод, который сигнализирует о подготовке датчика (горит 1 минуту). По истечении минуты загорится желтый светодиод , а красный погаснет, это означает что, датчик готов к обнаружению движения. Как только датчик обнаружит движение движение, загорится зеленый светодиод, который будет светится в течение трех секунд.

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

Практическая работа № 4

Подключение и программирование датчика температуры и влажности (климатконтроль)

Цель работы: научиться подключать и программировать датчик температуры и влажности к микроконтроллеру Ардуино

Выполнив работу, Вы будете:

уметь:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем
- производить тестирование и отладку микропроцессорных систем (МПС)
- выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

Теоретические сведения

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 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.

Датчик температуры и влажности DHT11 (рис.16) состоит из двух частей – емкостного датчика температуры и гигрометра. Первый используется для измерения температуры, второй – для влажности воздуха. Находящийся внутри чип может выполнять аналого-цифровые преобразования и выдавать цифровой сигнал, который считывается посредством микроконтроллера.

В большинстве случаев DHT11 или DHT22 доступен в двух вариантах: как отдельный датчик в виде пластикового корпуса с металлическими контактами или как готовый модуль с датчиком и припаянными элементами обвязки. Второй вариант гораздо проще использовать в реальных проектах и крайне рекомендуется для начинающих.

Рисунок 16

Характеристики DHT11:

- Потребляемый ток – 2,5 мА (максимальное значение при преобразовании данных);
- Измеряет влажность в диапазоне от 20% до 80%. Погрешность может составлять до 5%;
- Применяется при измерении температуры в интервале от 0 до 50 градусов (точность – 2%)
- Габаритные размеры: 15,5 мм длина; 12 мм широта; 5,5 мм высота;
- Питание – от 3 до 5 Вольт;
- Одно измерение в единицу времени (секунду). То есть, частота составляет 1 Гц;
- 4 коннектора. Между соседними расстояние в 0,1 ”.

Практическое задание

Соберите схему, представленную на рисунке 17.

Описание контактов DHT11:

1. Питание;
2. Вывод данных;
3. Не используется;
4. Земля (GND).

Контакты нумеруются слева на право, если корпус датчика находится перед вами со стороны решетки, и «ноги» расположены внизу. модуль датчика, то подключение его к Arduino предельно упрощается: подключаете VCC к +5В, GND – к земле, третий контакт – к любому свободному пину (например, 2) на плате Arduino. Номер пина нужно будет затем указать скетче.

Внимание! Обязательно соблюдайте полярность подключения. В случае неправильного подключения датчик почти неминуемо выйдет из строя. Кроме того, при неправильном подключении пластиковый корпус датчик очень сильно нагреется и может обжечь вам руки. Будьте внимательны, не торопитесь!

Загрузите скетч из листинга 4.1. Перед запуском скетча нужно убедиться, что установлена библиотека для работы с датчиками влажности и температуры. Скачать ее можно по ссылке https://github.com/adafruit/DHT-sensor-library. Загрузится папка под названием «DHT-sensor-library-master». Ее необходимо переименовать в DHT и переместить в папку libraries, что находится в корневой папке Arduino IDE.

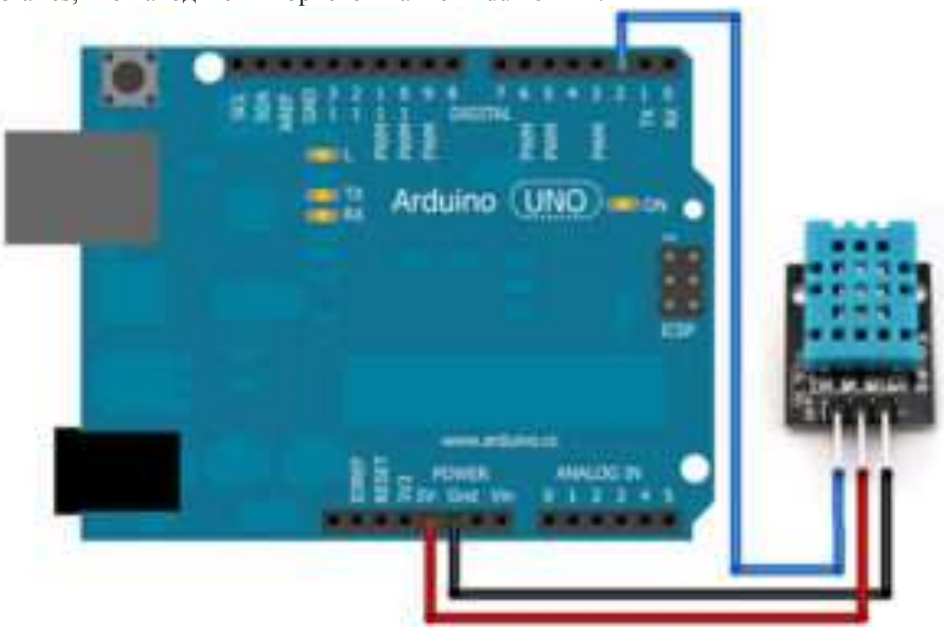


Рисунок 17

После загрузки скетча и подключения датчика, результат измерений можно посмотреть в окне монитора порта. Там будут выводиться значения температуры и влажности. Если что-то пошло не так, проверьте правильность подключения датчика, соответствие номера порта на плате Arduino и в скетче, надежность контактов.

Если все работает и датчик дает показания, можете провести эксперименты. Например, поместить датчик в более холодное место или подышать на него, отслеживая при этом изменения. Если при запотевании уровень влажности увеличивается, значит датчик работает исправно. Подуйте на него тонкой струйкой – влажность уменьшится и температура вернется в норму.

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

Практическая работа № 5

Подключение и программирование герметичного датчика температуры для влажной среды

Цель работы: научиться подключать и программировать герметичный датчик температуры к микроконтроллеру Ардуино

Выполнив работу, Вы будете:

уметь:

уметь:

– составлять программы на языке ассемблера для микропроцессорных систем
– производить тестирование и отладку микропроцессорных систем (МПС)
– выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

Теоретические сведения

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 климатическую 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. 11). Это хорошо известный цифровой датчик температуры DS18B20 водонепроницаемом корпусе из нержавейки. Приемущества водонепроницаемого корпуса – возможность измерить температуру в неблагоприятной для микросхем среде: в почве, на дожде или даже в аквариуме.

Рисунок 11

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 1-Wire протоколу. 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).

Практическое задание

В данном задании будет рассмотрен популярный цифровой датчик температуры DS18B20, работающий по протоколу 1-Wire, и создадим проект вывода показаний датчика на экран ЖКИ WH1602.

Необходимые компоненты:

- контроллер Arduino UNO R3;
- плата для прототипирования;
- датчик DS18B20;
- LCD-экран WH1602;
- резистор 50 Ом;
- потенциометр 1 кОм;
- провода папа-папа.
- внешний блок питания +5 В.

DS18B20 – цифровой термометр с программируемым разрешением от 9 до 12 битов, которое может сохраняться в EEPROM-памяти прибора. DS18B20 обменивается данными по шине 1-Wire и при этом может быть как единственным устройством на линии, так и работать в группе. Все процессы на шине управляются центральным микропроцессором.

Диапазон измерений датчика: от –55 °C до +125 °C с точностью 0,5 °C в диапазоне от –10 °C до +85 °C. В дополнение DS18B20 может питаться напряжением линии данных (так называемое питание от паразитного источника) при отсутствии внешнего источника напряжения.

Каждый датчик типа DS18B20 имеет уникальный 64-битный последовательный код, который позволяет общаться со множеством датчиков DS18B20, установленных на одной шине. Первые 8 битов – код серии (для DS18B20 – 28h), затем 48 битов уникального номера, и в конце 8 битов CRC-кода. Такой принцип позволяет использовать один микропроцессор, чтобы контролировать множество датчиков DS18B20, распределенных по большому участку.

В нашем эксперименте мы будем считывать данные с датчика температуры DS18B20 и выводить на экран ЖКИ WH1602. Схема подключения датчика температуры DS18B20 и WH1602 к плате Arduino показана на рис. 11.

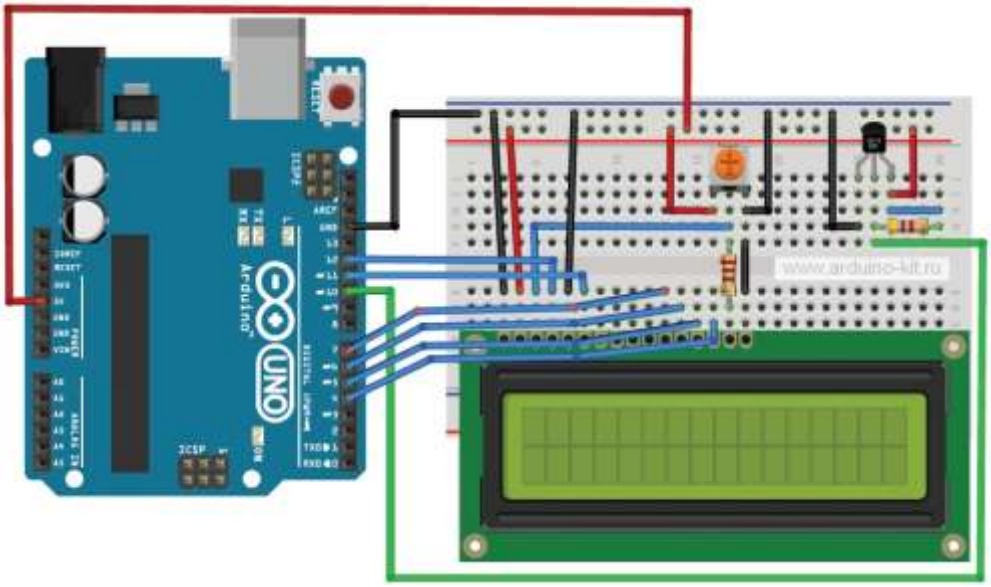


Рисунок 11

Для работы с устройствами 1-Wire в Arduino есть стандартная библиотека OneWire. Содержимое скетча для чтения данных с датчика и вывода на экран индикатора WH1602 показано в листинге 5.1. Последовательность данных для чтения данных с устройств 1-Wire следующая:

1. Произвести RESET и поиск устройств на линии 1-Wire.
2. Выдать команду 0x44, чтобы запустить конвертацию температуры датчиком.
3. Подождать не менее 750 мс.
4. Выдать команду 0xBE, чтобы считать ОЗУ датчика (данны о температуре будут в первых двух байтах).

Порядок подключения:

1. Подключаем датчик DS18B20 и WH1602 по схеме на рис. 11.
2. Загружаем в плату Arduino скетч из листинга 5.1.
3. Смотрим на экране дисплея показания датчика температуры.

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

Практическая работа № 6

Подключение и программирование ультразвукового дальномера (парктроник)

Цель работы: научиться подключать и программировать ультразвуковой дальномер к микроконтроллеру Ардуино

Выполнив работу, Вы будете:

уметь:

уметь:

– составлять программы на языке ассемблера для микропроцессорных систем
– производить тестирование и отладку микропроцессорных систем (МПС)
– выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

Теоретические сведения

Ультразвуковой дальномер HC-SR04 (см. рис.12) – это помещенные на одну плату приемник и передатчик ультразвукового сигнала. Излучатель генерирует сигнал, который, отразившись от препятствия, попадает на приемник. Измерив время, за которое сигнал проходит до объекта и обратно, можно оценить расстояние. Кроме самих приемника и передатчика, на плате находится еще и необходимая обвязка, чтобы сделать работу с этим датчиком простой и удобной.

Рисунок 12

Характеристики ультразвукового дальномера HC-SR04:

- измеряемый диапазон – от 2 до 500 см;
- точность – 0,3 см;
- угол обзора – < 15°;
- напряжение питания – 5 В.

Датчик имеет 4 вывода стандарта 2,54 мм:

- VCC – питание +5 В;
- Trig (T) – вывод входного сигнала;
- Echo (R) – вывод выходного сигнала;
- GND – земля.

Последовательность действий для получения данных такова:

- подаем импульс продолжительностью 10 мкс на вывод Trig;
- внутри дальномера входной импульс преобразуется в 8 импульсов частотой 40 кГц и посылается вперед через излучатель T;
- дойдя до препятствия, посланные импульсы отражаются и принимаются приемником R, в результате получаем выходной сигнал на выводе Echo;
- непосредственно на стороне контроллера переводим полученный сигнал в расстояние по формулам:
  - ширина импульса (мкс) / 58 = дистанция (см);
  - ширина импульса (мкс) / 148 = дистанция (дюйм).

Практическое задание

В задании нужно построить звуковую сигнализацию, которая будет включаться при приближении к плате Arduino на расстояние меньше 1 м, при этом появляется звуковой сигнал на пьезоизлучателе и начинает вращение сервопривод. Такая МПС может использоваться, например, для построения автоматических ворот.

Необходимые компоненты:

- контроллер Arduino UNO R3;
- плата для прототипирования;
- ультразвуковой датчик расстояния HC-SR04;
- пьезоизлучатель;
- резистор 100 Ом;
- сервопривод;
- провода.

Схема сборки представлена на рисунке 13.

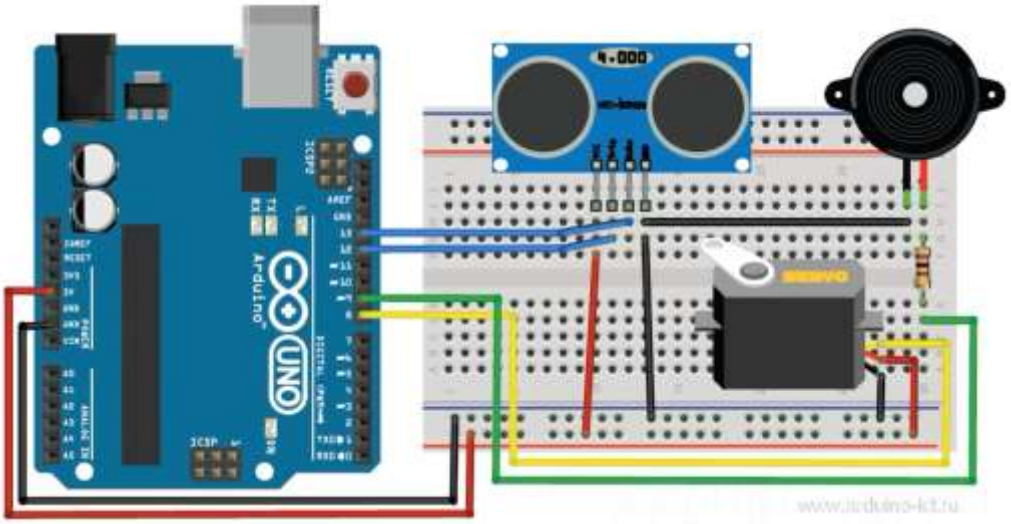


Рисунок 13– Схема сборки

Програмирование

При написании скетча будем использовать библиотеку Servo для работы с сервоприводом и библиотеку Ultrasonic для работы с ультразвуковым датчиком.

Датчик Ultrasonic принимает два параметра: номера пинов, к которым подключены выводы Trig и Echo.

В эксперименте была создана звуковая сигнализация, которая будет включаться при приближении к плате Arduino на расстояние меньше 1 м. Датчик размещен на кронштейне вращающегося сервопривода и контролирует пространство с углом обзора 180°. Если датчик обнаруживает объект в радиусе 1 м, подается звуковой сигнал на пьезоизлучатель, вращение сервопривода прекращается.

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

Практическая работа № 7
Подключение и программирование аналогового датчика шума

Цель работы: научиться подключать и программировать аналоговый датчик шума к микроконтроллеру Ардуино

Выполнив работу, Вы будете:

уметь:

уметь:

– составлять программы на языке ассемблера для микропроцессорных систем
– производить тестирование и отладку микропроцессорных систем (МПС)
– выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

Теоретические сведения

Датчик звука (микрофон) для Arduino из платы (орис. 14) на котором смонтированы порты подключения к Arduino Nano, усилитель звука, подстроечный резистор и электронный микрофон, чувствительный к звуку, приходящему во всех направлениях. Регулятором чувствительности (переменным резистором) можно настраивать чувствительность микрофона и выбирать от какого уровня шума будет срабатывать датчик. Данная плата расширения для Arduino позволяет перевести звуковые колебания в цифровой сигнал. При колебании мембраны в микрофоне от звуковых волн, изменяется емкость его конденсатора, вследствие чего проявляется изменение напряжения на выходах датчика звука, соответствующее звуковому сигналу. Сенсор справа на картинке может отправлять цифровой и аналоговый сигнал.

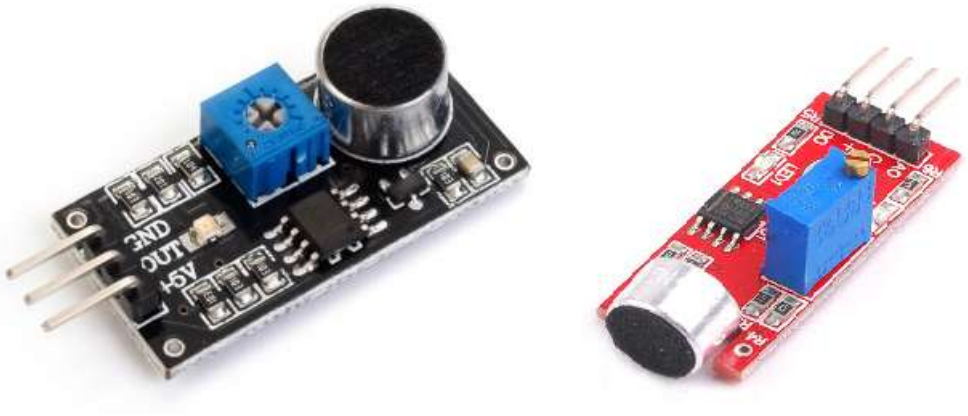


Рисунок 14

Практическое задание

Подключение датчика шума

Необходимое оборудование:

- плата Arduino Uno / Arduino Nano / Arduino Mega;
- макетная плата;
- датчик звука (микрофон);
- 1 светодиод и 1 резистор 220 Ом;
- провода «папа-папа» и «папа-мама».

Датчик звука для Ардуино имеет на плате подписанные выходы (обозначение у каждого производителя может отличаться), но проблем с подключением датчика к Ардуино возникнуть не должно. Питание датчика производится от 5V, выход (OUT, S или A0) подключается к любому аналоговому входу на Arduino Uno, а выход D0 к Pin 2, если требуется получать цифровой сигнал на Ардуино с датчика микрофона.

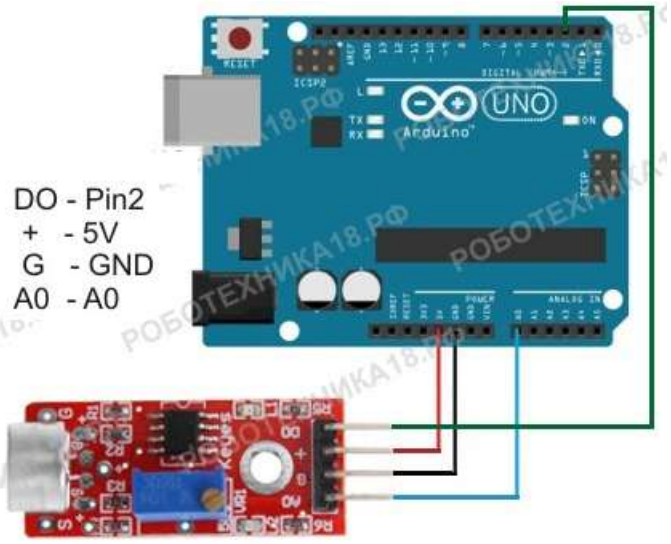


Рисунок 15

Чтобы сделать шумометр, который будет включать светодод по хлопку в ладоши необходимо собрать электрическую схему из следующих элементов: светодиод с резистором, плата Arduino и датчик звука для включения света своими руками. Светодиод можно подключить к любому выходу, в скетче мы использовали Pin 11. После сборки схемы, подключите Ардуино к компьютеру и загрузите скетч 7.1.

Программирование:

- спецификатор boolean используется для объявления логических значений (истина/ложь) в языке программирования C++;

- в строчке statuslamp=!statuslamp; мы меняем статус светодиода при хлопке;

- в строчке if(analogRead(A0)>60) вместо значения 60 можно подставить любое значение. Узнайте показания датчика звука при хлопке в ладоши на мониторе порта и поставьте свои значения в скетч, при необходимости.

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

Тема 3.3. Умный дом и город
Практическая работа № 8
Подключение и программирование умного светильника

Цель работы: научиться подключать и программировать подсистему контроля освещения на базе МК Ардуино

Выполнив работу, Вы будете:

уметь:

уметь:

– составлять программы на языке ассемблера для микропроцессорных систем
– производить тестирование и отладку микропроцессорных систем (МПС)
– выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

Теоретические сведения

В данной практической работе будет использоваться аналоговый датчик для измерения освещенности – фоторезистор (рис. 16). С его помощью можно зажигать и гасить освещение, в зависимости от уровня освещенности.

Рисунок 16.

Практическое задание 1

Необходимые компоненты:

– контроллер Arduino UNO R3;
– плата для прототипирования;
– фоторезистор;
– резистор 10 кОм;
– резистор 220 Ом – 8 штук;
– светодиод;
– провода папа-папа.

Распространённое использование фоторезистора – измерение освещённости. В темноте его сопротивление довольно велико. Когда на фоторезистор попадает свет, сопротивление падает пропорционально освещенности. Схема подключения фоторезистора к Arduino показана на рис. 17. Для измерения освещённости необходимо собрать делитель напряжения, в котором верхнее плечо будет представлено фоторезистором, нижнее – обычным резистором достаточно большого номинала. Будем использовать резистор 10 кОм. Среднее плечо делителя подключаем к аналоговому входу A0 Arduino.

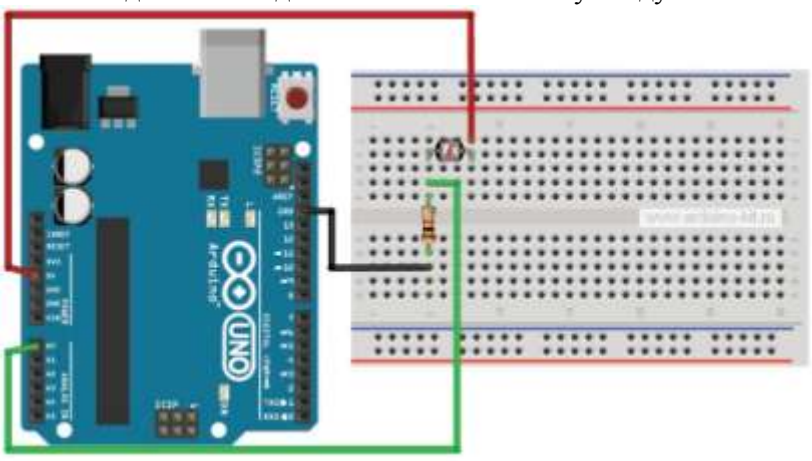


Рисунок 17

Порядок подключения:

1. Подключаем фоторезистор по схеме на рис. 17.

2. Загружаем в плату Arduino скетч из листинга 8.1.

3. Регулируем рукой освещенность фоторезистора и наблюдаем вывод в последовательный порт изменяющихся значений, запоминаем показания при полной освещенности помещения и при полном перекрывании светового потока.

Практическое задание 2

Создадим индикатор освещенности с помощью светодиодного ряда из 8 светодиодов. Количество горящих светодиодов пропорционально текущей освещенности. Собираем светодиоды по схеме на рис. 18, используя ограничительные резисторы номиналом 220 Ом.

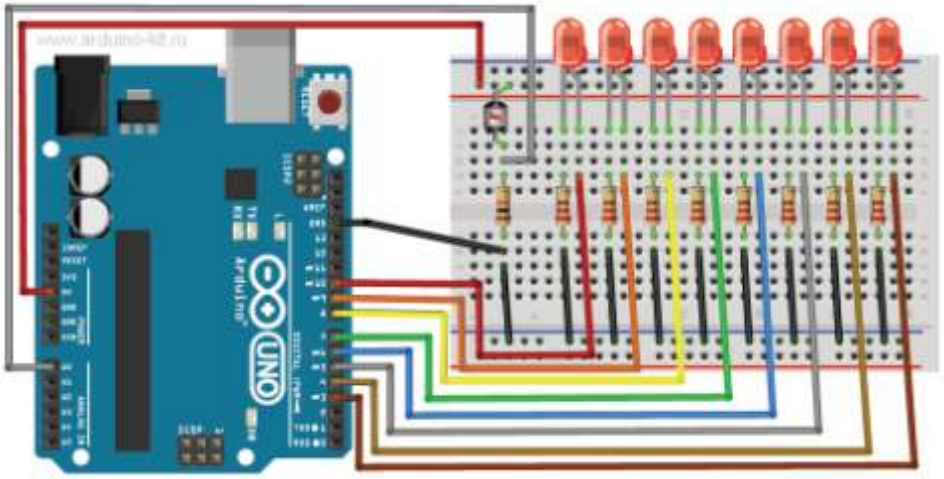


Рисунок 18

Порядок подключения:

1. Подключаем фоторезистор и светодиоды по схеме на рис. 18.

2. Загружаем в плату Arduino скетч из листинга 8.2.

3. Регулируем рукой освещенность фоторезистора и по количеству горящих светодиодов определяем текущий уровень освещенности (рис. 18).

Нижний и верхний пределы освещенности мы берем из запомненных значений при проведении эксперимента по предыдущему скетчу (листинг 18.1). Промежуточное значение освещенности мы масштабируем на 8 значений (8 светодиодов) и зажигаем количество светодиодов пропорциональное значению между нижней и верхней границами.

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;

б) схемы подключения устройств ввод-вывода;

в) выводы по работе.

Практическая работа № 9
Подключение и программирование датчика влажности почвы

Цель работы: научиться подключать и программировать датчик влажности почвы к микроконтроллеру Ардуино

Выполнив работу, Вы будете:

уметь:

уметь:

– составлять программы на языке ассемблера для микропроцессорных систем

– производить тестирование и отладку микропроцессорных систем (МПС)

– выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

Теоретические сведения

омашний 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 в создании уюта имеет использование комнатных цветов. Они доступны каждому из нас и при этом лучше любой мебели помогут создать уют и комфорт, и как ни что 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.19) 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 создаётся 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").

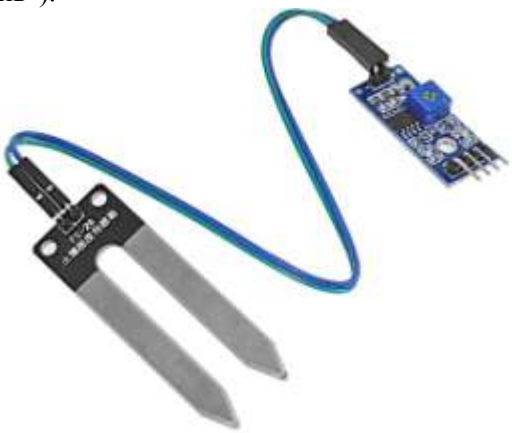


Рисунок 19

Кроме контактов соединения с 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 наличие поступающего на датчик питания и уровня цифрового сигналы на выходе D0. Наличие цифрового вывода D0 и светодиода уровня D0 позволяет использовать модуль автономно, без подключения к контроллеру.

Практическое задание

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. 20.

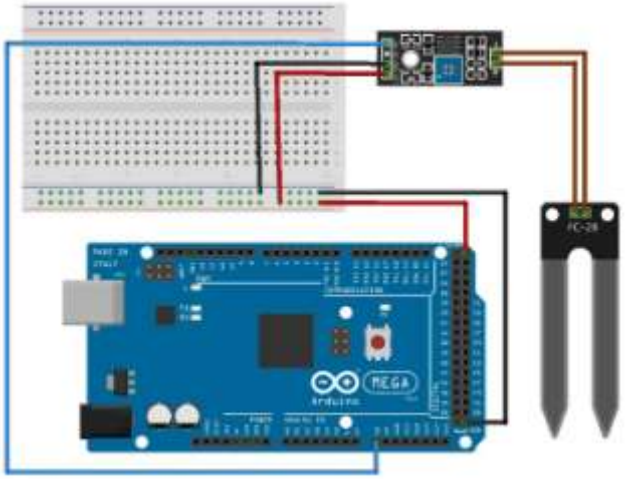


Рисунок 20

Загрузим на плату Arduino Mega скетч получения данных с датчика и вывода в последовательный порт Arduino. Получение данных влажности оформим в виде отдельной процедуры get_data_soilmoisture(). Содержимое скетча представлено в листинге 9.1.

Загрузим скетч на плату Arduino Mega, откроем монитор последовательного порта и видим вывод данных, получаемых с датчика Soil Moisture (рис.21). Подберите практическим путем аналоговые значения для констант MINVALUESOILMOISTURE (полный полив) и MINVALUESOILMOISTURE (критическая сухость).

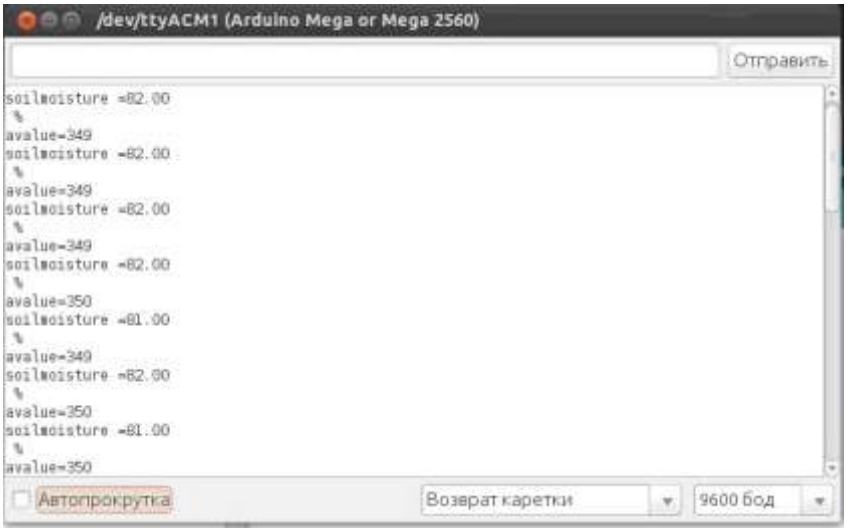


Рисунок 21

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

Практическая работа № 10
Подключение и программирование датчика уровня воды

Цель работы: научиться подключать и программировать датчик уровня воды к микроконтроллеру Ардуино

Выполнив работу, Вы будете:

уметь:

уметь:

– составлять программы на языке ассемблера для микропроцессорных систем
– производить тестирование и отладку микропроцессорных систем (МПС)
– выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

Теоретические сведения

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. 22) – погружной. Чем больше погружение датчика в воду, тем меньше сопротивление между двумя соседними проводами.

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, сигнализирующих о наличие поступающего на датчик питания.

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

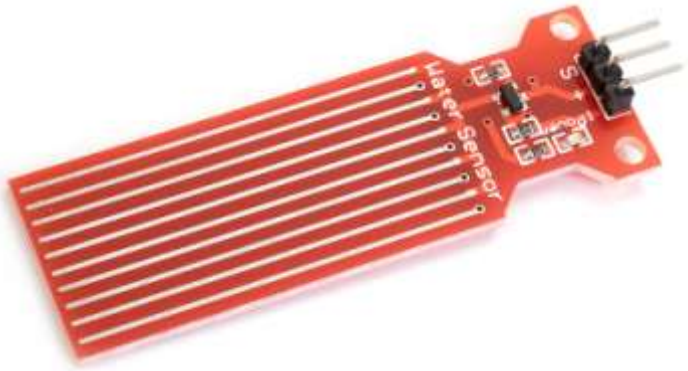


Рисунок 22

Практическое задание

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. 23.

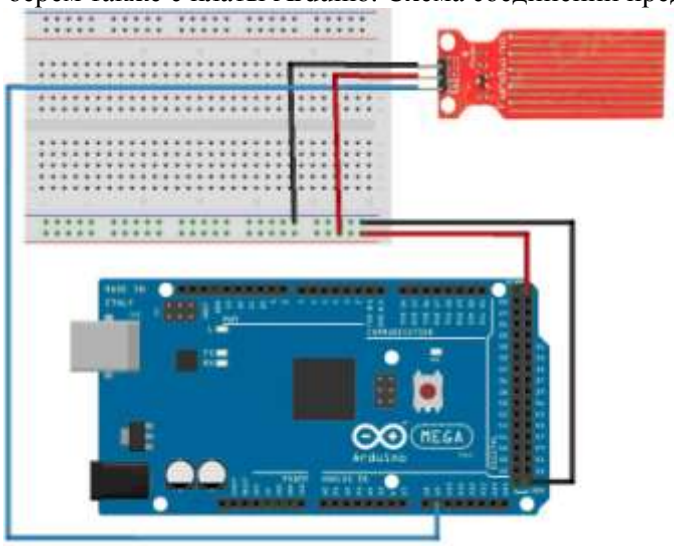


Рисунок 23

Загрузим на плату Arduino Mega скетч получения данных с датчика уровня воды, перевода аналогового значения в см (0 – 4) и вывода в последовательный порт Arduino. Получение данных влажности оформим в виде отдельной процедуры get_data_levelwater(). Содержимое скетча представлено в листинге 10.1.

Загрузим скетч на плату Arduino Mega, откроем монитор последовательного порта и видим вывод данных, получаемых с датчика уровня воды (рис. 24). При попадание воды на датчик выводим пообщение о протекании. Подберите практическим путем аналоговое значения для константы LEVELWATER (пороговое значение погружения).

Рисунок 24

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

Практическая работа № 11
Подключение и программирование датчика углеводородных газов

Цель работы: научиться подключать и программировать датчик углеводородных газов к микроконтроллеру Ардуино

Выполнив работу, Вы будете:

уметь:

уметь:

– составлять программы на языке ассемблера для микропроцессорных систем
– производить тестирование и отладку микропроцессорных систем (МПС)
– выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

Теоретические сведения

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 встроенный 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 занимать около минуты.

Рисунок 25

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 об единицах измерения. На территории бывшего Советского Союза, показатели принято измерять в процентах (%) или же непосредственно в массе к объему (мг/м3). 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.

Практическое задание

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. 26.

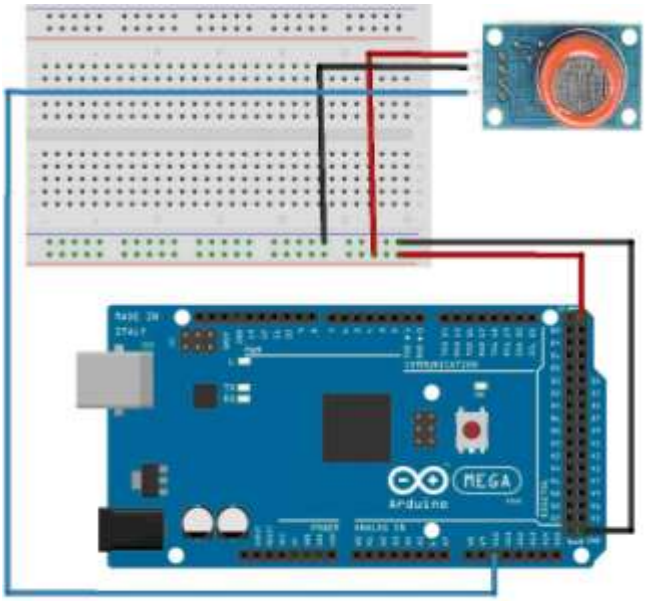


Рисунок 26

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.

Содержимое скетча представлено в листинге 11.1.

Загрузим скетч на плату Arduino Mega, откроем монитор последовательного порта и увидите вывод данных о содержании пропана, метана и дыма (рис. 27).

Рисунок 27

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

Практическая работа № 12

Подключение и программирование датчика угарных газов

Цель работы: научиться подключать и программировать датчик угарных газов к микроконтроллеру Ардуино

Выполнив работу, Вы будете:

уметь:

уметь:

– составлять программы на языке ассемблера для микропроцессорных систем
– производить тестирование и отладку микропроцессорных систем (МПС)
– выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

Теоретические сведения

Основным источником выделения угарного гоза CO, является сгорание углеродного топлива при недостаточном количестве кислорода. Углерод "не догорает" и вместо углекислого газа CO2, 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/м³ (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 первым страдает головной мозг и нервная система.Во-вторых, при определенных концентрациях он образует взрывоопасную смесь.

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

Рисунок 28

Практическое задание

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. 29.

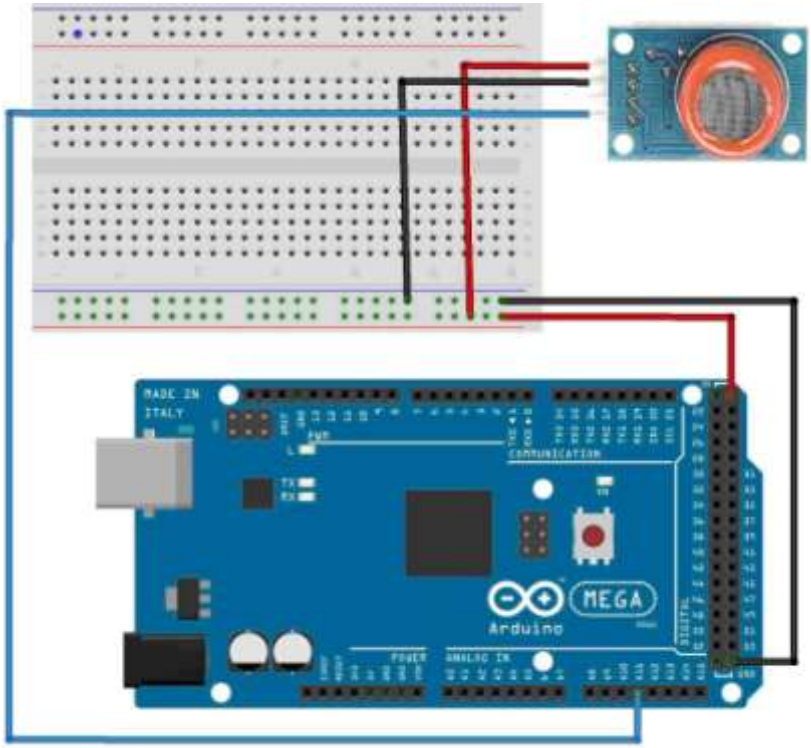


Рисунок 29

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 12.1.

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

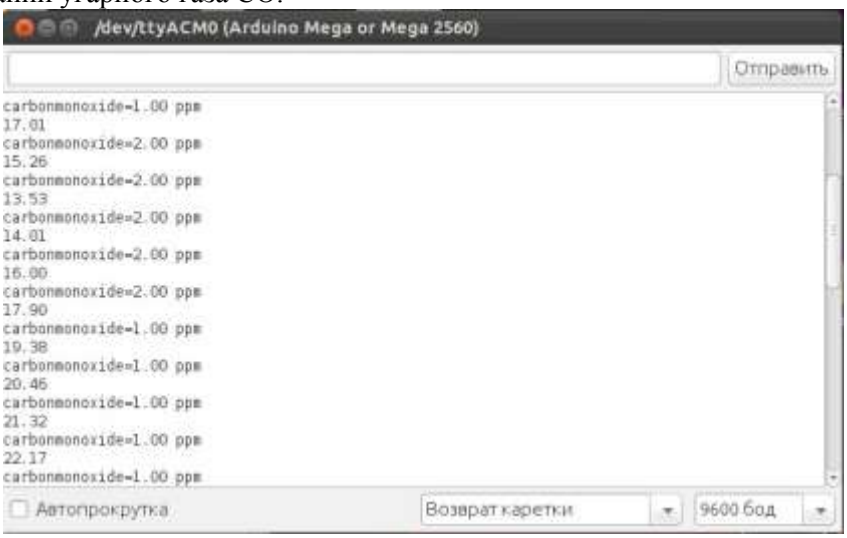


Рисунок 30

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

Практическая работа № 13
Подключение и программирование модуля датчика огня

Цель работы: научиться подключать и программировать модуль датчика огня к микроконтроллеру Ардуино

Выполнив работу, Вы будете:

уметь:

уметь:

– составлять программы на языке ассемблера для микропроцессорных систем
– производить тестирование и отладку микропроцессорных систем (МПС)
– выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino, ПК с ПО Arduino IDE

Теоретические сведения

Модуль датчика огня Flame Sensor (рис. 31.) 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 градусов по Цельсию.

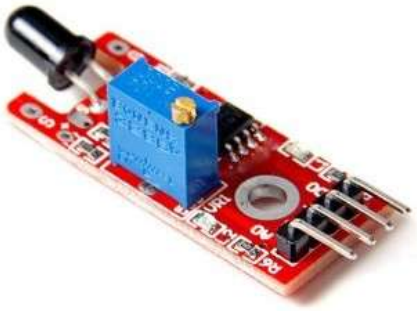


Рисунок 31

Практическое задание

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. 32.

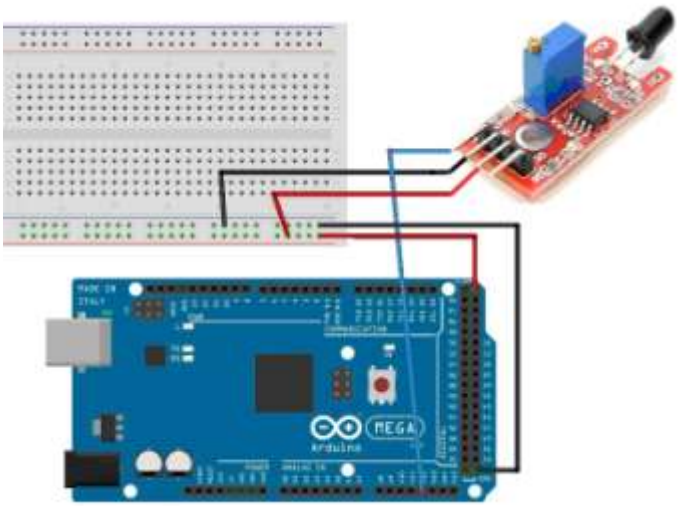


Рисунок 32

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 13.1.

Загрузим скетч на плату Arduino Mega, откроем монитор последовательного порта и увидите вывод данных с модуля датчика огня Flame Sensor (рис. 33).

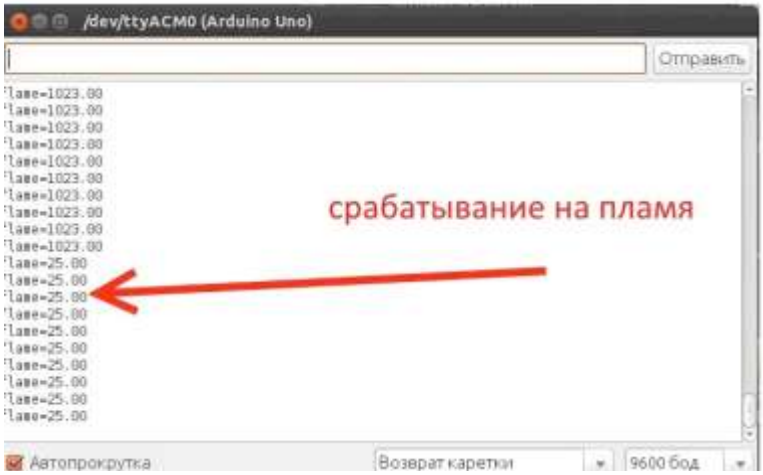


Рисунок 33

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

Практическая работа № 14
Управление умным домом через мобильное приложение

Цель работы: научиться подключать и программировать устройства умного дома на базе МК Ардуино с использованием приложений

Выполнив работу, Вы будете:

уметь:

уметь:

– составлять программы на языке ассемблера для микропроцессорных систем
– производить тестирование и отладку микропроцессорных систем (МПС)
– выбирать микроконтроллер/микропроцессор для конкретной системы управления

Материальное обеспечение: Стартовый набор для изучения Arduino (в комплекте с модулем NodeMCU!), ПК с ПО Arduino IDE, мобильное устройство с IoT Manager

Теоретические сведения

IoT Manager – это мобильное приложение для телефонов и планшетов, совмещающего 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 www.iotmanager.ru. 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.34).

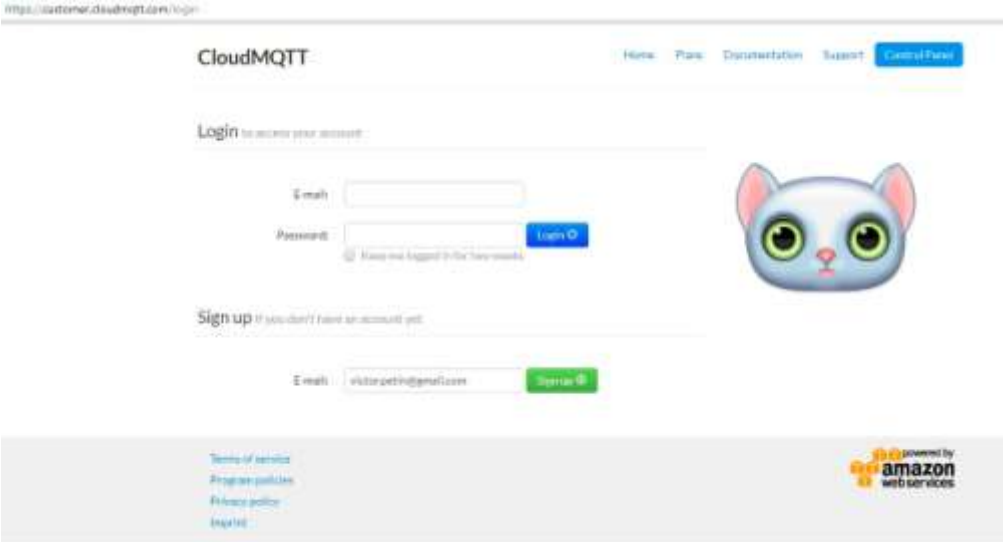


Рисунок 34

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. 35). Можно создать несколько брокеров.

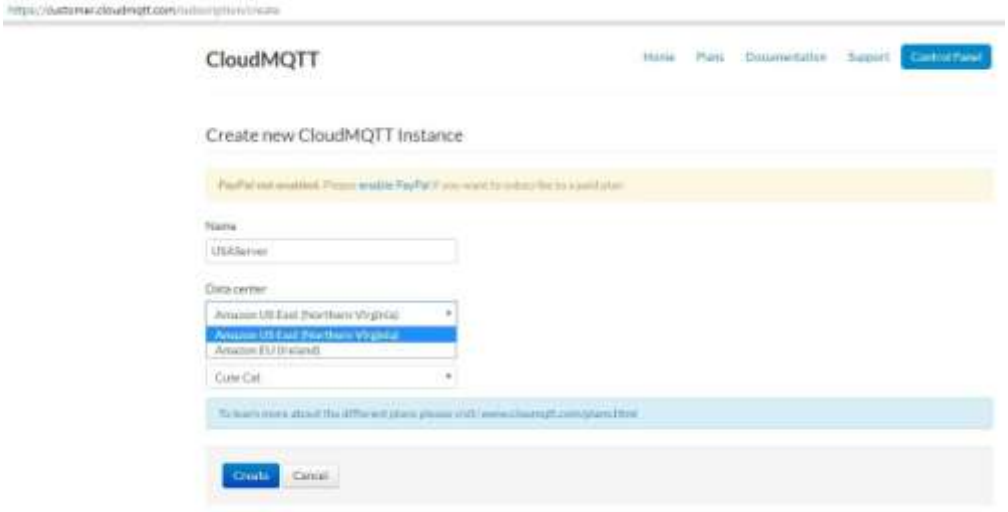


Рисунок 35

Теперь нажимаем на кнопку Details (см. рис. 36).

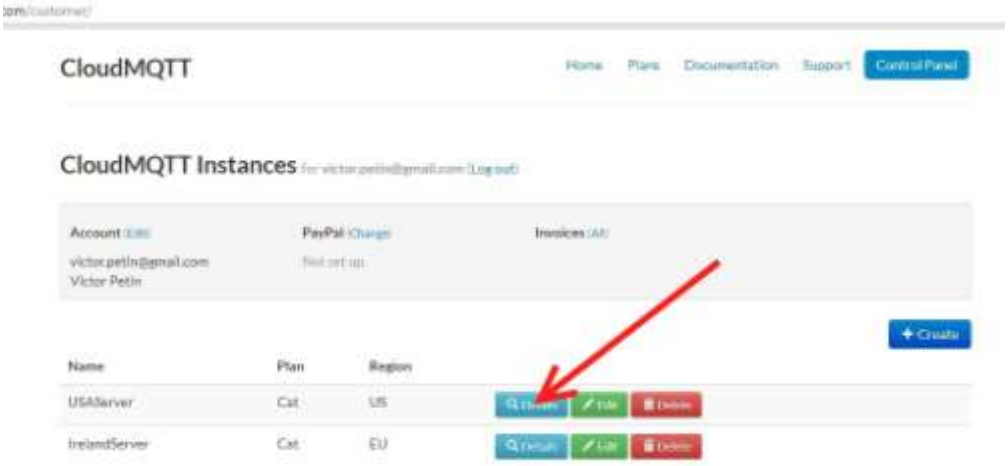


Рисунок 36

Попадаем в настройки (рис. 37). Нам необходимы следующие:

- имя хоста m13.cloudmqtt.com;
- порт 18274 (для скетча Arduino IDE –листинг 14.1);
- 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. 38).

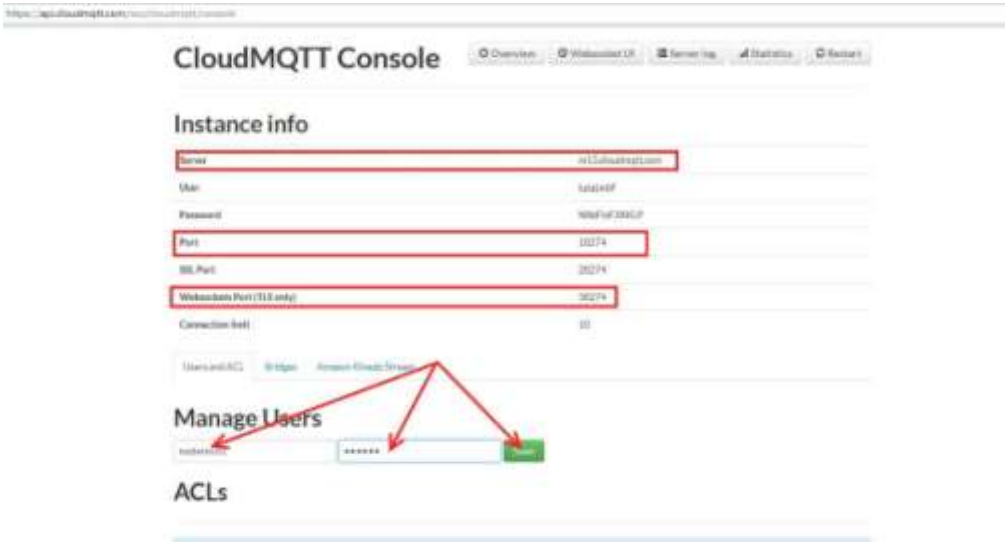


Рисунок 37

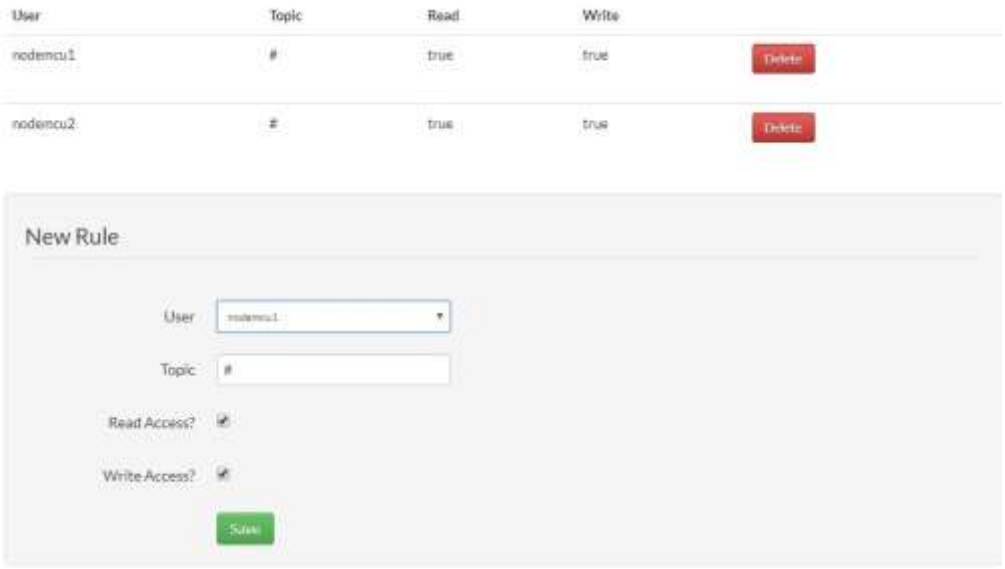


Рисунок 38

Теперь можно скачать и установить мобильное приложение IoTManager. Запускаем. Необходимо произвести настройку. Нажимаем на Settings (рис.39) и в появившейся форме вносим данные своего брокера (рис.40):

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

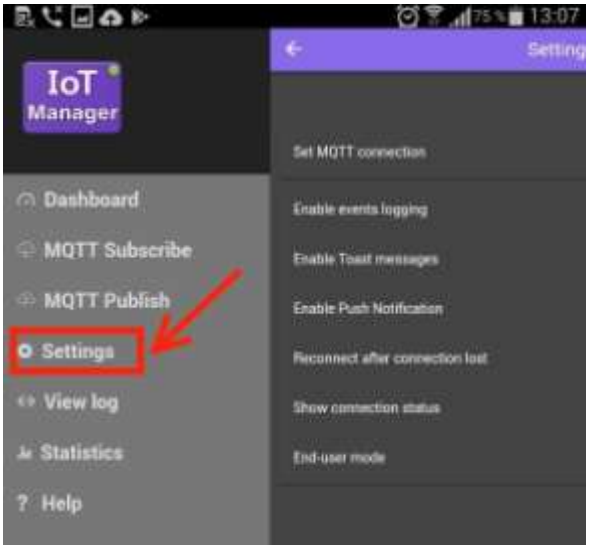


Рисунок 39

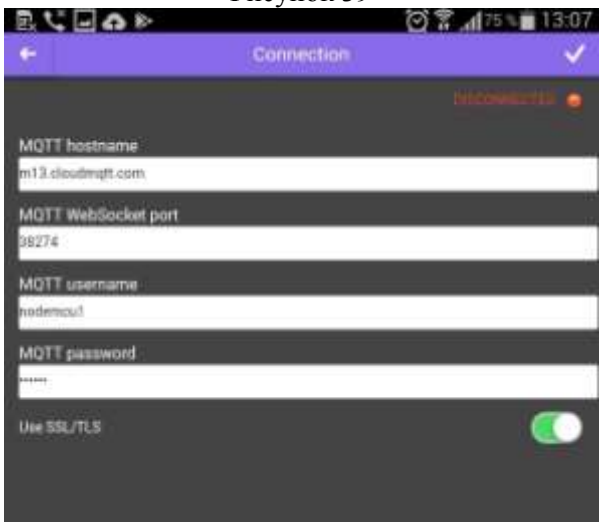


Рисунок 40

Теперь выходим на страницу Dashboard и должны увидеть установленое соединение (рис. 41). Надпись No data не должна вас смущать – данные в топики еще не передавались.

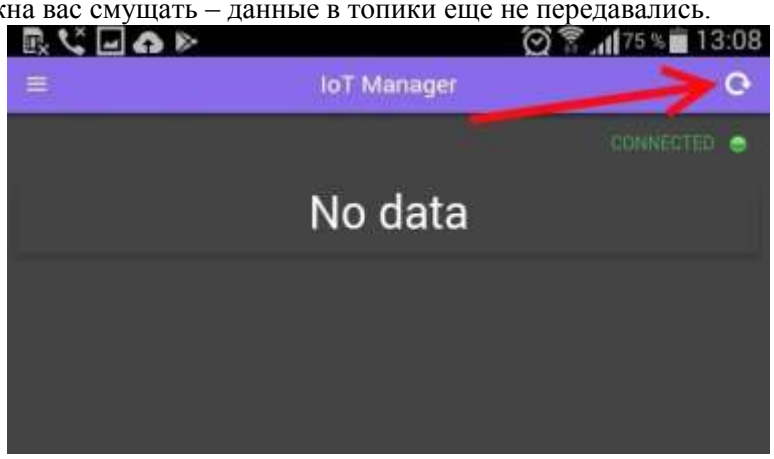


Рисунок 41

Практическое задание 1

Для проверки работы брокера загрузим на плату NodeMcu тестовый пример, который приведен в листинге 14.2. 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 www.arduino-kit.ru по ссылке. Загружаем скетч на плату NodeMCU и открываем монитор последовательного порта, где видим отправку данных брокеру с платы (рис. 42).

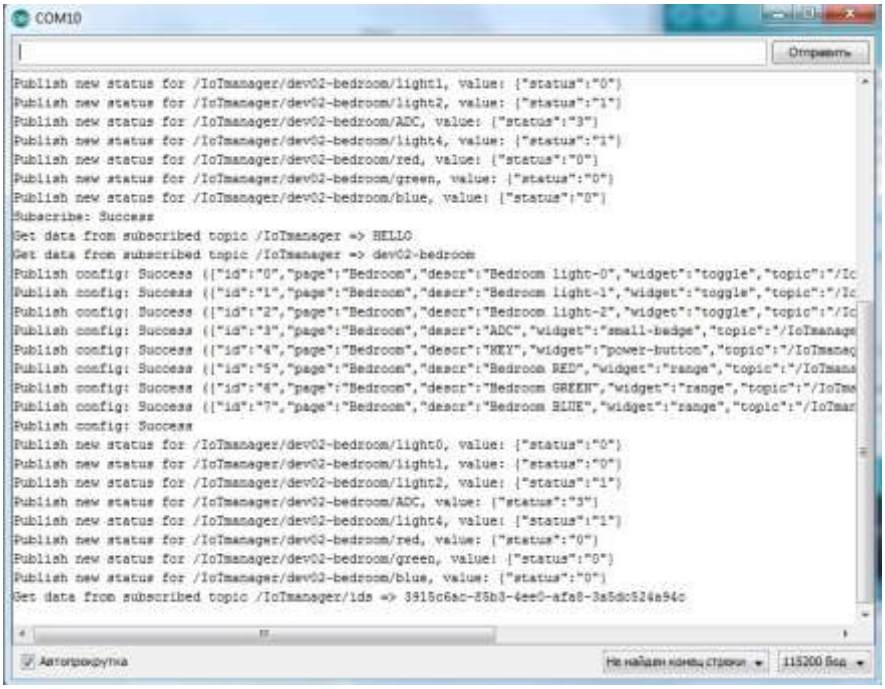


Рисунок 42

Как только началась отправка данных с платы NodeMCU брокеру, в мобильном приложении появятся эти данные (рисунок 43).

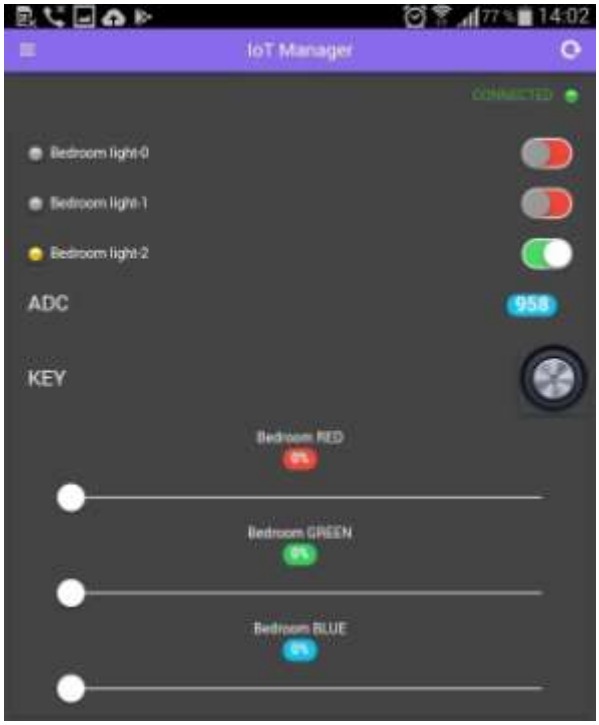


Рисунок 43

Мы можем также посмотреть и список тем, на которые подписано мобильное устройство (рис. 44).

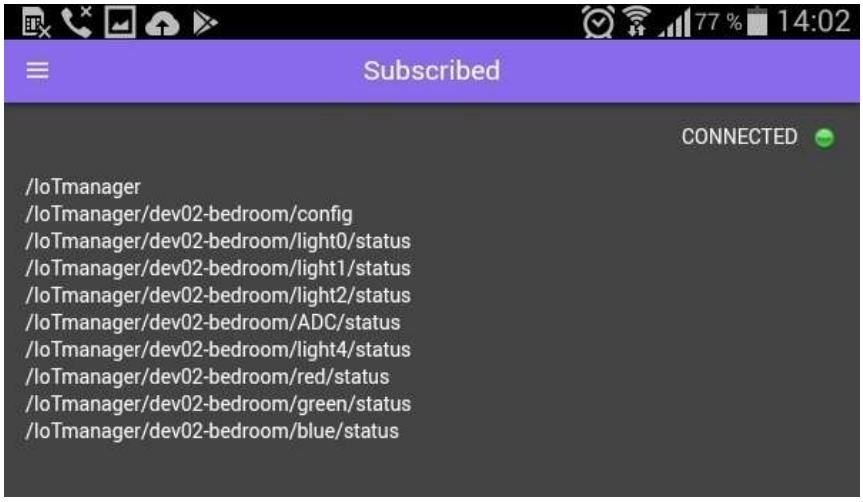


Рисунок 44

Но IoTManager не только подписан на темы, но также выступает в роли publisher – публикует данные в темы. Это значения слайдеров и статус кнопки. Эти данные плата NodeMCU, подписанная в качестве subscriber на эти темы, может использовать для управления, подключенными к плате устройствами.

Скетч 9.1

Практическое задание 2

Рассмотрим подробнее отправку данных с датчиков нашего умного дома брокеру. Будем отправлять брокеру данные с двух датчиков DHT22 и DS18B20. Правки осуществляем в скетче из предыдущей главы. 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 SensorsNodemcu, 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().

Полный скетч приведен в листинге 14.3. Загружаем его на плату NodeMCU, открываем монитор последовательного порта и видим отправку данных с датчиков (рис. 45).

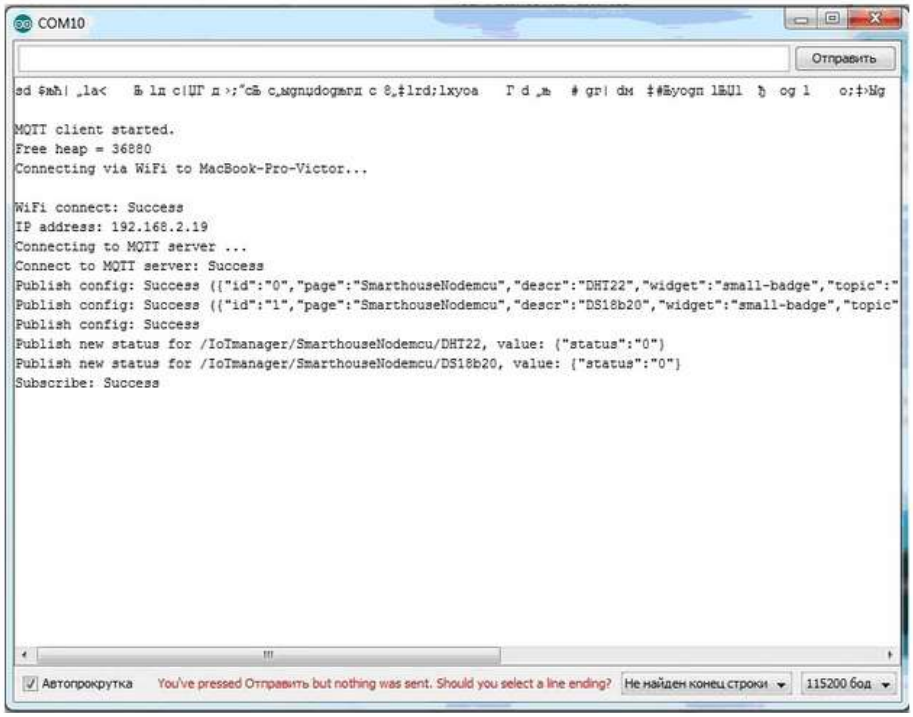


Рисунок 45

И смотрим данные на смартфоне в приложении IoT Manager (рис. 46).

Рисунок 46

Практическое задание 3

В данном задании 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]+"\",\"descr\":\"" + 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
}
```

Полный скетч приведен в листинге 14.4. Загружаем его на плату NodeMCU, открываем монитор последовательного порта и видим отправку данных с датчиков (рис. 47).

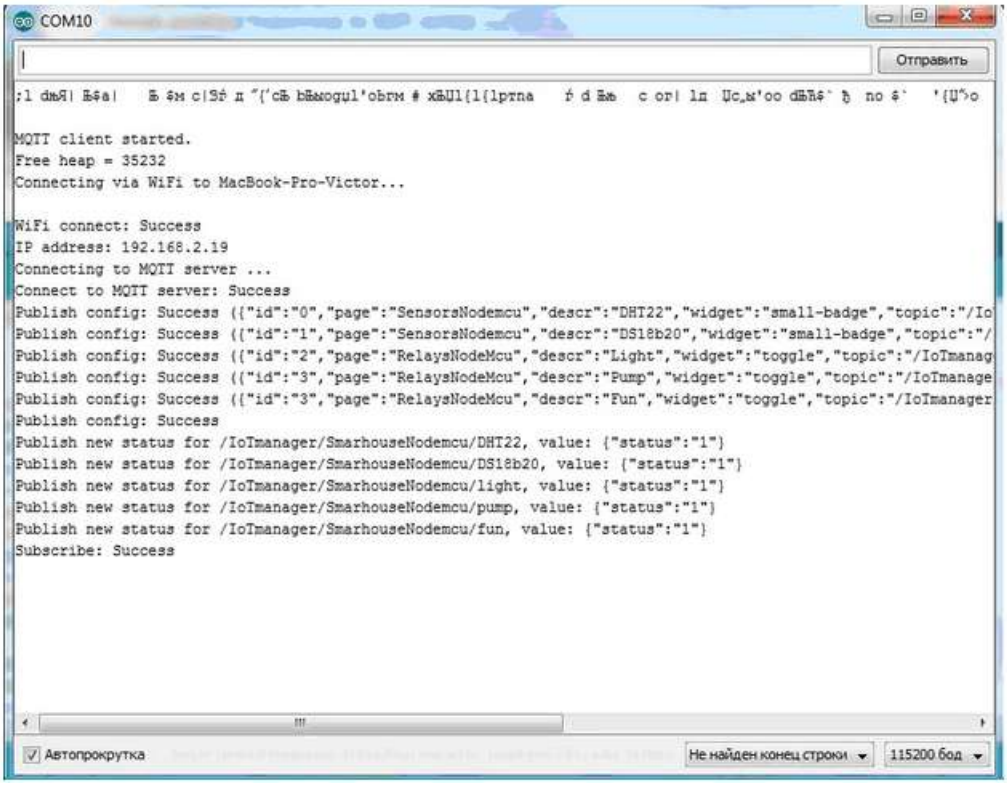


Рисунок 47

И смотрим данные на смартфоне в приложении IoT Manager (рис. 48). Виджеты получения данных и управления исполнительными устройствами расположены в разных вкладках (рис.49).

Рисунок 48

Рисунок 49

Изменение состояния toggle к публикации измененных данных в тему /IoTmanager/SmarhouseNodemcu/xxx и плата NodeMCU, получая эти данные (см. рис. 50), изменяет состояние соответствующего исполнительного устройства.

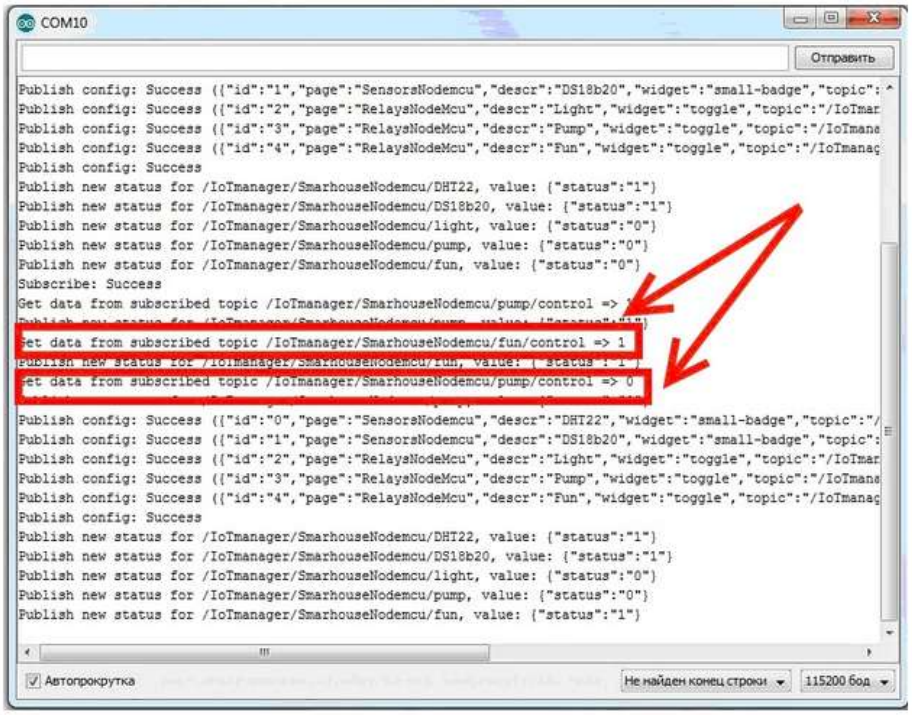


Рисунок 50

Форма представления результата:

Отчет по работе должен содержать:

а) наименование работы и цель работы;
б) схемы подключения устройств ввод-вывода;
в) выводы по работе.

ПРИЛОЖЕНИЕ 1.

Листинги программ

Листинг 1.1

```
const int LED=10; // вывод для подключения светодиода 10 (D10)
void setup()
{
// Конфигурируем вывод подключения светодиода как выход (OUTPUT)
pinMode(LED, OUTPUT);
}
void loop()
{
// включаем светодиод, подавая на вывод 1 (HIGH)
digitalWrite(LED,HIGH);
// пауза 1 сек (1000 мс)
delay(1000);
// выключаем светодиод, подавая на вывод 0 (LOW)
digitalWrite(LED,LOW);
// пауза 1 сек (1000 мс)
delay(1000);
}
```

Листинг 1.2

```
const int LED=10; // Контакт 10 для подключения светодиода
const int BUTTON=2; // Контакт 2 для подключения кнопки
int tekButton = LOW; // Переменная для сохранения текущего состояния
кнопки
int prevButton = LOW; // Переменная для сохранения предыдущего состояния
// к нопки
boolean ledOn = false; // Текущее состояние светодиода (включен/выключен)
void setup()
{
// Сконфигурировать контакт светодиода как выход
pinMode (LED, OUTPUT);
// Сконфигурировать контакт кнопки как вход
pinMode (BUTTON, INPUT);
}
void loop()
{
tekButton=digitalRead(BUTTON);
if (tekButton == HIGH && prevButton == LOW)
{
// нажатие кнопки – изменить состояние светодиода
ledOn=!ledOn;
digitalWrite(LED, ledOn);
}
prevButton=tekButton;
}
```

Листинг 1.3

```
const int LED=10; // Контакт 10 для подключения светодиода
const int BUTTON=2; // Контакт 2 для подключения кнопки
int tekButton = LOW; // Переменная для сохранения текущего состояния кнопки
int prevButton = LOW; // Переменная для сохранения предыдущего состояния
// к нопки
boolean ledOn = false; // Текущее состояние светодиода (включен/выключен)
void setup()
{
// Сконфигурировать контакт светодиода как выход
pinMode (LED, OUTPUT);
```

```
// Сконфигурировать контакт кнопки как вход
pinMode (BUTTON, INPUT);
}
// Функция сглаживания дребезга. Принимает в качестве
// аргумента предыдущее состояние кнопки и выдает фактическое.
boolean debounce(boolean last)
{
boolean current = digitalRead(BUTTON); // Считать состояние кнопки,
if (last != current) // если изменилось...
{
delay(5); // ждем 5 м с
current = digitalRead(BUTTON); // считываем состояние кнопки
return current; // возвращаем состояние кнопки
}
}
void loop()
{
tekButton = debounce(prevButton);
if (prevButton == LOW && tekButton == HIGH) // если нажатие...
{
ledOn = !ledOn; // инвертировать значение состояния светодиода
}
prevButton = tekButton;
digitalWrite(LED, ledOn); // изменить статус состояния светодиода
}
```

Листинг 1.4

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);

void setup()
{
lcd.begin(16, 2);
lcd.clear();
lcd.print("Egor !");
}
void loop()
{
lcd.setCursor(0,1);
lcd.print(millis()/1000);
```

Листинг 2.1

```
// Подключение библиотек
#include <SPI.h>
#include <MFRC522.h>
// константы подключения контактов SS и RST
#define RST_PIN 9
#define SS_PIN 10
// Инициализация MFRC522
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.
void setup()
{
Serial.begin(9600); // инициализация последовательного порта
SPI.begin(); // инициализация SPI
mfrc522.PCD_Init(); // инициализация MFRC522
}
void loop()
{
if ( ! mfrc522.PICC_IsNewCardPresent())
return;
// чтение карты
if ( ! mfrc522.PICC_ReadCardSerial())
return;
```

```
// показать результат чтения UID и тип метки
Serial.print(F("Card UID:"));
dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
Serial.println();
Serial.print(F("PICC type: "));
byte piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
Serial.println(mfrc522.PICC_GetTypeName(piccType));
delay(2000);
}
// Вывод результата чтения данных в HEX-виде
void dump_byte_array(byte *buffer, byte bufferSize)
{
for (byte i = 0; i < bufferSize; i++)
{
Serial.print(buffer[i] < 0x10 ? " 0" : " ");
Serial.print(buffer[i], HEX);
}
}
```

Листинг 2.2

```
// Подключение библиотек
#include <SPI.h>
#include <MFRC522.h>
// константы подключения контактов SS и RST
#define RST_PIN 9
#define SS_PIN 10
// Инициализация MFRC522
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.
MFRC522:MIFARE_Key key;
byte sector = 1;
byte blockAddr = 4;
byte dataBlock[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
byte trailerBlock = 7;
byte status;
byte buffer[18];
byte size = sizeof(buffer);
void setup()
{
Serial.begin(9600); // инициализация последовательного порта
SPI.begin(); // инициализация SPI
mfrc522.PCD_Init(); // инициализация MFRC522
// Значение ключа (A или B) – FFFFFFFFFFFFh значение с завода
for (byte i = 0; i < 6; i++)
key.keyByte[i] = 0xFF;
}
void loop()
{
if ( ! mfrc522.PICC_IsNewCardPresent())
return;
// чтение карты
if ( ! mfrc522.PICC_ReadCardSerial())
return;
// показать результат чтения UID и тип метки
Serial.print(F("Card UID:"));
dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
Serial.println();
Serial.print(F("PICC type: "));
byte piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
Serial.println(mfrc522.PICC_GetTypeName(piccType));
// Чтение данных из блока 4
Serial.print(F("Reading data from block "));
Serial.print(blockAddr);
```

```
Serial.println(F(" ..."));
Serial.print(F("Data for count ")); Serial.print(blockAddr);
Serial.println(F(":"));
dump_byte_array(buffer, 2); Serial.println();
Serial.println();
for (byte i = 0; i < 16; i++) // запись в buffer[]
dataBlock[i]=buffer[i];
// получение байт счетчика (0 и 1)
int count1=(buffer[0]<<8)+buffer[1];
Serial.print("count1=");Serial.println(count1);
count1=count1+1; // инкремент счетчика
dataBlock[0]=highByte(count1);
dataBlock[1]=lowByte(count1);
// Аутентификация key B
Serial.println(F("Authenticating again using key B..."));
// Запись данных в блок
Serial.print(F("Writing data into block "));
Serial.print(blockAddr);
Serial.println(F(" ..."));
dump_byte_array(dataBlock, 2); Serial.println();
}
// Вывод результата чтения данных в HEX-виде
void dump_byte_array(byte *buffer, byte bufferSize)
{
for (byte i = 0; i < bufferSize; i++)
{
Serial.print(buffer[i] < 0x10 ? " 0" : " ");
Serial.print(buffer[i], HEX);
}
}
```

Листинг 3.1

```
int detectedLED = 13;                 // Указываем пин
int readyLED = 12;                    // Указываем пин
int waitLED = 11;                     // Указываем пин

int pirPin = 7;                    // Указываем пин датчика
int motionDetected = 0;            // Переменная для обнаружения движения
int pirValue;                      // Переменная для сохранения значения из PIR

void setup()
{
  pinMode(detectedLED, OUTPUT);       // Установка пин как выход
  pinMode(readyLED, OUTPUT);          // Установка пин как выход
  pinMode(waitLED, OUTPUT);           // Установка пин как выход
  pinMode(pirPin, INPUT);             // Установка пин как вход

// Начальная задержка 1 минута, для стабилизации датчика//
  digitalWrite(detectedLED, LOW);
  digitalWrite(readyLED, LOW);
  digitalWrite(waitLED, HIGH);
  delay(60000);
  digitalWrite(readyLED, HIGH);
  digitalWrite(waitLED, LOW);
}

void loop()
{
  pirValue = digitalRead(pirPin);// Считываем значение от датчикадвижения
  if (pirValue == 1)             // Если движение есть, делаем задержку в 3с.
   {
    digitalWrite(detectedLED, HIGH);
    motionDetected = 1;
```

```
    delay(3000);
    }
    else
    {
    digitalWrite(detectedLED, LOW);
    }

// Задержка после срабатывания //
  if (motionDetected == 1)
   {
    digitalWrite(detectedLED, LOW);
    digitalWrite(readyLED, LOW);
    digitalWrite(waitLED, HIGH);
    delay(6000);
    digitalWrite(readyLED, HIGH);
    digitalWrite(waitLED, LOW);
    motionDetected = 0;
   }
}
```

Листинг 4.1

```
#include "DHT.h"
#define DHTPIN 2 // Тот самый номер пина, о котором упоминалось выше
// Одна из следующих строк закоментирована. Снимите комментарий, если подклю-
чаете датчик DHT11 к arduino
DHT dht(DHTPIN, DHT22); //Инициация датчика
//DHT dht(DHTPIN, DHT11);
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  delay(2000); // 2 секунды задержки
  float h = dht.readHumidity(); //Измеряем влажность
  float t = dht.readTemperature(); //Измеряем температуру
  if (isnan(h) || isnan(t)) {  // Проверка. Если не удается считать показания,
выводится «Ошибка считывания», и программа завершает работу
    Serial.println("Ошибка считывания");
    return;
  }
  Serial.print("Влажность: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Температура: ");
  Serial.print(t);
  Serial.println(" *C "); //Вывод показателей на экран
}
```

Листинг 5.1

```
#include <OneWire.h>
OneWire ds(10); // линия 1-Wire будет на pin 10
#include <LiquidCrystal.h>
// инициализация с указанием контактов подключения
LiquidCrystal lcd(12, 11, 7, 6, 5, 4);
void setup(void)
{
Serial.begin(9600);
// установить размерность дисплея
lcd.begin(16, 2);
lcd.clear();
}
void loop(void)
{
int t=get_temp();
lcd.setCursor(0,1);lcd.print(" ");
```

```
lcd.setCursor(0,1);
lcd.print(t/16);lcd.print(".");lcd.print((t%16)*100/16);
}
// получение данных с датчика DS18B20
int get_temp()
{
byte i;
byte present = 0;
byte data[12];
byte addr[8];
int Temp;
if ( !ds.search(addr))
{Serial.print("No more addresses.\n");
ds.reset_search();
return -1;
}
// вывод в монитор уникального адреса 1-Wire устройства
lcd.setCursor(0,0);
lcd.print("R=");
for( i = 0; i < 8; i++)
{lcd.print(addr[i], HEX);lcd.print(" ");}
if ( OneWire:crc8( addr, 7) != addr[7])
{
Serial.print("CRC is not valid!\n");
return -1;
}
if ( addr[0] != 0x28)
{
Serial.print("Device is not a DS18S20 family device.\n");
return -1;
}
ds.reset();
// 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];
return Temp;
}
```

Листинг 6.1

```
#include <Servo.h> // подключение библиотеки Servo
Servo servo1;
const int pinServo=8; // пин для подключения сервопривода
int pos = 0; // переменная для хранения позиции сервопривода
int dir =1; // направление перемещения сервопривода
// Выводы для подключения HC-SR04 Trig - 12, Echo - 13
Ultrasonic ultrasonic(12, 13);
float dist_cm; // переменная для дистанции, см
// подключить динамик к pin 9
int speakerPin = 9;
void setup()
{
// подключить переменную servo1 к выводу pinServo1
servo1.attach(pinServo1);
pinMode(speakerPin, OUTPUT);
}
void loop()
{
servo1.write(pos); // поворот сервоприводов на полученный угол
delay(15); // пауза для ожидания поворота сервоприводов
```

```
float dist_cm = ultrasonic.Ranging(CM);
if(dist_cm<100 && dist_cm>20)
tone(speakerPin,); // включить пьезозуммер
else
{
tone(speakerPin,0); // отключить пьезозуммер
pos=pos+dir; // изменение переменной положения сервопривода
if(pos==0 || pos==180)
dir=dir*(-1); // изменение направления движения
}
}
```

Листинг 7.1

```
boolean statuslamp; // состояние лампы: true - включено, false - выключено

void setup() {
   pinMode(12,OUTPUT); // пин 12 со светодиодом будет выходом (англ. «output»)
   pinMode(A0,INPUT); // к аналоговому входу A0 подключим датчик (англ. «intput»)
   statuslamp=false; // начальное состояние - лампа выключена
   Serial.begin(9600); // подключаем монитор порта
}

void loop() {
   Serial.println (analogRead(A0)); // выводим значение датчика на монитор

   if(analogRead(A0)>60) {
      statuslamp=!statuslamp; // меняем статус лампы при регистрации хлопка
      digitalWrite(12,statuslamp); // переключаем светодиод на выходе 12
      delay(20); // задержка, "дребезга" хлопков
   }
}
```

Листинг 8.1

```
int light; // переменная для хранения данных фоторезистора
void setup()
{
Serial.begin(9600);
}
void loop()
{
light = analogRead(0);
Serial.println(light);
delay(100);
}
```

Листинг 8.2

```
// Контакт подключения светодиодов
const int leds[]={3,4,5,6,7,8,9,10};
const int LIGHT=A0; // Контакт A0 для входа фоторезистора
const int MIN_LIGHT=200; // Нижний порог освещенности
const int MAX_LIGHT=900; // верхний порог освещенности
// Переменная для хранения данных фоторезистора
int val = 0;
void setup()
{
// Сконфигурировать контакты светодиодов как выход
for(int i=0;i<8;i++)
pinMode(leds[i],OUTPUT);
}
void loop()
{
val = analogRead(LIGHT); // Чтение показаний фоторезистора
// Применение функции map()
val = map(val, MIN_LIGHT, MAX_LIGHT, 8, 0);
// ограничиваем, чтобы не превысило границ
val = constrain(val, 0, 8);
// зажечь кол-во светодиодов, пропорциональное освещенности,
```

```
// остальные потушить
for(int i=1;i<9;i++)
{
if(i>=val) // зажечь светодиоды
digitalWrite(leds[i-1],HIGH);
else // потушить светодиоды
digitalWrite(leds[i-1],LOW);
}
delay(1000); // пауза перед следующим измерением
}
```

Листинг 9.1

```
#define INTERVAL_GET_DATA 2000  // интервала 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;
}
```

Листинг 10.1

```
#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;
}
```

Листинг 11.1

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

#define INTERVAL_GET_DATA 2000  // интервала измерений, мс
// пин, к которому подключен датчик
#define MQ2PIN         A10
// создаём 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;
}
```

Листинг 12.1

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

#define INTERVAL_GET_DATA 2000  // интервала измерений, мс
// пин, к которому подключен датчик
#define MQ7PIN         A11
// создаём 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;
}
```

Листинг 13.1

```
#define INTERVAL_GET_DATA 2000  // интервала 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;
}
```

Листинг 14.1

```
/*
 IoT Manager mqtt device client
https://play.google.com/store/apps/details?id=ru.esp8266.iotmanager
 Based on Basic MQTT example with Authentication
 PubSubClient library v 1.91.1 https://github.com/Imroy/pubsubclient
  - connects to an MQTT server, providing userdescr and password
  - publishes config to the topic "/IoTmanager/config/deviceID/"
  - subscribes to the topic "/IoTmanager/hello" ("hello" messages from mobile de-
vice)
  Tested with Arduino IDE 1.6.6 + ESP8266 Community Edition v 2.0.0-stable and
PubSubClient library v 1.91.1 https://github.com/Imroy/pubsubclient
  ESP8266 Community Edition v 2.0.0-stable have some HTTPS issues. Push notifica-
tion temporary disabled.
  sketch version : 1.5
  IoT Manager    : any version
  toggle, range, small-badge and power-button widgets demo
//////////////////////////////////////////////////////
//  Attention for ESP-01 users!
//  at line 167 change value 15 (GPIO15) to another.
//  (on ESP-01 GPIO15 pin connected directly to GND and "red" PWM out may overload
GPIO15 line). Repoted by grigorygn
http://www.esp8266.com/viewtopic.php?p=40407#p40407
//////////////////////////////////////////////////////
*/

#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
```

```
const char *ssid =  "MacBook-Pro-Victor";            // cannot be longer than 32
characters!
const char *pass =  "19101966";       // WiFi password

String prefix   = "/IoTmanager";     // global prefix for all topics - must be
some as mobile device
String deviceID = "dev02-bedroom";   // thing ID - unique device id in our project

WiFiClient wclient;

// config for cloud mqtt broker by DNS hostname ( for example, cloudmqtt.com use:
m20.cloudmqtt.com - EU, m11.cloudmqtt.com - USA )
String mqttServerName = "m13.cloudmqtt.com";            // for cloud broker - by
hostname, from CloudMQTT account data
int    mqttport = 18274;                                // default 1883, but
CloudMQTT.com use other, for example: 13191, 23191 (SSL), 33191 (WebSockets) - use
from CloudMQTT account data
String mqttuser =  "nodemcu1";                             // from CloudMQTT ac-
count data
String mqttpass =  "191066";                             // from CloudMQTT ac-
count data
PubSubClient client(wclient, mqttServerName, mqttport); // for cloud broker - by
hostname


// config for local mqtt broker by IP address
//IPAddress server(192, 168, 1, 100);                       // for local broker -
by address
//int    mqttport = 1883;                                   // default 1883
//String mqttuser =  "";                                    // from broker config
//String mqttpass =  "";                                    // from broker config
//PubSubClient client(wclient, server, mqttport);           // for local broker -
by address

String val;
String ids = "";
int oldtime, newtime, pushtime;
int newValue;

const String stat1 = "{\"status\":\"1\"}";
const String stat0 = "{\"status\":\"0\"}";

const int nWidgets = 8;
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];
int    defaultVal  [nWidgets];
bool   inverted    [nWidgets];

// Push notifications
const char* host = "onesignal.com";
WiFiClientSecure httpClient;
const int httpsPort = 443;
String url = "/api/v1/notifications";

void push(String msg) {
```

```
  Serial.println("PUSH: try to send push notification...");
  if (ids.length() == 0) {
     Serial.println("PUSH: ids not received, push failed");
     return;
  }
  if (!httpClient.connect(host, httpsPort)) {
     Serial.println("PUSH: connection failed");
     return;
  }
  String data = "{\"app_id\": \"8871958c-5f52-11e5-8f7a-
c36f5770ade9\",\"include_player_ids\":[\"" + ids + "\"],\"android_group\":\"IoT
Manager\",\"contents\": {\"en\": \"" + msg + "\"}}";
  httpClient.println("POST " + url + " HTTP/1.1");
  httpClient.print("Host:");
  httpClient.println(host);
  httpClient.println("User-Agent: esp8266.Arduino.IoTmanager");
  httpClient.print("Content-Length: ");
  httpClient.println(data.length());
  httpClient.println("Content-Type: application/json");
  httpClient.println("Connection: close");
  httpClient.println();
  httpClient.println(data);
  httpClient.println();
  Serial.println(data);
  Serial.println("PUSH: done.");
}
String setStatus ( String s ) {
  String stat = "{\"status\":\"" + s + "\"}";
  return stat;
}
String setStatus ( int s ) {
  String stat = "{\"status\":\"" + String(s) + "\"}";
  return stat;
}
void initVar() {
  id    [0] = "0";
  page  [0] = "Bedroom";
  descr [0] = "Bedroom light-0";
  widget[0] = "toggle";
  pin[0] = 4;                                              // GPIO4 - toggle
  defaultVal[0] = 1;                                       // defaultVal status
  inverted[0] = true;
  sTopic[0]   = prefix + "/" + deviceID + "/light0";
  color[0]   = "\"color\":\"red\"";                       // black, blue, green,
orange, red, white, yellow (off - grey)

  id    [1] = "1";
  page  [1] = "Bedroom";
  descr [1] = "Bedroom light-1";
  widget[1] = "toggle";
  pin[1] = 5;                                              // GPIO5 - toggle
  defaultVal[1] = 1;                                       // defaultVal status
  inverted[1] = true;
  sTopic[1] = prefix + "/" + deviceID + "/light1";
  color [1] = "\"color\":\"white\"";                       // black, blue, green,
orange, red, white, yellow (off - grey)

  id    [2] = "2";
  page  [2] = "Bedroom";
  descr [2] = "Bedroom light-2";
  widget[2] = "toggle";
  pin[2] = 0;                                              // GPIO0 - toggle
  defaultVal[2] = 1;                                       // defaultVal status
  inverted[1] = true;
  sTopic[2]   = prefix + "/" + deviceID + "/light2";
```

```
  color [2] = "\"color\":\"yellow\"";                      // black, blue, green,
orange, red, white, yellow (off - grey)

  id    [3] = "3";
  page  [3] = "Bedroom";
  descr  [3] = "ADC";
  widget[3] = "small-badge";
  pin   [3] = A0;                                          // ADC
  sTopic[3] = prefix + "/" + deviceID + "/ADC";
  badge [3] = "\"badge\":\"badge-calm\"";                  // see
http://ionicframework.com/docs/components/#colors
  style [3]   = "\"style\":\"font-size:150%;\"";

  id    [4] = "4";
  page  [4] = "Bedroom";
  descr [4] = "KEY";
  widget[4] = "power-button";
  pin   [4] = 2;                                           // GPIO2
  inverted[4] = true;
  sTopic[4] = prefix + "/" + deviceID + "/light4";
  style [4]   = "\"style\":\"font-size:150%;\"";

  // RED
  id    [5] = "5";
  page  [5] = "Bedroom";
  descr [5] = "Bedroom RED";
  widget[5] = "range";
  // ATTENTION! if you use ESP-01 module, then not use GPIO15
  pin   [5] = 15;                                          // GPIO15 - range
  defaultVal[5] = 0;                                       // defaultVal 0%, not
inverted
  sTopic[5]   = prefix + "/" + deviceID + "/red";
  style[5]   = "\"style\":\"range-assertive\"";            // see
http://ionicframework.com/docs/components/#colors
  badge[5]   = "\"badge\":\"badge-assertive\"";            // see
http://ionicframework.com/docs/components/#colors


  // GREEN
  id    [6] = "6";
  page  [6] = "Bedroom";
  descr [6] = "Bedroom GREEN";
  widget[6] = "range";
  pin[6] = 12;                                        // GPIO12 - range
  defaultVal[6] = 0;                                       // defaultVal 0%
  sTopic[6]   = prefix + "/" + deviceID + "/green";
  style[6]   = "\"style\":\"range-balanced\"";         // see
http://ionicframework.com/docs/components/#colors
  badge[6]   = "\"badge\":\"badge-balanced\"";          // see
http://ionicframework.com/docs/components/#colors

  // BLUE
  id    [7] = "7";
  page  [7] = "Bedroom";
  descr [7] = "Bedroom BLUE";
  widget[7] = "range";
  pin[7] = 13;                                        // GPIO13 - range
  defaultVal[7] = 0;                                       // defaultVal status 0%
  sTopic[7]  = prefix + "/" + deviceID + "/blue";
  style[7]   = "\"style\":\"range-calm\"";             // see
http://ionicframework.com/docs/components/#colors
  badge[7]   = "\"badge\":\"badge-calm\"";              // see
http://ionicframework.com/docs/components/#colors

  for (int i = 0; i < nWidgets; i++) {
```

```
    if (inverted[i]) {
      if (defaultVal[i]>0) {
         stat[i] = setStatus(0);
      } else {
         stat[i] = setStatus(1);
      }
    } else {
       stat[i] = setStatus(defaultVal[i]);
    }
  }

  thing_config[0] = "{\"id\":\"" + id[0] + "\",\"page\":\"" +
page[0]+"\",\"descr\":\"" + descr[0] + "\",\"widget\":\"" + widget[0] +
"\",\"topic\":\"" + sTopic[0] + "\"," + color[0] + "}";   // GPIO switched On/Off
by mobile widget toggle
  thing_config[1] = "{\"id\":\"" + id[1] + "\",\"page\":\"" +
page[1]+"\",\"descr\":\"" + descr[1] + "\",\"widget\":\"" + widget[1] +
"\",\"topic\":\"" + sTopic[1] + "\"," + color[1] + "}";   // GPIO switched On/Off
by mobile widget toggle
  thing_config[2] = "{\"id\":\"" + id[2] + "\",\"page\":\"" +
page[2]+"\",\"descr\":\"" + descr[2] + "\",\"widget\":\"" + widget[2] +
"\",\"topic\":\"" + sTopic[2] + "\"," + color[2] + "}";   // GPIO switched On/Off
by mobile widget toggle
  thing_config[3] = "{\"id\":\"" + id[3] + "\",\"page\":\"" +
page[3]+"\",\"descr\":\"" + descr[3] + "\",\"widget\":\"" + widget[3] +
"\",\"topic\":\"" + sTopic[3] + "\"," + badge[3] + "," + style[3] + "}";  // ADC
  thing_config[4] = "{\"id\":\"" + id[4] + "\",\"page\":\"" +
page[4]+"\",\"descr\":\"" + descr[4] + "\",\"widget\":\"" + widget[4] +
"\",\"topic\":\"" + sTopic[4] + "\"," + style[4] + "}";
  thing_config[5] = "{\"id\":\"" + id[5] + "\",\"page\":\"" +
page[5]+"\",\"descr\":\"" + descr[5] + "\",\"widget\":\"" + widget[5] +
"\",\"topic\":\"" + sTopic[5] + "\"," + style[5] + ","+ badge[5] + "}";    //
GPIO15 R
  thing_config[6] = "{\"id\":\"" + id[6] + "\",\"page\":\"" +
page[6]+"\",\"descr\":\"" + descr[6] + "\",\"widget\":\"" + widget[6] +
"\",\"topic\":\"" + sTopic[6] + "\"," + style[6] + ","+ badge[6] + "}";    //
GPIO12 G
  thing_config[7] = "{\"id\":\"" + id[7] + "\",\"page\":\"" +
page[7]+"\",\"descr\":\"" + descr[7] + "\",\"widget\":\"" + widget[7] +
"\",\"topic\":\"" + sTopic[7] + "\"," + style[7] + ","+ badge[7] + "}";    //
GPIO13 B

}
// send confirmation
void pubStatus(String t, String payload) {
    if (client.publish(t + "/status", payload)) {
       Serial.println("Publish new status for " + t + ", value: " + payload);
    } else {
       Serial.println("Publish new status for " + t + " FAIL!");
    }
}
void pubConfig() {
  bool success;
  success = client.publish(MQTT::Publish(prefix, deviceID).set_qos(1));
  if (success) {
      delay(500);
      for (int i = 0; i < nWidgets; i = i + 1) {
        success = client.publish(MQTT::Publish(prefix + "/" + deviceID +
"/config", thing_config[i]).set_qos(1));
        if (success) {
          Serial.println("Publish config: Success (" + thing_config[i] + ")");
        } else {
          Serial.println("Publish config FAIL! ("    + thing_config[i] + ")");
        }
        delay(150);
```

```
      }
  }
  if (success) {
     Serial.println("Publish config: Success");
  } else {
     Serial.println("Publish config: FAIL");
  }
  for (int i = 0; i < nWidgets; i = i + 1) {
      pubStatus(sTopic[i], stat[i]);
      delay(100);
  }
}
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") {
    if (sub.payload_string() == "0") {
       newValue = 1; // inverted
       stat[0] = stat0;
    } else {
       newValue = 0;
       stat[0] = stat1;
    }
    digitalWrite(pin[0],newValue);
    pubStatus(sTopic[0], stat[0]);
 } else if (sub.topic() == sTopic[1] + "/control") {
    if (sub.payload_string() == "0") {
       newValue = 1; // inverted
       stat[1] = stat0;
    } else {
       newValue = 0; // inverted
       stat[1] = stat1;
    }
    digitalWrite(pin[1],newValue);
    pubStatus(sTopic[1], stat[1]);
 } 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;
    }
    digitalWrite(pin[2],newValue);
    pubStatus(sTopic[2], stat[2]);
 } else if (sub.topic() == sTopic[3] + "/control") {
   // ADC : nothing, display only
 } else if (sub.topic() == sTopic[4] + "/control") {
   // nothing, display only
 } else if (sub.topic() == sTopic[5] + "/control") {
    String x = sub.payload_string();
    analogWrite(pin[5],x.toInt());
    stat[5] = setStatus(x);
    pubStatus(sTopic[5], stat[5]);
 } else if (sub.topic() == sTopic[6] + "/control") {
    String x = sub.payload_string();
    analogWrite(pin[6],x.toInt());
    stat[6] = setStatus(x);
    pubStatus(sTopic[6], stat[6]);
 } else if (sub.topic() == sTopic[7] + "/control") {
    String x = sub.payload_string();
    analogWrite(pin[7],x.toInt());
```

```
    stat[7] = setStatus(x);
    pubStatus(sTopic[7], stat[7]);
 } else if (sub.topic() == prefix + "/ids") {
    ids = sub.payload_string();
    //push();
 } else if (sub.topic() == prefix) {
    if (sub.payload_string() == "HELLO") {
      pubConfig();
    }
 }
}

void setup() {
  initVar();
  pinMode(pin[0], OUTPUT);
  digitalWrite(pin[0],defaultVal[0]);
  pinMode(pin[1], OUTPUT);
  digitalWrite(pin[1],defaultVal[1]);
  pinMode(pin[2], OUTPUT);
  digitalWrite(pin[2],defaultVal[2]);
  stat[3] = setStatus(analogRead(pin[3]));
  pinMode(pin[4], INPUT);
  stat[4] = setStatus(digitalRead(pin[4]));
  pinMode(pin[5], OUTPUT);
  analogWrite(pin[5],defaultVal[5]);  // PWM
  pinMode(pin[6], OUTPUT);
  analogWrite(pin[6],defaultVal[6]);  // PWM
  pinMode(pin[7], OUTPUT);
  analogWrite(pin[7],defaultVal[7]);  // PWM
  // Setup console
  Serial.begin(115200);
  delay(10);
  Serial.println();
  Serial.println();
  Serial.println("MQTT client started.");
  Serial.print("Free heap = ");
  Serial.println(ESP.getFreeHeap());
}

void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.print("Connecting via WiFi to ");
    Serial.print(ssid);
    Serial.println("...");

    WiFi.begin(ssid, pass);

    if (WiFi.waitForConnectResult() != WL_CONNECTED) {
      return;
    }

    Serial.println("");
    Serial.println("WiFi connect: Success");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
  }

  if (WiFi.status() == WL_CONNECTED) {
    if (!client.connected()) {
      Serial.println("Connecting to MQTT server ...");
      bool success;
      if (mqttuser.length() > 0) {
        success = client.connect( MQTT::Connect( deviceID ).set_auth(mqttuser,
mqttpass) );
      } else {
```

```
        success = client.connect( deviceID );
      }
      if (success) {
        client.set_callback(callback);
        Serial.println("Connect to MQTT server: Success");
        pubConfig();
        client.subscribe(prefix);                  // for receiving HELLO messages
        client.subscribe(prefix + "/ids");         // for receiving IDS  messages
        client.subscribe(sTopic[0] + "/control"); // for receiving GPIO messages
        client.subscribe(sTopic[1] + "/control"); // for receiving GPIO messages
        client.subscribe(sTopic[2] + "/control"); // for receiving GPIO messages
        // 3 - display only, no control
        client.subscribe(sTopic[4] + "/control"); // for receiving GPIO messages
        client.subscribe(sTopic[5] + "/control"); // for receiving GPIO messages
        client.subscribe(sTopic[6] + "/control"); // for receiving GPIO messages
        client.subscribe(sTopic[7] + "/control"); // for receiving GPIO messages

        Serial.println("Subscribe: Success");
      } else {
        Serial.println("Connect to MQTT server: FAIL");
        delay(1000);
      }
    }

    if (client.connected()) {
      newtime = millis();
      if (newtime - oldtime > 10000) { // 10 sec
        int x = analogRead(pin[3]);
        val = "{\"status\":\"" + String(x)+ "\"}";
        client.publish(sTopic[3] + "/status", val );  // widget 3
        oldtime = newtime;
        if ((millis()-pushtime > 10000) && (x > 100)) {
           String msg = "Bedroom ADC more then 100! (" + String(x) + ")";
           //
           // ESP8266 Community Edition v 2.0.0-stable have some HTTPS issues.
Push notification temporary disables. Please, uncomment next line if you use fu-
ture versions.
           //
           // push(msg);
           //
           //
           pushtime = millis();
        }
      }
      int key;
      key = digitalRead(pin[4]);
      if ( stat[4] != setStatus(key) ) {
        stat[4] = setStatus(key);
        pubStatus(sTopic[4], stat[4] );  // widget 4
      }
      client.loop();
    }
  }
}
```

Листинг 14.2

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>

const char *ssid =  "MacBook-Pro-Victor";            // cannot be longer than 32
characters!
const char *pass =  "19101966";       // WiFi password

String prefix   = "/IoTmanager";     // global prefix for all topics - must be
some as mobile device
```

```
String deviceID = "SmarhouseNodemcu";   // thing ID - unique device id in our
project

WiFiClient wclient;

// config for cloud mqtt broker by DNS hostname ( for example, cloudmqtt.com use:
m20.cloudmqtt.com - EU, m11.cloudmqtt.com - USA )
String mqttServerName = "m13.cloudmqtt.com";            // for cloud broker - by
hostname, from CloudMQTT account data
int    mqttport = 18274;                                // default 1883, but
CloudMQTT.com use other, for example: 13191, 23191 (SSL), 33191 (WebSockets) - use
from CloudMQTT account data
String mqttuser =  "nodemcu1";                             // from CloudMQTT ac-
count data
String mqttpass =  "191066";                             // from CloudMQTT ac-
count data
PubSubClient client(wclient, mqttServerName, mqttport); // for cloud broker - by
hostname

// подключение библиотеки DTH
#include "DHT.h"
// константы
#define DHTPIN 4           // пин (D2) подключения контакта DATA
#define DHTTYPE DHT22      // датчик DHT 22
// создание экземпляра объекта DHT
DHT dht(DHTPIN, DHTTYPE);

#define DSD18B20PIN 5      // пин подключения контакта DATA (D1, GPIO5)
// подключение библиотеки
#include <OneWire.h>
// создание экземпляра OneWire
OneWire  ds(DSD18B20PIN);


String val;
String ids = "";
int oldtime, newtime, pushtime, oldtime2;
int newValue;

const String stat1 = "{\"status\":\"1\"}";
const String stat0 = "{\"status\":\"0\"}";

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];

// Push notifications
const char* host = "onesignal.com";
WiFiClientSecure httpClient;
const int httpsPort = 443;
String url = "/api/v1/notifications";

void push(String msg) {
  Serial.println("PUSH: try to send push notification...");
```

```
  if (ids.length() == 0) {
     Serial.println("PUSH: ids not received, push failed");
     return;
  }
  if (!httpClient.connect(host, httpsPort)) {
     Serial.println("PUSH: connection failed");
     return;
  }
  String data = "{\"app_id\": \"8871958c-5f52-11e5-8f7a-
c36f5770ade9\",\"include_player_ids\":[\"" + ids + "\"],\"android_group\":\"IoT
Manager\",\"contents\": {\"en\": \"" + msg + "\"}}";
  httpClient.println("POST " + url + " HTTP/1.1");
  httpClient.print("Host:");
  httpClient.println(host);
  httpClient.println("User-Agent: esp8266.Arduino.IoTmanager");
  httpClient.print("Content-Length: ");
  httpClient.println(data.length());
  httpClient.println("Content-Type: application/json");
  httpClient.println("Connection: close");
  httpClient.println();
  httpClient.println(data);
  httpClient.println();
  Serial.println(data);
  Serial.println("PUSH: done.");
}
String setStatus ( String s ) {
  String stat = "{\"status\":\"" + s + "\"}";
  return stat;
}
String setStatus ( int s ) {
  String stat = "{\"status\":\"" + String(s) + "\"}";
  return stat;
}
void initVar() {


  id    [0] = "0";
  page  [0] = "SensorsNodemcu";
  descr  [0] = "DHT22";
  widget[0] = "small-badge";
  //pin   [0] = A0;                                        // ADC
  sTopic[0] = prefix + "/" + deviceID + "/DHT22";
  badge [0] = "\"badge\":\"badge-calm\"";                  // see
http://ionicframework.com/docs/components/#colors
  style [0]   = "\"style\":\"font-size:150%;\"";

  id    [1] = "1";
  page  [1] = "SensorsNodemcu";
  descr  [1] = "DS18b20";
  widget[1] = "small-badge";
  //pin   [1] = A0;                                        // ADC
  sTopic[1] = prefix + "/" + deviceID + "/DS18b20";
  badge [1] = "\"badge\":\"badge-calm\"";                  // see
http://ionicframework.com/docs/components/#colors
  style [1]   = "\"style\":\"font-size:150%;\"";


  for (int i = 0; i < nWidgets; i++) {
    if (inverted[i]) {
      if (defaultVal[i]>0) {
         stat[i] = setStatus(0);
      } else {
         stat[i] = setStatus(1);
      }
    } else {
```

```
       stat[i] = setStatus(defaultVal[i]);
    }
  }

  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
  //thing_config[2] = "{\"id\":\"" + id[2] + "\",\"page\":\"" +
page[2]+"\",\"descr\":\"" + descr[2] + "\",\"widget\":\"" + widget[2] +
"\",\"topic\":\"" + sTopic[2] + "\"," + color[2] + "}";   // GPIO switched On/Off
by mobile widget toggle

}
// send confirmation
void pubStatus(String t, String payload) {
    if (client.publish(t + "/status", payload)) {
       Serial.println("Publish new status for " + t + ", value: " + payload);
    } else {
       Serial.println("Publish new status for " + t + " FAIL!");
    }
}
void pubConfig() {
  bool success;
  success = client.publish(MQTT::Publish(prefix, deviceID).set_qos(1));
  if (success) {
      delay(500);
      for (int i = 0; i < nWidgets; i = i + 1) {
        success = client.publish(MQTT::Publish(prefix + "/" + deviceID +
"/config", thing_config[i]).set_qos(1));
        if (success) {
          Serial.println("Publish config: Success (" + thing_config[i] + ")");
        } else {
          Serial.println("Publish config FAIL! ("    + thing_config[i] + ")");
        }
        delay(150);
      }
  }
  if (success) {
     Serial.println("Publish config: Success");
  } else {
     Serial.println("Publish config: FAIL");
  }
  for (int i = 0; i < nWidgets; i = i + 1) {
      pubStatus(sTopic[i], stat[i]);
      delay(100);
  }
}
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() == prefix) {
    if (sub.payload_string() == "HELLO") {
      pubConfig();
    }
 }
}

void setup() {
  initVar();
  pinMode(pin[0], INPUT);
  stat[0] = setStatus(digitalRead(pin[0]));
  pinMode(pin[1], INPUT);
  stat[1] = setStatus(digitalRead(pin[1]));
  // Setup console
  Serial.begin(115200);
  delay(10);
  Serial.println();
  Serial.println();
  Serial.println("MQTT client started.");
  Serial.print("Free heap = ");
  Serial.println(ESP.getFreeHeap());
  dht.begin();          // запуск DHT
}

void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.print("Connecting via WiFi to ");
    Serial.print(ssid);
    Serial.println("...");

    WiFi.begin(ssid, pass);

    if (WiFi.waitForConnectResult() != WL_CONNECTED) {
      return;
    }

    Serial.println("");
    Serial.println("WiFi connect: Success");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
  }

  if (WiFi.status() == WL_CONNECTED) {
    if (!client.connected()) {
      Serial.println("Connecting to MQTT server ...");
      bool success;
      if (mqttuser.length() > 0) {
        success = client.connect( MQTT::Connect( deviceID ).set_auth(mqttuser,
mqttpass) );
      } else {
        success = client.connect( deviceID );
      }
      if (success) {
        client.set_callback(callback);
        Serial.println("Connect to MQTT server: Success");
        pubConfig();


        Serial.println("Subscribe: Success");
```

```
      } else {
        Serial.println("Connect to MQTT server: FAIL");
        delay(1000);
      }
    }

    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;
      }
      if (newtime - oldtime2 > 60000) { // 60 sec
        pubConfig();
        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
        oldtime2 = newtime;
      }
      client.loop();
    }
  }
}

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

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;
}
```

Листинг 14.3

```
/*
 IoT Manager mqtt device client
https://play.google.com/store/apps/details?id=ru.esp8266.iotmanager
 Based on Basic MQTT example with Authentication
 PubSubClient library v 1.91.1 https://github.com/Imroy/pubsubclient
  - connects to an MQTT server, providing userdescr and password
  - publishes config to the topic "/IoTmanager/config/deviceID/"
  - subscribes to the topic "/IoTmanager/hello" ("hello" messages from mobile de-
vice)
  Tested with Arduino IDE 1.6.6 + ESP8266 Community Edition v 2.0.0-stable and
PubSubClient library v 1.91.1 https://github.com/Imroy/pubsubclient
  ESP8266 Community Edition v 2.0.0-stable have some HTTPS issues. Push notifica-
tion temporary disabled.
  sketch version : 1.5
```

```
  IoT Manager    : any version
  toggle, range, small-badge and power-button widgets demo
//////////////////////////////////////////////////////
//  Attention for ESP-01 users!
//  at line 167 change value 15 (GPIO15) to another.
//  (on ESP-01 GPIO15 pin connected directly to GND and "red" PWM out may overload
GPIO15 line). Repoted by grigorygn
http://www.esp8266.com/viewtopic.php?p=40407#p40407
//////////////////////////////////////////////////////
*/

#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>

#include <Wire.h>

const char *ssid =  "MacBook-Pro-Victor";            // cannot be longer than 32
characters!
const char *pass =  "19101966";       // WiFi password

String prefix   = "/IoTmanager";     // global prefix for all topics - must be
some as mobile device
String deviceID = "SmarhouseNodemcu";   // thing ID - unique device id in our
project

WiFiClient wclient;

// config for cloud mqtt broker by DNS hostname ( for example, cloudmqtt.com use:
m20.cloudmqtt.com - EU, m11.cloudmqtt.com - USA )
String mqttServerName = "m13.cloudmqtt.com";            // for cloud broker - by
hostname, from CloudMQTT account data
int    mqttport = 18274;                                // default 1883, but
CloudMQTT.com use other, for example: 13191, 23191 (SSL), 33191 (WebSockets) - use
from CloudMQTT account data
String mqttuser =  "nodemcu1";                             // from CloudMQTT ac-
count data
String mqttpass =  "191066";                             // from CloudMQTT ac-
count data
PubSubClient client(wclient, mqttServerName, mqttport); // for cloud broker - by
hostname


// config for local mqtt broker by IP address
//IPAddress server(192, 168, 1, 100);                       // for local broker -
by address
//int    mqttport = 1883;                                   // default 1883
//String mqttuser =  "";                                    // from broker config
//String mqttpass =  "";                                    // from broker config
//PubSubClient client(wclient, server, mqttport);           // for local broker -
by address

String val;
String ids = "";
int oldtime, newtime, pushtime, oldtime2;
int newValue;

const String stat1 = "{\"status\":\"1\"}";
const String stat0 = "{\"status\":\"0\"}";

const int nWidgets = 5;
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];

// Push notifications
const char* host = "onesignal.com";
WiFiClientSecure httpClient;
const int httpsPort = 443;
String url = "/api/v1/notifications";

int datarelays[]={0,0,0,0,0,0,0,0};

void push(String msg) {
  Serial.println("PUSH: try to send push notification...");
  if (ids.length() == 0) {
     Serial.println("PUSH: ids not received, push failed");
     return;
  }
  if (!httpClient.connect(host, httpsPort)) {
     Serial.println("PUSH: connection failed");
     return;
  }
  String data = "{\"app_id\": \"8871958c-5f52-11e5-8f7a-
c36f5770ade9\",\"include_player_ids\":[\"" + ids + "\"],\"android_group\":\"IoT
Manager\",\"contents\": {\"en\": \"" + msg + "\"}}";
  httpClient.println("POST " + url + " HTTP/1.1");
  httpClient.print("Host:");
  httpClient.println(host);
  httpClient.println("User-Agent: esp8266.Arduino.IoTmanager");
  httpClient.print("Content-Length: ");
  httpClient.println(data.length());
  httpClient.println("Content-Type: application/json");
  httpClient.println("Connection: close");
  httpClient.println();
  httpClient.println(data);
  httpClient.println();
  Serial.println(data);
  Serial.println("PUSH: done.");
}
String setStatus ( String s ) {
  String stat = "{\"status\":\"" + s + "\"}";
  return stat;
}
String setStatus ( int s ) {
  String stat = "{\"status\":\"" + String(s) + "\"}";
  return stat;
}
void initVar() {


  id    [0] = "0";
  page  [0] = "SensorsNodemcu";
  descr  [0] = "DHT22";
  widget[0] = "small-badge";
  //pin   [0] = A0;                                        // ADC
  sTopic[0] = prefix + "/" + deviceID + "/DHT22";
  badge [0] = "\"badge\":\"badge-calm\"";                  // see
http://ionicframework.com/docs/components/#colors
  style [0]   = "\"style\":\"font-size:150%;\"";
```

```
  id    [1] = "1";
  page  [1] = "SensorsNodemcu";
  descr  [1] = "DS18b20";
  widget[1] = "small-badge";
  //pin   [1] = A0;                                          // ADC
  sTopic[1] = prefix + "/" + deviceID + "/DS18b20";
  badge [1] = "\"badge\":\"badge-calm\"";                  // see
http://ionicframework.com/docs/components/#colors
  style [1]   = "\"style\":\"font-size:150%;\"";

  id    [2] = "2";
  page  [2] = "RelaysNodeMcu";
  descr [2] = "Light";
  widget[2] = "toggle";
  //pin[0] = 4;
  defaultVal[2] = 0;                                        // defaultVal status
  inverted[2] = false;
  sTopic[2]   = prefix + "/" + deviceID + "/light";
  color[2]   = "\"color\":\"red\"";                        // black, blue, green,
orange, red, white, yellow (off - grey)

  id    [3] = "3";
  page  [3] = "RelaysNodeMcu";
  descr [3] = "Pump";
  widget[3] = "toggle";
  //pin[0] = 4;
  defaultVal[3] = 0;                                        // defaultVal status
  inverted[3] = false;
  sTopic[3]   = prefix + "/" + deviceID + "/pump";
  color[3]   = "\"color\":\"red\"";                        // black, blue, green,
orange, red, white, yellow (off - grey)

  id    [4] = "4";
  page  [4] = "RelaysNodeMcu";
  descr [4] = "Fun";
  widget[4] = "toggle";
  //pin[0] = 4;
  defaultVal[4] = 0;                                        // defaultVal status
  inverted[4] = false;
  sTopic[4]   = prefix + "/" + deviceID + "/fun";
  color[4]   = "\"color\":\"red\"";                        // black, blue, green,
orange, red, white, yellow (off - grey)


  for (int i = 0; i < nWidgets; i++) {
    if (inverted[i]) {
      if (defaultVal[i]>0) {
         stat[i] = setStatus(0);
      } else {
         stat[i] = setStatus(1);
      }
    } else {
       stat[i] = setStatus(defaultVal[i]);
    }
  }

  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
```

```
  thing_config[2] = "{\"id\":\"" + id[2] + "\",\"page\":\"" +
page[2]+"\",\"descr\":\"" + 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] + "}";
  }
// send confirmation
void pubStatus(String t, String payload) {
    if (client.publish(t + "/status", payload)) {
       Serial.println("Publish new status for " + t + ", value: " + payload);
    } else {
       Serial.println("Publish new status for " + t + " FAIL!");
    }
}
void pubConfig() {
  bool success;
  success = client.publish(MQTT::Publish(prefix, deviceID).set_qos(1));
  if (success) {
      delay(500);
      for (int i = 0; i < nWidgets; i = i + 1) {
        success = client.publish(MQTT::Publish(prefix + "/" + deviceID +
"/config", thing_config[i]).set_qos(1));
        if (success) {
          Serial.println("Publish config: Success (" + thing_config[i] + ")");
        } else {
          Serial.println("Publish config FAIL! ("    + thing_config[i] + ")");
        }
        delay(150);
      }
  }
  if (success) {
     Serial.println("Publish config: Success");
  } else {
     Serial.println("Publish config: FAIL");
  }
  for (int i = 0; i < nWidgets; i = i + 1) {
      pubStatus(sTopic[i], stat[i]);
      delay(100);
  }
}
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();
    }
 }
}

void setup() {
  initVar();
  pinMode(pin[0], INPUT);
  stat[0] = setStatus(digitalRead(pin[0]));
  pinMode(pin[1], INPUT);
  stat[1] = setStatus(digitalRead(pin[1]));
  // Setup console
  Serial.begin(115200);
  delay(10);

  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();

  Serial.println();
  Serial.println();
  Serial.println("MQTT client started.");
  Serial.print("Free heap = ");
  Serial.println(ESP.getFreeHeap());
}
```

```
void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.print("Connecting via WiFi to ");
    Serial.print(ssid);
    Serial.println("...");

    WiFi.begin(ssid, pass);

    if (WiFi.waitForConnectResult() != WL_CONNECTED) {
      return;
    }

    Serial.println("");
    Serial.println("WiFi connect: Success");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
  }

  if (WiFi.status() == WL_CONNECTED) {
    if (!client.connected()) {
      Serial.println("Connecting to MQTT server ...");
      bool success;
      if (mqttuser.length() > 0) {
        success = client.connect( MQTT::Connect( deviceID ).set_auth(mqttuser,
mqttpass) );
      } else {
        success = client.connect( deviceID );
      }
      if (success) {
        client.set_callback(callback);
        Serial.println("Connect to MQTT server: Success");
        pubConfig();
        client.subscribe(sTopic[2] + "/control"); // for receiving GPIO messages
        client.subscribe(sTopic[3] + "/control"); // for receiving GPIO messages
        client.subscribe(sTopic[4] + "/control"); // for receiving GPIO messages


        Serial.println("Subscribe: Success");
      } else {
        Serial.println("Connect to MQTT server: FAIL");
        delay(1000);
      }
    }

    if (client.connected()) {
      newtime = millis();
      if (newtime - oldtime > 10000) { // 10 sec
        float x = random(40,45);
        val = "{\"status\":\"" + String(x)+ "\"}";
        client.publish(sTopic[0] + "/status", val );  // widget 0
        x = random(18,20);
        x=x+(x+random(100))/100;
        val = "{\"status\":\"" + String(x)+ "\"}";
        client.publish(sTopic[1] + "/status", val );  // widget 1
        oldtime = newtime;
      }
      if (newtime - oldtime2 > 60000) { // 60 sec
        pubConfig();
        float x = random(40,45);
        val = "{\"status\":\"" + String(x)+ "\"}";
        client.publish(sTopic[0] + "/status", val );  // widget 0
        x = random(18,20);
        x=x+(x+random(100))/100;
        val = "{\"status\":\"" + String(x)+ "\"}";
        client.publish(sTopic[1] + "/status", val );  // widget 1
```

```
        oldtime2 = newtime;
      }
      client.loop();
    }
  }
}

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
}
```

Листинг 14.4

```
/*
 Basic MQTT example

  - connects to an MQTT server
  - publishes "hello world" to the topic "potato"
  - subscribes to the topic "batata"
*/

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

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

sim800Client s800;
char imeicode[16];

void callback(char* topic, byte* payload, unsigned int length) {
  // handle message arrived
  char mypl[48];
  Serial.println(length);
  memcpy(mypl,payload,length);
  mypl[length]=char(0);
  Serial.print("receive: ");
  Serial.print(topic);
  Serial.print("->");
  Serial.println(mypl);
}

String prefix   = "/IoTmanager";     // global prefix for all topics - must be
some as mobile device
String deviceID = "SmarhouseArduinoMega";   // thing ID - unique device id in our
project
char server[] = "m13.cloudmqtt.com";  //Mqtt Server
int    mqttport = 18274;                                  // default 1883, but
CloudMQTT.com use other, for example: 13191, 23191 (SSL), 33191 (WebSockets) - use
from CloudMQTT account data
String mqttuser =  "nodemcu1";                               // from CloudMQTT ac-
count data
String mqttpass =  "191066";                                // from CloudMQTT ac-
count data
```

```
PubSubClient client(server, mqttport, callback, s800); // Pass the server, port,
callback, and client.
unsigned long millis1=0;

// подключение библиотеки DTH
#include "DHT.h"
// константы
#define DHTPIN 22           // пин (D2) подключения контакта DATA
#define DHTTYPE DHT22      // датчик DHT 22
// создание экземпляра объекта DHT
DHT dht(DHTPIN, DHTTYPE);

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];

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
}

void setup()
{

  Serial.begin(9600);
  Serial.println("SIM800 Shield testing.");
  for (int i=0; i<10; i++){
    delay(5000);
    Serial.println("try to init sim800");

#ifdef HARDWARESERIAL
    if (s800.init( 7, 6)) break; // RX, Tx GPRS IcomsatV1.1
#else
    if (s800.init(&Serial1)) break;
#endif

  }
```

```
  initVar();
  Serial.println("try to setup sim800");
  s800.setup();
  s800.stop();
  s800.TCPstop();
  s800.getIMEI(imeicode);
  Serial.print("IMEI: ");
  Serial.println(imeicode);


  while (s800.TCPstart(GSMAPN,GSMUSER,GSMPASSWORD)) {  //receive the response 7
true 0  false.
    Serial.println("TCPstart failed");
    s800.TCPstop();
    delay(1000);
  }
  Serial.println("TCPstart started");

 while (client.connect("arduinokit", "nodemcu1", "191066")) {
    Serial.println("connect failed");
    delay(1000);
  }

  Serial.println("connected");
  client.publish("potato","hello world");// topico potato, msg helloworld.
  client.subscribe("potato"); // topic potato.
  Alarm.timerRepeat(5, pub);             // timer

}

void loop()
{

    client.loop();
    Alarm.delay(10000);

}

void initVar() {


  id    [0] = "0";
  page  [0] = "SensorsNodemcu";
  descr  [0] = "DHT22";
  widget[0] = "small-badge";
  //pin   [0] = A0;                                          // ADC
  sTopic[0] = prefix + "/" + deviceID + "/DHT22";
  badge [0] = "\"badge\":\"badge-calm\"";                  // see
http://ionicframework.com/docs/components/#colors
  style [0]   = "\"style\":\"font-size:150%;\"";


    if (inverted[0]) {
      if (defaultVal[0]>0) {
         stat[0] = setStatus(0);
      } else {
         stat[0] = setStatus(1);
      }
    } else {
       stat[0] = setStatus(defaultVal[0]);
    }
```

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

}
String setStatus ( String s ) {
  String stat = "{\"status\":\"" + s + "\"}";
  return stat;
}
String setStatus ( int s ) {
  String stat = "{\"status\":\"" + String(s) + "\"}";
  return stat;
}

float get_data_humidity() {
   //float h = dht.readHumidity();
   return 43;
}
```