О. Ю. Макаров, А. В. Турецкий, М. В. Хорошайлова

ЭЛЕКТРОНИКА И МИКРОПРОЦЕССОРНАЯ ТЕХНИКА

Практикум

Воронеж 2019

УДК 628.33

О. Ю. Макаров, А. В. Турецкий, М. В. Хорошайлова «Электроника и микропроцессорная техника» Практикум : учеб. пособие [Электронный ресурс]. – Электрон. текстовые и граф. данные (5,0 Мб) / О. Ю. Макаров, -Воронеж: ФГБОУ ВО «Воронежский государственный технический университет», 2019. – 1 электрон. опт. диск (CD-ROM) : цв. – Систем. требования : ПК 500 и выше ; 256 Мб ОЗУ ; Windows XP ; SVGA с разрешением 1024×768 ; MS Word 2007 или более поздняя версия или (Adobe Acrobat) ; CD-ROM дисковод ; мышь. – Загл. с экрана. – Диск и сопровод. материал помещены в контейнер 12×14 см.

Учебное пособие содержит сведения о большом количестве датчиков, работающих на разных физических эффектах с использованием распространенных протоколов обмена данными.

Издание предназначено для студентов, обучающихся по направлениям 11.03.03 «Конструирование и технология ЭС» и 12.03.01 «Приборостроение», может быть использовано в самостоятельной работе, при подготовке к лабораторным и практическим занятиямзанятиям, а также в курсовом и дипломном проектировании как справочное пособие по перспективной элементной базе САУ и РЭС.

Табл. 2. Ил. 43. Библиогр: 12 назв.

Рецензенты: кафедра основ радиотехники и электроники ФГОУ ВПО Воронежского института ФСИН (зав. кафедрой к. т. н., доцент Р.Н. Андреев);
д-р техн. наук, проф. В.М. Питолин

ВВЕДЕНИЕ

Микроконтроллерная система может рассматриваться как частный случай электронной системы, предназначенной для обработки входных сигналов и выдачи выходных сигналов. В качестве входных и выходных сигналов при этом могут использоваться аналоговые сигналы, одиночные цифровые сигналы, цифровые коды, последовательности цифровых кодов.

Пособие содержит 2 части, первая из которого содержит 15 лабораторных работ, а вторая 14 практических работ. Каждая работа посвящена изучению одного из узлов микроконтроллерных систем, подключаемых к платформе Arduino. Приведены теоретичесие сведения о принципах действия этих электронных модулей, применяемых программах обработки сигналов, поступающих с них, а также задания для самостоятельного выполнения.

Данное пособие содержит сведения о большом количестве датчиков, работающих на разных физических эффектах (фотоэлектрическом, пироэлектрическом, термоэлектрическом и пр.) с использованием распространенных протоколов обмена данными (SPI, IIC, UART, Bluetooth и др.). Широко также представлены устройства отображения информации: семисегментные индикаторы, LCD индикаторы, светодиодные матрицы и линейки. Подробно описаны исполнительные устройства: электродвигатели, шаговые двигатели, сервоприводы и реле.

1. ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа №1
Основы работы с платформой Arduino. Среда разработки Arduino IDE. Управление светодиодом

Цель работы: изучить аппаратную часть платформы Arduino, познакомиться со средой разработки Arduino IDE, научиться составлять, компилировать и загружать в микроконтроллер простейшие программы на Arduino.

Оборудование: персональный компьютер, среда разработки Arduino IDE, микроконтроллер Arduino UNO, макетная плата, провода, светодиоды, кнопки.

Теоретические сведения

Arduino Uno контроллер построен на ATmega328. Платформа имеет 14 цифровых вход/выходов (6 из которых могут использоваться как выходы ШИМ), 6 аналоговых входов, кварцевый генератор 16 МГц, разъем USB, силовой разъем, разъем ICSP и кнопку перезагрузки. Для работы необходимо подключить платформу к компьютеру посредством кабеля USB, либо подать питание при помощи адаптера AC/DC или батареи.

В отличие от всех предыдущих плат, использовавших FTDI USB микроконтроллер для связи по USB, новый Ардуино Uno использует микроконтроллер ATmega8U2. Технические характеристики платформы, представлены в таблице 1.1.1 [1].

Таблица 1.1.1

Технические характеристики Ардуино Uno

Микроконтроллер	ATmega328
Рабочее напряжение	5 В
Входное напряжение (рекомендуемое)	7-12 В
Входное напряжение (предельное)	6-20 В
Цифровые Входы/Выходы	14 (6 из которых могут использоваться как выходы ШИМ)
Аналоговые входы	6
Постоянный ток через вход/выход	40 мА
Постоянный ток для вывода 3.3 В	50 мА
Флеш-память	32 Кб (ATmega328) из которых 0.5 Кб используются для загрузчика
ОЗУ	2 Кб (ATmega328)
EEPROM	1 Кб (ATmega328)
Тактовая частота	16 МГц

Arduino Uno может получать питание через подключение USB или от внешнего источника питания. Источник питания выбирается автоматически.

Внешнее питание (не USB) может подаваться через преобразователь напряжения AC/DC (блок питания) или аккумуляторной батареей. Преобразователь напряжения подключается посредством разъема 2.1 мм с центральным положительным полюсом. Провода от батареи подключаются к выводам Gnd и Vin разъема питания.

Платформа может работать при внешнем питании от 6 В до 20 В. При напряжении питания ниже 7 В, вывод 5V может выдавать менее 5 В, при этом платформа может работать нестабильно. При использовании напряжения выше 12 В регулятор напряжения может перегреться и повредить плату. Рекомендуемый диапазон от 7 В до 12 В.

Выводы питания:

1) VIN. Вход используется для подачи питания от внешнего источника (в отсутствие 5 В от разъема USB или другого регулируемого источника питания). Подача напряжения питания происходит через данный вывод.

2) 5V. Регулируемый источник напряжения, используемый для питания микроконтроллера и компонентов на плате. Питание может подаваться от вывода VIN через регулятор напряжения, или от разъема USB, или другого регулируемого источника напряжения 5 В.

3) 3V3. Напряжение на выводе 3.3 В генерируемое встроенным регулятором на плате. Максимальное потребление тока 50 мА.

4) GND. Выводы заземления.

- Память. Микроконтроллер ATmega328 располагает 32 кБ флэш памяти, из которых 0.5 кБ используется для хранения загрузчика, а также 2 кБ ОЗУ (SRAM) и 1 Кб EEPROM.(которая читается и записывается с помощью библиотеки EEPROM).
- Входы и Выходы. Каждый из 14 цифровых выводов Uno может настроен как вход или выход, используя функции pinMode(), digitalWrite(), и digitalRead(). Выводы работают при напряжении 5 В. Каждый вывод имеет нагрузочный резистор (по умолчанию отключен) 20-50 кОм и может пропускать до 40 мА. Некоторые выводы имеют особые функции:
- Последовательная шина: 0 (RX) и 1 (TX). Выводы используются для получения (RX) и передачи (TX) данных TTL. Данные выводы подключены к соответствующим выводам микросхемы последовательной шины ATmega8U2 USB-to-TTL.
- Внешнее прерывание: 2 и 3. Данные выводы могут быть сконфигурированы на вызов прерывания либо на младшем значении, либо на переднем или заднем фронте, или при изменении значения. Подробная информация находится в описании функции attachInterrupt().
- ШИМ: 3, 5, 6, 9, 10, и 11. Любой из выводов обеспечивает ШИМ с разрешением 8 бит при помощи функции analogWrite().

- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Посредством данных выводов осуществляется связь SPI, для чего используется библиотека SPI.
- LED: 13. Встроенный светодиод, подключенный к цифровому выводу 13. Если значение на выводе имеет высокий потенциал, то светодиод горит.
- На платформе Uno установлены 6 аналоговых входов (обозначенных как A0 .. A5), каждый разрешением 10 бит (т.е. может принимать 1024 различных значения). Стандартно выводы имеют диапазон измерения до 5 В относительно земли, тем не менее имеется возможность изменить верхний предел посредством вывода AREF и функции analogReference(). Некоторые выводы имеют дополнительные функции:
- I2C: 4 (SDA) и 5 (SCL). Посредством выводов осуществляется связь I2C (TWI), для создания которой используется библиотека Wire.

Дополнительная пара выводов платформы:

1) AREF. Опорное напряжение для аналоговых входов. Используется с функцией analogReference().

2) Reset. Низкий уровень сигнала на выводе перезагружает микроконтроллер. Обычно применяется для подключения кнопки перезагрузки на плате расширения, закрывающей доступ к кнопке на самой плате Arduino.

3) Обратите внимание на соединение между выводами Arduino и портами ATmega328.

4) Связь. На платформе Arduino Uno установлено несколько устройств для осуществления связи с компьютером, другими устройствами Arduino или микроконтроллерами. ATmega328 поддерживают последовательный интерфейс UART TTL (5 В), осуществляемый выводами 0 (RX) и 1 (TX). Установленная на плате микросхема ATmega8U2 направляет данный интерфейс через USB, программы на стороне компьютера "общаются" с платой через виртуальный COM порт. Прошивка ATmega8U2 использует стандартные драйвера USB COM, никаких сторонних драйверов не требуется, но на Windows для подключения потребуется файл ArduinoUNO.inf. Мониторинг последовательной шины (Serial Monitor) программы Arduino позволяет посылать и получать текстовые данные при подключении к платформе. Светодиоды RX и TX на платформе будут мигать при передаче данных через микросхему FTDI или USB подключение (но не при использовании последовательной передачи через выводы 0 и 1).

Библиотекой SoftwareSerial возможно создать последовательную передачу данных через любой из цифровых выводов Uno.

ATmega328 поддерживает интерфейсы I2C (TWI) и SPI. В Arduino включена библиотека Wire для удобства использования шины I2C.

Широтно-импульсная модуляция. Широтно-Импульсная модуляция, или ШИМ, это операция получения изменяющегося аналогового значения посредством цифровых устройств. Устройства используются для получения прямоугольных импульсов - сигнала, который постоянно переключается между максимальным и минимальным значениями. Данный сигнал моделирует напряжение между максимальным значением (5 В) и минимальным (0 В), изменяя при

этом длительность времени включения 5 В относительно включения 0 В. Длительность включения максимального значения называется шириной импульса. Для получения различных аналоговых величин изменяется ширина импульса. При достаточно быстрой смене периодов включения-выключения можно подавать постоянный сигнал между 0 и 5 В на светодиод, тем самым управляя яркостью его свечения.

На графике (рис. 1.1.1) [1] зеленые линии отмечают постоянные временные периоды. Длительность периода обратно пропорциональна частоте ШИМ. Т.е. если частота ШИМ составляет 500 Гц, то зеленые линии будут отмечать интервалы длительностью в 2 миллисекунды каждый. Вызов функции analogWrite() с масштабом 0 – 255 означает, что значение analogWrite(255) будет соответствовать 100% рабочему циклу (постоянное включение 5 В), а значение analogWrite(127) – 50% рабочему циклу.

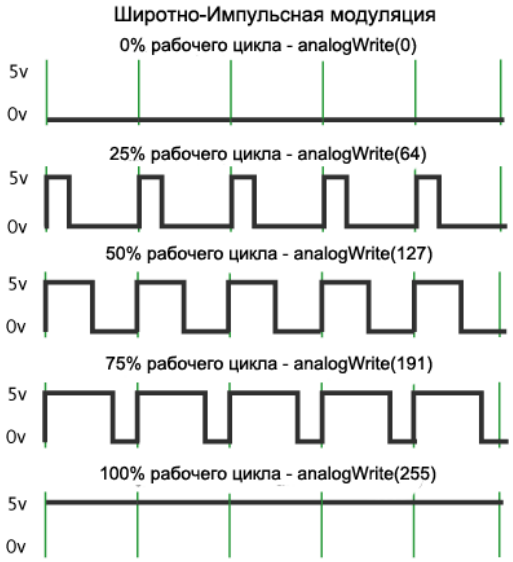


Рис. 1.1.1. Схема работы ШИМ [1]

Для примера можно взять платформу и начать трясти ее взад и вперед. Для наших глаз данное движение превращает в светящиеся линии мигание светодиода. Наращивание или уменьшение ширины импульса на светодиоде будет увеличивать или уменьшать светящиеся линии светодиода. Для написания программ и загрузки в платформу используется среда программирования Arduino IDE. Внешний вид рабочего окна программы представлен на рис. 1.1.2.

Среда разработки Arduino состоит из:

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

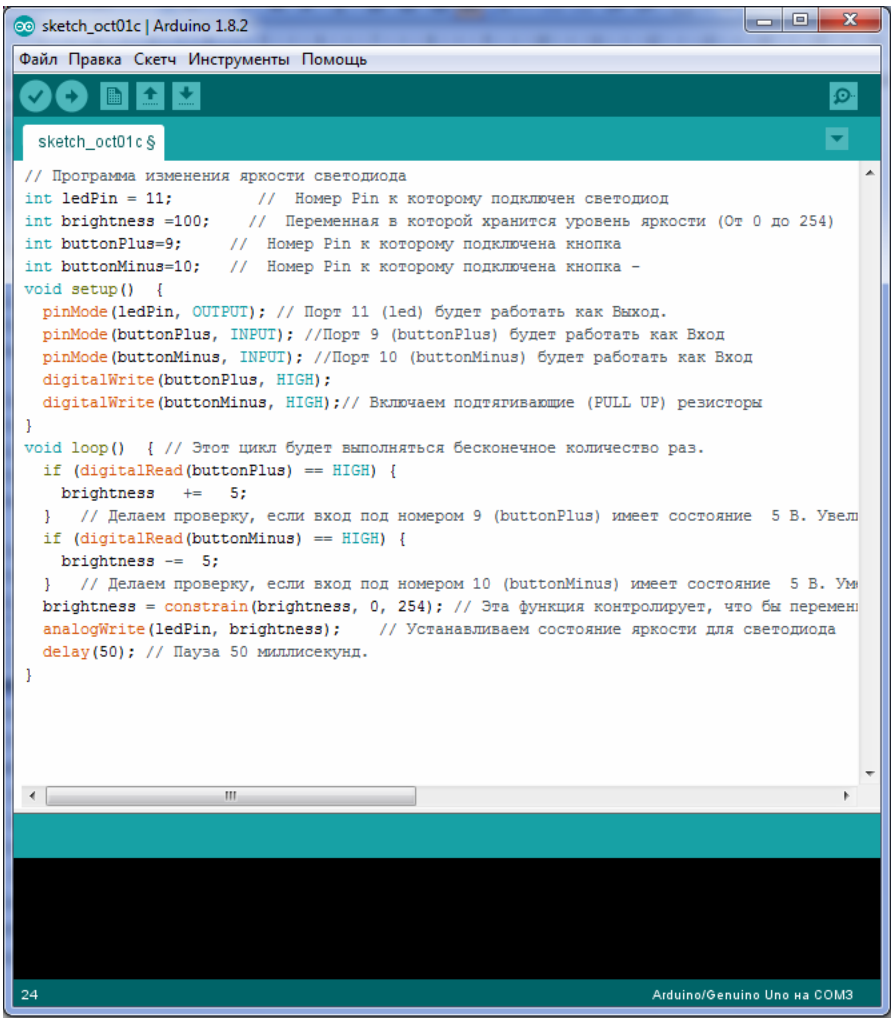


Рис. 1.1.2. Внешний вид программы Arduino IDE

Программа, написанная в среде разработки Arduino, называется скетчем. Скетч пишется в текстовом редакторе, который имеет цветовую подсветку создаваемого программного кода.

Дополнительная функциональность может быть добавлена с помощью библиотек, представляющих собой оформленный специальным образом код. В основном он находится в закрытом от разработчика доступе. Среда обычно поставляется со стандартным набором, который можно постепенно пополнять. Они находятся в подкаталоге libraries каталога Arduino.

Многие библиотеки снабжаются примерами, расположенными в папке example.Выбор библиотеки в меню приведет к добавлению в исходный код строчки:

#include <library.h>

Это директива — некая инструкция, заголовочный файл с описанием объектов, функций, и констант библиотеки. Многие функции уже разработаны для большинства типовых задач.

После подключения платформы к компьютеру необходимо настроить плату Arduino и COM порт, как показано на рис. 1.1.3.

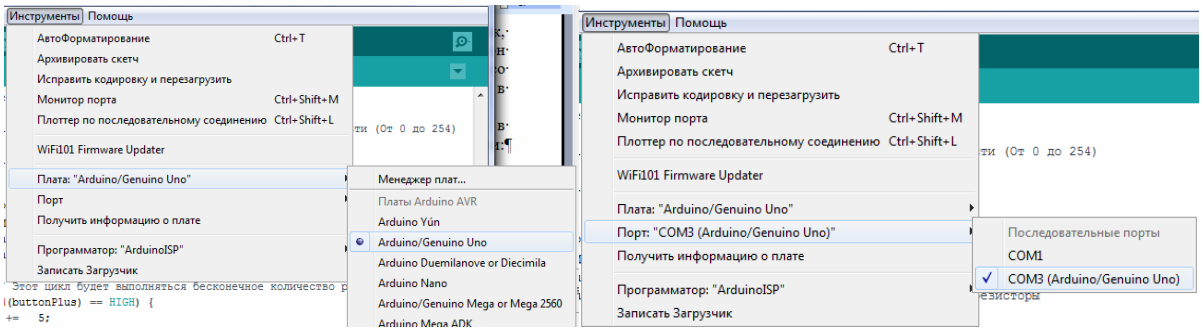


Рис. 1.1.3. Внешний вид окна программы Arduino IDE
для настройки COM порта

Для запуска монитора последовательного порта (Serial) нажимается кнопка справа вверху. Появляется окно, в верхней части которого находится строка для ввода значений с клавиатуры, а ниже данные, которые выводятся в последовательный порт (рис. 1.1.4).

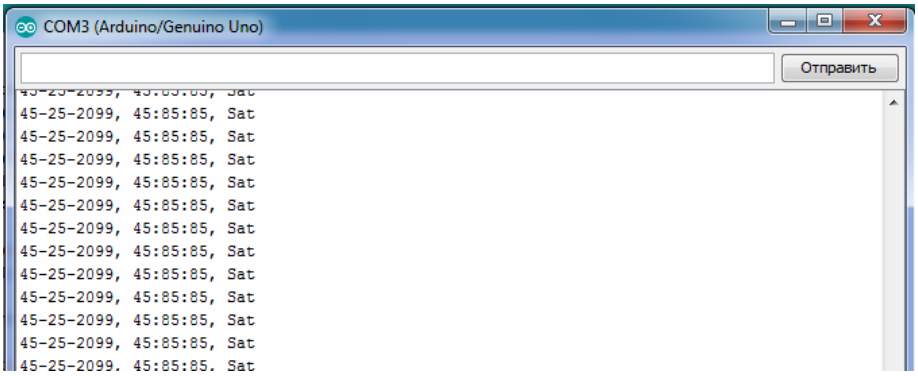


Рис. 1.1.4. Внешний вид окна монитора последовательного порта

Подключение библиотек осуществляется следующим образом:
- с помощью Library Manager;
- с помощью импорта в виде файла .zip;
- установка вручную.

Рассмотрим первый способ, с помощью Library Manager. В рабочем окне программы выбираем вкладку «Скетч». После этого нажимаем на кнопку «Подключить библиотеку». Перед нами откроется менеджер библиотек. В окне будут отображаться уже установленные файлы с подписью installed, и те, которые можно установить (рис. 1.1.5).

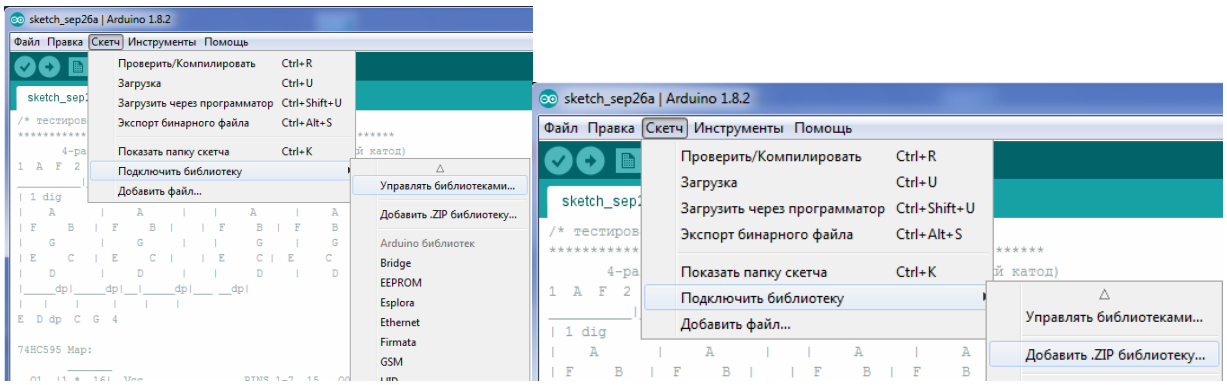


Рис. 1.1.5. Внешний вид окна программы Arduino IDE в режиме подключения библиотек

При втором способе, с помощью импорта в виде файла .zip. выбирается пункт «Добавить .ZIP библиотеку». В нём содержится заголовочный файл .h и файл кода .cpp. При установке не нужно распаковывать архив.

Третьим способом сначала надо закрыть программу Arduino IDE. Архив с библиотекой надо сначала распаковать. Затем папку с библиотекой надо перенести в каталог libraries, где установлена среда «Путь\Arduino\libraries».

Лабораторное задание

В работе необходимо подключить светодиод и две кнопки к портам. Необходимо путем нажатия кнопок добиться изменения яркости свечения светодиода. Схема имеет следующий вид (рис. 1.1.6).

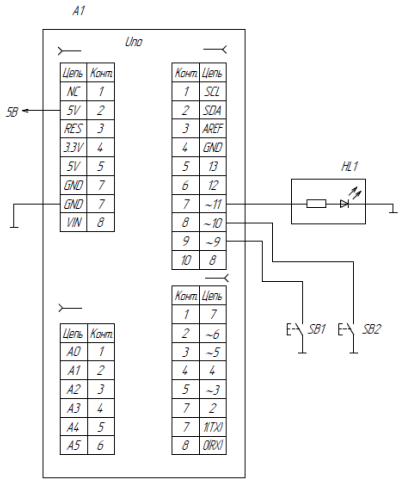


Рис. 1.1.6. Схема подключения светодиода и кнопок к платформе Arduino UNO

Программа имеет следующий вид.

```
// Программа изменения яркости светодиода
int ledPin = 11;        //  Номер Pin к которому подключен светодиод
```

```
int brightness =100;    //  Переменная в которой хранится уровень яркости (От 0 до 254)
int buttonPlus=9;     //  Номер Pin к которому подключена кнопка
int buttonMinus=10;   //  Номер Pin к которому подключена кнопка -
void setup()  {
  pinMode(ledPin, OUTPUT); // Порт 11 (led) будет работать как Выход.
  pinMode(buttonPlus, INPUT); //Порт 9 (buttonPlus) будет работать как Вход
  pinMode(buttonMinus, INPUT); //Порт 10 (buttonMinus) будет работать как Вход
  digitalWrite(buttonPlus, HIGH);
  digitalWrite(buttonMinus, HIGH);// Включаем подтягивающие (PULL UP) резисторы
}
void loop()  { // Этот цикл будет выполняться бесконечное количество раз.
  if (digitalRead(buttonPlus) == LOW) {
    brightness   +=   5;
  }    // Делаем проверку, если вход под номером 9 (buttonPlus) имеет состояние 5 В. Увеличиваем значение переменной яркости на 5 единиц.
  if (digitalRead(buttonMinus) == LOW) {
    brightness -=  5;
  }    // Делаем проверку, если вход под номером 10 (buttonMinus) имеет состояние  5 В. Уменьшаем значение переменной яркости на 5 единиц.
  brightness = constrain(brightness, 0, 254); // Эта функция контролирует, что бы переменная brightness не стала больше 254 и меньше 0, если значение вылазит за границу то функция 0 или 254
  analogWrite(ledPin, brightness);     // Устанавливаем состояние яркости для светодиода
  delay(50); // Пауза 50 миллисекунд.
}
```

Задания для самостоятельного решения

1. Написать программу, которая бы включала и выключала светодиод с определенной частотой.

2. Сделать так, чтобы в устройстве были 2 светодиода, которые переключались бы на манер железнодорожного светофора: горел бы то один то другой

3. Возьмите еще один светодиод с резистором и соберите аналогичную схему, подключив светодиод в противофазе. Надо сделать так, чтобы первый выключен, второй горит максимально ярко и до противоположного состояния

Контрольные вопросы

1. Что входит в состав программной части платформы Arduino?
2. Что входит в состав аппаратной части платформы Arduino?

3. Перечислите основные преимущества платформы Arduino.

4. Поясните работу программ, приведенных в отчете.

5. Поясните основные конструкции языка Си, приведенные в кратких теоретических сведениях.

6. Аналоговые порты ввода и вывода.

Лабораторная работа №2
Измеритель влажности и температуры. Подключение индикатора LCD 1602 к микроконтроллеру, вывод информации на индикатор.

Цель работы: познакомиться с синтаксисом языка, научиться программировать и выводить информацию на LCD индикатор с использованием микроконтроллера Arduino UNO.

Оборудование: персональный компьютер, среда разработки Arduino IDE, микроконтроллер Arduino UNO, макетная плата, провода, индикатора LCD 1602**,** датчик DHT11.

Теоретические сведения

LCD дисплей размерности 1602, на базе контроллера HD44780, являются одними из самых простых, доступных и востребованных дисплеев для разработки различных электронных устройств.

Данные дисплеи имеют два исполнения: желтая подсветка с черными буквами либо, что встречается чаще, синюю подсветку с белыми буквами.

Размерность дисплеев на контроллере HD44780 может быть различной, управляться они будут одинаково. Самые распространенные размерности 16x02 (т.е. по 16 символов в двух строках) либо 20x04. Разрешение же самих символов – 5x8 точек.

Большинство дисплеев не имеют поддержку кириллицы, имеют её лишь дисплеи с маркировкой СТК. Но данную проблему можно попытаться частично решить.

Выводы дисплея:

На дисплее имеется вилка PLS-16 для подключения. Выводы промаркированы на тыльной стороне платы.

1 (VSS) – Питание контроллера (-)

2 (VDD) – Питание контроллера (+)

3 (VO) – Вывод управления контрастом

4 (RS) – Выбор регистра

5 (R/W) – Чтение/запись (режим записи при соединении с землей)

6 (E) – Enable (строб по спаду)

7-10 (DB0-DB3) – Младшие биты 8-битного интерфейса

11-14 (DB4-DB7) – Старшие биты интерфейса

15 (A) – Анод (+) питания подсветки

16 (K) – Катод (-) питания подсветки

Внешний вид индикатора и выводов представлен на рис. 1.2.1 [2].

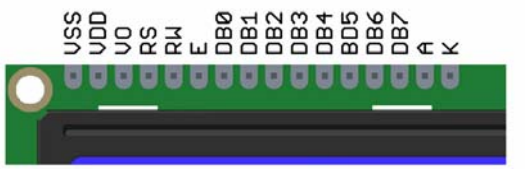


Рис. 1.2.1. Внешний вид индикатора и выводов дисплея LCD 1602 [2]

У дисплея имеется режим самотестирования, чтобы его включить необходимо подать напряжение на сам контроллер (VSS и VDD), запитать подсветку (A и K), а также настроить контрастность.

Для настройки контрастности следует использовать потенциометр на 10 кОм. На крайние выводы подается +5V и GND, центральная ножка соединяется с выводом VO

После подачи питания на схему необходимо добиться правильного контраста, если он будет настроен не верно, то на экране ничего не будет отображаться. Для настройки контрастности следует с помощью отвертки крутить вал потенциометра.

При правильной сборке схемы и правильной настройке контрастности, на экране должна заполниться прямоугольниками верхняя строка.

Вывод информации:

Для работы дисплея используется встроенная с среду Arduino IDE библиотека LiquidCrystal.h

Вот основные команды библиотеки:

begin(cols, rows,[char_size]); – Инициализация дисплея с указанием количества столбцов, строк и размера символа.

clear(); – Очистка дисплея с установкой курсора в положение 0,0 (Занимает много времени!).

home(); – Установка курсора в положение 0,0 (Занимает много времени!).

display(); – Быстрое включение дисплея (без изменения данных в ОЗУ).

noDisplay(); – Быстрое выключение дисплея (без изменения данных в ОЗУ).

blink(); – Включение мигающего курсора (с частотой около 1 Гц).

noBlink(); – Выключение мигающего курсора.

cursor(); – Включение подчеркивания курсора.

noCursor(); – Выключение подчеркивания курсора.

scrollDisplayLeft(); – Прокрутка дисплея влево. Сдвиг координат дисплея на один столбец влево (без изменения ОЗУ).

scrollDisplayRight(); – Прокрутка дисплея вправо. Сдвиг координат дисплея на один столбец вправо (без изменения ОЗУ).

leftToRight(); – Указывает в дальнейшем сдвигать положение курсора, после вывода очередного символа, на один столбец вправо.

rightToLeft(); – Указывает в дальнейшем сдвигать положение курсора, после вывода очередного символа, на один столбец влево.

noAutoscroll(); – Указывает в дальнейшем выравнивать текст по левому краю от позиции курсора (как обычно).

autoscroll(); – Указывает в дальнейшем выравнивать текст по правому краю от позиции курсора.

createChar(num,array); – Запись пользовательского символа в CGRAM дисплея под указанным номером.

setCursor(col,row); – Установка курсора в позицию указанную номером колонки и строки.

print(text); – Вывод текста, символов или цифр на экран дисплея. Синтаксис схож с одноимённой функцией класса Serial.

Сам же дисплей может работать в двух режимах :

- 8-битный режим – для этого используются и младшие и старшие биты (BB0- DB7);

- 4-битный режим – для этого используются и только младшие биты (BB4- DB7).

Использование 8-битного режима на данном дисплее не целесообразно. Для его работы требуется на 4 ноги больше, а выигрыша в скорости практически нет т.к. частота обновления данного дисплея упирается в предел < 10раз в секунду.

Для вывода текста необходимо подключить выводы RS, E, DB4, DB5, DB6, DB7 к выводам контроллера. Их можно подключать к либым пинам Arduino, главное в коде задать правильную последовательность.

Дисплей, используемый в работе, имеет модуль IIC, который позволяет его подключить, используя всего 4 провода (SCL, SDA, VCC, GND). Модуль также имеет потенциометр для настройки контрастности. Его внешний вид представлен на рис. 1.2.2 [3].

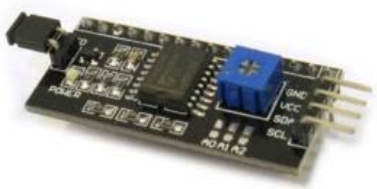


Рис. 1.2.2. Внешний вид модуля IIC [3]

Для работы с этим модулем, используется библиотека LiquidCrystal_I2C. Команды библиотеки те же, что и у предыдущей, и к ним добавлены следующие команды.

init(); – Инициализация дисплея. Должна быть первой командой библиотеки LiquidCrystal_I2C после создания объекта. На самом деле данная функция есть и в библиотеке LiquidCrystal, но в той библиотеке она вызывается автоматически (по умолчанию) при создании объекта.

backlight(); – Включение подсветки дисплея.

noBacklight(); – Выключение подсветки дисплея.

setBacklight(flag); – Управление подсветкой (true – включить / false – выключить), используется вместо функций noBacklight и backlight.

Ну и так как используется протокол IIC необходимо подключить библиотеку Wire.h.

Создание собственных символов. Представленный индикатор не имеет в памяти символов кириллицы, поэтому для вывода русских текстов необходимо использовать буквы латинского, похожих на кириллицу, а быквы и смволы, которых нет можно создать самому (всего до 8 символов). Знакоместо, в рассматриваемом дисплеи, имеет разрешение 5x8 точек. Все, к чему сводится задача создания символа, это написать битовую маску и расставить в ней единички в местах, где должны гореть точки и нолики где нет.

В ниже приведенном примере отображается смайлик, как на рис. 1.2.3 [2].

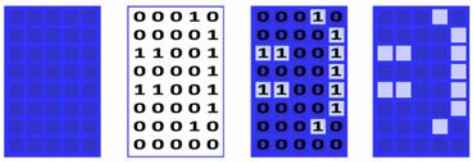


Рис. 1.2.3. Пример использования битовой маски для формирования символа [2]

На занятии используется датчик DHT11, смонтированный на плате. DHT11 — это цифровой датчик, состоящий из терморезистора и емкостного датчика влажности. Наряду с невысокой стоимостью DHT11 имеет следующие характеристики:

- питание осуществляется от 3,5-5V;
- определение температуры от 0 до 50 градусов с точностью 2 град;
- определение влажности от 20% до 95% с 5% точностью.

Внешний вид датчика DHT11 представлен на рис. 1.2.4 [4].

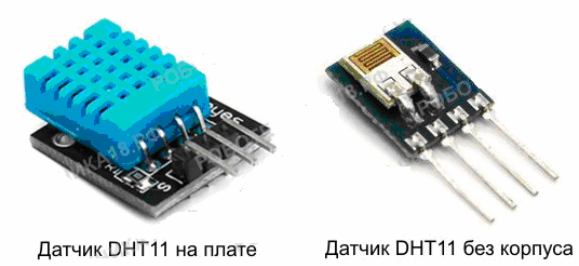


Рис. 1.2.4. Внешний вид датчика DHT11 [4]

Модуль DHT11 оборудован трехвыводной штыревой вилкой и подключается по следующей схеме:

G (-) – Подключается к выводу GND

V (+) – Подключается к выводу +5V

S – Подключается к цифровому выводу.

Для оаботы с этм датчиком надо установить библиотеку DHT11, поскольку ее нет в стандартных библиотеках программы Arduino IDE.

Лабораторное задание

Необходимо подключить LCD дисплей к платформе и определить адрес модуля IIC. Для этого в платформу записывают следующую программу. Затем в мониторе последовательного порта можно считать адрес. Если адрес не определился, то необходимо проверить правильность соединения проводов. Надо загрузить следующую программу.

```
//Программа для определения адреса на I2C
#include <Wire.h>
void setup(){
   Wire.begin();
    Serial.begin(9600);
 }
void loop(){
   byte error, address;
   int nDevices;

   Serial.println("Scanning...");
   nDevices = 0;
   for(address = 1; address < 127; address++ ){
      Wire.beginTransmission(address);
      error = Wire.endTransmission();
      if (error == 0){
         Serial.print("I2C device found at address 0x");
         if (address<16)
            Serial.print("0");
         Serial.print(address,HEX);
         Serial.println();

         nDevices++;
      }
     }
   if (nDevices == 0)
      Serial.println("No I2C devices found\n");
  else
     Serial.println("done\n");
   delay(5000);
```

```
}
```

Необходимо подключить к платформе DHT11. Вывод S надо соединить с выводом №2 на платформе. Программа имеет следующий вид.

```
//Программа для измерителя влажности и температуры
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F, 16, 2);
#include <DHT.h>
DHT dht(2, DHT11);
byte degree[8] = // кодируем символ градуса
{
B00111,
B00101,
B00111,
B00000,
B00000,
B00000,
B00000,
};
void setup()
{
lcd.init();
lcd.backlight();
lcd.createChar(1, degree); // Создаем символ под номером 1
  dht.begin(); // запускаем датчик влажности DHT11
  lcd.createChar(1, degree); // Создаем символ под номером 1
}
void loop()
{
  // считываем температуру (t) и влажность (h) каждые 250 мс
int h = dht.readHumidity();
int t = dht.readTemperature();
   lcd.setCursor(0, 0);
  lcd.print("Hum:          %");
  lcd.setCursor(11, 0);
  lcd.print(h);
  lcd.setCursor(0, 1);
  lcd.print("temp:         \1C");
  lcd.setCursor(11, 1);
  lcd.print(t);
  delay(2000);
```

}

Задания для самостоятельного решения

1. Написать программу, выводящие на экран слова «ВЛАЖНОСТЬ» и «ТЕМПЕРАТУРА».

Контрольные вопросы

1. Каким образом формируется символ в ЖК дисплее.
2. Как использовать русские символы на ЖК-дисплее?
3. Каким образом определяется адрес устройства на шине IIC?
4. Каков принцип действия датчика DHT11.
5. Назовите основные команды библиотек LiquidCrystal и LiquidCrystal_I2C?

Лабораторная работа №3
Часы реального времени (RTC). Библиотека для работы с RTC

Цель работы: познакомиться с библиотекой для работы с RTC, научиться программировать и выводить время на LCD индикатор с использованием микроконтроллера Arduino UNO, разработать будильник.

Оборудование: персональный компьютер, среда разработки Arduino IDE, микроконтроллер Arduino UNO, макетная плата, провода, модуль часов реального времени DS1302, индикатор LCD 1602, зуммер, кнопка.

Теоретические сведения

Контроллер Arduino не имеет своих собственных часов. Поэтому в случае необходимости нужно дополнять специальной микросхемой, например DS1302 (рис. 1.3.1) [5].

Рис. 1.3.1. Внешний вид модуля часов реального времени DS1302 [5]

По питанию эти платы могут использовать свой элемент питания, или питаться непосредственно с платформы Arduino.

Модуль имеет следующие выводы:

- VCC -питание (+5V)
- GND земля;

- CLK вход тактовых импульсов(к цифровому выводу Arduino);

- DAT вход данных к цифровому выводу Arduino);

- RST вход защелки (к цифровому выводу Arduino).

Для работы с модулем часов используется библиотека iarduino_RTC.h. Библиотека позволяет считывать и записывать параметры реального времени.

Описание функций библиотеки.

Подключение библиотеки: #include <iarduino_RTC.h>.

iarduino_RTC time(RTC_DS3231); // Создаём объект time, если модуль создан на базе чипа DS3231.

iarduino_RTC time(RTC_DS1307); // Создаём объект time, если модуль создан на базе чипа DS1307.

iarduino_RTC time(RTC_DS1302, RST, CLK, DAT); // Создаём объект time, если модуль создан на базе чипа DS1302.

Функция begin();

Назначение: инициализация RTC модуля.

Пример: time.**begin**(); // Инициируем работу с модулем реального времени.

Функция вызывается 1 раз в коде setup.

Проверяет регистры модуля, запускает генератор модуля и т.д.

Функция settime();

Назначение: Установка времени.

Синтаксис: settime(СЕКУНДЫ [, МИНУТЫ [, ЧАСЫ [, ДЕНЬ [, МЕСЯЦ [, ГОД [, ДЕНЬ_НЕДЕЛИ]]]]]]);

Параметры:

СЕКУНДЫ – число от 0 до 59, или -1.

МИНУТЫ – число от 0 до 59, или -1.

ЧАСЫ – число от 0 до 23, или -1.

ДЕНЬ – число от 1 до 31, или -1.

МЕСЯЦ – число от 1 до 12, или -1.

ГОД – число от 0 до 99 (без учёта века), или -1.

ДЕНЬ_НЕДЕЛИ – число от 0 (воскресение) до 6 (суббота), или -1.

Значение параметров -1 означает, что данный параметр устанавливать не нужно.

Все параметры, кроме секунд, являются необязательными.

Пример: time.settime(-1, -1, 10); // установит 10 часов, а секунды, минуты и дату, оставит без изменений.

time.settime(30, -1, -1, 2); // установит 30 секунд и 2 день месяца, остальные данные оставит без изменений.

Функция gettime();

Назначение: Чтение времени.

Синтаксис: gettime(["СТРОКА_ШАБЛОН"]);

Параметры:

СТРОКА_ШАБЛОН – задаёт шаблон для вывода строки времени, как у функции date() в PHP:

s – секунды от 00 до 59 (два знака);

i – минуты от 00 до 59 (два знака);

h – часы в 12-часовом формате от 01 до 12 (два знака);

H – часы в 24-часовом формате от 00 до 23 (два знака);

d – день месяца от 01 до 31 (два знака);

w – день недели от 0 до 6 (один знак: 0-воскресенье, 6-суббота);

D – день недели наименование от Mon до Sun (три знака: Mon Tue Wed Thu Fri Sat Sun);

m – месяц от 01 до 12 (два знака);

M – месяц наименование от Jan до Dec (три знака: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec);

Y – год от 2000 до 2099 (четыре знака);

y – год от 00 до 99 (два знака);

a – полдень am или pm (два знака, в нижнем регистре);

A – полдень AM или PM (два знака, в верхнем регистре);

Возвращаемые значения: Строка шаблона, указанные символы которой заменены на время.

Примечание:

Строка не должна превышать 50 символов.

Функцию можно вызывать без параметра, это приведёт к выводу пустой строки и обновлению переменных (см. ниже).

Пример: Serial.println(time.gettime("d-m-Y, H:i:s, D")); // выведет строку "01-10-2015, 14:00:05, Thu" и обновит значение переменных.

Serial.println(time.gettime("s")); // выведет строку "05" и обновит значение переменных.

Переменная seconds

Значение: Возвращает количество секунд, от 0 до 59;

Тип данных: uint8_t.

Переменная minutes

Значение: Возвращает количество минут, от 0 до 59;

Тип данных: uint8_t.

Переменная hours

Значение: Возвращает количество часов, от 1 до 12;

Тип данных: uint8_t.

Переменная Hours

Значение: Возвращает количество часов, от 0 до 23;

Тип данных: uint8_t.

Переменная midday

Значение: Возвращает значение 0 или 1 (0-am, 1-pm);

Тип данных: uint8_t.

Переменная day

Значение: Возвращает день месяца, от 1 до 31;
Тип данных: uint8_t.

Переменная weekday

Значение: Возвращает день недели, от 0 до 6 (0-воскресенье, 6-суббота);
Тип данных: uint8_t.

Переменная month

Значение: Возвращает текущий месяц, от 1 до 12;
Тип данных: uint8_t.

Переменная year

Значение: Возвращает текущий год, от 0 до 99;
Тип данных: uint8_t.

time.gettime(); // читаем время, обновляя значения всех переменных.
Serial.println(time.seconds); // выведет число – количество секунд
Serial.println(time.minutes); // выведет число – количество минут
Serial.println(time.hours); // выведет число – количество часов в 12 часовом формате
Serial.println(time.Hours); // выведет число – количество часов в 24 часовом формате
Serial.println(time.midday); // выведет число – 0 (до полудня), или 1 (после полудня)
Serial.println(time.day); // выведет число – текущий день месяца
Serial.println(time.weekday); // выведет число – текущий день недели: 0 (воскресение), 1 (понедельник), ... , 5 (пятница), 6 (суббота)
Serial.println(time.month); // выведет число – текущий месяц
Serial.println(time.year); // выведет число – текущий год

Дополнительные функции:

Функция blinktime();

Назначение: мигание одним из параметров времени.
Указывает функции gettime(), каждую чётную секунду, заменять указанный параметр времени на пробел(ы).
Может быть полезна, для отображения устанавливаемого параметра времени на дисплее.

Синтаксис: blinktime(ПАРАМЕТР_ВРЕМЕНИ [, ЧАСТОТА_МИГАНИЯ]);

Параметры:
ПАРАМЕТР_ВРЕМЕНИ – число от 0 до 8:
0 не мигать
1 мигают секунды
2 мигают минуты
3 мигают часы
4 мигает день
5 мигает месяц
6 мигает год

7 мигает день недели

8 мигает полдень (AM/PM)

ЧАСТОТА_МИГАНИЯ – число типа float определяющее частоту мигания в Гц (по умолчанию 1 Гц).

Возвращаемые значения: Нет.

Примечание: Значение по умолчанию 0.

Пример:

```
void setup(){
  Serial.begin( 9600 );    // Инициируем передачу данных в монитор последовательного порта.
  time.begin();            // Инициируем работу с модулем RTC
  time.blinktime( 2 );     // Заставляем функцию gettime мигать минутами с частотой по умолчанию.
//time.blinktime( 2 , 5 ); // Заставляем функцию gettime мигать минутами с частотой 5 Гц (5 раз в секунду).
}
void loop(){
  Serial.println( time.gettime("H:i:s") ); // будет выводить время, но минуты будут мигать.
}
```

Функция period();

Назначение: установка минимального периода обращения к модулю в минутах.

Означает что в указанный период времени, к модулю может быть отправлен только 1 запрос на получение времени.

Ответом на все остальные запросы будет результат последнего полученного времени + время прошедшее с этого запроса.

Функция может быть полезна, если шина передачи данных сильно нагружена другими устройствами.

Синтаксис: period (ПЕРИОД);

Параметры:

ПЕРИОД – количество минут от 0 до 255, в течении которого, к модулю может быть отправлен только 1 запрос. Если указан 0, значит каждый вызов функции gettime() генерирует запрос к модулю.

Возвращаемые значения: Нет.

Примечание: Значение по умолчанию 0.

Лабораторное задание

Необходимо к платформе подключить LCD индикатор и модуль часов реального времени (RST-6, CLK-7, DAT-8). Необходимо записать следующую программу:

```
//  УСТАНОВКА ВРЕМЕНИ МОДУЛЯ:
#include <iarduino_RTC.h> // Подключаем библиотеку.
iarduino_RTC time(RTC_DS1302,6,7,8); // Объявляем объект time для модуля на базе чипа DS1302 RST-6, CLK-7, DAT-8.
void setup() {
  Serial.begin(9600); // Инициируем передачу данных в монитор последовательного порта.
  time.begin(); // Инициируем RTC модуль.
  time.settime(0,51,21,27,10,15,2);   // Устанавливаем время: 0 сек, 51 мин, 21 час, 27, октября, 2015 года, вторник.
}
void loop(){
  if(millis()%1000==0){ // если прошла 1 секунда.
    Serial.println(time.gettime("d-m-Y, H:i:s, D"));
// выводим время в монитор, одной строкой.
    delay(2);    // приостанавливаем программу на 2 мс.
  }
}
```

Задания для самостоятельного решения

1. Написать программу, которая бы выводила данные на LCD индикатор.
2. Сделать программу, моделирующую работу будильника, который включается при совпадении времени и выключается при нажатии кнопки.

Контрольные вопросы

1. Для чего нужен модуль часов реального времени
2. Каким образом RTC подключается к платформе?
3. Назовите основные функции библиотеки для работы с RTC.
4. Каково назначение функций settime и gettime.
5. В каком формате переменная weekday представляет дни недели?

Лабораторная работа №4
Динамическая индикация. Семисегментные индикаторы. Сдвиговый регистр 74HC595

Цель работы: изучить принцип работы семисегментной матрицы и сдвигового регистра, научиться составлять, компилировать и загружать в микроконтроллер программы на Arduino.

Оборудование: персональный компьютер, среда разработки Arduino IDE, платформа Arduino UNO, макетная плата, провода, одноразрядный и 4-разрядный семисегментный индикаторы, микросхема 74HC595.

Теоретические сведения

В настоящее время для отображения информации всё чаще используются графические дисплеи, однако, семисегментные индикаторы также не утратили своего значения. Если требуется лишь отображение чисел, то они могут стать более предпочтительным вариантом, т.к. просты в управлении и могут использоваться совместно с любым микроконтроллером с достаточным количеством выводов. Жидкокристаллические семисегментные индикаторы обладают сверхнизким энергопотреблением (например, в электронных часах, вместе со схемой управления работают от одной батарейки в течении нескольких лет).

Они имеют предельно простую конструкцию, дёшевы, надёжны. Обеспечивают высокую яркость и контрастность отображаемой информации. Существует большое разнообразие индикаторов: с разным цветом свечения сегментов, разного размера, отличающиеся схемой подключения светодиодов (с общим катодом или общим анодом). При необходимости отображения нескольких разрядов можно установить несколько одноразрядных индикаторов рядом на печатной плате либо выбрать нужный вариант многоразрядного индикатора.

Свое название семисегментные индикаторы получили в связи с тем, что изображение символа формируется с помощью семи отдельно управляемых (подсвечиваемых светодиодом) элементов – сегментов. Эти элементы позволяют отобразить любую цифру 0..9, а также некоторые другие символы, например: «-«, «A», «b», «C», «d», «E», «F» и другие (рис. 1.4.1) [6]. Это даёт возможность использовать индикатор для вывода положительных и отрицательных десятичных и шестнадцатеричных чисел и даже текстовых сообщений. Обычно индикатор имеет также восьмой элемент – точку, используемую при отображении чисел с десятичной точкой. Сегменты индикатора обозначают буквами a, b, ..., g (a – верхний элемент, далее буквы присваиваются сегментам по часовой стрелке; g – центральный сегмент; dp – точка).

8 независимых элементов, каждый из которых может находиться в одном из двух состояний – горит или не горит, дают всего 2^8=256 возможных комбинаций. Или 128 комбинаций, каждая из которых может быть с горящей точкой или без неё.

Рис. 1.4.1. Схема формирования изображения

в семисегментном индикаторе [6]

Существует два варианта одноразрядных индикаторов: с общими катодами или общими анодами. Всего для подключения используется 9 выводов – общий и 8 отдельных выводов светодиодов (рис. 1.4.2) [6].

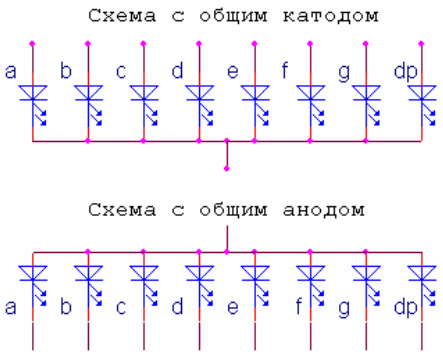


Рис. 1.4.2. Схема подключения светодиодов в индикаторах [6]

Статическая индикация. В том случае, если светодиоды в индикаторе имеют соединённые вместе аноды (схема с общим анодом), общий анод подключается к источнику напряжения $+V_{DD}$, а катоды светодиодов – сегментов подключаются к схеме управления (например, микроконтроллеру), которая отвечает за формирование изображения на индикаторе. Зажигаются сегменты низким уровнем (логический 0) на выводе схемы управления (рис. 1.4.3) [6]. По отношению к схеме управления ток светодиодов является втекающим, так что могут использоваться интегральные схемы, которые имеют выходы с открытым стоком. Изменяя величину питающего индикатор напряжения V_{DD}, можно регулировать яркость свечения.

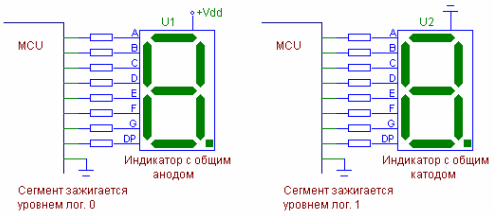


Рис. 1.4.3. Схема подключения индикаторов к платформе [6]

Если в индикаторе соединены вместе катоды (схема с общим катодом), то общий катод подключается к общему проводу схемы, а аноды светодиодов подключаются к схеме управления. В этом случае сегмент зажигается высоким уровнем на выходе схемы управления, для которой ток светодиода является вытекающим, что не позволяет использовать выходы с открытым стоком, необходим выход, выполненный по двухтактной схеме. Регулировать яркость можно, подключив общий вывод индикатора к источнику смещающего напряжения

0..V_{DD}, рассчитанного на втекающий ток, например к эмиттерному повторителю на транзисторе структуры p-n-p. Увеличивая смещение, будем уменьшать яркость свечения рис. 1.4.4 [6].

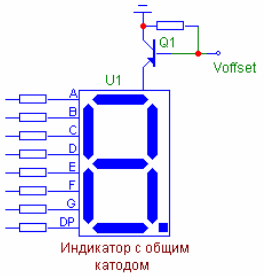


Рис. 1.4.4. Схема подключения индикаторов к платформе [6]

Для формирования изображения символа на индикаторе используют таблицу, которая ставит в соответствие коду символа набор отображаемых сегментов табл. 1.4.1. В этой таблице код символа – его порядковый номер в таблице. Набор сегментов, формирующих символ, рассматривается как двоичное число, сегменту A соответствует младший бит числа. Если бит числа равен 0, то соответствующий сегмент не зажигается при отображении символа, а если равен 1, то зажигается. В таблице также приводится запись числа, определяющего набор зажигаемых сегментов, в шестнадцатеричной форме.

Таблица 1.4.1

Таблица формирования символов в семисегментном индикаторе

№	Символ	g	f	e	d	c	b	a	HEX
0	0	0	1	1	1	1	1	1	0x3F
1	1	0	0	0	0	1	1	0	0x06
2	2	1	0	1	1	0	1	1	0x5B
3	3	1	0	0	1	1	1	1	0x4F
4	4	1	1	0	0	1	1	0	0x66
5	5	1	1	0	1	1	0	1	0x6D
6	6	1	1	1	1	1	0	1	0x7D
7	7	0	0(1)	0	0	1	1	1	0x07 (0x27)
8	8	1	1	1	1	1	1	1	0x7F
9	9	1	1	0	1	1	1	1	0x6F
10	A	1	1	1	0	1	1	1	0x77
11	b	1	1	1	1	1	0	0	0x7C
12	C	0	1	1	1	0	0	1	0x39
13	d	1	0	1	1	1	1	0	0x5E
14	E	1	1	1	1	0	0	1	0x79
15	F	1	1	1	0	0	0	1	0x71
16	<Пробел>	0	0	0	0	0	0	0	0x00
17	-	1	0	0	0	0	0	0	0x40

Для символа «7» в таблице даны два возможных варианта отображения.

Динамическая индикация. Обычно требуется отображение чисел, состоящих более чем из одного разряда. Например, для цифрового вольтметра понадобится хотя бы 4..5 разрядов, а для RLC-метра, отображающего две величины или для частотомера, требуемое количество разрядов может составить 8..10. Количество разрядов в калькуляторе может превышать 12. Проблема в том, что если каждым разрядом управлять индивидуально, то с увеличением их количества, рост числа выводов микроконтроллера и количества проводников для подключения индикатора будет просто катастрофическим. Для управления N разрядами требуется 8×N управляющих линий и 1 общий провод. В случае 5 разрядов, количество управляющих линий составит 40, а в 10-разрядном индикаторе – уже 80. А между тем свободных 80 выходов может просто не быть даже у микроконтроллера в 100-выводном корпусе (с учётом выводов питания, отладки, подключения кварцевых резонаторов и реализации важных альтернативных функций). А в 100-выводных корпусах выпускаются далеко не самые дешёвые микроконтроллеры.

К счастью, динамическая индикация позволяет решить проблему, во много раз сократив требуемое для подключения количество выводов. Идея динамической индикации состоит в том, что информация отображается не во всех разрядах индикатора сразу, а поочерёдно, в каждый момент времени только в одном разряде. В связи с тем, что зрение инерционно, необязательно чтобы все элементы изображения светились непрерывно и одновременно. Если с достаточно высокой частотой последовательно переключаться от отображения одного разряда к следующему, а когда будет достигнут последний разряд индикатора, снова переходить к отображению первого и т.д., то глазом это будет восприниматься так, как если бы каждый разряд отображал информацию статично. Этот метод похож на использовавшийся в кинескопах способ формирования изображения с помощью развёртки.

Так как требуется, чтобы в каждый момент времени работал только один разряд индикатора, то количество выводов можно существенно уменьшить: выводы одноимённых сегментов всех разрядов соединяются вместе, образуя общую шину для управления сегментами. Включение нужного разряда производится с помощью вывода общего анода (или катода, в зависимости от варианта исполнения индикатора) этого разряда.

Как правило, индикаторы, содержащие несколько разрядов, выпускаются именно в расчёте на динамическую индикацию и все необходимые соединения выполнены внутри устройства. N-разрядный индикатор в этом случае имеет 8 выводов для управления сегментами и N выводов для управления включением разрядов (общий анод или катод разряда). Всего требуется 8+N выводов, что намного лучше, чем 8×N+1 при статической индикации. На рис. 1.4.5 [6] представлена схема четырехразрядного индикатора FYQ-3641AX.

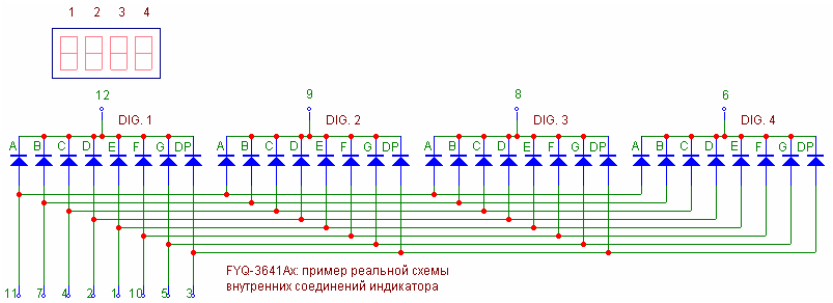


Рис. 1.4.5. Схема индикатора FYQ-3641AX [6]

В индикаторах, разряды которых выполнены по схеме с общим катодом, сегменты зажигаются высоким уровнем на выводах управления сегментами, а разряд включается низким уровнем на соответствующем выводе.

Предположим, что на изображённом выше индикаторе мы хотим вывести цифру «1» в младшем разряде (зажечь сегменты b, c). Для этого на выводе 6 управления младшим разрядом 4 устанавливаем низкий уровень; на выводах 8, 9, 12 управления остальными разрядами устанавливаем высокий уровень; на выводах 4, 7 управления сегментами c и b устанавливаем высокий уровень, а на выводах управления остальными сегментами – низкий (выводы управления сегментами подключаем через токоограничительные резисторы). В результате только светодиоды B, C разряда 4 будут смещены в прямом направлении, через них будет течь ток и они будут светиться. Сегменты B и C остальных разрядов светиться не будут, так как и на их анодах, и катодах установлен высокий уровень, т.е. напряжение смещения отсутствует. Все сегменты разряда 4, кроме b и c не будут гореть, так как на анодах и катодах соответствующих светодиодов установлен низкий уровень, т.е. напряжение смещение и ток через них отсутствуют. Все остальные светодиоды – светодиоды сегментов a, d, e, f, g, dp разрядов 1..3 не будут светиться, так как они вообще смещены в обратном направлении (на катоде установлен высокий уровень, на аноде низкий).

Это очень важная особенность схемы с динамической индикацией – в определённые моменты времени на светодиоды индикаторов неизбежно подаётся обратное напряжение. Максимально допустимое обратное напряжение для светодиодов очень невелико, типичное значение – единицы вольт. Для приводимых здесь в качестве примера индикаторов FYQ-3641Ax/Bx, в соответствии с документацией, допускается обратное напряжение 5 В. Это означает, что напряжение на выходах схемы управления этим индикатором не должно превышать 5 В. В противном случае потребуются меры по преобразованию уровней сигналов. При подключении индикатора к микроконтроллерам с напряжением питания до 5 В проблем не возникает.

В индикаторах с разрядами по схеме с общим анодом, наоборот, сегменты зажигаются низким уровнем, а разряд включается высоким уровнем на выводе.

Как уже было сказано, динамическая индикация предполагает последовательное поразрядное отображение информации с большой частотой переключения. Для этого в цикле выполняются следующие действия.

1. Гасятся все разряды индикатора – для предотвращения появления артефактов на выводимом изображении при смене состояния шины управления сегментами; если используется схема с общими катодами, для этого на общие катоды всех разрядов подаётся высокий уровень (лог. 1); в схеме с общими анодами, на аноды подаётся лог. 0.
2. На шину управления сегментами выдаются сигналы для отображения символа в очередном разряде.
3. Зажигается очередной разряд.

Либо можно погасить сегменты с помощью шины управления сегментами, переключиться на очередной разряд и выставить на шине сегментов индикатора сигналы для формирования символа. Возможны различные варианты, так что можно выбрать тот, который в данной ситуации проще реализовать.

Затем делается пауза, в течении которой происходит отображение информации на текущем разряде, после чего процесс повторяется. В результате происходит последовательное отображение от первого разряда до последнего, после чего вновь возвращаемся к первому и т.д.

Если разряды переключаются с частотой f, то время отображения одного разряда составит максимум 1/f. Максимум – потому что время горения разряда может быть и меньше периода переключения. Мы можем изменять время горения от 0 до 1/f, и тем самым регулировать яркость разряда за счёт эффекта от широтно-импульсной модуляции.

При количестве разрядов N, полное время регенерации изображения на индикаторе в целом составит $N \times 1/f$, соответственно частота регенерации $F=1/T=f/N$. Для того, чтобы не было заметно мерцания изображения, частота регенерации F должна быть не менее 50 Гц, а лучше не менее 100 Гц.

Каждый разряд горит в течении не более чем 1/N от периода регенерации. При быстром переключении разрядов глаз не будет замечать мерцания, но воспринимать он будет усреднённую яркость. Усреднённая за период регенерации, она составит 1/N от величины в случае статической индикации при тех же токах через светодиоды. Поэтому силу тока во время импульсов при динамической индикации потребуется увеличивать по сравнению с силой тока при статической. Естественно, предназначенные для динамической индикации индикаторы рассчитаны на это: они имеют достаточно большой максимально допустимый импульсный ток, в несколько раз превышающий максимальный средний ток. Сложнее обстоит дело с управляющей индикатором схемой. Не каждый микроконтроллер сможет обеспечить достаточный ток для управления сегментами, и тем более не каждый сможет непосредственно управлять включением разрядов – ток через общий вывод разряда может превышать ток сегмента в 8 раз, если горят все элементы разряда. Но это не большая проблема: подключить

индикатор к микроконтроллеру можно через микросхему-драйвер с мощными выходами или можно использовать ключи на транзисторах.

Вместо обычных повторителей или инверторов для подключения выводов управления разрядами индикатора может использоваться дешифратор n x N или демультиплексор. Помимо увеличения нагрузочной способности, это даёт возможность ещё уменьшить количество занятых управлением индикатором выводов микроконтроллера. На входы дешифратора подаётся двоичный код, и только на одном выходе, определяемом этим двоичным кодом, будет лог. 1, а на всех остальных будет лог. 0 (или, если выходы инверсные, то наоборот). Дешифратор с трёхбитовым входом имеет до 2^3=8 выходов и может использоваться до 8-разрядных индикаторов включительно, а с 4-битовым входом может переключать до 16 разрядов. Демультиплексор осуществляет коммутирование входного сигнала E на один из выходов, задаваемых адресными входами и полностью эквивалентен дешифратору при E=1, а при E=0 на всех выходах будет лог. 0 (или 1, если выходы инверсные).

В некоторых случаях, особенно если количество разрядов у индикатора небольшое, при подключении удаётся обойтись без дополнительных микросхем. Современные светодиоды имеют высокий КПД и обеспечивают достаточно высокую яркость при малых токах, в то время как при токах, близких к максимальным яркость становится чрезмерно, некомфортно высокой. Имеет смысл экспериментально подобрать режим работы индикатора.

Например, FYQ-3641A8 (четырёхразрядный, красный цвет свечения) обеспечивает хорошую видимость символов при токе в импульсе всего 2 мА на сегмент. Это позволяет использовать индикатор совместно с микроконтроллерами STM32F100xx непосредственно, без драйверов (максимальный ток вывода микроконтроллера составляет 25 мА, этого более чем достаточно даже для управления включением разрядов, так как максимальный ток общего вывода для разряда в данном случае не превышает 8×2 мА=16 мА).

Индикаторы с большим количеством разрядов (существенно большим, чем 8) используются реже, но следует иметь в виду существование второго варианта динамической индикации, который выгодно использовать для таких многоразрядных устройств. Можно осуществлять «развёртку» не по разрядам индикатора, а по сегментам. Это означает, что сначала зажигаются сегменты «A» во всех разрядах, где они должны гореть. В следующий интервал времени зажигаются сегменты «B» в нужных разрядах, и т.д. по всем 8 элементам. При таком способе соотношение между периодом отображения одного элемента и периодом полной регенерации индикатора всегда 1/8, независимо от «длины» индикатора. В этом случае ток одного разряда будет либо нулевым, либо равным току одного элемента I_0. Ток в линиях управления сегментами может достигать $N \times I_0$, где N – количество разрядов, I_0 – ток одного сегмента; максимальной величины ток достигает, когда один и тот же сегмент горит во всех разрядах.

В данной работе используется одноразрядный индикатор 5161AS, Внешний вид и схема подключения светодиодов представлена на рис. 1.4.6 [6].

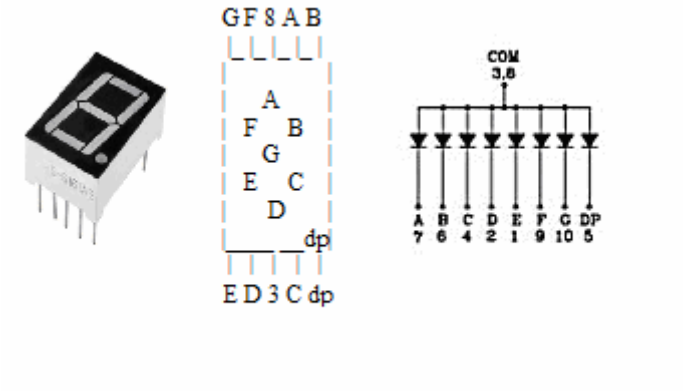


Рис. 1.4.6. Внешний вид и схема индикатора 5161AS [6]

Четырехразрядный индикатор 5461AS имеет схему соединения светододов с общим катодом (рис. 1.4.7) [6].

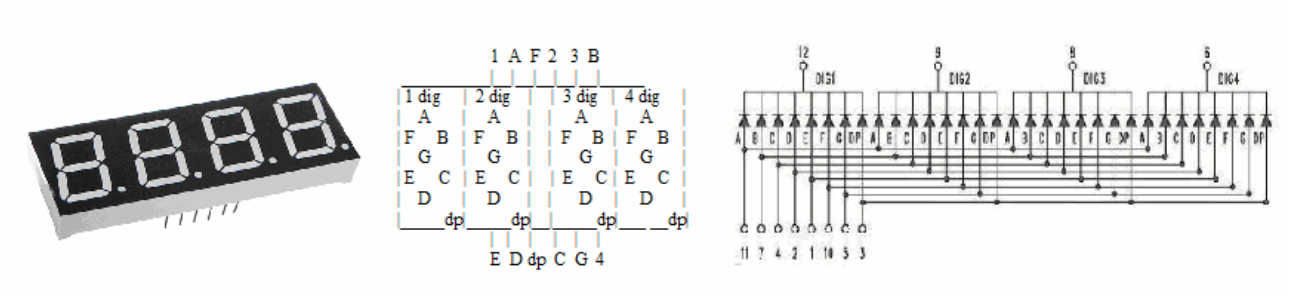


Рис. 1.4.6. Внешний вид и схема индикатора 5461AS [6]

Сдвиговый регистр 74HC595

Сдвиговый регистр – это набор последовательно соединённых триггеров (обычно их 8 штук). В отличии от стандартных регистров, сдвиговые поддерживают функцию сдвига вправо и влево. (т. е. переписывание данных с каждого предыдущего триггера на следующий по счёту) (рис. 1.4.7) [7].

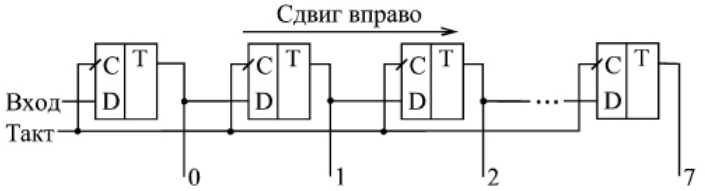


Рис. 1.4.7. Схема соединения триггеров [7]

Функционал и назначение у сдвиговых регистров довольно велик. Наверное самая популярная микросхема, представляющая собой такой регистр – это 74HC595. Внешний вид и схема выводов представлены на рис. 1.4.8 [7].

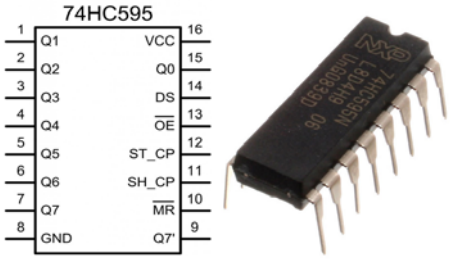


Рис. 1.4.8. Внешний вид и схема микросхемы 74HC595 [7]

Преимущества использования сдвигового регистра 74HC595:

- не требует никакой обвязки кроме конденсатора по питанию;
- работает через широкораспостраненный интерфейс SPI;
- для самого простого включения достаточно двух выходов микроконтроллера;
- возможность практически неограниченного расширения количества выходов без увеличения занятых выходов микроконтроллера;
- частота работы до 100 МГц;
- напряжение питания от 2 В до 6 В;
- выпускается как в планарных корпусах (74HC595D для производства), так и в DIP16 (74HC595N удобен для радиолюбителей и макетирования).

Для понимания работы регистра стоит взглянуть на функциональную схему (рис. 1.4.9) [7]. Она состоит из:

- 8-битного регистра сдвига;
- 8-битного регистра хранения;
- 8-битного выходного регистра.

Рассмотрим, какие выводы есть у сдвигового регистра 74hc595. Вывод подключения минуса и плюса питания объяснений не требуют:

- GND – земля;
- VCC – питание 5 вольт.

Входы 74HC595:

OE – включение высокоимпедансного (Z состояния).

Вход, переводящий выходы из высокоимпедансного состояние в рабочее состояние. При логической единице на этом входе выходы 74HC595 будут отключены от остальной части схемы. Это нужно, например, для того чтобы другая микросхема могла управлять этими сигналами.

Если нужно включить в рабочее состояние микросхеме подайте логический ноль на этот вход. А если в принципе не нужно переводить выходы в высокоимпедансное состояние – смело заземляйте этот вывод.

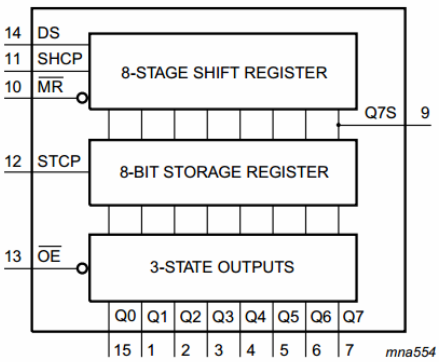


Рис. 1.4.9. Схема функциональная микросхемы 74HC595 [7]

MR – сброс регистра.

Переводить все выходы в состояние логического нуля. Чтобы сбросить регистр нужно подать логический ноль на этот вход и подать положительный импульс на вход STCP.

Подключаем этот выход через резистор к питанию микросхемы и при необходимости замыкаем на землю.

DS – вход данных.

Последовательно подаваемые сюда данные будут появляются на 8-ми выходах регистра в параллельной форме.

SHCP (CLK) – вход для тактовых импульсов.

Когда на тактовом входе SHCP появляется логическая единица, бит находящийся на входе данных DS считывается и записывается в самый младший разряд сдвигового регистра. При поступлении на тактовый вход следующего импульса высокого уровня, в сдвиговый регистр записывается следующий бит со входа данных. Тот бит который был записан ранее сдвигается на один разряд (из Q0 в Q1) , а его место занимает вновь пришедший бит. И так далее по цепочке.

STCP (SS) – вход «защёлкивающий» данные.

Что бы данные появились на выходах Q0…Q7 нужно подать логическую единицу на вход STCP. Данные поступают в параллельный регистр который сохряняет их до следующего импульса STCP.

Выходы 74HC595.

Q0…Q7 – выходы, которыми будем управлять. Они могут находится в трёх состояниях: логическая единица, логический ноль и высокоимпедансное состояние.

Q7′ – выход, предназначенный для последовательного соединения регистров.

Временная диаграмма, на которой показано движение логической единицы по всем выходам регистра показано на рис. 1.4.10 [7].

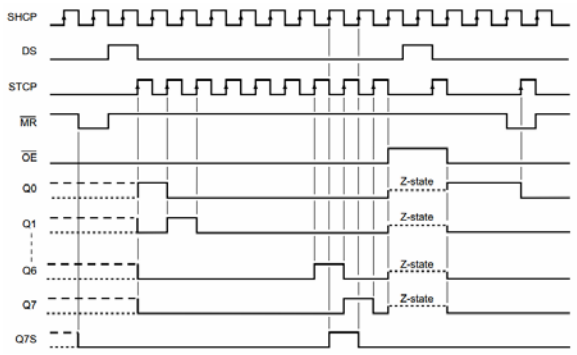


Рис. 1.4.10. Временная диаграмма работы микросхемы 74HC595 [7]

Лабораторные задания

Подключение 1 разрядного семисегментного индикатора напрямую к портам, реализация цифр и символов.

Для тестирования одноразрядного индикатора надо собрать следующую схему (рис. 1.4.11).

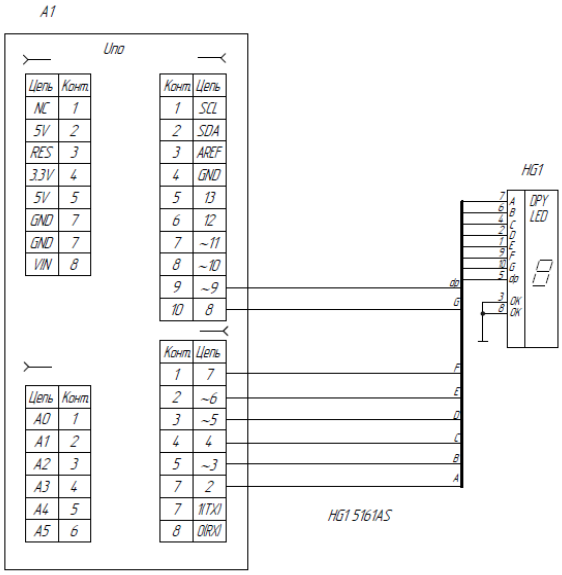


Рис. 1.4.11. Схема подключения одноразрядного индикатора

Программа имеет следующий вид:

```
/*ПРОГРАММА ТЕСТИРОВАНИЯ LED семисегментного 1 разряд
// список выводов Arduino для подключения к разрядам a-g,dp семисегментного индикатора*/
int pins[8]={2,3,4,5,6,7,8,9};// 2-A...dp-9
// значения для вывода цифр 0-9
byte numbers[31] = {
 B00111111, B00000110,    // 0 1
```

```
 B01011011, B01001111,    // 2 3
 B01100110, B01101101,    // 4 5
 B01111101, B00000111,    // 6 7
 B01111111, B01101111,    // 8 9
 B10000000, B00000000,    // горит точка, все сегменты выключены
 B01110111, B01111100,    // буквы a, b
 B00111001, B01011110,    // буквы c, d
 B01111001, B01110001,    // буквы e, f
 B00111101, B01110110,    // буквы g, h
 B00000110, B00011110,    // буквы i, j
 B00111000, B01010100,    // буквы l, n
 B01011100, B01110011,    // буквы o, p
 B01010000, B01101101,    // буквы r, s
 B01111000, B00011100,    // буквы t, u
 B01101110};              // буква y
// переменная для хранения значения текущей цифры
int number=0;
void setup(){
// Сконфигурировать контакты как выходы
for(int i=0;i<=7;i++)
pinMode(pins[i],OUTPUT);}
void loop(){
showNumber(number);
delay(500);
number=(number+1)%31;}
// функция вывода цифры на семисегментный индикатор
void showNumber(int num){
for(int i=0;i<=7;i++){
if(bitRead(numbers[num],i)==HIGH) // зажечь сегмент
digitalWrite(pins[i],HIGH);
else // потушить сегмент
digitalWrite(pins[i],LOW);
}
}
```

Она поочередно выводит цифры и некоторые буквы.

Для тестирования 4 разрядного индикатора необходимо собрать следующую схему (рис. 1.4.12).

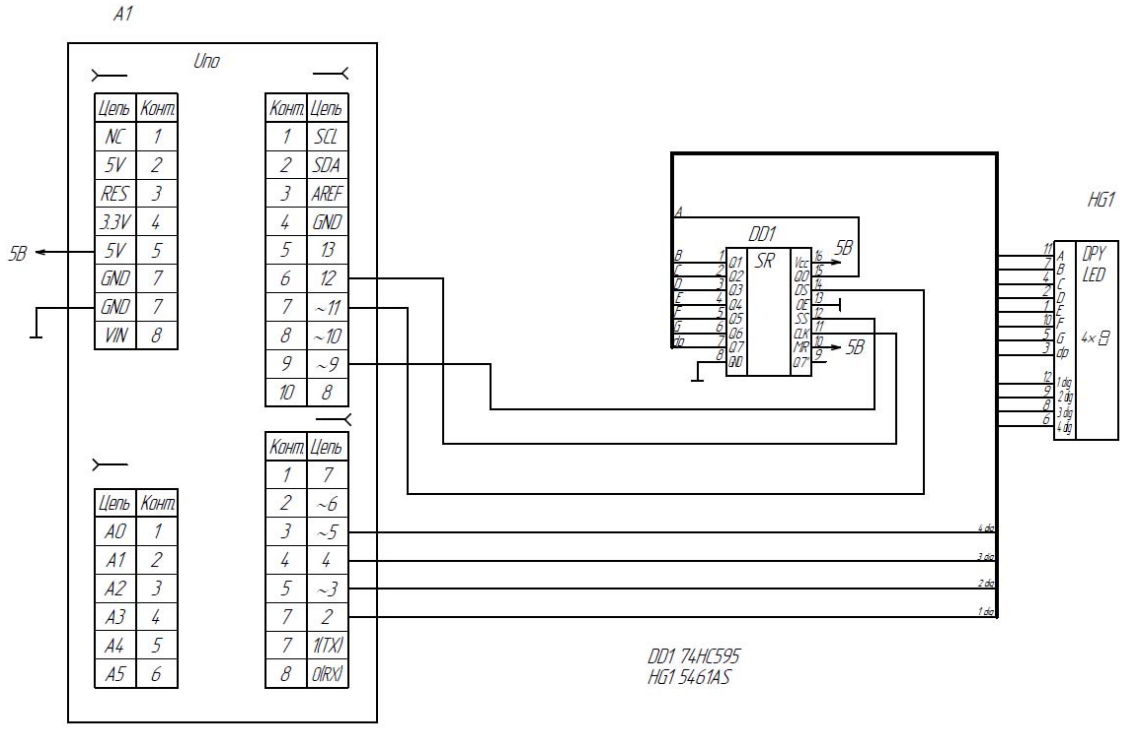


Рис. 1.4.11. Схема подключения четырехразрядного индикатора

Программа имеет следующий вид:

```
/* тестирование LED индикатора
74HC595 Map:

  Q1  |1 *  16|  Vcc      PINS 1-7, 15   Q0 - Q7   Output Pins
  Q2  |2    15|  Q0       PIN 8      GND       Ground, Vss
  Q3  |3    14|  DS       PIN 9      Q7"       Serial Out
  Q4  |4    13|  OE       PIN 10     MR        Master Reclear, active hi
  Q5  |5    12|  ST_CP(SS)  PIN 11     SH_CP   clock pin
  Q6  |6    11|  SH_CP (CLK) PIN 12   ST_CP  Storage reg. clock
  Q7  |7    10|  MR           pin(latch pin)
GND |8___9|  Q7"     PIN 13     OE        Output enable, active low
                     PIN 14     DS        Serial data input
                     PIN 16     Vcc       Positive supply voltage

  LED B -|1 *  16|-5V
  LED C -|2    15|-LED A
  LED D -|3    14|-PIN 11
  LED E -|4    13|-GND
  LED F -|5    12|-PIN 9
  LED G -|6    11|-PIN 12 ; 1uF TO GND
  LED dp-|7    10|-5V
    GND-|8____9|-NILL
```

```
    */
int latchPin = 9;//Пин «защелки» регистра подключен к ST_CP (SS) входу
int clockPin = 12;    //Пин подключен к SH_CP (CLK) входу 74HC595
int dataPin = 11;     //Пин подключен к DS входу 74HC595
int TimeLight = 500;  //время для разогрева сегментов
byte SegDisplay; // переменная для вывода символов на индикаторе
int RazrDisplay; // переменная для включения разрядов
// Настройка комбинации для отображения каждого номера на дисплее.
byte g_digits[12]={
 B00111111, B00000110,    // 0 1
 B01011011, B01001111,    // 2 3
 B01100110, B01101101,    // 4 5
 B01111101, B00000111,    // 6 7
 B01111111, B01101111,    // 8 9
 B00000000, B10000000};   // все сегменты выключены, горит точка
int g_registerArray[4]={2,3,4,5}; //массив цифр, указывающий пины разрядов цифр (катоды LED)
void setup() {
  // обозначаем все пины защелок как выходы
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  // настраиваем пины пины разрядов цифр (катоды LED) как выходы и гасим все разряды
 for(int i=0;i<4;i++){
 pinMode (g_registerArray[i],OUTPUT);
 digitalWrite(g_registerArray[i], 1);
  }
}
void loop() {
 // вложенный цикл параметр k- разряд цифры, i- формируемая цифра
for (int k=0; k<=3; k++){
for (int i=0; i<=11; i++){
SegDisplay=g_digits[i]; // получаем цифру и выводим символ, из массива цифр, соответствующий этой цифре.
RazrDisplay=g_registerArray[k];  // получаем цифру и выводим номер регистра, из массива разрядов цифр.
   digitalWrite(latchPin, LOW);  // устанавливаем синхронизацию «защелки» на LOW
     shiftOut(dataPin, clockPin, MSBFIRST, SegDisplay); // Записываем информацию для регистра (формируем цифру)
     digitalWrite(latchPin, HIGH);  // «защелкиваем» регистр, тем самым устанавливая значения на выходах
```

```
    digitalWrite (RazrDisplay,0); // включаем цифру в заданном разряде
   delay(TimeLight); // пауза, чтобы сегменты «разгорелись»
  digitalWrite (RazrDisplay,1); // гасим цифру
}
}
}
```

Задания для самостоятельного решения

1 уровень. Необходимо выводить на все разряды информацию одновременно.

2 уровень. Необходимо на индикатор вывести значения времени, получаемое от модуля часов реального времени.

Контрольные вопросы

1. Как подключается модуль семисегментного индикатора?
2. Каково назначение динамической индикации?
3. Зачем нужен токоограничивающий резистор?
4. Какое максимальное число можно вывести на четырехразрядном семисегментном индикаторе?
5. Что за порядок сдвига MSBFIRST и LSBFIRST?
6. Что такое сдвиговый регистр, для чего он используется?
7. Для чего служит функция shiftOut?
8. Какой алгоритм работы с микросхемой 74HC595?

Лабораторная работа №5
Динамическая индикация. Светодиодные матрицы. SPI интерфейс

Цель работы: изучить библиотеку SPI, подключить светодиодную матрицу через 2 два сдвиговых регистра 74HC595.

Оборудование: персональный компьютер, среда разработки Arduino IDE, платформа Arduino UNO, макетная плата, провода, светодиодная матрица 8×8, микросхема 74HC595 – 2шт.

Теоретические сведения

Как известно, сегментные индикаторы, будь то шкалы или цифры, состоят из отдельных светодиодов, соединенных вместе. Например, у группы светодиодов могут быть соединены все катоды. Такой индикатор имеет название «с общим катодом», в противном случае — «с общим анодом».

Существуют также конструкции в виде матрицы светодиодов либо одного цвета (красного, зеленого и пр.) либо трехцветные.

Такой индикатор называется матричным. Разрешение матричного индикатора — это количество точек по горизонтали и вертикали. Например, самые распространенные индикаторы имеют разрешение 8×8 точек (рис. 1.5.1) [8].

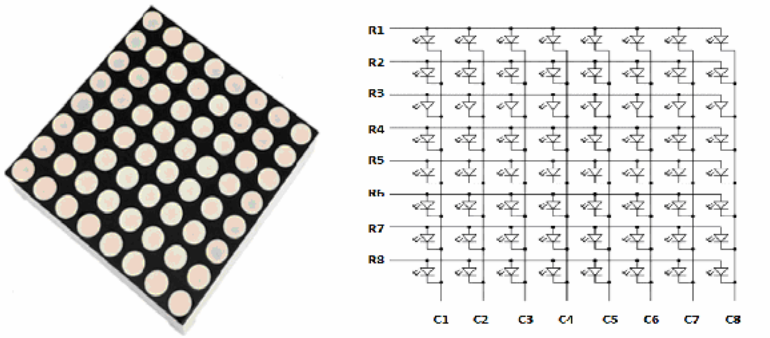


Рис. 1.5.1. Внешний вид и схема светодиодной матрицы [8]

Если требуется светодиодная матрица с большим разрешением, то её составляют из нескольких 8×8 индикаторов. Рассмотрим способ соединения светодиодов внутри матрицы.

Конечно, можно было бы как и в случае семисегментного индикатора соединить все светодиоды общим катодом или анодом. В этом случае нам бы потребовалось либо 64 вывода контроллера, либо 8 сдвиговых регистров. Оба варианта весьма расточительны.

Более правильный вариант — объединить светодиоды в группы по 8 штук с общим катодом. Пусть это будут столбцы матрицы. Затем, параллельные светодиоды в этих столбцах объединить снова в группы по 8 штук уже с общим анодом. Получится, как показано на рис. 1.5.1 [8]:

Предположим, стоит задача зажечь светодиод R6C3. Для этого нам потребуется подать высокий уровень сигнала на вывод R6, а вывод C3 соединить с землей. Получится, как показано на рис. 1.5.2 [8].

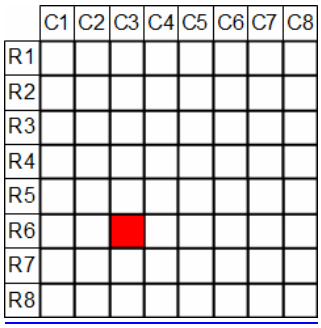


Рис. 1.5.2. Схема включения точки R6C3 [8]

Не выключая эту точку, попробуем зажечь другую — R3C7. Положительный контакт питания соединим с R3 и землю с C7. Но в таком случае стро-

ки R6 и R3 будут пересекаться с колонками C3 и C7 не в двух, а в четырех местах. Следовательно, и зажжется не две, а четыре точки (рис. 1.5.3) [8].

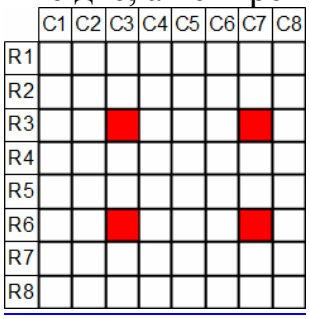


Рис. 1.5.3. Схема включения точки R6C3 и R3C7 [8]

Очевидно, что помочь сможет всё та же динамическая индикация. Если мы будем включать точки R6C3 и R3C7 по очереди очень быстро, то сможем использовать персистентность зрения – способность интерпретировать быстро сменяющиеся изображения как одно целое.

В данной работе используется светодиодная матрица 1588AS – это матрица имеет размерность 8×8 светодиодов. Цвет свечения – красный. Светодиоды в строках подключены по схеме с общим катодом. Габаритные размеры матрицы и схема представлены на рис. 1.5.4 [8].

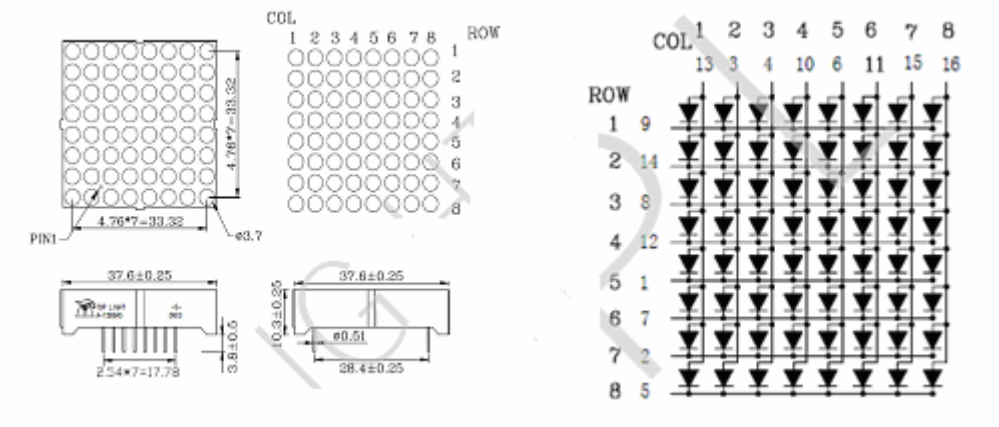


Рис. 1.5.4. Габаритные размеры и схема матрицы 1588AS [8]

Снизу матрицы, находится два ряда выводов, со стандартным шагом 2.54, что позволяет удобно монтировать сетодиодную матрицу в две беспаечные макетные платы. В работе представлен модуль, содержащий токоограничительные резисторы и позволяющий монтировать матрицу в одну макетную плату.

Осчёт выводов ведётся от угла, на котором сходятся стороны матрицы с выступами. Принципиальная схема матрицы представлена на рис. 1.5.4 [8].

Интерфейс SPI. Библиотека SPI.

SPI (Serial Peripheral Interface), или последовательный периферийный интерфейс, был разработан компанией Motorola для организации быстрого и про-

стого в реализации обмена данными между компонентами системы – микроконтроллерами и периферийными устройствами. На шине может быть одно ведущее устройство (master) и несколько ведомых (slave).

Интерфейс использует 4 линии для обмена данными:

SCLK – Serial Clock: тактовый сигнал (от ведущего) Другие обозначения: SCK, CLK Arduino: пин 13.

MOSI – Master Output, Slave Input: данные от ведущего к ведомому Другие обозначения: SDI, DI, SI Arduino: пин 11.

MISO – Master Input, Slave Output: данные от ведомого к ведущему Другие обозначения: SDO, DO, SO Arduino: пин 12.

SS – Slave Select: выбор ведомого; устанавливается ведущим Другие обозначения: nCS, CS, CSB, CSN, nSS, STE Arduino: по умолчанию пин 10.

Линия SS обычно для каждого ведомого своя, но некоторых ведомых возможно подключить к одной SS – такой способ используется для каскадного подключения устройств.

Стандартный алгоритм работы SPI таков:

Ведущий устанавливает низкий уровень на той линии SS, к которой подключен нужный ведомый.

Ведущий задаёт такт, предавая сигнал типа «меандр» на SCLK, и одновременно с каждым фронтом импульса SCLK выставляет нужный уровень на MOSI, передавая ведомому по биту за такт.

Ведомый на каждый фронт сигнала SCLK выставляет нужный уровень на MISO, передавая ведущему по биту за такт.

Для завершения передачи ведущий устанавливает высокий уровень на SS.

SPI является полнодуплексной шиной – данные передаются одновременно в обе стороны. Типичная скорость работы шины лежит в пределах 1-50 МГц. Благодаря исключительной простоте алгоритма передачи SPI получил широчайшее распространение в самых различных электронных устройствах – например, в датчиках, чипах памяти, радиомодулях, и т.д.

Вообще, у SPI есть четыре режима передачи, которые основаны на комбинации «полярности» тактового сигнала (clock polarity, CPOL) и фазы синхронизации (clock phase, CPHA). Проще говоря, CPOL – это уровень на тактовой линии до начала и после окончания передачи: низкий (0) или высокий (1). А фаза определяет, на фронте или спаде тактового сигнала передавать биты:

Режим 0: CPOL=0, CPHA=0 Чтение бита происходит на фронте тактового сигнала (переход $0 \rightarrow 1$), а запись – на спаде ($1 \rightarrow 0$).

Режим 1: CPOL=0, CPHA=1 Чтение – на спаде, запись – на фронте.

Режим 2: CPOL=1, CPHA=0 Чтение – на спаде, запись – на фронте.

Режим 3: CPOL=1, CPHA=1Чтение – на фронте, запись – на спаде.

Временные диаграммы этих четырех режимов работы представлены на рис. 1.5.5 [9].

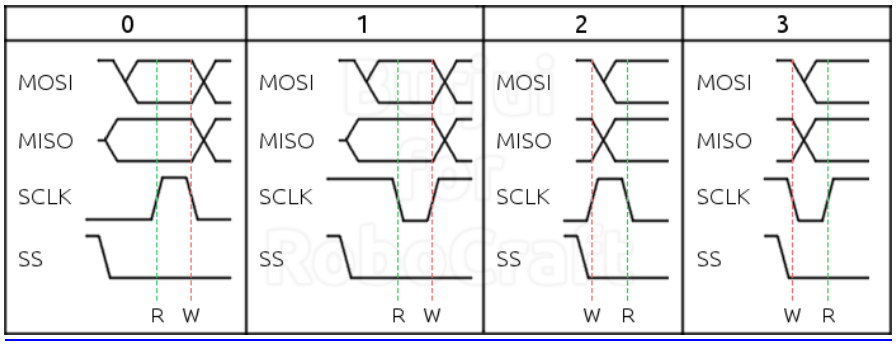


Рис. 1.5.5. Временные диаграммы режимов интерфейса SPI [9]

Данные по SPI можно передавать либо старшим битом вперёд (по умолчанию для Arduino), либо младшим. Обычно используется первый вариант, но перед началом работы с устройством следует уточнять этот момент в документации.

Кратко о библиотеке SPI. Эта библиотека использует аппаратные возможности AVR для работы по SPI на Arduino, причём только в режиме ведущего (SPI master). Функций в ней совсем немного:

1) ***begin() и end().*** Инициализация и завершение работы с SPI. При инициализации линии SCLK (13), MOSI (11) и SS (10) настраиваются на вывод, выставляя на SCK и MOSI низкий, а на SS – высокий уровень. Вызов end() линии не трогает, оставляя в том же состоянии, что и до вызова – просто выключает блок SPI микроконтроллера.

2) ***setBitOrder(order).*** Устанавливает порядок посылки битов данных (order): MSBFIRST – первым идёт старший бит посылки (по умолчанию) LSBFIRST – первым идёт младший бит

3) ***setClockDivider(divider)*** Устанавливает делитель тактов для SPI относительно основной частоты. Доступны делители 2, 4, 8, 16, 32, 64 и 128. Соответствующие константы имеют имена вида SPI_CLOCK_DIVn, где n – делитель, например, SPI_CLOCK_DIV32. По умолчанию делитель равен 4 – при обычной тактовой частоте МК на Arduino в 16 МГц SPI будет работать на частоте 4 МГц.

На заметку: если устройство поддерживает частоту, скажем, 1.25 МГц, то нужно выставить делитель, соответствующий этой или меньшей частоте – 16, например.

4) ***setDataMode(mode)*** Задаёт режим работы SPI, используя константы SPI_MODE0 (по-умолчанию), SPI_MODE1, SPI_MODE2 и SPI_MODE3. Это те самые режимы с параметрами CPOL и CPHA.

5) ***transfer(value)*** Осуществляет двустороннюю передачу: передаёт байт value и возвращает байт, принятый от ведомого.

Кроме того, доступны функции ***shiftIn(miso_pin, sclk_pin, bit_order)*** *и* ***shiftOut(mosi_pin, sclk_pin, order, value)****,* они предоставляют программную полудуплексную передачу данных по SPI – этакие половинки метода transfer(): shiftIn() только принимает, а shiftOut()только передаёт данные. Как видно по их

аргументам, они позволяют использовать любые цифровые пины Arduino в качестве линий SPI, но вы сами должны настроить их как входы/выходы, функции shiftIn() и shiftOut() этого не делают.

Лабораторные задания

Подключение светодиодной матрицы через два сдвиговых регистра 74HC595 с использованием SPI интерфейса, реализация вывода символов.

Схема включения следующая (рис. 1.5.6).

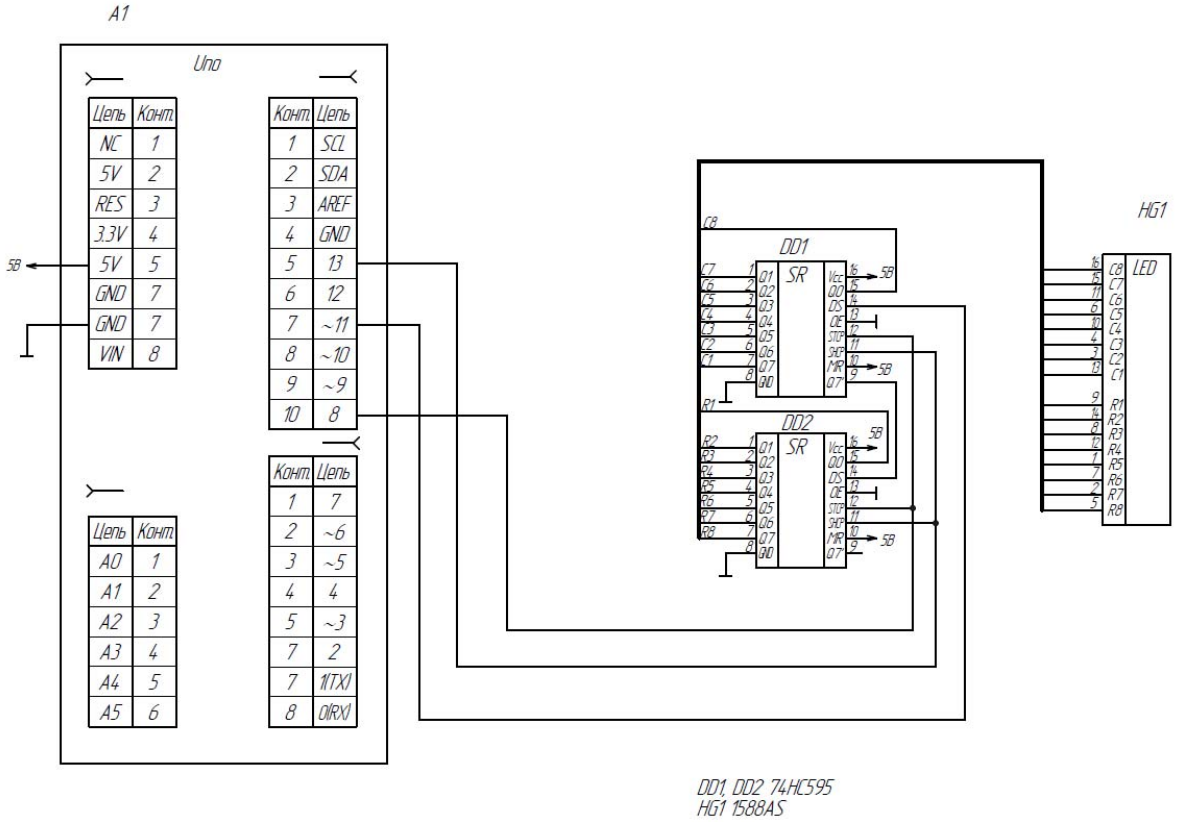


Рис. 1.5.6. Схема подключениясветодиодной матрицы

```
// Программа вывода изображения на светодиодную матрицу
// подключение библиотеки SPI
#include <SPI.h>
int ss_pin=8; // пин SS
int pos=0; //
int offfigure=0; // текущая фигура для отображения
unsigned long millis1=0;
// массив с данными фигур для отображения
byte figure[3][8]={
 {B00111100,
 B01000010,
  B10100101,
```

```
  B10000001,
  B10000001,
  B10100101,
  B01011010,
  B00111100},
 {B00111100,
  B01000010,
  B10100101,
  B10000001,
  B10000001,
  B10011001,
  B01000010,
  B00111100},
  {B00111100,
  B01000010,
  B10100101,
  B10000001,
  B10011001,
  B10100101,
  B01000010,
  B00111100}};
  void setup()
{
SPI.begin();
// Сконфигурировать контакт SS как выход
pinMode(ss_pin, OUTPUT);
}
void loop()
{
digitalWrite(ss_pin, LOW); // Защелку вниз
//передаем строки
SPI.transfer((B00000001<<pos)^B11111111);
// передаем столбцы
SPI.transfer(figure[offfigure][pos]);
digitalWrite(ss_pin,HIGH); // защелку вверх вывести данные на выводы 74HC595
delay(1);
pos=(pos+1)%8;
if(millis()-millis1>1000) // через 1 секунду – новая фигура
{
offfigure=(offfigure+1)%3;
millis1=millis();
}
}
```

Задания для самостоятельного решения

1. Написать программу, реализующую другое изображение.
2. Модифицировать предыдущую программу и сделать так, чтобы фигура менялась при нажатии кнопки.

Контрольные вопросы

1. Каким образом формируется изображение на светодиодной матрице?
2. Каковы особенности применения каскадного соединения сдвиговых регистров?
3. Какие линии обмена данными использует интерфейс?
4. Почему интерфейс SPI называется полнодуплексный?
5. Каким образом в программе реализовано сканирование строк матрицы?

Лабораторная работа №6
Динамический опрос клавиатуры. Входной сдвиговый регистр 74HC165

Цель работы: изучить принцип работы матричной клавиатуры, подключить клавиатуру через выходной 74HC595 и входной 74HC165 сдвиговые регистры

Оборудование: персональный компьютер, среда разработки Arduino IDE, платформа Arduino UNO, макетная плата, провода, матричная клавиатура, модуль с подтягивающими резисторами, микросхема 74HC595 – 1шт, микросхема 74HC165 – 1шт.

Теоретические сведения

При использовании большого числа кнопок подключение их к портам напрямую нецелесообразно. Чаще всего это относится к моделям с небольшим количеством выводов. Для этого обычно используется матричная клавиатура. Такая система работает в компьютерных клавиатурах, калькуляторах, телефонах и других устройств, в которых используется большое количество кнопок.

Для Arduino чаще всего используются клавиатуры, представленные на рис. 1.6.1 [10].

Самыми распространенными являются 16 кнопочные клавиатуры 4x4. Схема подключения кнопок в матричной клавиатуре представлена на рис. 1.6.2 [10]. Принцип их работы достаточно прост, Arduino поочередно подает логический ноль на каждый из 4 столбцов (на остальные столбцы подается единица), и в этот момент отслеживается появление этого нуля на строках.

Рис. 1.6.1. Внешний вид матричных клавиатур [10]

Если кнопка оказалась нажатой, то ноль появится, если ненажатой, то не появится. По номеру столбца и строки вычисляется нажатая кнопка. Этот способ называется динамическим опросом, он аналогичен динамической индикации, за исключением того, что здесь надо выставлять числа и опрашивать кнопки.

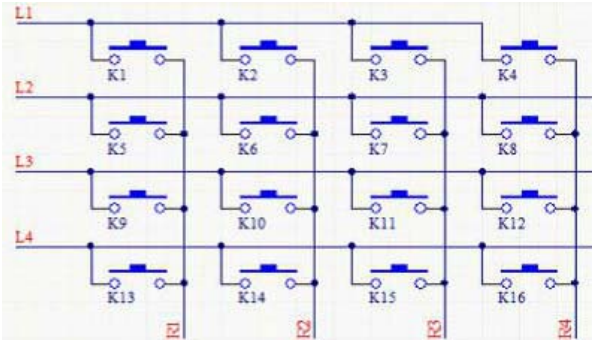


Рис. 1.6.2. Схема подключения кнопок в матричной клавиатуре [10]

Для подачи сигналов логического «0» на столбцы можно использовать сдвиговый регистр 74HC595. А вот для опроса понадобится входной сдвиговый регистр 74HC165.

Микросхема 74HC165 позволяет увеличить количество цифровых входов микроконтроллера, иногда их не хватает для решения каких то задач.

Микросхема 74HC165 – сдвиговый регистр, преобразующий параллельный входной сигнал в последовательный выходной. Из трёх выводов микроконтроллера (в том числе ардуино) можно получить 8 цифровых входов. Из регистров 74HC165 можно делать каскады, подключая один за другим, и таким образом из всё тех же 3 входящих линий получать 16, 24, 32 и т.д. цифровых входов.

Данная микросхема выпускается в корпусах SOIC, SSOP, PDIP, SO, CDIP, CFP, TSSOP. Схема расположения выводов представлена на рис. 1.6.3 [11].

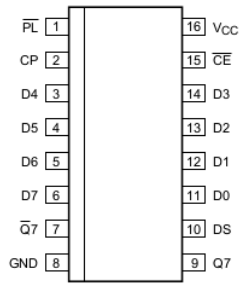


Рис. 1.6.3. Схема расположения выводов 74HC165 [11]

Назначение выводов следующее:

D0-D7– входы, состояние которых считывается в регистр;

Q7 – последовательный вывод;

Q7 – инверсный вывод, на нём идут биты с Q7, но инвертированные;

DS – последовательный ввод; к нему можно подсоединить вывод Q7 второго регистра, получив каскадное подключение;

Vcc – питание;

GND – земля;

PL (SS) – защёлка;

CP (CLK) – тактовый вход;

CE – когда на нём 1 – тактирование выключено.

Структура микросхемы 74HC165 представлена на рис. 1.6.4 [11].

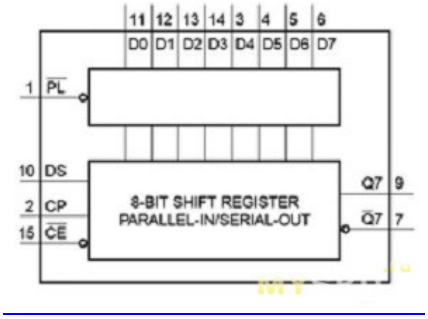


Рис. 1.6.4. Структура микросхемы 74HC165 [11]

74HC165 и 74HC595 при использовании вместе могут использовать общий вывод синхронизации. В итоге, при подключении к микроконтроллеру в сумме используют 5 выводов.

Типовое использование данной микросхемы с Arduino требует SPI и выглядит как на рис. 1.6.5 [12].

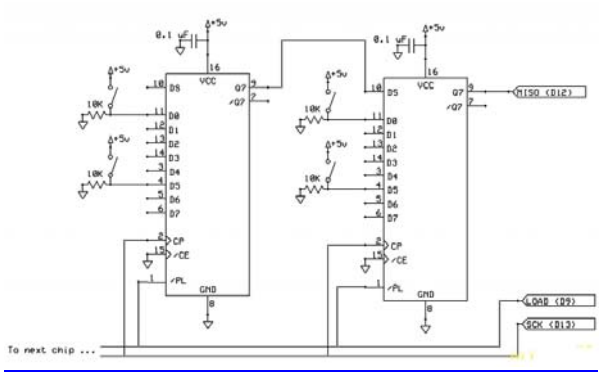


Рис. 1.6.5. Типовая **с**хема подключения микросхемы 74HC165 [12]

На этом рисунке видно, что входы регистра надо подтянуть к питанию через резисторы. Рассмотри подробно этот вопрос.

В цифровой схемотехнике есть не только понятие «подтягивающий резистор», но и «заземляющий». В иностранной литературе они называюися соответственно pull-up и pull-down резисторами. Это просто резисторы, которые подключаются между входом/выходом цифровой микросхемы и питанием/землей (рис. 1.6.6).

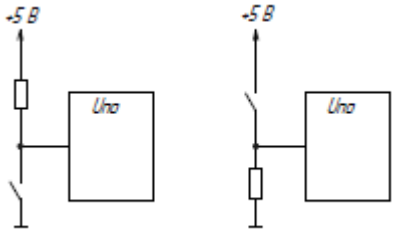


Рис. 1.6.6. Схема подключения «подтягивающих» резисторов

Каково назначение подтягивающих резисторов? Случается так, что не все вывод микросхемы используются в схеме. Иногда они либо вовсе не задействованы, либо включаются в работы только в опредленных ситуациях. Выводы цифровых микросхем обладают достаточно большим сопротивлением и если свободные выводы никуда не подключить, то от посторонних электромагнитных полей на них могут образоваться достаточно большие потенциалы, которые будут восприниматься микросхемой как полезный сигнал, от чего произойдёт её ложное срабатывание. По этой причине инженеры придумали фиксировать потенциалы таких выводов с помощью хитрого, но простого трюка.

Фиксация производится с помощью обычного резистора, включенного между выводом (будь то вход или выход) микросхемы и питанием/землёй. Такой резистор как-бы «подтягивает» потенциал вывода до потенциала питания или земли. При этом, помимо фиксации потенциала, сохраняется возможность использовать вывод по назначению.

Подтягивающий резистор выбирается таким, чтобы ток через него был очень маленьким. Такой ток называется слабым сигналом. И если вывод будет «подтянут» с помощью подтягивающего резистора к питанию/земле, а затем на него подать полезный сигнал (сильный сигнал), то соотношение между их мощностями будет на столько большим, что слабый сигнал будет не заметен на фоне сильного. Т.е. на вывод микросхемы подаются оба сигнала, но ведущую роль играет только сильный.

Таким образом, подтягивающий резистор решает задачу защиты цифровой микросхемы от ложных срабатываний.

Лабораторные задания

Реализация ввода символов и цифр с клавиатуры и выдача в последовательный порт. Необходимо собрать схему, представленную на рис. 1.6.7.

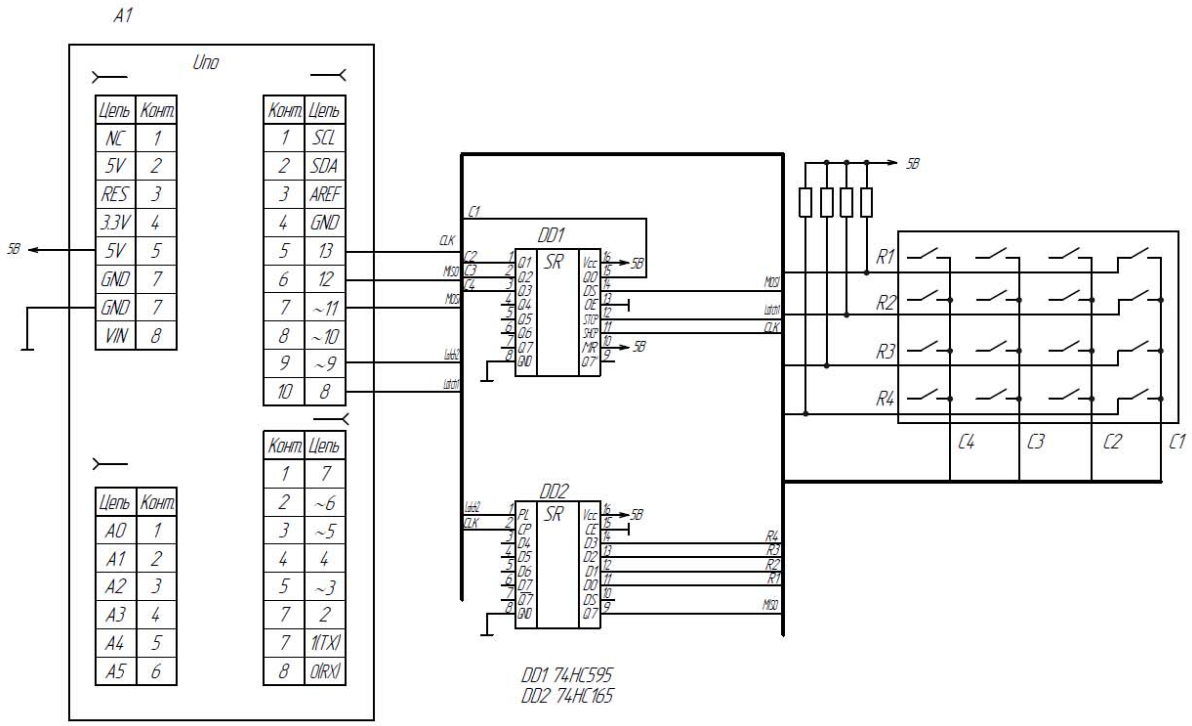


Рис. 1.6.7. Схема подключения матричной клавиатуры

Необходимо записать в платформу следующую программу.

```
/* Подключение клавиатуры через 74HC595 и 74HC165 вывод в Serial
74HC595 Map:
Q1   |1 * 16|  Vcc          PINS 1-7, 15   Q0 - Q7   Output Pins
Q2   |2   15|  Q0           PIN 8      GND      Ground, Vss
Q3   |3   14|  DS           PIN 9      Q7"      Serial Out
Q4   |4   13|  OE           PIN 10     MR       Master Reclear, active hi
```

```
Q5    |5   12|  ST_CP
Q6    |6   11|  SH_CP
Q7    |7   10|  MR
GND |8___9|  Q7"
```

PIN 11 SH_CP Shift register clock pin
PIN 12 ST_CP latch pin
PIN 13 OE Output enable, active low
PIN 14 DS Serial data input
PIN 16 Vcc Positive supply voltage

74HC165 Map:

```
PL    |1 *  16|  Vcc
CP    |2   15|  CE
D4    |3   14|  D3
D5    |4   13|  D2
D6    |5   12|  D1
D7    |6   11|  D0
Q!7   |7   10|  DS
GND |8___9|  Q7
```

PIN 11-14,3-6 D0-D7 Iput Pins
PIN 8 GND Ground, Vss
PIN 9 Q7 Serial Output
PIN 10 DS Serial Input
PIN 2 CP Shift register clock pin
PIN 1 PL (latch pin)
PIN 13 OE Output enable, active low
PIN 14 DS Serial data input
PIN 16 Vcc Positive supply voltage
PIN 15 CE Output enable, active low
PIN 7 !Q7 invers Serial Output

74HC595 №1 Подключаем к столбцам

```
     _______
    C2 -|1 *  16|-5V
    C3 -|2    15|-C1
    C4 -|3    14|-PIN 11 (MOSI)UNO
        -|4    13|-GND
        -|5    12|-PIN 8 (Latch (защелка))UNO
        -|6    11|-PIN 13 (CLK)UNO
        -|7    10|-5V
  GND-|8___9|-NILL
```

74HC165 №2 Подключаем к строкам

```
                  _______
 PIN 9(защелка)) UNO -|1 *  16|-5V
    PIN 13 (CLK)UNO -|2    15|-GND
                    -|3    14|- R4
                    -|4    13|- R3
                    -|5    12|- R2
                    -|6    11|- R1
                    -|7    10|-
                GND-|8___9|-PIN 12 (MISO)UNO
*/

// подключение библиотеки SPI
#include <SPI.h>
int ssout_pin=8; // пин SS out
```

```
int ssin_pin=9; // пин SS in
const char value[4][4]{
  {'1','2','3','+'},
  {'4','5','6','-'},
  {'7','8','9','*'},
  {'C','0','=','/'}
};
// двойной массив, обозначающий кнопку
  byte str=B11111111; // записываем в переменную строк изначально все единицы (не нажаты кнопки)
  void setup(){
SPI.begin();
Serial.begin(9600);
// Сконфигурировать защелки как выход и записать в них 1 (запрещен прием и передача)
pinMode(ssout_pin, OUTPUT);
pinMode(ssin_pin, OUTPUT);
digitalWrite(ssout_pin, HIGH);
digitalWrite(ssin_pin, HIGH);
}
void loop(){
for (int i=0; i<=3; i++ ){    // цикл сканирования столбцов
digitalWrite(ssout_pin, LOW); // Защелку выход вниз
SPI.transfer((B00000001<<i)^B11111111); // передаем маску столбца
digitalWrite(ssout_pin, HIGH); // Защелку выход вверх
digitalWrite(ssin_pin, LOW); // Защелку вход вниз
digitalWrite(ssin_pin, HIGH); // Защелку вход вверх
str=SPI.transfer(0);          // получаем маску строки
delay(50);
for (int j=0; j<=3; j++){ // цикл сканирования маски строки
if(bitRead(str,j)==LOW){        // проверяем, если в полученной маске строки на месте столбца 0
Serial.println(value[j][i]); // выводим символ
 }
}
}
}
```

Задания для самостоятельного решения

1. Необходимо выводить данные с матричной клавиатуры на LCD индикатор.

2. Необходимо реализовать программу часов с управлением от матричной клавиатуры.

Контрольные вопросы

1. Как происходит динамический опрос матричной клавиатуры?
2. Каково назначение выводов микросхемы 74HC165, и каков принцип работы сдвигового регистра?
3. Для чего нужны подтягивающие резисторы?
4. Как выбирается вид подтяжки резисторов?

Лабораторная работа №7
Расширитель I2C

Цель работы: изучить принцип действия интерфейса I2C, познакомиться с библиотекой Wire, изучить структуру и принцип работы микросхемы PCF8574.

Оборудование: персональный компьютер, среда разработки Arduino IDE, платформа Arduino UNO, макетная плата, провода, линейка светодиодов, потенциометр, LCD экран.

Теоретические сведения

Ранее мы подробно рассматривали расширители портов на сдвиговых регистрах. Это решение довольно распространенное, однако, у него есть и определенные недостатки.

Во-первых, как и все SPI-устройства, сдвиговый регистр требует для себя отдельной линии «chip select» (CS), и это при том, что шина как таковая и так занимает три свободных порта ввода/вывода (MOSI, MISO, SCK).

Во-вторых, сдвиговый регистр может увеличивать либо входы, либо выходы, но не то и другое стразу. А если нужны и входы, и выходы – то нужно ставить два регистра и выделять два вывода CS.

Для работы также можно использовать расширители портов на шине I2C. Одна из распространенных микросхем, реализующая это PCF8574. Для работы этого расширителя требуются всего два вывода (причем вне зависимости от количества устройств на шине). Устройство, с которым осуществляется обмен данными, определяется адресом, а не прижатием к нулю физической линии. Порты могут работать как порты ввода, так и порты вывода (то есть являются двунаправленными, хотя и с ограничениями). Структурная схема микросхемы, представлена на рис. 1.7.1 [12].

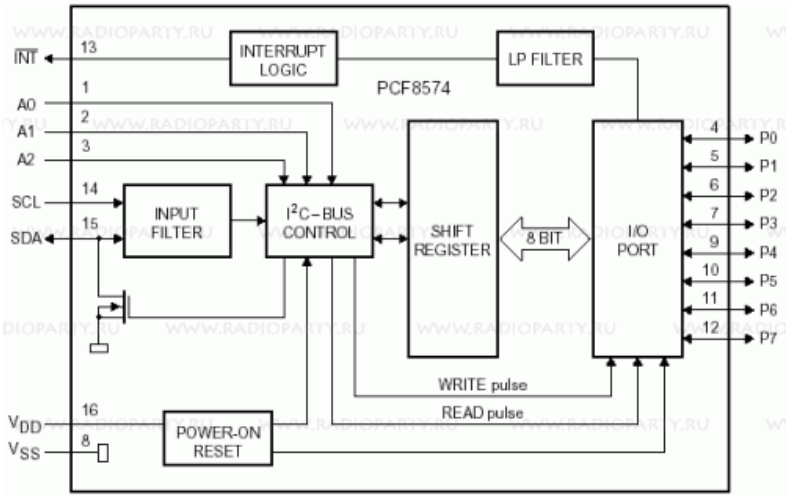


Рис. 1.7.1. Структурная схема PCF8574 [12]

Адрес устройства формируется из неизменного для всех чипов префикса и 3-битной переменной части, которую можно задать, подтягивая три специально выделенные выводы микросхемы (A0, A1, A2) к земле («0» в соответствующей позиции адреса) или к питанию («1»).

Подтягивать адресные выводы и к земле, и к питанию можно без какого бы то ни было дополнительного резистора.

Известно, что, существует две модификации этой микросхемы – собственно PCF8574 и PCF8574A, которые отличаются друг от друга только значением адресного префикса. У PCF8574 он равен 0100, а у PCF8574A – 0111. Сделано это для того, чтобы к одной шине можно было подключить до 16-ти расширителей с разными адресами – максимум 8 штук PCF8574 и 8 штук PCF8574A. Так как у каждой такой микросхемы по 8 портов, можно получить до 128 портов ввода-вывода.

Рассмотрим порядок формирования адреса устройства. По стандарту I2C адреса у устройств не 8-ми, а 7-ми битные. То есть всего на шине I2C может быть до 127 устройств. Младший же бит адреса используется для указания направления передачи данных. «0» в этом бите соответствует передачи данных в устройство («записи»), «1» – чтению.

Так вот, если организовывать обмен по I2C шине не самому, а использовать какую-нибудь библиотеку, то все эти особенности протокола библиотека уже учитывает и про значение последнего бита адреса можно не думать. Библиотека сама выставит его правильно, в зависимости от того, какую функцию – чтения или записи в шину – вы вызвали.

Таким образом, адрес PCF8574 для библиотеки и при чтении, и при записи будет один и тот же – 0100<A2><A1><A0> (где <A1>, <A2> и <A3> – настраевыемые биты адреса, выставленные путем подтяжки адресных выводов).

Вот соответствующая картинка из документации на микросхему PCF8574A фирмы NXP, только там еще и опечатка в скобке, показывающей ад-

рес для PCF8574A – в адрес включен R/W бит и адрес получился 8-ми битным, чего быть вообще не может. Схема образования адреса представлена на рис. 1.7.2 [12].

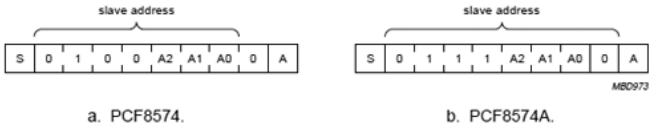


Рис. 1.7.2. Образование адреса микросхемы PCF8574 [12]

Порты у PCF8574 названются «квази-двунаправленными» (Quasi-bidirectional I/Os). На рис. 1.7.3 [12] представлена схема структурная одной линии порта PCF8574.

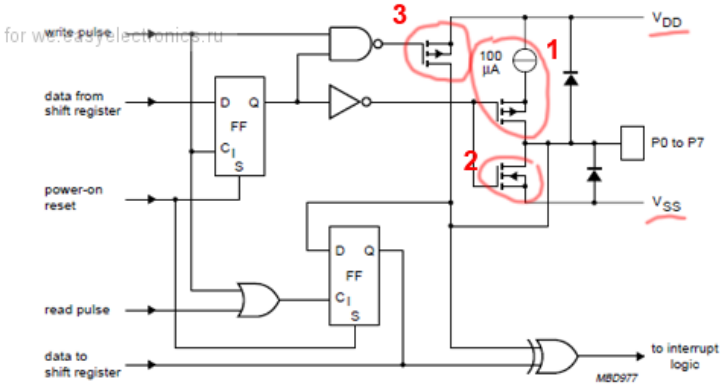


Рис. 1.7.3. Структурная схема одной линии порта PCF8574 [12]

Фактически, при операции записи одного байта в PCF8574 происходит выставление подтяжек портов ввода/вывода исходя из значения соответствующего порту бита данных.

Причем, если бит установлен в ноль, происходит подтяжка порта к земле (транзистор 2 на рис. 1.7.3).

А вот если бит установлен в единицу, порт будет подтянут к питанию, но с ограничением тока в 100 мкА (транзистор 1 на рис. 1.7.3). То есть логический уровень есть, но светодиод с такой подтяжкой гореть небудет.

Еще на схеме порта виден транзистор (транзистор 3 на рис. 1.7.3), который как раз может подтянуть порт к питанию напрямую. Но увы, он включается только на короткое время, чтобы обеспечить быстрое переключение порта в логическую единицу. И управлять поведением этого транзистора мы не можем.

Чтение из PCF8574 – операция более простая для понимания. Какие уровни установлены на портах с учетом текущей подтяжки и того, что подключено к портам извне – то и будет отправлено микроконтроллеру.

Не то, чтобы такое построение порта сильно ограничивало возможности, но оказаться неприятным сюрпризом оно вполне может. Так, светодиоды к PCF8574 надо подключать так, чтобы к порту подключался катод (минус) све-

тодиода. На светодиод, подключенный анодом (плюсом) поступает ограниченный ток в 100 мкА, что для него явно мало.

Определенные ухищрения придется предпринимать и при подключении семисегментных индикаторов. Индикаторы с общим катодом лучше вообще не использовать (или подключать через PNP транзисторы), а вот с общим анодом можно использовать без ограничений. Катоды сегментов можно напрямую подключить к PCF8574, а аноды знаков запитать, скажем, через полевые транзисторы P-типа, затворы которых будут управляться 100 мкА подтяжкой PCF8574. Использование каких-нибудь транзисторов на питании общих анодов вполне оправдано, так как с каждого из них питаются восемь сегментов.

Очевидно, что для управления 2-х и более значным семисегментным индикатором понадобится не менее двух PCF8574 – одна для 8ми катодов сегментов, а вторая – для управления транзисторами питания знаков.

Особенности подтяжки надо учитывать и при работе портов на ввод. Например, если подключить к порту PCF8574 тактовую кнопку, которая при нажатии подсоединяет этот порт к земле, то все будет работать – операцией записи логической «1» в PCF8574 заранее подтягивает этот порт к питанию и, как только кнопка будет нажата и порт окажется соединен с землей, слабую подтяжку сорвет и при очередном опросе в бите, соответствующем порту кнопки получится логический «0» вместо «1».

Однако, если при нажатии кнопка подтягивает порт к питанию в случае заранее записи в него логической «1» его состояние не будет меняться при нажатии кнопки.

А если подтянуть его заранее к земле, то при нажатии кнопки питание через кнопку и порт окажется напрямую подключенным к земле. Будет короткое замыкание.

Инструкция на микросхему PCF8574 в явном виде требует перед использованием порта в качестве порта ввода предварительно записать в него единицу. Вот именно по указанной причине.

Также микросхема PCF8574 поддерживает генерацию сигнала прерывания, прижимая к земле вывод INT при изменении уровня на одном из портов. INT возвращается к высокому уровню при следующей же операции чтения из PCF8574.

Использование этой возможности занимает дополнительный порт микроконтроллера, но иногда это необходимо – например, если в контроллере необходимо организовать аппаратное прерывание.

Расширитель может быть представлен в виде готового модуля, например Wavgat PCF8574 (рис. 1.7.4) [13].

Рис. 1.7.4. Внешний вид модуля Wavgat PCF8574 [13]

Основные технические характеристики:
- входной интерфейс: I2C (100 кГц);
- количество выводов I/O: 8;
- количество выводов прерывания: 1 (с открытым стоком);
- напряжение питания: 3,3-5,5В;
- максимальный ток вывода при состоянии HIGH: 300мкА;
- максимальный ток вывода при состоянии LOW: 25мА;
- ток покоя: <10мкА
- I2C адрес: на выбор 8 вариантов.

Для работы с шиной I2C используется библиотека Wire.

Рассмотрим ее команды.

Wire.begin()

Wire.begin(address)

Инициализирует библиотеку Wire и подключает Ардуино к шине I2C в роли ведущего (master) или ведомого (slave) устройства. Как правило, эта функция вызывается только один раз.

Параметры:

address: 7-битный адрес ведомого устройства (не обязательный параметр); если адрес не указан, то Ардуино выступает в роли ведущего устройства (master).

Возвращаемые значения: нет.

Wire.beginTransmission(address)

Начинает процедуру передачи данных по интерфейсу I2C ведомому устройству с указанным адресом. Для последующей отправки данных, необходимо сначало поставить их в очередь с помощью функции write(), после чего осуществить, непосредственно, передачу функцией endTransmission().

Параметры: address: 7-битный адрес принимающего устройства.

Возвращаемые значения: Нет.

Wire.endTransmission()

Описание.

Завершает процедуру передачи данных ведомому устройству, инициированную функцией beginTransmittion(). При этом функция отправляет байты, поставленные в очередь функцией write().

Начиная с версии Ардуино 1.0.1, функция endTransmission() может принимать логический параметр, способствующий лучшей совместимости с некоторыми I2C-устройствами.

Если этот параметр равен true, то функция requestFrom() отправит запрос со стоповым битом, что позволит освободить шину I2C.

Если этот параметр равен false, то после отправки запроса шина по-прежнему будет занята, что предотвратит отправку посторонних сообщений другими ведущими устройствами. Этот режим позволяет Мастеру отправлять по несколько запросов за один сеанс.

Значение по умолчанию – true.

Синтаксис:

Wire.endTransmission()

Wire.endTransmission(stop)

Параметры.

stop: boolean. При значении true будет отправлен запрос со стоповым битом, что позволит освободить шину. При значении false - соединение будет поддерживаться в активном состоянии.

Возвращаемые значения. byte, байт данных, характеризующий статус передачи:

0: передача успешна

1: объем данных слишком велик для буфера передачи

2: получен NACK при передаче адреса

3: получен NACK при передаче данных

4: другая ошибка

write()

На ведомом устройстве (Slave) данная функция отправляет данные в ответ на запрос ведущего устройства. На ведущем устройстве (Master) функция добавляет данные в очередь отправки для последующей передачи ведомому устройству (в этом случае функция write() должна вызываться между beginTransmission() и endTransmission()).

Синтаксис

Wire.write(value)

Wire.write(string)

Wire.write(data, length)

Параметры.

value: значение, которое необходимо отправить в виде одиночного байта

string: строка, которую необходимо отправить в виде последовательности байт

data: массив данных, который необходимо отправить в виде нескольких байт

length: количество передаваемых байт

Возвращаемые значения

byte

Функция write() возвращает количество записанных байт. Считывание этого значения не обязательно

Пример

```
#include <Wire.h>
byte val = 0;
void setup(){
  Wire.begin(); // подключаемся к шине i2c
}
void loop()
```

```
{
  Wire.beginTransmission(44); // начинаем процедуру передачи устройству с ад-
ресом #44 (0x2c)
                        // адрес устройства указан в даташите
  Wire.write(val);            // отправляем байт данных
  Wire.endTransmission();     // завершаем процедуру передачи
  val++;        // увеличиваем значение
  if(val == 64) // при достижении значения 64 (максимум)
  {
    val = 0;    // начинаем счет заново  }
  delay(500);
}
```

Wire.requestFrom()

Функция запрашивает данные у ведомого устройства (slave); как правило, используется только ведущим устройством (Master). После вызова requestFrom() запрашиваемые данные должны быть считаны с помощью функций available() и read().

Начиная с версии Ардуино 1.0.1, функция requestFrom() может принимать третий параметр – логическое значение, обеспечивающее лучшую совместимость с некоторыми I2C-устройствами.

Если этот параметр равен true, то функция requestFrom() отправит запрос со стоповым битом, что позволит освободить шину I2C.

Если этот параметр равен false, то после отправки запроса шина по прежнему будет занята, что предотвратит отправку посторонних сообщений другими ведущими устройствами. Этот режим позволяет Мастеру отправлять по несколько запросов за один сеанс.

Значение по умолчанию – true.

Синтаксис.

Wire.requestFrom(address, quantity)

Wire.requestFrom(address, quantity, stop)

Параметры.

address: 7-битный адрес ведомого устройства, у которого запрашиваются данные.

quantity: количество запрашиваемых байт.

stop: boolean. При значении true будет отправлен запрос со стоповым битом, что позволит освободить шину. При значении false – соединение будет поддерживаться в активном состоянии.

Возвращаемые значения.

byte : количество байт, возвращенных ведомым устройством.

Wire.available()

Возвращает количество байт, доступных для считывания функцией read(). На ведущем устройстве (Master) данная функция должна вызываться после функции requestFrom(), а на ведомом (Slave) - внутри обработчика onReceive().

Функция available() является наследником вспомогательного класса Stream.

Параметры: нет.

Возвращаемые значения: Количество байт, доступных для считывания.

read()

Данная функция считывает байт данных, полученный ведущим устройством от ведомого (либо наоборот) в результате выполнения функции requestFrom(). Функция read() является наследником вспомогательного класса Stream.

Синтаксис.

Wire.read()

Параметры: нет.

Возвращаемые значения: очередной полученный байт.

Пример

```
#include <Wire.h>
 void setup(){
  Wire.begin();        // подключаемся к шине i2c (для ведущего устройства адрес не обязательный)
  Serial.begin(9600);  // инициализируем последовательный порт для вывода информации}
 void loop(){
  Wire.requestFrom(2, 6);    // запрашиваем у ведомого устройства #2 6 байт
   while(Wire.available())    // ведомое устройство может отправить не все запрашиваемые байты
  {
    char c = Wire.read();    // считываем байт данных в виде символа
    Serial.print(c);         // выводим символ
  }

  delay(500);
}
```

Wire.onReceive(handler)

На ведомом устройстве позволяет назначить функцию, которая будет автоматически вызываться при поступлении данных от Мастера.

Параметры.

handler: функция, которую необходимо вызвать при поступлении данных от ведущего устройства; эта функция не должна возвращать никаких значений и может принимать только один параметр (int), описывающий количество поступивших байт. Например: void myHandler(int numBytes).

Возвращаемые значения

нет

Wire.onRequest(handler)

На ведомом устройстве позволяет назначить функцию, которая будет автоматически вызываться при получения запроса от Мастера.

Параметры.

handler: вызываемая функция без параметров, не должна возвращать никаких значений. Например: void myHandler()

Возвращаемые значения

нет.

Лабораторные задания

Аналоговая линейка. В зависимости от положения ручки потенциометра поочередно зажигаются светодиоды и сегменты на LCD.

Схема представлена на рис. 1.7.5.

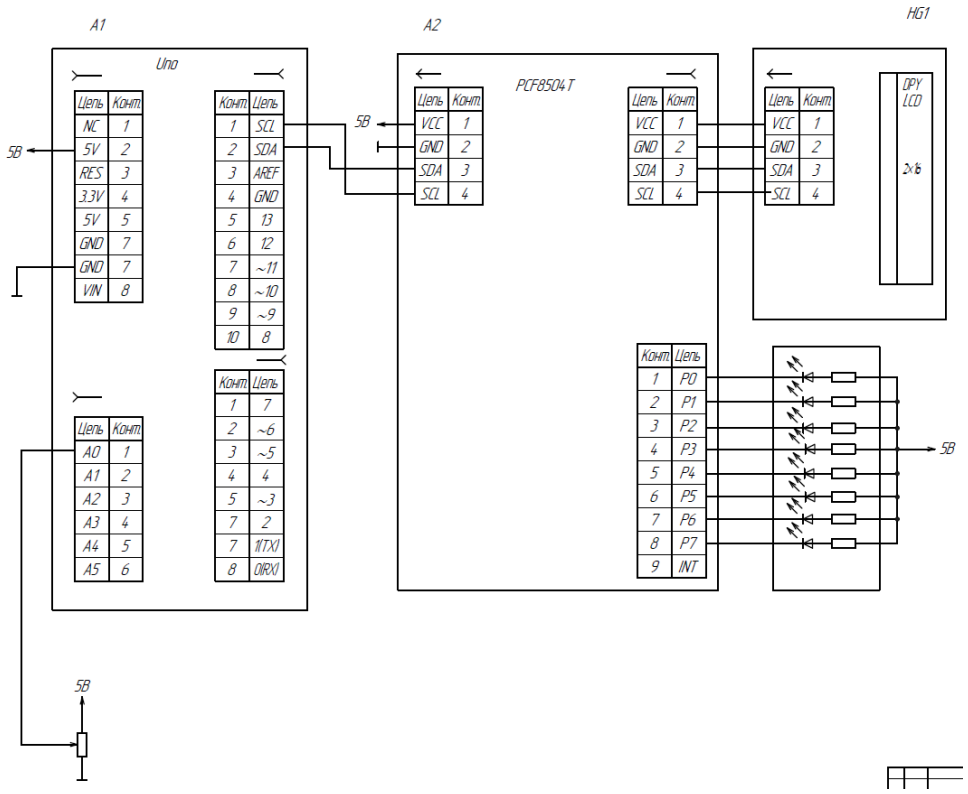


Рис. 1.7.5. Схема подключения модуля Wavgat PCF8574, светодиодной линейки и LCD индикатора

Необходимо в платформу записать следующую программу.

```
// АНАЛОГОВАЯ ЛИНЕЙКА НА I2C и ЖК
#include <Wire.h>                    //  Подключаем библиотеку для работы с шиной I2C
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
uint8_t symbol[8] = {31,31,31,31,31,31,31,31};//  Определяем массив который содержит полностью закрашенный символ
```

```
int     pinSensor = A0;              //  Вывод переменного резистора
int      valSensor;                  //  Объявляем  переменную для хранения данных с аналогового входа pinSensor
int     lin1,lin2;
int     address=0x20;
void setup(){
   lcd.init();                                //  Инициируем работу с LCD дисплеем
   lcd.backlight();                           //  Включаем подсветку LCD дисплея
   lcd.createChar(1, symbol);                 //  Загружаем символ из массива symbol в первую ячейку ОЗУ дисплея
    Wire.begin();                 // Инициализация библиотеки Wire
    Serial.begin(9600);
}
 void loop(){                                 //
   valSensor = analogRead(pinSensor);         //  Читаем данные с аналогового входа pinSensor в переменную valSensor
   lin1=map(valSensor,0,1024,0,9);            //  Преобразуем переменную от диапазона 0...1024 к диапазону 0...8
  //  val=B11111110<<valSensor
   Wire.beginTransmission(address);           //Начинаем процедуру передачи данных по интерфейсу I2C
   if (lin1==0)Wire.write(B11111111);
   else Wire.write(B11111110<<lin1-1);        // отправляем байт данных маска B11111110 сдвигается побитово влево на значение valSensor
   Wire.endTransmission();                    // Завершаем процедуру передачи данных
  // Serial.println(B11111110<<valSensor, BIN);
  // delay (10);
   lcd.setCursor(6, 1); lcd.print("    ");    //  Устанавливаем курсор дисплея на 6 символ 1 строки и выводим 4 пробела начиная с той позиции, куда ранее был установлен курсор.
   lcd.setCursor(6, 1); lcd.print(valSensor); //  Устанавливаем курсор дисплея на 6 символ 1 строки и выводим значение переменной valSensor на дисплей (начиная с той позиции, куда ранее был установлен курсор)
   lcd.setCursor(0, 0);                       //  Устанавливаем курсор дисплея на 0 символ 0 строки для вывода шкалы, дальнейший вывод символов начнётся именно с этой позиции
   lin2=map(valSensor,0,1023,0,17);           //  Определяем переменную j которой присваиваем значение valSensor преобразованное от диапазона 0...1023 к диапазону 0...17
   for(int i=0; i<16; i++){                   //  Выполняем цикл 16 раз для вывода шкалы из 16 символов начиная с позиции в которую ранее был установлен курсор
```

```
    lcd.write(lin2>i? 1:32);                    //  Выводим на дисплей символ по
его коду, либо 1 (символ из 1 ячейки ОЗУ дисплея), либо 32 (символ пробела)
  }
  }
```

Клавиатура на I2C. Программа опрашивает клавиатуру и выводит символы в Serial. Схема представлена на рис. 1.7.6.

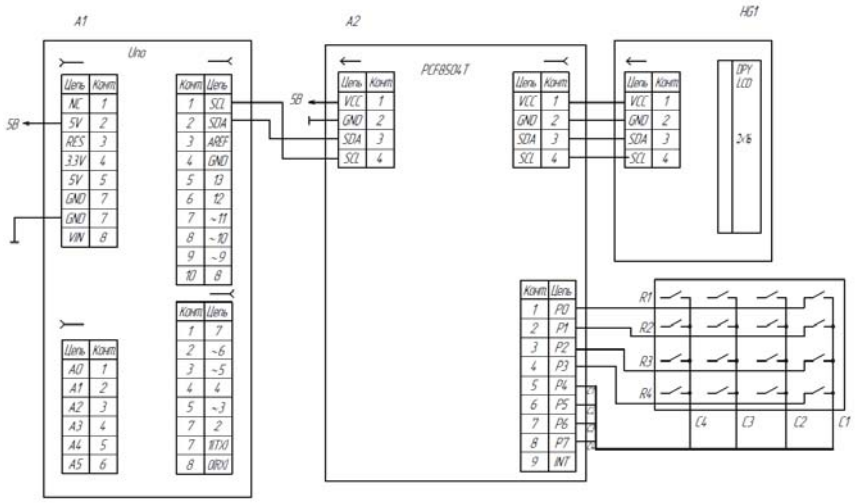


Рис. 1.7.6. Схема подключения модуля Wavgat PCF8574,
клавиатуры и LCD индикатора

Необходимо в платформу записать следующую программу.

```
// КЛАВИАТУРА НА I2C
#include <Wire.h>                    //  Подключаем библиотеку для работы с шиной
I2C
#include <LiquidCrystal_I2C.h>       // Подключаем библиотеку LCD
LiquidCrystal_I2C lcd(0x27,16,2);    // Задаем размерность LCD и адрес
int    address=0x20;                 // Адрес расширителя I2C
byte str=B11111111;                  // Маска
const char value[4][4]{              // Массив символов
  {'1','2','3','+'},
  {'4','5','6','-'},
  {'7','8','9','*'},
  {'C','0','=','/'}
};
void setup(){
    Wire.begin();                    // Инициализация библиотеки Wire
    Serial.begin(9600);
     Wire.beginTransmission(address);    //Начинаем процедуру передачи данных
по интерфейсу I2C
     Wire.write(B11111111);          // Записываем 1 во все разряды
```

```
    Wire.endTransmission();          // Завершаем процедуру передачи данных
}
 void loop(){                                  //
  for (int i=0; i<=3; i++ ){    // цикл сканирования столбцов
Wire.beginTransmission(address);
Wire.write((B00000001<<i)^B11111111); // передаем маску столбца с 0 по 3 разряд
//Serial.println((B00000001<<i)^B11111111, BIN); // это я для отладки выводил маску столбцов в Serial
 Wire.endTransmission();
Wire.requestFrom(address, 1);                  // Получаем 1 байт
str=Wire.read();                               // получаем маску строки (строки начинаются с 4 разряда)
//Serial.println(str,BIN);
delay(50);
for (int j=4; j<=7; j++){ // цикл сканирования маски строки начиная с 4 разряда (с 0 по 3 передается маска столбца)
//Serial.println (str, BIN);  // это я для отладки выводил маску строк в Serial
if(bitRead(str,j)==LOW){        // проверяем, если в полученной маске строки на месте столбца 0
Serial.println(value[i][j-4]); // выводим символ
 }
}
}
}
```

Задания для самостоятельного решения

1. Необходимо подключить через шину IIC кнопки и реализовать регулировку яркости светодиода, посредством ШИМ.

2. Необходимо подключить одноразрядный семисегментный индикатор через шину IIC.

Контрольные вопросы

1. Каковы преимущества шины I2C?
2. Каким образом формируется адрес устройства на шине I2C?
3. Почему линии портов в микросхеме PCF8574 называются «квазидвунаправленные»?
4. Почему не требуется применение подтягивающих резисторов при использовании микросхемы PCF8574?
5. Каковы основные команды библиотеки Wire?

Лабораторная работа №8
Использование ИК пульта для управления

Цель работы: познакомиться с библиотекой iarduino_IR, составить, компилировать и загрузить в микроконтроллер программы на Arduino.

Оборудование: персональный компьютер, среда разработки Arduino IDE, платформа Arduino UNO, макетная плата, провода, светодиоды, IR приемник, пульт ДУ, транзисторный модуль, светодиодная линейка.

Теоретические сведения

Инфракрасный пульт дистанционного управления – один из самых простых способов взаимодействия с электронными приборами. Так, практически в каждом доме есть несколько таких устройств: телевизор, музыкальный центр, видеоплеер, кондиционер.

Для передачи сигналов ИК требуется пульт дистанционного управления. Можно использовать обычный пульт от телевизора, а можно использовать миниатюрный пульт от автомагнитолы (рис. 1.8.1) [14]. Именно такие пульты часто используются для управления.

Рис. 1.8.1. Внешний вид ИК пульта [14]

На таком пульте есть 10 цифровых кнопок и 11 кнопок для манипуляции с музыкой: громкость, перемотка, play, stop, и т.д.

Для приема ИК сигнала существует специализированный ИК приемник. Он в себе объединяет:

- фильтр светового потока;
- фотодиод;
- предусилитель;
- полосовой фильтр несущей частоты – 38 кГц;
- демодулятор;
- операционный усилитель;
- блоки защиты от помех (электромагнитных, световых, пульсаций напряжения).

На выходе приемник дает сигнал уровня TTL, который может быть обработан микроконтроллером. Приемники рассчитаны на определенный диапазон частот. Наиболее распространенными являются 36 и 38 кГц. В названии приемника имеются 4 цифры, например 1838, при этом 18 это номер модели а 38 частота. На рис. 1.8.2 [14] представлен приемник VS1838B. Он имеет следующие характеристики:

- несущая частота: 38 кГц;
- напряжение питания: 2,7-5,5 В;
- потребляемый ток: 50 мкА.

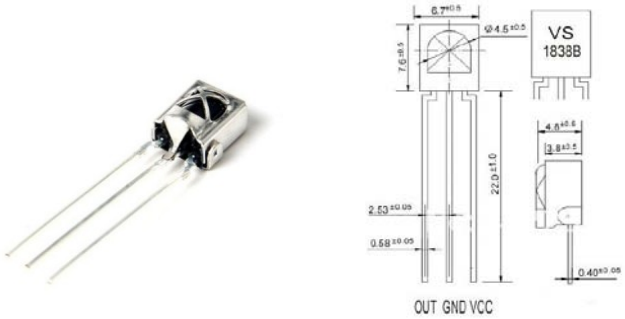


Рис. 1.8.2. Внешний вид и назначение выводов
ИК приемника VS1838B [14]

А на рис. 1.8.3 [14] представлен TSOP1738. У них некоторая разница в назначении выводов.

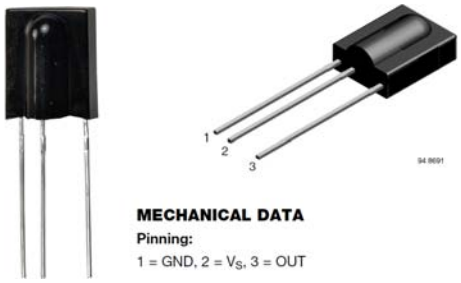


Рис. 1.8.3. Внешний вид и назначение выводов ИК приемника TSOP1738 [14]

Для работы с ИК приемником используется библиотека iarduino_IR.

Библиотека позволяет работать с ИК-приёмником и (или) ИК-передатчиком.

Для работы с ИК-приёмником, нужно создать объект класса iarduino_IR_RX.

Для работы с ИК-передатчиком, нужно создать объект класса iarduino_IR_TX.

Для работы с ИК-приёмником и ИК-передатчиком, нужно создать оба объекта.

Библиотека использует второй аппаратный таймер, как для раскодирования данных с ИК-приёмника «в фоновом режиме», так и для формирования несущей частоты ИК-передатчика. Не выводите сигнал ШИМ на 3 или 11 вывод, это помешает корректной работе библиотеки.

Управление ИК-приёмником:

#include <iarduino_IR_RX.h> // Подключаем библиотеку.

iarduino_IR_RX ОБЪЕКТ (№_ВЫВОДА [, ИНВЕРСИЯ]); // Объявляем объект.

Функция ***begin()***; // Инициализация работы с ИК-приёмником.

Назначение: инициализация работы с ИК-приёмником

Синтаксис: begin();

Параметры: Нет.

Возвращаемые значения: Нет.

Примечание: Вызывается 1 раз в коде setup.

Пример:

IR.begin(); // Инициируем работу с ИК-приёмником

Функция ***check([УДЕРЖАНИЕ])***; // Проверка наличия принятых с пульта данных.

Назначение: Проверка наличия принятых с пульта данных.

Синтаксис: check([УДЕРЖАНИЕ]);

Параметры:

УДЕРЖАНИЕ – необязательный параметр, типа bool – указывающий, что необходимо реагировать на удержание кнопок пульта.

Возвращаемые значения: bool – приняты или нет, данные с пульта.

Примечание: Если функция вызвана без параметра, или он равен false, то функция будет реагировать только на сигналы с пульта при нажатии его кнопок, а если указать true, то функция будет реагировать, как на нажатие, так и на удержание кнопок пульта.

Пример:

if(IR.check()){ ... ;} // Если приняты данные с пульта, при нажатии его кнопки

if(IR.check(true)){ ... ;} // Если принимаются данные с пульта, при удержании кнопки

Функция ***protocol([ПАРАМЕТР])***; // Получение, установка или сброс протокола передачи данных.

Назначение: Получение, установка или сброс протокола передачи данных.

Синтаксис: protocol([ПАРАМЕТР]);

Получение протокола: Если функция вызвана без параметра, то она вернёт строку из 25 символов + символ конца строки. Биты данной строки, несут информацию о типе протокола передачи данных пульта, данные которого были приняты последними. Данную строку можно использовать для установки протокола ИК-передатчику, или ИК-приёмнику (см.ниже).

Установка протокола: Если функция вызвана с параметром в виде строки из 25 символов протокола + символ конца строки, то после этого, функция chek(), будет реагировать только на пульты, соответствующие указанному протоколу передачи данных.

Сброс протокола: Если функция вызвана с параметром IR_CLEAN, то функция chek() опять станет реагировать на сигналы с любых пультов.

Получение параметров протокола: Если функция вызвана с параметром int, от 0 до 17, то она вернёт не строку протокола, а значение типа int с одним из параметров протокола передачи данных пульта, данные которого были приняты последними:

0 – тип кодировки:

IR_UNDEFINED – тип кодировки не определён;

IR_PAUSE_LENGTH –кодирование длинной паузы;

IR_PULSE_LENGTH – кодирование длинной (шириной) импульса (ШИМ);

IR_BIPHASIC – бифазное кодирование;

IR_BIPHASIC_INV – бифазное кодирование с инверсными битами;

IR_NRC – пакеты повтора идентичны, а первый и последний пакеты специальные;

IR_RS5 – кодировка PHILIPS с битом toggle;

IR_RS5X – кодировка PHILIPS с битом toggle;

IR_RS6 – кодировка PHILIPS с битом toggle.

1 – несущая частота передачи данных (в кГц);

2 – заявленное количество информационных бит в 1 пакете;

3 – заявленное количество информационных бит в пакете повтора;

4 – длительность паузы между пакетами (в мс);

5 –длительность импульса в стартовом бите (в мкс);

6 – длительность паузы в стартовом бите (в мкс);

7 – длительность импульса в стоповом бите (в мкс);

8 – длительность паузы в стоповом бите (в мкс);

9 – длительность импульса в бите рестарт или toggle (в мкс);

10 – длительность паузы в бите рестарт или toggle (в мкс);

11 – позиция бита рестарт или toggle в пакете (№ бита);

12 – максимальная длительность импульса в информационных битах (в мкс);

13 – минимальная длительность импульса в информационных битах (в мкс);

14 – максимальная длительность паузы в информационных битах (в мкс);

15 – минимальная длительность паузы в информационных битах (в мкс);

16 – флаг наличия стартового бита (true/false);

17 – флаг наличия стопового бита (true/false);

18 – флаг наличия бита рестарт или toggle (true/false);

19 – тип пакета повтора (0 – нет, 1 – с инверсными битами, 2 – идентичен информационному, 3 – уникален);

Возвращаемые значения: Зависят от наличия и типа параметра.

Примечание: Если ранее был установлен протокол, то попытка получения протокола, или параметров протокола, вернёт значения установленного ранее протокола, а не протокола передачи данных пульта, данные которого были приняты последними.

Пример:

IR.protocol("AeQQV~zK]Kp^KJp[@@@@@@@@Bp"); // Устанавливаем протокол. Теперь приёмник будет получать данные, только от пультов телевизора ELENBERG.

IR.protocol(IR_CLEAN); // Сбрасываем ранее установленный протокол. Теперь приёмник снова будет реагировать на любые пульты.

if(IR.check()){ Serial.println(IR.protocol()); } // Получаем протокол. Как только приёмник получит данные, в мониторе высветится строка из 25 символов протокола.

if(IR.check()){ Serial.println(IR.protokol(12)); } // Получаем один из параметров протокола. Как только приёмник получит данные, в мониторе отобразится максимальная длительность импульса информационного бита в микросекундах.

Переменная ***data*** // Возвращает код кнопки, принятый с пульта.

Значение: Возвращает код кнопки, принятый с пульта;

Тип данных: uint32_t.

if(IR.check()){ Serial.println(IR.data); } // Выводим код нажатой кнопки, если он принят

Переменная ***length*** // Возвращает размер принятого кода, в битах.

Значение: Возвращает размер кода кнопки, в битах;

Тип данных: uint8_t.

if(IR.check()){ Serial.println(IR.length); } // Выводим размер кода нажатой кнопки, если он принят

Переменная ***key_press*** // Возвращает флаг, указывающий на то, что кнопка нажимается а не удерживается.

Значение: Возвращает флаг, указывающий на то, что кнопка пульта нажимается а не удерживается;

Тип данных: bool.

if(IR.check(true)){

if(IR.key_press){Serial.println("PRESS");} // Текст будет выведен 1 раз, когда кнопка нажимается else {Serial.println("HOLD ");} // Текст будет выводиться постоянно, пока кнопка удерживается

Управление ИК-передатчиком:

#include <iarduino_IR_TX.h> // Подключаем библиотеку iarduino_IR_TX ОБЪЕКТ (№_ВЫВОДА [, ИНВЕРСИЯ]); // Объявляем объект.

Функция begin(); // Инициализация работы с ИК-передатчиком.

Функция send(ДАННЫЕ [, УДЕРЖАНИЕ]); // Передача данных.

Функция protocol(СТРОКА); // Установка протокола передачи данных.

Переменная frequency // Устанавливает несущую частоту передачи данных в кГц.

Лабораторные задания

Прием сигнала с ИК пульта и выдача кодов кнопок в последовательный порт.

Необходимо подключить ИК приемник к платформе, как показано на рис. 1.8.4.

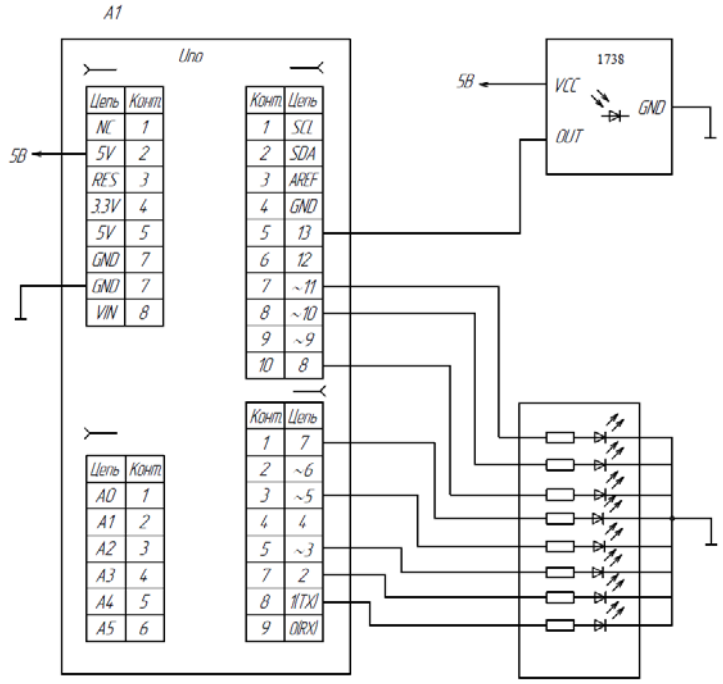


Рис. 1.8.4. Схема подключения ИК приемника и светодиодной линейки

В платформу надо загрузить следующую программу.

```
// Программа определения кода нажатой кнопки
#include <iarduino_IR_RX.h>                    // Подключаем библиотеку для работы с ИК-приёмником
iarduino_IR_RX IR(7);                          // Объявляем объект IR, с указанием вывода к которому подключён ИК-приёмник
void setup(){
  Serial.begin(9600);                          // Инициируем передачу данных в монитор последовательного порта, на скорости 9600 бит/сек
  IR.begin();                                  // Инициируем работу с ИК-приёмником
}
```

```
void loop(){
  if(IR.check()){                          // Если в буфере имеются данные, принятые с
пульта (была нажата кнопка)
    Serial.println(IR.data, HEX);              // Выводим код нажатой кнопки
    Serial.println(IR.length    );             // Выводим количество бит в коде
  }
}
```

Управление от пульта скоростью переключения 8 светодиодов типа «бегущая волна». Схема прежняя. Программа следующая.

```
/*,БЕГУЩАЯ СТРОКА УПРАВЛЕНИЕ ОТ ИК
*/int pins[8]={1,2,3,5,7,8,10,11};// 8 светодиодов
int timedown=5;
int speedlight=100;
#include <iarduino_IR_RX.h>                    // Подключаем библиотеку для работы
с ИК-приёмником
iarduino_IR_RX IR(13);                         // Объявляем объект IR, с указанием вы-
вода к которому подключён ИК-приёмник
void setup(){
 IR.begin();                                   // Инициируем работу с ИК-приёмником
// Сконфигурировать контакты как выходы
for(int i=0;i<=7;i++){
pinMode(pins[i],OUTPUT);
digitalWrite (pins[i],0);
}
}
void loop(){
  if(IR.check()){                          // Если в буфере имеются данные, принятые с
пульта (была нажата кнопка)
    switch(IR.data){
      case 0xFFA857:
      speedlight-=15;     // do something
      break;
      case 0xFFE01F:
      speedlight+=15;     // do something else...
      break;
      }
    }
    speedlight=constrain (speedlight, 2,400);
 //Serial.println (speedlight);
  for(int i=0;i<=7;i++){
digitalWrite(pins[i],HIGH);
delay (speedlight);
```

```
digitalWrite(pins[i],LOW);
delay (timedown);
}
}
```

Задания для самостоятельного решения

1. Необходимо модифицировать программу, предусматривающую возможность изменения яркости свечения светодиодов с управлением от ИК пульта и использованием транзистора.
2. Необходимо задействовать большее число кнопок, реализующих разный алгоритм зажигания светодиодов в линейке.

Контрольные вопросы

1. Опишите принцип работы ИК пульта и приемника ИК сигналов.
2. Какие бывают разновидности ИК пультов и приемников?
3. Что содержит в себе ИК приемник?
4. Каким образом можно осуществить двустороннюю передачу и прием сигнала по ИК?

Лабораторная работа №9
Графический индикатор LCD 5110. Подключение индикатора LCD 5110 к микроконтроллеру, вывод информации на индикатор

Цель работы: познакомиться с индикатор LCD 5110, научиться программировать и выводить информацию на LCD индикатор с использованием микроконтроллера Arduino UNO.

Оборудование: персональный компьютер, среда разработки Arduino IDE, платформа Arduino UNO, макетная плата, провода, индикатор LCD 5110, потенциометр.

Теоретические сведения

Дисплей Nokia 5110 LCD (blue screen) Модуль состоит из печатной платы, на которой размещается жидкокристаллический индикатор Nokia 5110 содержащий контроллер PCD8544 фирмы Philips. Внешний вид дисплея представлен на рис. 1.9.1 [2].

Для соединения модуля с другими устройствами плата содержит вилку соединителя и отверстия для проводов. Также имеются 4 установочных отверстия, расположенные по углам платы. Монохроматический дисплей Nokia 5110 LCD (blue screen) имеет подсветку синими светодиодами.

За годы производства Nokia 5110 применялся и применяется в широком спектре приложений. Изначально применявшийся как индикатор мобильного телефона дисплей распространился в другие категории приборов. Его удобно использовать в измерительных приборах: вольтметрах, амперметрах, омметрах и других. Пригодится дисплей и для индикации показаний медицинских мобильных приборов. Интересно его применить для шкалы радиоприемника или индикатора уровня сигнала в звуковоспроизводящей аппаратуре.

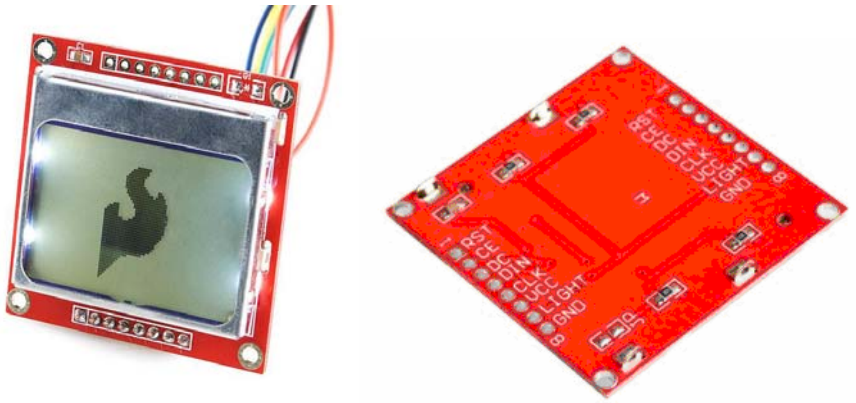


Рис. 1.9.1. Внешний вид дисплея Nokia 5110 [2]

Экран разрешением 84x48 точек позволяет выводить символьную и графическую информацию.

Характеристики модуля Nokia 5110 LCD:

- питание напряжение 2,7…3,3 В;
- ток подсветка отключена 5 мА;
- подсветка включена 20 мА;
- температура воздуха во время работы 0…50;
- температура хранения -10…70.

Структура дисплея следующая. Дисплей представляет собой матрицу ЖК-элементов и микросхему PCD8544 для их управления, размещенные в корпусе установленном на плате. На ней так же размещены четыре светодиода подсветки экрана. Информация о состоянии точек дисплея хранится в оперативной памяти контроллера PCD8544, каждой точке соответствует один бит памяти. Так же встроен счетчик адреса, который автоматически увеличивается при записи очередного байта информации в память.

Управление дисплеем Осуществляется по интерфейсу SPI, дисплей является ведомым устройством. Однако, вместо обычных четырех линий управления здесь лишь три. Это линии тактирования CLK, выбора кристалла SCE и входных данных MOSI. Линия выхода MISO отсутствует. Это приводит к необходимости применять специальные методы управления.

Самый простой вариант подключения к Arduino напрямую. В этом случае надо использовать платы Arduino Pro 3.3V/8MHz или 3.3V Arduino Pro Mini. Вариант, предложенный ниже работает с платами Arduino 5V. Это рабочий вариант, но срок работы дисплея может несколько сократиться.

В широком распространении можно найти два варианта исполнения модулей: на синей и красной плате. Оба модуля имеют восемь выводов

Предназначение:

RST – Вывод для перезагрузки контроллера дисплея.

CE – Состояние данного вывода разрешает или запрещает ввод данных в контроллер дисплея

DC – Вывод выбора режима ввода данных – Данные/Команды

DIN – Вход данных последовательного интерфейса SPI

CLK – Тактирующий вывод для последовательного интерфейса SPI

VCC – Питание контроллера дисплея 2,7-3,3Вольт

BL (LIGHT) –Подсветка

GND – GND

Внутри дисплея находится контроллер PCD8544. **Его питание должно лежать в пределах 2,7- 3,3 В (максимум 3,3 В, при подаче 5 В на вывод VCC дисплей может выйти из строя)**.

Сигнальные же выводы поддерживают подключение к 5 В и подключаются к любым цифровым выводам Arduino. Необходимо подключить индикатор как указано в табл. 1.9.1:

Таблица 1.9.1

Порядок подключения дисплея Nokia 5110

Nokia 5110 модуль	Arduino
RST	D7
CE	D6
DC	D5
Din	D4
CLK	D3
VCC	3.3B
GND	GND

При использовании синего модуля, подсветка дисплея "BL" активизируется подачей 3.3 либо 5 В. При использовании красного модуля, подсветка дисплея "LIGHT" активизируется подачей минуса (GND).

Для работы с индикатором разработана библиотека Adafruit_GFX. Основные функции и команды библиотеки следующие:

Adafruit_GFX functions

display - отображение на экране. Вызывается после каждого изменения экрана. КАЖДОГО!

clearDisplay – очищает экран;

fillScreen(color) **–** закрашиваем весь дисплей в цвет color (константы BLACK или WHITE);

drawPixel(x, y, color) **–** рисует пиксель по координатам x, y с цветом color;

drawLine(x, y, x1, y1, color) **–** рисует линию с началом в координатах x, y и концом в координатах x1, y1 цветом color;

drawFastVLine(x, y, length, color) **–** рисует горизонтальную линию с началом в координатах x, y, длиной length и цветом color (быстрее чем drawLine);

drawFastHLine(x, y, length, color) **–** рисует вертикальную линию с началом в координатах x, y, длиной length и цветом color (быстрее чем drawLine);

drawRect(x, y, width, height, color) **–** рисует прямоугольник с началом в координатах x, y, высотой height, шириной width и цветом color;

fillRect(x, y, width, height, color) **–** рисует закрашенный прямоугольник с началом в координатах x, y, высотой height, шириной width и цветом color;

drawCircle(x, y, radius, color) **–** рисует круг с началом в координатах x, y, радиусом radius и цветом color;

fillCircle(x, y, radius, color) **–** рисует закрашенный круг с началом в координатах x, y, радиусом radius и цветом color;

drawTriangle(x, y, x1, y1, x2, y2, color) **–** рисует треугольник с углами в x, y; x1, y1; x2, y2 и цветом color;

fillTriangle(x, y, x1, y1, x2, y2, color) **–** рисует закрашенный треугольник с углами в x, y; x1, y1; x2, y2 и цветом color;

drawRoundRect(x, y, width, height, radius, color) **–** рисует прямоугольник с началом в координатах x, y, высотой height, шириной width, радиусом radius и цветом color;

fillRoundRect(x, y, width, height, radius, color) – рисует закрашенный прямоугольник с началом в координатах x, y, высотой height, шириной width, радиусом radius и цветом color;

setCursor(x, y) **–** устанавливает курсор в позицию x, y;

setTextColor(color) **–** устанавливает цвет текста;

setTextSize(size) **–** устанавливает размер текста;

write(symbol) **–** выводит на экран символ symbol.

drawBitmap – позволяет выводить двоичный массив закодированного изображения;

drawBitmap**(x, y, array, w, h, BLACK)**; // x, y, logo, w, h, color

начало в координатах x, y; array – массив изображения; w, h – ширина и высота изображения в пикселях; color – цвет.

Лабораторные задания

Задание №1. Необходимо посмотреть возможности библиотеки Adafruit_GFX , запустив тестовую программу. Схема подключения показана на рис. 1.9.2. Необходимо записать в платформу следующую программу.

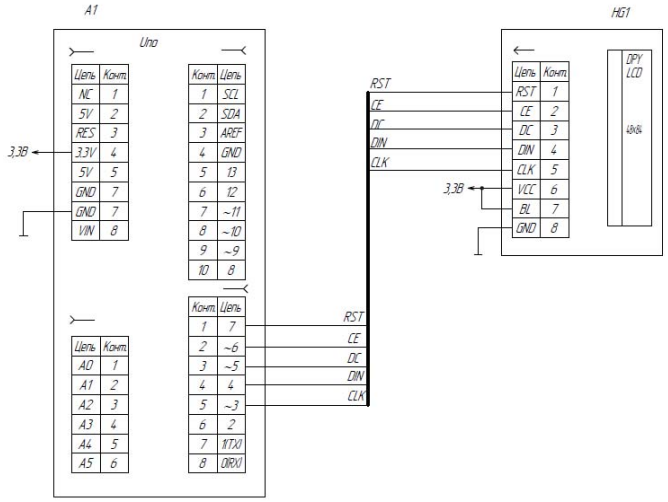


Рис. 1.9.2. Схема подключения дисплея Nokia 5110

```
// Тестировалось Adafruit_GFX
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>

// pin 3 - Serial clock out (SCLK)
// pin 4 - Serial data out (DIN)
// pin 5 - Data/Command select (D/C)
// pin 6 - LCD chip select (CS)
// pin 7 - LCD reset (RST)
Adafruit_PCD8544 display = Adafruit_PCD8544(3, 4, 5, 6, 7);
const static unsigned char PROGMEM logoBmp[] =
{
  B11111111, B11111111, B10000000,
  B11111111, B11111111, B10000000,
  B11111111, B11111111, B10000000,
  B11111100, B00000011, B10000000,
  B11111000, B00000001, B10000000,
  B11111100, B00000011, B10000000,
  B11111111, B11000011, B10000000,
  B11111111, B10000111, B10000000,
  B11111111, B10001111, B10000000,
  B11111111, B00001111, B10000000,
  B11111110, B00011111, B10000000,
  B11111110, B00011111, B10000000,
  B11111100, B00111111, B10000000,
  B11111100, B01111111, B10000000,
  B11111000, B00000011, B10000000,
```

```
  B11111000, B00000001, B10000000
};
void setup() {
  Serial.begin(9600);
  display.begin();             // Инициализация дисплея
  display.setContrast(60);      // Устанавливаем контраст
  display.setTextColor(BLACK);  // Устанавливаем цвет текста
  display.setTextSize(1);       // Устанавливаем размер текста
  display.clearDisplay();       // Очищаем дисплей
  display.display();
  delay(1000);
}

void loop() {
  // Рисуем заранее подготовленное лого
  // Подготовлен массив из 16 пар байтов
  // каждый байт состоит из 8 битов, соответсвенно
  // получаем матрицу 16x16 битов, 1-черный цвет, 0-белый цвет
  display.drawBitmap(LCDWIDTH/2-8, LCDHEIGHT/2-8, logoBmp, 24, 16,
BLACK); // x, y, logo, w, h, color
  display.display();
  delay(2000);

    // Очищаем дисплей
  display.clearDisplay();
  display.display();
  delay(1000);

  // Рисуем несколько пикселей (точек)
  display.drawPixel(0, 0, BLACK);
  display.drawPixel(1, 1, BLACK);
  display.drawPixel(2, 2, WHITE); // Посередине белый пиксель
  display.drawPixel(3, 3, BLACK);
  display.drawPixel(4, 4, BLACK);
  display.display();
  delay(2000);

  display.clearDisplay();
  display.display();
  delay(1000);

  // Рисуем диагональ
```

```
  display.drawLine(0, LCDHEIGHT-1, LCDWIDTH, 0, BLACK); // x0, y0, x1, y1,
color
  display.display();
  delay(2000);

  display.clearDisplay();
  display.display();
  delay(1000);

  // Для рисования вертикальных и горизонтальных линий лучше использовать
  // более быстрые функции
  display.drawFastVLine(LCDWIDTH/2, 0, LCDHEIGHT, BLACK); // x, y, h, color
  display.drawFastHLine(0, LCDHEIGHT/2, LCDWIDTH, BLACK); //x, y, w, color
  display.display();
  delay(2000);

  display.clearDisplay();
  display.display();
  delay(1000);

  // Рисуем прямоугольник
  display.drawRect(LCDWIDTH/4, LCDHEIGHT/4, LCDWIDTH/2,
LCDHEIGHT/2, BLACK); // x, y, w, h, color
  display.display();
  delay(2000);

  display.clearDisplay();
  display.display();
  delay(1000);

  // Рисуем закрашенный прямоугольник
  display.fillRect(LCDWIDTH/4, LCDHEIGHT/4, LCDWIDTH/2, LCDHEIGHT/2,
BLACK); // x, y, w, h, color
  display.display();
  delay(2000);

  display.clearDisplay();
  display.display();
  delay(1000);

  // Закрашиваем весь дисплей
  display.fillScreen(BLACK);
  display.display();
```

```
  delay(2000);

  display.clearDisplay();
  display.display();
  delay(1000);

  // Рисуем окружность
  display.drawCircle(LCDWIDTH/2, LCDHEIGHT/2, LCDHEIGHT/2, BLACK); //
x, y, r, color
  display.display();
  delay(2000);

  display.clearDisplay();
  display.display();
  delay(1000);

  // Рисуем закрашенную окружность
  display.fillCircle(LCDWIDTH/2, LCDHEIGHT/2, LCDHEIGHT/2, BLACK); // x,
y, r, color
  display.display();
  delay(2000);

  display.clearDisplay();
  display.display();
  delay(1000);

  // Рисуем треугольник
  display.drawTriangle(LCDWIDTH/4, LCDHEIGHT/4, 3*LCDWIDTH/4,
LCDHEIGHT/4, LCDWIDTH/2, 3*LCDHEIGHT/4, BLACK); // x0, y0, x1, y1, x2,
y2, color
  display.display();
  delay(2000);

  display.clearDisplay();
  display.display();
  delay(1000);

  // Рисуем закрашенный треугольник
  display.fillTriangle(LCDWIDTH/4, LCDHEIGHT/4, 3*LCDWIDTH/4,
LCDHEIGHT/4, LCDWIDTH/2, 3*LCDHEIGHT/4, BLACK); // x0, y0, x1, y1, x2,
y2, color
  display.display();
  delay(2000);
```

```
  display.clearDisplay();
  display.display();
  delay(1000);

  // Рисуем прямоугольник с закругленными углами
  display.drawRoundRect(LCDWIDTH/4, LCDHEIGHT/4, LCDWIDTH/2,
LCDHEIGHT/2, 10, BLACK); // x, y, w, h, r, color
  display.display();
  delay(2000);

  display.clearDisplay();
  display.display();
  delay(1000);

  // Рисуем закрашенный прямоугольник с закругленными углами
  display.fillRoundRect(LCDWIDTH/4, LCDHEIGHT/4, LCDWIDTH/2,
LCDHEIGHT/2, 10, BLACK); // x, y, w, h, r, colordisplay.display();
  display.display();
  delay(2000);

  display.clearDisplay();
  display.display();
  delay(1000);

  // Рисуем заранее подготовленное лого
  // Подготовлен массив из 16 пар байтов
  // каждый байт состоит из 8 битов, соответсвенно
  // получаем матрицу 16x16 битов, 1-черный цвет, 0-белый цвет
  display.drawBitmap(LCDWIDTH/2-8, LCDHEIGHT/2-8, logoBmp, 24, 16,
BLACK); // x, y, logo, w, h, color
  display.display();
  delay(2000);

  display.clearDisplay();
  display.display();
  delay(1000);

  // Выведем текст
  display.print("Zelectro");
  delay(3000);

  display.clearDisplay();
```

```
  display.display();
  delay(5000);
}
```

Необходимо вывести картинку на LCD индикатор. Схема подключения прежняя. Программа имеет следующий вид.

```
// Вывод картинки
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>

Adafruit_PCD8544 display = Adafruit_PCD8544(3, 4, 5, 6, 7);

const unsigned char PROGMEM logoBmp[] ={
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11000000, B00011111, B10000000, B00000111,
B11111100, B00001111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11110011, B11001111, B10111100, B11110111,
B11111110, B00011111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11110011, B11100111, B11111100, B11111111,
B11111111, B00111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11110011, B11100111, B11111100, B11111111,
B11110000, B00001111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11100111, B11100111, B11111001, B11111111,
B10011110, B01111011, B11111111, B11111111, B11110000,
B11111111, B11111111, B11100111, B11000111, B11111001, B11111111,
B00111110, B01111001, B11111111, B11111111, B11110000,
B11111111, B11111111, B11100111, B11001111, B11111001, B11111110,
B01111100, B01111001, B11111111, B11111111, B11110000,
```

B11111111, B11111111, B11000111, B10011111, B11110011, B11111110,
B01111100, B11111001, B11111111, B11111111, B11110000,
B11111111, B11111111, B11000000, B00111111, B11110011, B11111100,
B11111100, B11110011, B11111111, B11111111, B11110000,
B11111111, B11111111, B11001111, B11111111, B11110011, B11111100,
B11111001, B11110011, B11111111, B11111111, B11110000,
B11111111, B11111111, B11001111, B11111111, B11100111, B11111110,
B11111001, B11100111, B11111111, B11111111, B11110000,
B11111111, B11111111, B10011111, B11111111, B11100111, B11111110,
B01110001, B11001111, B11111111, B11111111, B11110000,
B11111111, B11111111, B10011111, B11111111, B11100111, B11111111,
B10000000, B00111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B10011111, B11111111, B11000111, B11111111,
B11110011, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B00011111, B11111111, B11000111, B11111111,
B11100011, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111100, B00001111, B11111111, B00000011, B11111111,
B10000000, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111000, B00011111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11100000, B00011111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B10000000, B00110001,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B00000000, B11100000,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B00000001, B10000000,
B01111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111110, B00010111, B00000000,
B01111111, B11111111, B11111111, B11111111, B11110000,

```
B11111111, B11111111, B11111111, B11111110, B00011110, B00000000,
B01111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111100, B00010000, B00000000,
B00111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111100, B00010000, B00000000,
B00111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11110000, B00000000,
B00111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111100, B00010000, B00000000,
B00111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111100, B00011110, B00000000,
B00111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111110, B00010011, B00000000,
B01111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111110, B00010001, B11000100,
B01111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B00000000, B01110100,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B10000000, B00011101,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11100000, B01111111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111000, B00001111,
B11111111, B11111111, B11111111, B11111111, B11110000,
B11111111, B11111111, B11111111, B11111111, B11111111, B11111111,
B11111111, B11111111, B11111111, B11111111, B11110000
};

void setup() {
 Serial.begin(9600);
 // Инициализируем наш дисплей
 display.begin();
 // Делаем его пустым
 display.clearDisplay();
 display.display();
 // Устанавливаем контраст
 display.setContrast(60);
}

void loop()
{
 display.drawBitmap(0, 0, logoBmp, 84, 48, BLACK); // x, y, logo, w, h, color
 display.display();
```

```
}
```

Необходимо вывести текст на кириллице на LCD индикатор. Схема подключения прежняя. Программа имеет следующий вид.

```
// Вывод текста
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
Adafruit_PCD8544 display = Adafruit_PCD8544(3, 4, 5, 6, 7);
void setup() {
   // инициализация и очистка дисплея
   display.begin();
   display.clearDisplay();
   display.display();

   display.setContrast(50); // установка контраста
   delay(1000);
   display.setTextSize(2);  // установка размера шрифта
   display.setTextColor(BLACK); // установка цвета текста
   display.setCursor(0,0); // установка позиции курсора

   display.println(utf8rus("Привет мир"));
   display.display();
}
void loop() {
}
String utf8rus(String source) // подпрограмма перевода кодировки на русский
шрифт
{
  int i,k;
  String target;
  unsigned char n;
  char m[2] = { '0', '\0' };
  k = source.length(); i = 0;
  while (i < k) {
    n = source[i]; i++;
    if (n >= 0xC0) {
      switch (n) {
        case 0xD0: {
          n = source[i]; i++;
          if (n == 0x81) { n = 0xA8; break; }
          if (n >= 0x90 && n <= 0xBF) n = n + 0x30;
          break;
```

```
      }
      case 0xD1: {
        n = source[i]; i++;
        if (n == 0x91) { n = 0xB8; break; }
        if (n >= 0x80 && n <= 0x8F) n = n + 0x70;
        break;
      }
    }
   }
   m[0] = n; target = target + String(m);
  }
return target;
}
```

Необходимо вывести значение освещенности на LCD индикатор. Значение освещенности берутся с модуля фоторезистора.

```
// Вывод значения уровня освещенности
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
// PIN 7 - RST Pin 1 on LCD
// PIN 6 - CE Pin 2 on LCD
// PIN 5 - DC Pin 3 on LCD
// PIN 4 - DIN Pin 4 on LCD
// PIN 3 - CLK Pin 5 on LCD
Adafruit_PCD8544 display = Adafruit_PCD8544(3, 4, 5, 6, 7);
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 val1,val2 = 0;
void setup()
{
display.begin();
// установить контраст фона экрана
// очень важный параметр!
display.setContrast(50);
display.clearDisplay(); // очистить экран
delay(2000);
}
void loop()
{
val1 = analogRead(LIGHT); // Чтение показаний фоторезистора
```

```
drawText(val1,1); // вывести текст
// масштабирование значения потенциометра к 0–75
val2= map(val1, MIN_LIGHT, MAX_LIGHT, 0, 75);
// вывод черного прямоугольника в %
display.fillRect(5, 25, val2, 10, BLACK);
// вывод белой части прямоугольника
display.fillRect(5+val2,25, 75-val2, 10, WHITE);
display.display();
delay(100); // пауза перед новым измерением
drawText(val1,2); // стереть текст
}
// процедура вывода текста
void drawText(unsigned long num,int color)
{
display.setTextSize(2); // размер шрифта
display.setCursor(20,5); // позиция курсора
if(color==1)
display.setTextColor(BLACK); // вывести значение
else
display.setTextColor(WHITE); // стереть (белым по белому)
display.print(num);
}
```

Задания для самостоятельного решения

1. Необходимо вывести несколько картинок, организовав слайд-шоу.
2. С помощью графических средств создать линейную шкалу со стрелкой в виде черты. Стрелка должна двигаться по шкале при изменении положения потенциометра.

Контрольные вопросы

1. Каковы технические характеристики LCD 5110?
2. Каким образом подключается LCD 5110 к платформе?
3. Какие основные команды библиотеки Adafruit_GFX?
4. Каким образом кодируется картинка в LCD 5110?
5. Как использовать русские символы на LCD 5110?

Лабораторная работа №10
Радиопередатчик RF 315/433 МГц. Подключение радиопередатчика к микроконтроллеру, вывод информации на индикатор

Цель работы: познакомиться с радиопередатчиком RF 315/433 МГц., научиться программировать и передавать информацию по радиоканалу.

Оборудование: персональный компьютер, среда разработки Arduino IDE, платформа Arduino UNO, макетная плата, провода, передатчик FS100A, приемник MX-RM-5V, потенциометр.

Теоретические сведения

На этом занятии необходимо передать радиосигнал между двумя контроллерами с помощью приемопередатчика с частотой 433МГц. Устройство по передаче данных состоит из двух модулей: приемника и передатчика. Данные можно передавать только в одном направлении. Это важно понимать при использовании этих модулей. Например, можно сделать дистанционное управление любым электронным устройством или предавать данные. В этом случае информация будут передаваться от пульта управления к устройству. Другой вариант – передача сигналов с беспроводных датчиков на систему сбора данных. Здесь уже маршрут меняется, теперь передатчик стоит на стороне датчика, а приемник на стороне системы сбора.

Модули могут иметь разные названия: MX-05V, XD-RF-5V, XY-FST, XY-MK-5V, и т.п., но все они имеют примерно одинаковый внешний вид и нумерацию контактов. Также, распространены две частоты радиомодулей: 433 МГц и 315 МГц.

В качестве передатчика используется FS100A, в качестве приемника MX-RM-5V.

1. Подключение

Передатчик имеет всего три вывода: Gnd, Vcc и Data. Внешний вид передатчика представлен на рис. 1.10.1 [15].

Рис. 1.10.1. Внешний вид и назначения выводов передатчика FS1000A [15]

Характеристики FS1000A:
- Рабочее напряжение: DC 5В;
- Рабочий ток: 4 мА;
- Режим модуляции: OOK (AM);
- Рабочая температура: -10 ° C ~ +70 ° C;
- Получать чувствительность:-110дБ;
- Рабочая частота: 433МГц;
- Размер: 30 x 14 x 7 мм.

Вывод Data у передатчика может подключаться к любому цифровому выводу. Питание подается на уровне 5 В.

Характеристики MX-RM-5V:

- Рабочее напряжение: 3В ~ 12 В;
- Рабочий ток: 20 мА ~ 28мА;
- Резервное течение: 0мА;
- Рабочая частота: 433МГц;
- Расстояние передатчика: >500 м (чувствительность может быть выше-103 дБм на открытой местности);
- Выходная мощность: 16 дБм (40 мВт);
- Скорость Передатчик: Режим модуляции: OOK (AM);
- Рабочая температура: -10 ° C ~ +70 ° C;
- Размер: 19 x 19 x 8 мм.

Внешний вид приемника MX-RM-5V представлен на рис. 1.10.1 [15].

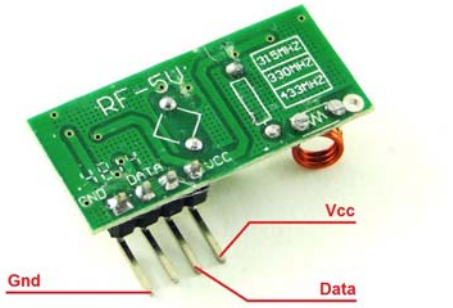


Рис. 1.10.2. Внешний вид и назначения выводов приемника MX-RM-5V [15]

У приемника надо подавать питание тоже на уровне 5 В. Вывод Data надо подключать к выводам 2 или 3 Arduino UNO, так как библиотека iarduino_RF433 использует прерывания.

Приёмник MX-RM-5V критичен даже к небольшим пульсациям на шине питания. Если Arduino управляет устройствами вносящими даже небольшие, но постоянные, пульсации в шину питания (сервоприводы, LED индикаторы, ШИМ и т.д.), то приёмник расценивает эти пульсации как сигнал и не реагирует на радиоволны передатчика. Влияние пульсаций на приёмник можно снизить одним из способов:

- использовать, для питания Arduino, внешний источник, а не шину USB. Так как напряжение на выходе многих внешних источников питания контролируется или сглаживается. В отличии от шины USB, где напряжение может существенно «проседать»;
- установить на шине питания приёмника сглаживающий конденсатор;
- использовать отдельное стабилизированное питание для приёмника;
- использовать отдельное питание для устройств вносящих пульсации в шину питания.

Рассмотрим команды библиотеки iarduino_RF433.

Функции для работы с передатчиком.

Функция ***begin();***

Назначение: Инициализация передатчика.

Синтаксис: begin([СКОРОСТЬ]);

Параметры:

СКОРОСТЬ – не обязательный параметр, целое число, указывающее скорость передачи данных (бит/сек).

Если скорость указана и её не требуется менять, то можно не вызывать функцию setDataRate().

Возвращаемые значения: Нет.

Примечание:

Функция вызывается 1 раз в коде setup.

Скорость передачи данных должна совпадать, как для передатчика так и для приёмника.

Функция ***setDataRate();***

Назначение: Указание или изменение скорости передачи данных.

Синтаксис: setDataRate(СКОРОСТЬ);

Параметры:

СКОРОСТЬ – целое число, указывающее скорость передачи данных (бит/сек).

Возвращаемые значения: Нет.

Примечание:

Если скорость указана при вызове функции begin() и её не требуется менять, то данную функцию можно не вызывать.

Скорость передачи данных должна совпадать, как для передатчика так и для приёмника.

Скорость можно указать как целым числом, так и константами:

i433_5KBPS
i433_4KBPS
i433_3KBPS
i433_2KBPS
i433_1KBPS
i433_500BPS
i433_100BPS

Скорость можно менять во время работы передатчика.

Функция ***openWritingPipe();***

Назначение: Открытие трубы для передачи данных

Синтаксис: openWritingPipe(НОМЕР_ТРУБЫ);

Параметры:

НОМЕР_ТРУБЫ – целое число, от 0 до 7.

Возвращаемые значения: Нет.

Примечание:

Номер трубы это адрес передатчика, по которому приёмник «решит» получать или нет отправленные данные.

Передатчик может передавать данные только по одной из труб (от 0 до 7)

Номер трубы можно менять во время работы передатчика.

Функция write();

Назначение: Передача данных.

Синтаксис: write(ССЫЛКА_НА_МАССИВ , КОЛИЧЕСТВО_БАЙТ);

Параметры:

ССЫЛКА_НА_МАССИВ – ссылка на массив любого типа (в том числе и на строку), данные которого требуется передать.

КОЛИЧЕСТВО_БАЙТ – число определяющее, какое количество байт требуется передать.

Возвращаемые значения: Нет.

Примечание: Если Вы используете несколько передатчиков, то используйте интервалы между передачей данных.

Функции для работы с приёмником:

Функция ***begin();***

Назначение: Инициализация приёмника

Синтаксис: begin([СКОРОСТЬ]);

Параметры:

СКОРОСТЬ – не обязательный параметр, целое число, указывающее скорость приёма данных (бит/сек).

Если скорость указана и её не требуется менять, то можно не вызывать функцию setDataRate().

Возвращаемые значения: Нет.

Примечание:

Функция вызывается 1 раз в коде setup.

Скорость приёма данных должна совпадать, как для передатчика так и для приёмника.

Функция ***setDataRate();***

Назначение: Указание или изменение скорости приёма данных.

Синтаксис: setDataRate(СКОРОСТЬ);

Параметры:

СКОРОСТЬ – целое число, указывающее скорость приёма данных (бит/сек).

Возвращаемые значения: Нет.

Примечание:

Если скорость указана при вызове функции begin() и её не требуется менять, то данную функцию можно не вызывать.

Скорость приёма данных должна совпадать, как для передатчика так и для приёмника.

Скорость можно указать как целым числом, так и константами:

i433_5KBPS
i433_4KBPS
i433_3KBPS
i433_2KBPS
i433_1KBPS
i433_500BPS
i433_100BPS

Скорость можно менять во время работы приёмника, но если во время изменения скорости принимался пакет данных, он будет пропущен.

Функция ***openReadingPipe();***

Назначение: Открытие трубы для приёма данных

Синтаксис: openReadingPipe([НОМЕР_ТРУБЫ]);

Параметры:

НОМЕР_ТРУБЫ - необязательный параметр, целое число, от 0 до 7.

Возвращаемые значения: Нет.

Примечание:

Номер трубы это адрес передатчика (от 0 до 7), открывая трубу мы соглашаемся принимать данные от указанного передатчика.

Функцию можно вызвать несколько раз, открывая несколько труб для приёма данных.

Если вызвать функцию без параметра, то будут открыты все трубы от 0 до 7.

Функция ***closeReadingPipe();***

Назначение: Закрытие трубы от приёма данных

Синтаксис: closeReadingPipe([НОМЕР_ТРУБЫ]);

Параметры:

НОМЕР_ТРУБЫ - необязательный параметр, целое число, от 0 до 7.

Возвращаемые значения: Нет.

Примечание:

Действие противоположное функции openReadingPipe().

Функцию можно вызвать несколько раз, закрывая несколько труб от приёма данных.

Если вызвать функцию без параметра, то будут закрыты все трубы от 0 до 7.

Функция ***startListening();***

Назначение: Включение прослушивания открытых труб.

Синтаксис: startListening();

Параметры: Нет.

Возвращаемые значения: Нет.

Примечание: Открывать и закрывать трубы, можно и после начала их прослушивания.

Функция ***stopListening();***

Назначение: Выключение прослушивания открытых труб.

Синтаксис: stopListening();

Параметры: Нет.

Возвращаемые значения: Нет.

Примечание: Данная функция отключает обработку внешнего прерывания, что может ускорить выполнение скетча.

Функция ***available();***

Назначение: Проверка наличия принятых данных, доступных для чтения функцией read().

Синтаксис: available([ССЫЛКА_НА_ПЕРЕМЕННУЮ]);

Параметры:

ССЫЛКА_НА_ПЕРЕМЕННУЮ – необязательный параметр, ссылка на переменную типа uint8_t, в которую будет записан номер трубы, если данные приняты и доступны для чтения.

Возвращаемые значения: true / false – имеются данные доступные для чтения или нет.

Примечание: По номеру трубы можно определить, данные какого передатчика были приняты.

Функция ***read();***

Назначение: Передача данных.

Синтаксис: read(ССЫЛКА_НА_МАССИВ , КОЛИЧЕСТВО_БАЙТ);

Параметры:

ССЫЛКА_НА_МАССИВ – ссылка на массив любого типа (в том числе и на строку), в который требуется прочитать данные.

КОЛИЧЕСТВО_БАЙТ – число определяющее, какое количество байт требуется прочитать.

Возвращаемые значения: Нет.

Примечание:

Если указанное количество байт больше размера массива, на который ссылается ссылка, то данные запишутся за пределы массива, что может вызвать некорректную работу скетча.

Если указанное количество байт больше чем реально принято, то в массив запишутся только реально принятые байты, а оставшиеся байты массива останутся без изменений.

Лабораторные задания

Необходимо обеспечить беспроводную передачу данных, поступающих с переменного резистора.

Схема подключения передатчика FS1000A представлена на рис. 1.10.3. Программа имеет следующий вид.

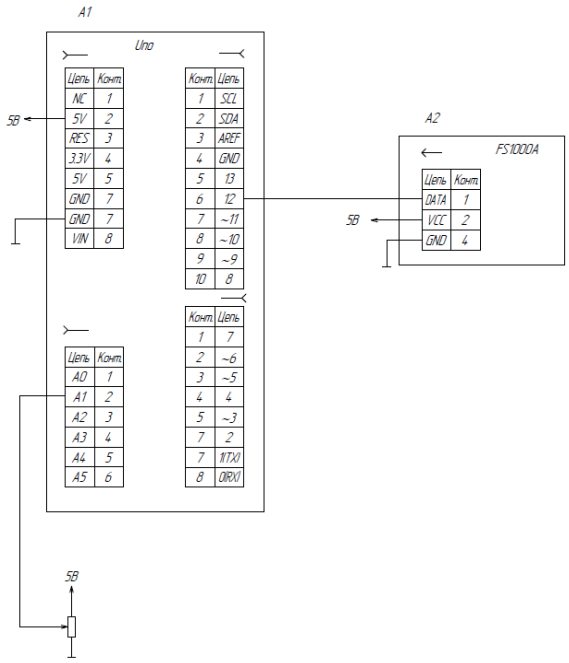


Рис. 1.10.3. Схема подключения передатчика FS1000A

```
#include <iarduino_RF433_Transmitter.h> // Подключаем библиотеку для работы с передатчиком FS1000A
iarduino_RF433_Transmitter radio(12); // Создаём объект radio для работы с библиотекой iarduino_RF433, указывая номер вывода к которому подключён передатчик
int data[1]; // Создаём массив для передачи данных
void setup(){
   radio.begin();// Инициируем работу передатчика FS1000A (в качестве параметра можно указать скорость ЧИСЛО бит/сек, тогда можно не вызывать функцию setDataRate)
   radio.setDataRate     (i433_1KBPS); // Указываем скорость передачи данных
radio.openWritingPipe (5); // Открываем 5 трубу для передачи данных
} // Если повторно вызвать функцию openWritingPipe указав другой номер трубы, то передатчик начнёт передавать данные по вновь указанной трубе
void loop(){
   data[0] = analogRead(A1); // считываем показания потенциометра с вывода A1 и записываем их в 0 элемент массива data
   radio.write(&data, sizeof(data)); // отправляем данные из массива data указывая сколько байт массива мы хотим отправить
   delay(10); // пауза между пакетами
```

Схема подключения приемника MX-RM-5V представлена на рис. 1.10.4. Программа имеет следующий вид.

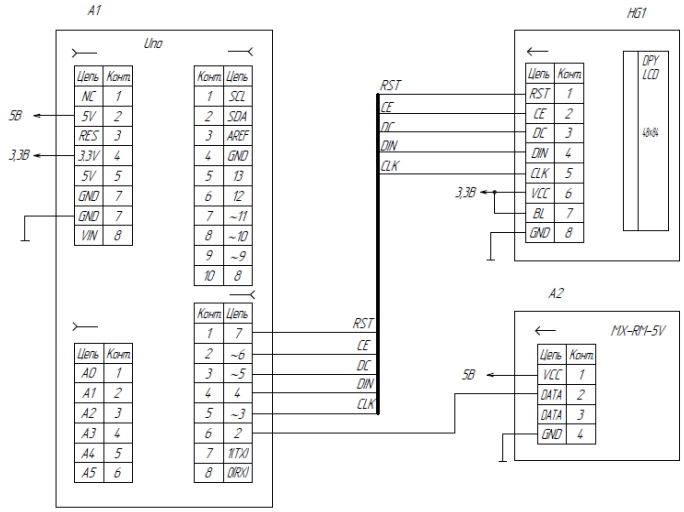


Рис. 1.10.4. Схема подключения приемника MX-RM-5V

// Вывод значения уровня освещенности

#include <Adafruit_GFX.h> // Подключаем библиотеку для работы с LCD Adafruit_GFX

#include <Adafruit_PCD8544.h>

#include <iarduino_RF433_Receiver.h> // Подключаем библиотеку для работы с приёмником MX-RM-5V

// PIN 7 - RST Pin 1 on LCD // Выводы подключения LCD

// PIN 6 - CE Pin 2 on LCD

// PIN 5 - DC Pin 3 on LCD

// PIN 4 - DIN Pin 4 on LCD

// PIN 3 - CLK Pin 5 on LCD

Adafruit_PCD8544 display = Adafruit_PCD8544(3, 4, 5, 6, 7);

// Переменная для хранения данных с передатчика

int val1,val2 = 0;

iarduino_RF433_Receiver radio(2); // Создаём объект radio для работы с библиотекой iarduino_RF433, указывая номер вывода к которому подключён приёмник (можно подключать только к выводам использующим внешние прерывания)

int data[1]; // Создаём массив для приёма данных

void setup()

{

radio.begin();// Инициируем работу приёмника MX-RM-5V (в качестве параметра можно указать скорость ЧИСЛО бит/сек, тогда можно не вызывать функцию setDataRate)

radio.setDataRate (i433_1KBPS); // Указываем скорость приёма данных (i433_5KBPS, i433_4KBPS, i433_3KBPS, i433_2KBPS, i433_1KBPS, i433_500BPS, i433_100BPS), i433_1KBPS - 1кбит/сек

```
radio.openReadingPipe (5); // Открываем 5 трубу для приема данных (если вызвать функцию без параметра, то будут открыты все трубы сразу, от 0 до 7)
radio.startListening  ();// Включаем приемник, начинаем прослушивать открытую трубу
display.begin();
// установить контраст фона экрана
// очень важный параметр!
display.setContrast(50);
display.clearDisplay(); // очистить экран
delay(2000);
}
void loop()
{
if(radio.available()){ // Если в буфере имеются принятые данные
radio.read(&data, sizeof(data)); // Читаем данные в массив data и указываем сколько байт читать
val1 = data[0]; // Чтение данных передатчика
}
drawText(val1,1); // вывести текст
// масштабирование значения потенциометра к 0–75
val2= map(val1, 0, 1024, 0, 75);
// вывод черного прямоугольника в %
display.fillRect(5, 25, val2, 10, BLACK);
// вывод белой части прямоугольника
display.fillRect(5+val2,25, 75-val2, 10, WHITE);
display.display();
delay(10); // пауза перед новым измерением
drawText(val1,2); // стереть текст
}
// процедура вывода текста
void drawText(unsigned long num,int color)
{
display.setTextSize(2); // размер шрифта
display.setCursor(20,5); // позиция курсора
if(color==1)
display.setTextColor(BLACK); // вывести значение
else
display.setTextColor(WHITE); // стереть (белым по белому)
display.print(num);
}
```

Задания для самостоятельного решения

1. Необходимо организовать беспроводную передачу данных с фоторезистора.

2. Необходимо организовать опрос 4 кнопок с передачей данных на приемник. На приемной стороне обеспечить включение светодиодов.

Контрольные вопросы

1. Каковы технические характеристики передатчика FS100A?
2. Каковы технические характеристики приемника MX-RM-5V?
3. Какие основные команды библиотеки iarduino_RF433?
4. Каким образом обеспечить работу одновременно нескольких комплектов приемопередатчиков?
5. Как увеличить дальность радиосвязи приемопередатчика?

Лабораторная работа №11
Датчик уровня жидкости

Цель работы: познакомиться с устройством датчика уровня жидкости резистивного типа, составить, компилировать и загрузить в микроконтроллер программы на Arduino.

Оборудование: персональный компьютер, среда разработки Arduino IDE, платформа Arduino UNO, макетная плата, провода, светодиодная линейка, датчик уровня жидкости, транзисторный модуль, светодиод, емкость для налива воды.

Теоретические сведения

Рассмотрим на данном занятии аналоговый датчик протечки воды (уровня жидкости Water Sensor). Внешний вид датчика представлен на рисунке. Он представляет собой пластину из стеклотекстолита с установленными электрорадиоэлементами и длинные полоски проводников, вытравленные на ней (рис. 1.11.1) [16].

Рис. 1.11.1. Внешний вид датчика уровня воды [16]

Схема электрическая сенсора представлена на рис. 1.11.2. Он работает следующим образом. На базу транзистора, через резистор, сопротивлением 100

Ом и сенсор, представляющий собой набор параллельно расположенных проводников подается напряжение питания +5В. Аналоговый сигнал снимается с эмиттера транзистора, а на коллектор подается +5В. Если датчик сухой, то сопротивление сенсора велико и транзистор закрыт, следовательно на выходе «S» будет напряжение близкое к 0В. По мере увеличения уровня воды сопротивление сенсора будет уменьшаться, что приведет к постепенному открыванию транзистора вплоть до полного. На выходе «S» будет напряжение, пропорциональное уровню воды. Для сигнализации подачи питания на датчик применяется светодиод с токоограничивающим резистором, сопротивлением 1 кОм.

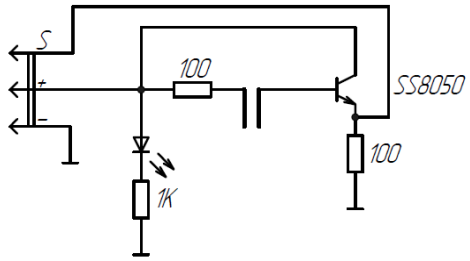


Рис. 1.11.2. Схема датчика уровня воды

Подключение к Ардуино датчика воды (Water Sensor).

Согласно схемы, датчик уровня жидкости имеет три контакта. Правый контакт (-) подключается к Земле (GND), средний к питанию 5В, а левый к аналоговому входу, например, A0. При полностью сухом датчике выходное напряжение и показания на аналоговом входе будут равны нулю, чем больше датчик будет погружен в воду, тем больше будут его показания (от 0 до 1023).

Лабораторные задания

Необходимо подключить датчик уровня воды к платформе UNO, Насос подключить через кнопку напрямую к блоку питания. **На блоке питания надо выставить напряжение 5В. Напряжение нельзя превышать, в противном случае насос выйдет из строя!** При нажатии на кнопку насос включается, и вода перекачивается в емкость с установленным датчиком воды.

Датчик подключается к аналоговому входу A0. С помощью представленной ниже программы выводятся данные с датчика в последовательный порт.

```
// Считывание данных с сенсора воды
int val;
void setup(){
  pinMode(A0, INPUT);
  Serial.begin(9600);
}
void loop(){
```

```
  val = analogRead(A0);
  Serial.println(val);
  delay(500);
}
```

Задания для самостоятельного решения

1. Необходимо смоделировать включение сигнализации при наличии воды (сенсор протечек). Сигнализацию протечки выводить с помощью светодиода.

2. Необходимо организовать процесс определения порогов уровня воды (4 порога) с выводом сигнализации на светодиоды.

3. Необходимо сделать автомат, включающий подкачку воды при понижении уровня. Насос запитать через транзистор.

Контрольные вопросы

1. Опишите принцип работы датчика уровня жидкости.
2. Какие достоинства и недостатки резистивных датчиков уровня жидкости?
3. Каким образом можно увеличить чувствительность датчика?
4. Каким способом еще можно измерять уровень жидкости?

Лабораторная работа №12
Измерение температуры с помощью термопары

Цель работы: познакомиться с принципом действия термопары и модулем для преобразования сигнала с термопары MAX 6675, составить, компилировать и загрузить в микроконтроллер программы на Arduino.

Оборудование: персональный компьютер, среда разработки Arduino IDE, платформа Arduino UNO, макетная плата, провода, модуль MAX 6675, ЖК индикатор 5110, зажигалка, реле.

Теоретические сведения

Для измерения высоких температур часто используют термопары. Работа термопары основана на термоэлектрическом эффекте, когда на спайке разнородных металлов образуется ЭДС, которая прямо пропорциональна температуре окружающей среды. Эту ЭДС возможно измерить, но она настолько маленькая (всего 2 мВ при разнице температур в 100 °С), что так просто ее на вход микроконтроллера не подашь. На помощь приходит ИС от фирмы Maxim - MAX6675. Она измеряет ЭДС термопары, усиливает сигнал, компенсирует температуру холодных концов, преводит в цифровой код (12 разрядов) и через

SPI интерфейс выдает в виде готового числа. Эта микросхема рассчитана на подключение термопары К-типа (Хомель-Алюмель).

Рассмотрим документацию на микросхему MAX6675 [17]. Напряжение питания может быть 3 или 5 вольта. Из интересного, заявлена защита от электростатики (ESD Protection) на два киловольта, т.е. щуп вполне можно трогать руками не опасаясь повредить микросхему. Структурная схема представлена на данном рис. 1.12.1 [17].

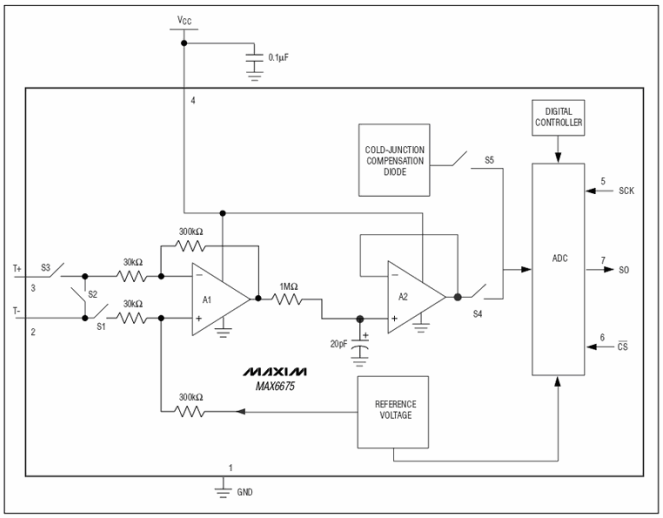


Рис. 1.12.1. Структурная схема микросхемы MAX6675 [17]

Устройство микросхемы. Видно, что входная ЭДС поступает на предварительный операционный усилитель, затем на основной, после чего сигнал идет на АЦП. Формат, передачи данных представлен на рис. 1.12.2 [17].

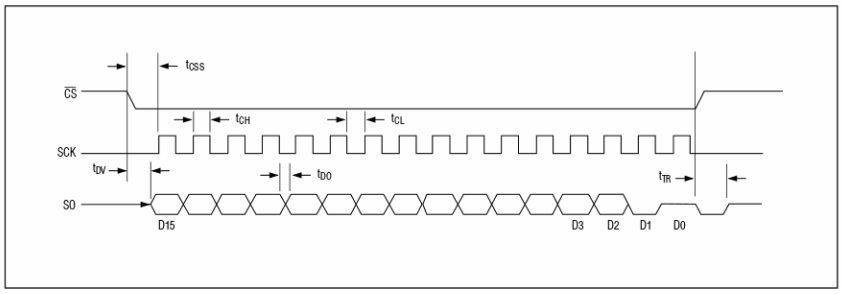


Figure 1b. Serial Interface Timing

BIT	DUMMY SIGN BIT	12-BIT TEMPERATURE READING												THERMOCOUPLE INPUT	DEVICE ID	STATE
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	MSB											LSB		0	Three-state

Figure 2. SO Output

Рис. 1.12.2. Формат, передачи данных микросхемы MAX6675 [17]

Видно что данные передаются пакетом по 16 бит. SPI интерфейс работает в одну сторону, т.о. получается, что задействовано три цифровые линии: SCK для тактирования, SC для управления, SO на передачу.

Для Arduino есть готовая библиотека для работы с микросхемой MAX6675. Создание объекта ведется следующей командой.

MAX6675 thermocouple (pinSCK, pinCS, pinSO);

У нее всего 2 команды:

thermocouple.readCelsius()

и

thermocouple.readFahrenheit()

Соответственно выдает значение в градусах Цельсия и Фаренгейта.

Лабораторные задания

Задание. Необходимо подключить термопару с модулем MAX6675 и записать следующую программу, выводящую значение температуры в последовательный порт.

```
// Измерение температуры термопарой
#include "max6675.h"
int thermoSO = 4;
int thermoCS = 5;
int thermoSCK = 6;
MAX6675 thermocouple(thermoSCK, thermoCS, thermoSO);
void setup() {
  Serial.begin(9600);
  Serial.println("MAX6675 test");
 }
void loop() {
  //Выводим показания в монитор порта
  Serial.print("C = ");
  Serial.print(thermocouple.readCelsius());
  Serial.print(";   F = ");
  Serial.println(thermocouple.readFahrenheit());
  delay(300);
}
```

Задания для самостоятельного решения

1. Необходимо вывести значение температуры на ЖК графический индикатор Nokia 5110.

2. Необходимо смоделировать систему, регулирующую температуру в печи, с помощью реле.

3. Необходимо получить данные с термопары без использования библиотеки max6675.h по SPI интерфейсу.

Контрольные вопросы

1. Каковы достоинства и недостатки термопар.
2. Каков принцип действия микросхемы MAX 6675?
3. Каким образом опрашивается микросхема MAX 6675?
4. С какой целью компенсируется температура холодного конца термопары?

Лабораторная работа №13
Пироэлектрический датчик HC-SR501

Цель работы: познакомиться с принципом работы пироэлектрического датчика, составить, компилировать и загрузить в микроконтроллер программы на Arduino.

Оборудование: персональный компьютер, среда разработки Arduino IDE, платформа Arduino UNO, макетная плата, провода, светодиоды, модуль HC-SR501, ЖК индикатор 5110, Н мост, электродвигатель.

Теоретические сведения

Тема сегодняшнего занятия — датчик движения на основе пироэлектрического эффекта (PIR, passive infrared motion sensor). Такие датчики часто используются в охранных системах и в быту для обнаружения движения в помещении. Например, на принципе детектирования движения основано автоматическое включение света в подъезде или в ванной. Пироэлектрические датчики достаточно просто устроены, недороги и неприхотливы в установке и обслуживании. Существуют также и другие способы детектирования движения.

Пироэлектрики — это диэлектрики, которые создают электрическое поле при изменении их температуры. На основе пироэлектриков делают датчики измерения температуры, например, LHI778 или IRA-E700. Каждый такой датчик содержит два чувствительных элемента размером 1×2 мм, подключенных с противоположной полярностью. И как мы увидим далее, наличие именно двух элементов поможет нам детектировать движение.

Внешний вид датчика IRA-E700 компании Murata представлен на рис. 1.13.1 [15].

Рис. 1.13.1. Внешний вид пиродатчика IRA-E700 [15]

На этом занятии рассматривается датчик движения HC-SR501, в котором установлен один такой пироэлектрический датчик. Сверху пироэлектрик окружен полусферой, разбитой на несколько сегментов. Каждый сегмент этой сферы представляет собой линзу, которая фокусирует тепловое излучение на разные участки ПИР-датчика. Часто в качестве линзы используют линзу Френеля. Линзы Френеля концентрируют излучение, значительно расширяя диапазон чувствительности пиродатчиков (рис. 1.13.2) [15].

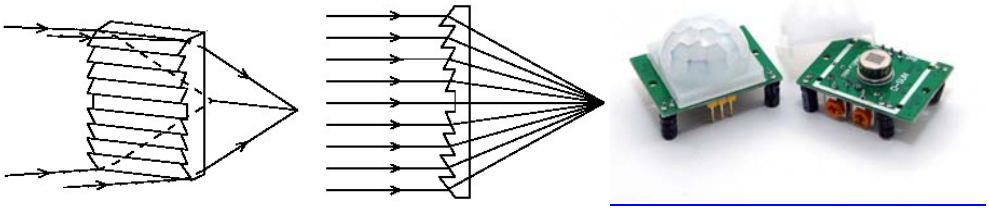


Рис. 1.13.2. Схема работы линзы Френеля и ее
внешний вид на модуле HC-SR501[15]

Принцип работы датчика движения следующий. Предположим, что датчик установлен в пустой комнате. Каждый чувствительный элемент получает постоянную дозу излучения, а значит и напряжение на них имеет постоянное значение (левая позиция на рис. 1.13.3) [15].

Как только в комнату заходит человек, он попадает сначала в зону обзора первого элемента, что приводит к появлению положительного электрического импульса на нем (центральная позиция на рис. 1.13.3).

Человек движется, и его тепловое излучение через линзы попадает уже на второй PIR-элемент, который генерирует отрицательный импульс. Электронная схема датчика движения регистрирует эти разнонаправленные импульсы и делает выводы о том, что в поле зрения датчика попал человек. На выходе датчика генерируется положительный импульс (правая позиция на рис. 1.13.3).

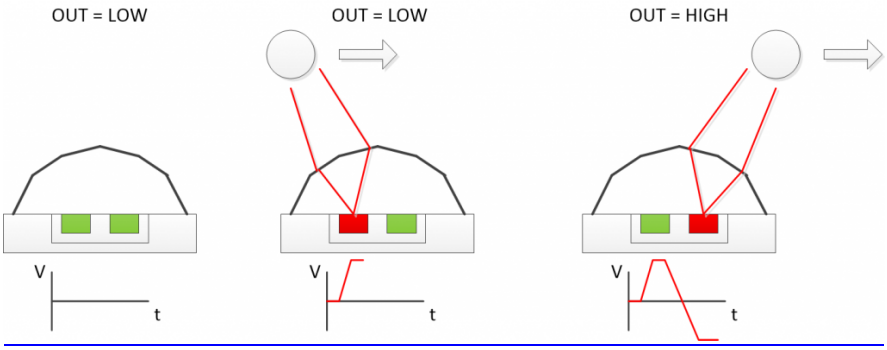


Рис. 1.13.3. Схема работы пиродатчика [15]

Характеристики пиродатчика:

- питание – постоянное напряжение 4,5-20 В;
- в режиме ожидания ток потребления менее 50 мкА;
- наибольший потребляемый ток во время работы 65 мА;
- напряжение логических уровней соответствует требованиям логики с питанием 3,3 В;
- расстояние обнаружения 3-7 м, по умолчанию 7 м;
- максимальный угол обнаружения 110° , на расстоянии 7 м 120°;
- время поддержания высокого уровня выхода при присутствии 20-300 с;
- время игнорирования событий после фиксации 0,2 с;
- температура окружающего воздуха при работе –15…70 °С;
- размеры 32 X 24 X 28 мм

Настройка HC-SR501.

Работой датчика управляет микросхема BISS0001. У датчика имеется два переменных резистора и перемычка для настройки режима (рис. 1.13.4) [15]. Один из потенциометров регулирует чувствительность прибора. Чем она больше, тем дальше область действия датчика. Также чувствительность влияет на размер детектируемого объекта. К примеру, можно исключить из срабатывания собаку или кошку.

Рис. 1.13.4. Назначение выводов пиродатчикаHC-SR501 [15]

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

Наконец, третий элемент управления – перемычка, которая переключает режим датчика. В положении L датчик ведет отсчет T от самого первого срабатывания. Допустим, мы хотим управлять светом в ванной комнате. Зайдя в комнату, человек вызовет срабатывание датчика, и свет включится ровно на время T. По окончании периода, сигнал на выходе вернется в исходное состояние, и датчик будет дать следующего срабатывания.

В положении H датчик начинает отсчет времени T каждый раз после обнаружения движения. Другими словами, любое шевеление человека вызовет обнуление таймера отсчета T. По умолчанию, перемычка находится в состоянии H.

Подключение HC-SR501 к платформе. Для соединения с микроконтроллером или напрямую с реле у HC-SR501 имеется три вывода. Выходной вывод можно подключать к любому цифровому входу платформы.

Лабораторные задания

Необходимо данные с пиродатчика вывести в последовательный порт на компьютере. Для этого необходимо подать питание и выход HC-SR501 подключить к восьмому порту на платформе. Необходимо записать следующую программу.

```
// Подключение пиродатчика
int pirPin=8;
int LedPin=13;
  void setup() {
  Serial.begin(9600); // Объявляем работу COM порта со скоростью 9600
  pinMode(pirPin, INPUT); //Объявляем вывод, к которому подключен датчик движения, входом
  pinMode(LedPin, OUTPUT); //Объявляем вывод, к которому подключен светодиод, выходом
}
  void loop() {
  int pirVal = digitalRead(pirPin); //Считываем значения с датчика движения. Если обнаружили движение,
                                    //то транслируем сигнал в монитор порта и включаем светодиод
  if(pirVal == HIGH)
  {
   digitalWrite(LedPin, HIGH);
   Serial.println("Тревога ");
```

```
    delay(2000);
  }
else
{
  Serial.println("Сканирую ");
  digitalWrite(LedPin,LOW);
  delay(1000);
}
}
```

Задания для самостоятельного решения

1. Необходимо смоделировать устройство, включающее освещение (светодиод) при обнаружении движения в зависимости от уровня освещения. В качестве датчика освещения использовать фоторезистор.

2. Необходимо смоделировать систему, включающую звуковой сигнал тревоги с выдачей информации на ЖК индикатор 5110.

3. Необходимо смоделировать устройство управления дверьми. В качестве привода двери использовать электродвигатель, подключенный через Н мост. При обнаружении движения двигатель 2 секунды вращает вал в одну сторону (дверь открыта). Далее пауза в несколько секунд, и, если нет движения, двигатель включается в обратную сторону на 2 секунды (дверь закрыта).

Контрольные вопросы

1. Каковы области применения пиродатчиков.
2. Каков принцип действия пиродатчика?
3. Какое устройство у модуля HC-SR501?
4. Каким образом можно увеличить чувствительность пиродатчика?

Лабораторная работа №14
Датчики температуры LM 35 и DS18B20

Цель работы: познакомиться с принципом работы аналогового (LM35) и цифрового (DS18B20) датчиков температуры, компилировать и загрузить в микроконтроллер программы на Arduino.

Оборудование: персональный компьютер, среда разработки Arduino IDE, платформа Arduino UNO, макетная плата, провода, датчики температуры LM 35 и DS18B20, индикатор ЖК 5110, светодиодный индикатор TM1637.

Теоретические сведения

Датчик температуры LM35. Еще один полезный элемент, который часто используется в современных устройствах – это датчик температуры. Самый популярный пример использования датчика температуры – термостат. Это устройство, которое постоянно следит за температурой воздуха, и регулирует подачу энергии в систему отопления. Смежный пример – котел для нагрева воды.

На данном занятии используется датчик LM35. Вместо него можно использовать любой другой похожий датчик: TMP35, TMP37, LM335 и подобные. Выглядит датчик как обычный транзистор, так как выполнен в корпусе TO-92 (рис. 1.14.1) [15].

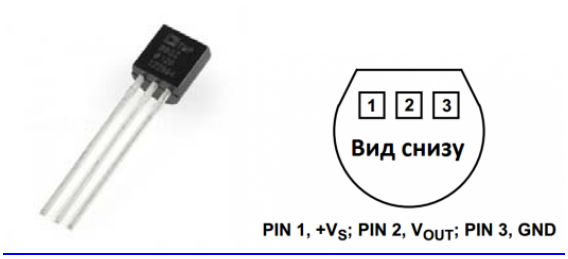


Рис. 1.14.1. Внешний вид и назначение выводов датчика температуры LM35 [15]

Конкретно этот датчик имеет следующие характеристики:

- напряжение питания: 2,7-5,5В;
- погрешность: 2 градуса;
- измеряемая температура: от 10°С до 125°С;
- потребляемый ток: 50мкА.

Датчик LM35 имеет три вывода. Если посмотреть на датчик со стороны этих выводов и срезом вверх, как показано на рисунке, то слева будет – положительный контакт питания (+2.7-5.5В), по центру – выход на контроллер, и справа – отрицательный контакт питания (земля), как показано на (рис. 1.14.1) [15].

Датчик аналоговый, а значит, на его выходе мы имеем не 0 или 1, а напряжение в диапазоне от 0 до 5 вольт. Для удобства датчик поставлен на плату. Назначение выводов представлены на рис. 1.14.2.

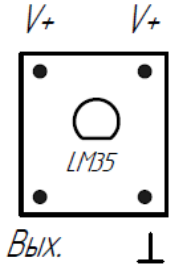


Рис. 1.14.2. Назначение выводов модуля датчика температуры LM35

Первая программа предназначена для считывания данных о температуре и выдаче в последовательный порт:

```
// Программа считывания данных температуры и вывод в Serial
float tempC; // определяем переменную для показаний температуры
int reading; // определяем переменную для считывания "сырых" данных с датчика
int tempPin = A0; // определяем контакт подключения датчика
 void setup()
{
  Serial.begin(9600);
analogReference(INTERNAL);
}
void loop()
{
reading = analogRead(tempPin);
tempC = (reading*1.1 /1023)*1000 /10 ;
Serial.println(tempC);
delay(1000);
}
```

В программе можно заметить команду ***analogReference(INTERNAL);***

Можно также воспользоваться функцией ***analogReference().***

Функция определяет опорное напряжение, относительно которого происходят аналоговые измерения. Функция ***analogRead()*** возвращает значение с разрешением 10 бит пропорционально входному напряжению на аналоговом входе, и в зависимости от опорного напряжения.

Возможные настройки:

DEFAULT: стандартное опорное напряжение 5 В (на платформах с напряжением питания 5 В) или 3.3 В (на платформах с напряжением питания 3.3 В)

INTERNAL: встроенное опорное напряжение 1.1 В на микроконтроллерах ATmega168 и ATmega328, и 2.56 В на ATmega8.

INTERNAL1V1: встроенное опорное напряжение 1.1 В (Arduino Mega).

INTERNAL2V56: встроенное опорное напряжение 2.56 (Arduino Mega).

EXTERNAL: внешний источник опорного напряжения, подключенный к выводу AREF.

Синтаксис

analogReference(type)

Параметры

type: определяет используемое опорное напряжение (DEFAULT, INTERNAL или EXTERNAL).

Возвращаемое значение – нет

Внешнее напряжение рекомендуется подключать к выводу AREF через резистор 5 кОм.

Таким образом уменьшается риск повреждения микросхемы Atmega если настройки analogReference не совпадают с возможностями платформы. Однако при этом произойдет небольшая просадка напряжения, вследствие того, что имеется встроенный резистор 32 кОм, подключенный к выводу AREF. В этом случае оба резистора работают как делитель напряжения. Подсоединение внешнего резистора позволяет быстро переключаться на напряжение 3.3 В вывода AREF с напряжения 5 В DEFAULT без конфигурации аппаратной части и АЦП.

В программе можно заметить выражение:

tempC = (reading*1.1 /1023)*1000 /10.

Оно необходимо для того, чтобы преобразовать аналоговый сигнал датчика в градусы Цельсия. Дело тут вот в чем. Все аналоговые датчики имеют важную характеристику — отношение количества вольт к единице измеряемой величины. Например, в спецификации к нашему датчику tmp35 написано, что каждый градус измеряемой температуры, соответствует 10 милливольтам напряжения на выходе. Исходя из этих рассуждений, прочитанное с помощью analogRead значение мы сначала преобразуем к количеству Вольт:

вольты = (значение АЦП * 1,1 / 1023)

Такая процедура называется нормировкой. Здесь 1023 – максимальное значение, которое может вернуть нам 10-битный АЦП, встроенный в Ардуино Уно.

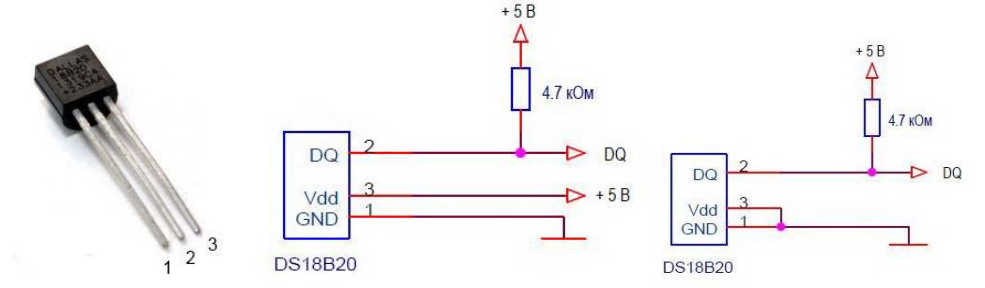
1,1 – рабочеенапряжение встроенного АЦП.

Затем преобразуем эти вольты в градусы Цельсия:

градусы = (вольты * 1000) / 10

Превращаем вольты в милливольты (*1000), и делим на 10 (то самое число из спецификации!).

Рассмотрим возможности цифрового датчика DS18B20. Подключение датчика DS18B20 к микроконтроллеру. Самый распространенный вариант исполнения DS18B20 в корпусе TO-92 (рис. 1.14.3) [15].



а) б) в)

Рис. 1.14.3. Внешний вид и схемы включения датчика температуры DS18B20
а) внешний вид; б) схема с внешним источником;
в) схема в режиме «паразитного питания» [15]

Выводы на схемах подключения указаны для такого варианта. Существуют две стандартные схемы подключения DS18B20 к микроконтроллеру.
Схема питания с внешним источником (рис. 1.14.3, б). Подтягивающий резистор строго необходим и устанавливается вблизи вывода микроконтроллера.

Схема в режиме «паразитного питания» (рис. 1.14.3, в) позволяет подключить термодатчик к микроконтроллеру всего двумя проводами, что особенно важно при размещении датчика на значительном расстоянии от контроллера.

В этом режиме сигнал шины данных заряжает внутренний конденсатор датчика и за счет энергии на нем происходит питание устройства при низком уровне на шине. У режима «паразитного питания» много особенностей. Главное, что в этом режиме не гарантируется работа датчика при температуре свыше 100 °С. Надо использовать схему с внешним питанием.

Для удобства использования датчик смонтирован на плате. Назначение выводов на модуле с датчиком DS18B20 показаны на рис. 1.14.4

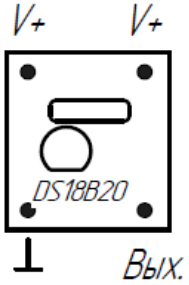


Рис. 1.14.4. Назначение выводов датчика температуры DS18B20

Обмен информацией с термодатчиком осуществляется по шине интерфейса 1-Wire.

Вот перечень необходимых операций работы с устройством:

1. Инициализация – последовательность импульсов, с которых начинается любая операция на шине.
2. Запись байта – передача байта данных в устройство DS18B20.
3. Чтение байта – прием данных из устройства DS18B20.

Этих трех операций достаточно для работы с термодатчиком, все они поддерживаются библиотекой OneWire.

Библиотека Ардуино OneWire для работы с интерфейсом 1-Wire.

Нас интересуют следующие команды.

OneWire(uint8_t pin);

Конструктор, Pin – номер вывода, к которому подключен датчик.

OneWire sensDs (10); // датчик подключен к выводу 10

uint8_t reset(void);

Инициализация операции на шине. С этой команды должна начинаться любая операция обмена данными. Возвращает:

- 1 – если устройство подключено к шине (был ответный импульс присутствия);

- 0 – если устройство отсутствует на шине (ответного импульса не было).

void write(uint8_t v, uint8_t power = 0);

Запись байта. Передает байт в устройство на шине.

v – байт;

power – признак выбора режима питания;

power=0 – питание от внешнего источника

power=1 – “паразитное” питание.

uint8_t read(void);

Чтение байта – принимает байт, переданный устройством. Возвращает значение принятого байта.

Этих команд вполне достаточно, чтобы работать с датчиком DS18B20. Можно добавить методы записи и чтения блоков байтов и функцию вычисления контрольной суммы.

void write_bytes(const uint8_t *buf, uint16_t count, bool power = 0);

Запись блока байтов.

buf – указатель на массив;

count – число байтов;

power – признак выбора режима питания.

void read_bytes(uint8_t *buf, uint16_t count);

Чтение блока байтов.

buf – указатель на массив;

count – число байтов.

static uint8_t crc8(const uint8_t *addr, uint8_t len);

Функция вычисления 8ми разрядной контрольной суммы.

addr – указатель на массив данных;

len – число байтов.

Возвращает вычисленную сумму.

Последовательность операций работы с DS18B20.

Для измерения температуры необходимо выполнить следующую последовательность действий:

1. **sensDs.reset()** – Инициализация. Выполняет сброс шины, готовит ее для выполнения новой операции.

2. **sensDs.write(0xCC, power)** – Команда пропуск ROM. У нас только один датчик на шине. Поэтому нет необходимости в поиске устройства с нужным адресом. Мы эту операцию пропускаем.

3. **sensDs.write(0x44, power) –** Команда инициирует измерение температуры.

Пауза 1 сек. Ожидание на время, необходимое для выполнения датчиком преобразования температуры. Это время зависит от выбранной разрешающей способности датчика. Мы используем максимальное разрешение 12 бит. Это разрешение установлено в датчике по умолчанию. Время преобразования для него – 750 мс. Если необходима другая разрешающая способность, то ее необходимо задать дополнительными командами.

4. **sensDs. reset() –** Инициализация. Мы собираемся выполнить новую операцию на шине 1-Wire.

5. **sensDs.write(0xCC, power) –** Команда пропуск ROM.

6. **sensDs.write(0xBE, power)** – Команда чтения памяти датчика. Команда используется для чтения всех 9-ти байтов памяти DS18B20 (таблица 1.14.1).

Таблица 1.14.1

Структура памяти DS18B20

Байт 0	Температура мл. байт
Байт 1	Температура ст. байт
Байт 2	Th регистр порога сигнализации
Байт 3	Tl регистр порога сигнализации
Байт 4	Регистр конфигурации
Байт 5	Зарезервирован
Байт 6	Зарезервирован
Байт 7	Зарезервирован
Байт 8	Циклический код CRC

7. **read_bytes(buf, 9)** – Чтение 9-ти байтов данных.

8. **crc8(addr, 8)** – Вычисление контрольного кода данных.

Сравнение контрольного кода с принятым.

После этой последовательности операций значение температуры содержится в первых двух байтах массива из принятых 9-ти байтов (рис. 1.14.5) [18].

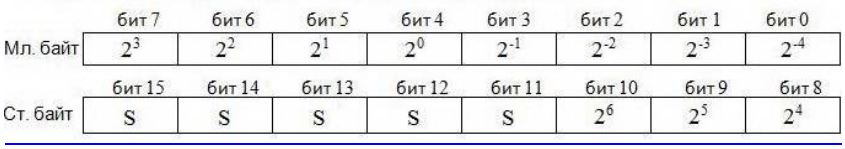


Рис. 1.14.5. Формат данных датчика температуры DS18B20[18]

Отрицательная температура записывается в дополнительном коде (рис. 1.14.6) [18].

ТЕМПЕРАТУРА	ЦИФРОВОЙ КОД (двоичный)	ЦИФРОВОЙ КОД (Hex)
+125°C	0000 0111 1101 0000	07D0h
+85°C	0000 0101 0101 0000	0550h
+25.0625°C	0000 0001 1001 0001	0191h
+10.125°C	0000 0000 1010 0010	00A2h
+0.5°C	0000 0000 0000 1000	0008h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1000	FFF8h
-10.125°C	1111 1111 0101 1110	FF5Eh
-25.0625°C	1111 1110 0110 1111	FE6Fh
-55°C	1111 1100 1001 0000	FC90h

Рис. 1.14.6. Цифровой код данных датчика температуры DS18B20 [18]

Младший разряд имеет вес 0,0625 °C.

Лабораторные задания

Необходимо подключить аналоговый датчик LM35, записать код, приведенный ранее и получить значения температуры.

Для проверки работы цифрового датчика температуры DS18B20 может использоваться следующая программа.

```
// Вывод значений данных в Serial
#include <OneWire.h> // Подключение библиотеки OneWire
OneWire ds(10);  // Создание объекта ds и указание вывода подключения DS18B20
 void setup() {
Serial.begin(9600);
}
 void loop() {
byte data[2];   // массив данных для температуры
ds.reset();       // инициализация шины
ds.write(0xCC); //команда пропуск ROM
ds.write(0x44); // команда инициирует измерение температуры
delay(750); // Задержка для получения температуры
ds.reset();
ds.write(0xCC);
ds.write(0xBE); // Команда чтения памяти датчика
data[0] = ds.read(); // чтение байта 0
data[1] = ds.read(); // чтение байта 1
int Temp = (data[1]<< 8)+data[0]; // Старший байт сдвигается на 8 добавляется младший
Temp = Temp>>4; // Сдвигается вправо, убираются дробные части
```

```
  Serial.println(Temp);
  }
```

Для подключения двух датчиков можно использовать следующий скетч и библиотеку DallasTemperature, при этом адреса датчиков надо узнать заранее. В библиотеке OneWire есть пример DS18x20_Temperature.pde с помощью которого, можно определить номер устройства.

```
#include <OneWire.h>
#include <DallasTemperature.h>
 #define ONE_WIRE_BUS 9
 OneWire oneWire(ONE_WIRE_BUS);
 DallasTemperature sensors(&oneWire);
 DeviceAddress Thermometer1 = {
  0x28, 0x00, 0x54, 0xB6, 0x04, 0x00, 0x00, 0x92 };  // адрес датчика DS18B20
280054B604000092
DeviceAddress Thermometer2 = {
  0x28, 0x9E, 0x95, 0xB5, 0x04, 0x00, 0x00, 0x57 };
 void setup() {
   sensors.begin();
  sensors.setResolution(Thermometer1, 10);
  sensors.setResolution(Thermometer2, 10);
   Serial.begin(9600);
}
 void printTemperature(DeviceAddress deviceAddress) {
  float tempC = sensors.getTempC(deviceAddress);
  Serial.println(tempC);
}
void loop() {
   sensors.requestTemperatures();
  Serial.print("Sensor1  ");
  printTemperature(Thermometer1);
   Serial.print("Sensor2  ");
  printTemperature(Thermometer2);
}
```

Задания для самостоятельного решения

1. Необходимо вывести значение температуры на ЖК индикатор Nokia 5110.

2. Необходимо вывести значение температуры на LCD индикатор с контроллером TM1637.

3. К полученным устройствам добавить возможность включение вентилятора (использовать выход на реле) при превышении температуры.

Контрольные вопросы

1. Каковы достоинства и недостатки датчика температуры LM35?
2. Каковы достоинства и недостатки датчика температуры DS18B20?
3. Какие достоинства и недостатки полупроводниковые датчики температуры имеют перед термопарой?
4. Каким образом используется «паразитное питание»?

Лабораторная работа №15
Bluetooth модуль HC06

Цель работы: познакомиться с принципом работы Bluetooth модуля HC06, составить, компилировать и загрузить в микроконтроллер программы на Arduino.

Оборудование: персональный компьютер, среда разработки Arduino IDE, платформа Arduino UNO, макетная плата, провода, Bluetooth модуль HC06, светодиод

Теоретические сведения

Bluetooth – это пожалуй самый распространенный тип связи для коротких дистанций, которым пользуются большинство современных электронных устройств. Главными достоинствами bluetooth можно назвать хорошую устойчивость к широкополосным помехам и простоту реализации. Первое ознчает, что много устройств, находящихся в одном месте, могут одновременно общаться между собой, не мешая друг другу.

Самыми доступными на сегодня Bluetooth модулями можно назвать HC-05 и HC-06. Отличия между ними в том, что первый может работать как в режиме ведущего (slave), так и в режиме ведомого (master). Второй же является чисто ведомым устройством. Другими словами, HC-06 не может сам обнаружить парное устройство и наладить с ним связь, он может лишь подчиниться ведущему.

Оба устройства базируются на чипе CSR BC417, который поддерживает Bluetooth версии 2.0 со скоростью до 3 Мбит/сек. Существует также совместимый вариант Jdy-30 с контроллером BK3231.

Варианты исполнения. Обычно модули продаются в виде двух спаянных вместе плат. Меньшая из них – заводской модуль, широко используемый в разных электронных устройствах. Внешний вид модулей HC-05, HC-06 и JDY-30 представлен на рис. 1.15.1 [19].

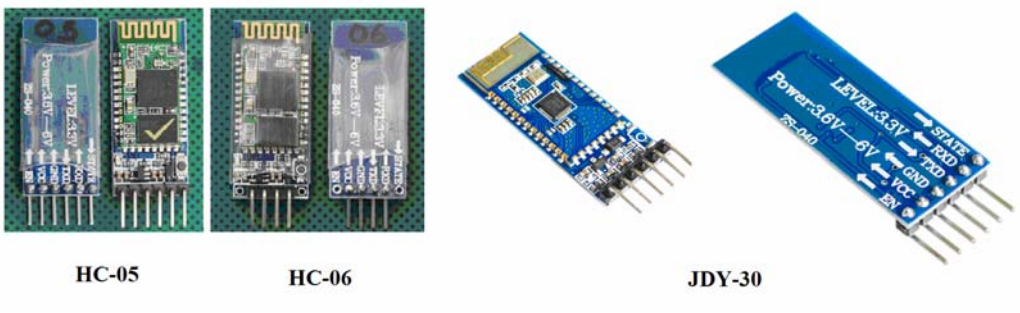


Рис. 1.15.1. Внешний вид модулей HC-05, HC-06 и JDY-30 [19]

Назначение выводов следующее^
EN – включение/выключение модуля;
VCC – питание +5В;
GND – земля;
TXD, RXD – UART интерфейс для общения с контроллером;
STATE – индикатор состояния;
KEY – вывод для входа в режим АТ-команд.

Настройка модуля осуществляется в режиме АТ-команд, который включается с помощью вывода KEY.

Подключение к платформе Ардуино. осуществляется по следующей схеме (табл. 1.15.1)

Таблица 1.15.1

Порядок подключения модуля Bluetooth

Bluetooth	GND	VCC	TXD	RXD	KEY
Ардуино Уно	GND	+5V	RX	TX	

Малая платка Bluetooth модуля имеет напряжение логики 3.3 Вольта. А это значит, что Ардуино Уно может либо сжечь у нее порты, либо просто неправильно передавать сигналы. В большинстве случаев большая плата имеет все необходимое, чтобы этого избежать. Указанные выше модули легко подключаются к Arduino, даже несмотря на то, что сами изготовители написали на плате предупреждение об уровне сигналов 3.3 Вольта.

Лабораторные задания

Теперь необходимо записать пробный код программы, приведенный ниже.

Во время загрузки скетча необходимо, чтобы Bluetooth модуль был отключен от микроконтроллера arduino. В противном случае скетч не запишется, потому что связь с Bluetooth модулем происходит по одному и тому же порту RX и TX, что и USB.

```
int val;
int LED = 13;
void setup()
{
 Serial.begin(9600);
 pinMode(LED, OUTPUT);
 digitalWrite(LED, LOW);
}
void loop()
{
 if (Serial.available())
  {
   val = Serial.read();
   // При символе "1" включаем светодиод
   if (val == '1')
   {
    digitalWrite(LED, HIGH);
   }
   // При символе "0" выключаем светодиод
   if ( val == '0')
   {
    digitalWrite(LED, LOW);
   }
  }

}
```

После того как программа записана и Bluetooth модуль подключен к Arduino, можно перейти к следующему шагу.

Подключение Bluetooth к телефону. Желательно в качестве источника питания для arduino использовать не USB, а внешний блок питания на 9 В.

Необходимо включить Bluetooth на телефоне и поискать новые устройства. Необходимо найти в списке устройств "HC-06" или «JDY-30» и подключиться к нему. Пин-код "1234" Теперь нужно скачать программу bluetooth terminal на телефон.

После того как установлен терминал, необходимо его запустить и выбрать bluetooth модуль HC-06 или «JDY-30» и подключиться к нему.

Надо написать в терминале цифру «1» и отправить. Светодиод L, который находится на плате arduino рядом с pin 13, должен загореться. Теперь если отправить через терминал цифру «0», светодиод L должен погаснуть.

Задания для самостоятельного решения

1. Изменить программу так, чтобы светодиод зажигался и потухал с помощью одной и той же команды, например "G".

2. Изменить программу так, чтобы статус светодиода выводился в bluetooth.

3. Дописать программу и научить ее преобразовывать текстовые данные, приходящие через bluetooth в цифровые и реализовать диммер, зажигать светодиод с помощью ШИМ, на заданную яркость от 0 до 254 приходящую через bluetooth.

Контрольные вопросы

1. Каковы области применения Bluetooth модуля HC06?

2. Каким образом осуществляется соединение с внешним устройством?

3. Почему во время загрузки команды Bluetooth модуль надо отключить от питания?

4. Каким образом можно изменить название модуля в сети и пароль?

2. ПРАКТИЧЕСКИЕ РАБОТЫ

Практическая работа №1
Логический анализатор Saleae Logic Analyzer

Логический анализатор – это хороший инструмент при отладке цифровой электроники. Технические характеристики логического анализатора Saleae logic Analyzer представлены в табл. 2.1.1.

Таблица 2.1.1

Технические характеристики Saleae Logic Analyzer

Параметр	Значение
число цифровых каналов	8
частота оцифровки на канал	до 24 МГц
количество сэмплов в выборке	до 1G (зависит от количества памяти ПК)
входное сопротивление	100 кОм
диапазон рабочих напряжений	–0,5…5,25 В
напряжение логического «0»	–0,5…0,8 В
напряжение логической «1»	2,0…5,25 В
защита от статики	+
защита по превышению напряжения	+/−15 В

Для работы с устройством используется программа Logic. Внешний вид окно программы представлен на рис. 2.1.1. Если устройство подключено, то в заголовке появится сообщение «Connected».

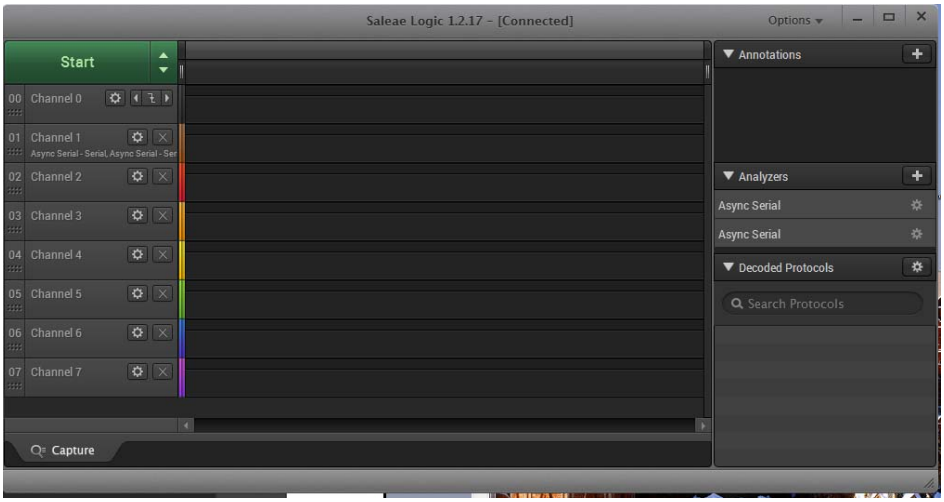


Рис. 2.1.1. Внешний вид окна программы Logic

В центре находятся 8 полос-каналов, на которых будет выводиться график сигнала. Слева находится кнопка Start и кнопки настроек каналов. Справа – окна измерений и анализа. Рассмотрим настройки (рис. 2.1.2). Здесь можно выбрать частоту выборки (24 млн/с) и длительность записи сигнала.

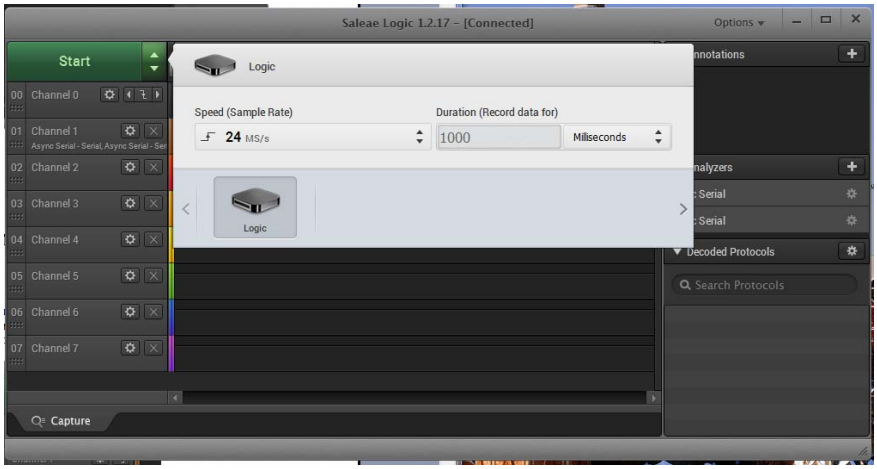


Рис. 2.1.2. Внешний вид окна настроек программы Logic

Чем больше частота и длительность, тем больше потребуется памяти для сбора и хранения данных. Для каждого канала можно выбрать ширину полосы, настроить фильтр помех или вообще скрыть канал (рис. 2.1.3).

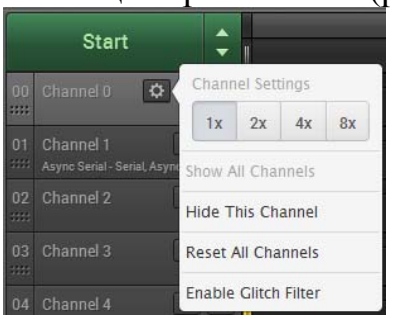


Рис. 2.1.3. Внешний вид окна настроек каналов программы Logic

Для одного из каналов можно настроить триггер: по фронту сигнала, по спаду, по ширине высокого или низкого импульса. Тогда при нажатии кнопки Start отсчёт времени записи начнётся только после срабатывания триггера (рис. 2.1.4).

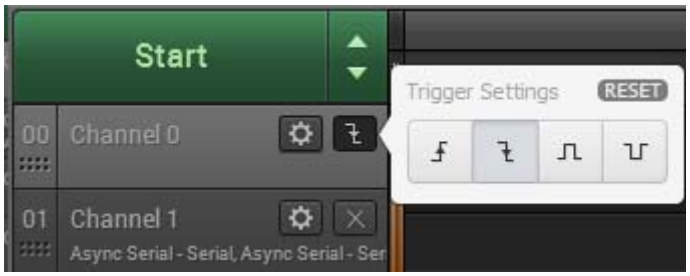


Рис. 2.1.4. Внешний вид окна настроек триггера программы Logic

Рассмотрим различные сигналы с платформы Ардуино UNO. Подадим на анализатор сигнал ШИМ с изменяющейся частотой. Для этого в платформу надо записать следующую программу.

```
// программа для тестирования логического анализатора ШИМ
int outPin=11;
int startPin=12;
void setup() {
 pinMode(outPin, OUTPUT);
 pinMode(startPin, OUTPUT);
 digitalWrite(startPin, LOW);
}
void loop() {
  digitalWrite(startPin, HIGH);
    for (int i=0; i<=256; i+=1){
 analogWrite (outPin, i);
 delay (10);
 }
 digitalWrite(startPin, LOW);
}
```

Вывод 11 используется как выход ШИМ, а вывод 12 в качестве источника сигнала для триггера, который запускает процесс записи. Триггер на канале «1» надо настроить на запуск по фронту, как показано на рис. 2.1.5.

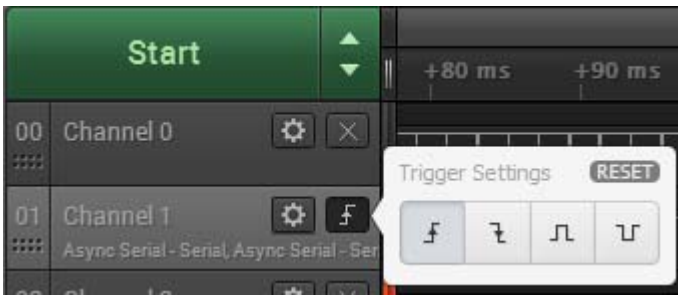


Рис. 2.1.5. Внешний вид окна установки триггера по фронту импульса

На этот канал надо подать сигнал с 12 вывода. Сигнал с 11 вывода надо подать на канал «0». Длительность записи выбрать 3 секунды (как на рис. 2.1.6).

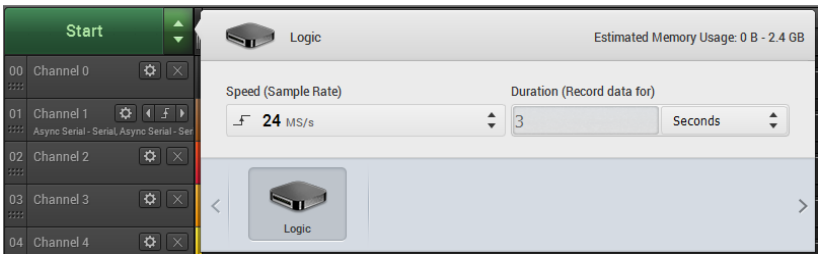


Рис. 2.1.6. Внешний вид окна установки длительности записи сигналов

Процесс записи начинается при нажатии на кнопку «Start». После записи отображаются состояния 2 каналов. Верхний (канал 0) содержит вид сигнала с ШИМ, а нижний (канал 1) сигнал с вывода 12. Его можно использовать в качестве опорной точки отсчета начала работы ШИМ (рис. 2.1.7).

Рис. 2.1.7. Внешний вид сигнала ШИМ

При наведении курсора мыши на отображаемый канал и вращая колесо мыши можно изменять масштаб отображения сигнала и посмотреть длительность импульса, а также частоту.

Несколько изменим программу и подадим на выход 11 меандр с изменяющееся частотой. Длительность можно установить 1 секунду.

```
// программа для тестирования логического анализатора Меандр
int outPin=11;
int startPin=12;
```

```
void setup() {
 pinMode(outPin, OUTPUT);
 pinMode(startPin, OUTPUT);
 digitalWrite(startPin, LOW);
}
void loop() {
  digitalWrite(startPin, HIGH);
    for (int i=200; i<=20000; i+=200){
 tone(11,i);
delay(1);
 }
 digitalWrite(startPin, LOW);
}
```

На анализаторе видно, как изменяется частота с минимального значения до максимального.

Устройство также может расшифровывать протоколы и показывать передаваемые данные. Ниже приведен список доступных протоколов:

Async Serial
I2C
SPI
1-Wire
Atmel SWI
BiSS C
CAN
DMX-512
HD44780
HDLC
HDMI CEC
I2S / PCM
JTAG
LIN
MDIO
Manchester
MIDI
Modbus
PS/2 Keyboard/Mouse
UNI/O
USB LS и FS

Для примера посмотрим на сигналы UART с платформы. Необходимо отправить по последовательному соединению фразу «Hello, world!». Для этого канал 0 надо подключить к выводу 1 (TX).

В анализаторе надо выбрать тип распознаваемого протокола «Async Serial», как на рис. 2.1.8.

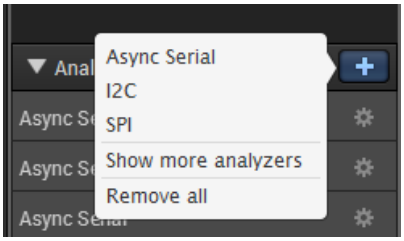


Рис. 2.1.8. Внешний вид окна установки анализируемых протоколов

Программа имеет следующий вид:

```
// программа для тестирования логического анализатора UART
int outPin=11;
int startPin=12;
void setup() {
 pinMode(outPin, OUTPUT);
 pinMode(startPin, OUTPUT);
 digitalWrite(startPin, LOW);
 Serial.begin(9600);
}
void loop() {
  digitalWrite(startPin, HIGH);
     Serial.print("Hello, world!");
  delay(200);
  digitalWrite(startPin, LOW);
}
```

В результате можно наблюдать как логический сигнал, так и расшифровка передаваемых символов (рис. 2.1.9).

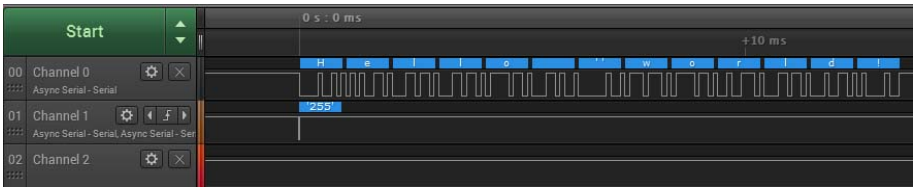


Рис. 2.1.9. Внешний вид передаваемого сигнала по UART

Задания для самостоятельного решения

1. Необходимо написать программу и получить изображение ШИМ с плавно изменяющейся скважностью от минимума до максимума и наоборот.

2. Подключив к анализатору кнопку, определить длительность дребезга контактов.

Практическая работа №2
Сдвиговый регистр 74HC595

Для понимания принципа работы сдвигового регистра 74HC595 необходимо проработать материал из лабораторной работы № 4 касательно этого регистра.

Для изучения работы сдвигового регистра необходимо собрать схему представленную на рис. 2.2.1. При нажатии на кнопку происходит поочередное зажигание светодиодов в линейке.

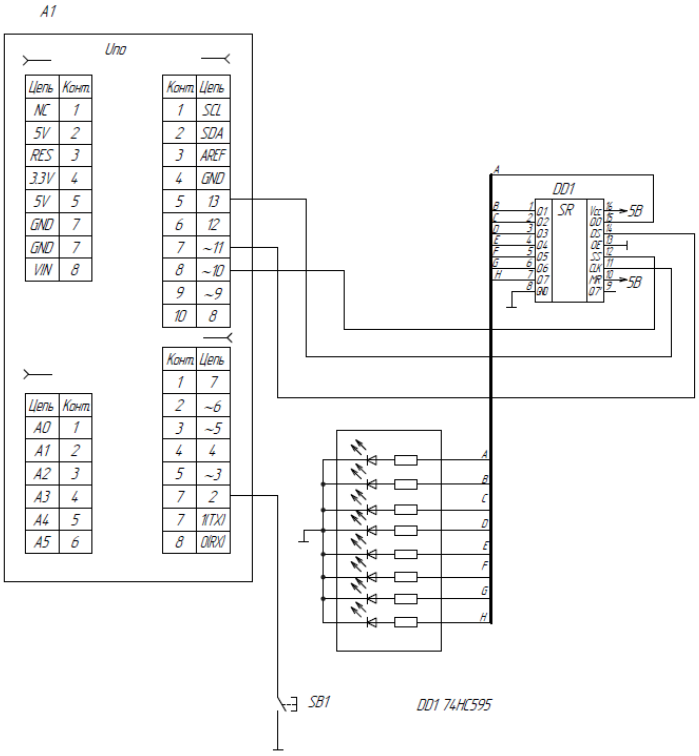


Рис. 2.2.1. Схема подключения сдвигового регистра 74HC595 и светодиодной линейки к платформе

Необходимо записать следующую программу:

```
// Программа тестирования 74HC595
// подключение библиотеки SPI
#include <SPI.h>
```

```
int ss_pin=10; // пин SS
int switch_pin=2; // пин кнопки
int pos=0;
  void setup()
{
SPI.begin();
pinMode(ss_pin, OUTPUT);// Сконфигурировать контакт SS как выход
pinMode(switch_pin, INPUT);// Сконфигурировать контакт кнопки как
вход
digitalWrite(switch_pin, HIGH);// Включаем подтягивающий резистор
//Serial.begin(9600);
digitalWrite(ss_pin, LOW); // Защелку вниз
//передаем битовую маску
SPI.transfer(B00000000);// Гасим все светодиоды
digitalWrite(ss_pin,HIGH); // защелку вверх вывести данные на выводы
74HC595
}
void loop()
{
  if (digitalRead (switch_pin)==LOW)
   {
digitalWrite(ss_pin, LOW); // Защелку вниз
//передаем битовую маску
SPI.transfer(B00000001<<pos);
digitalWrite(ss_pin,HIGH); // защелку вверх вывести данные на выводы
74HC595
pos=(pos+1)%8;
delay(250);
 }
}
```

Для просмотра логическим анализатором сигналов SPI необходимо назначить каналам выводы интерфейса, например, как на рис. 2.2.2.

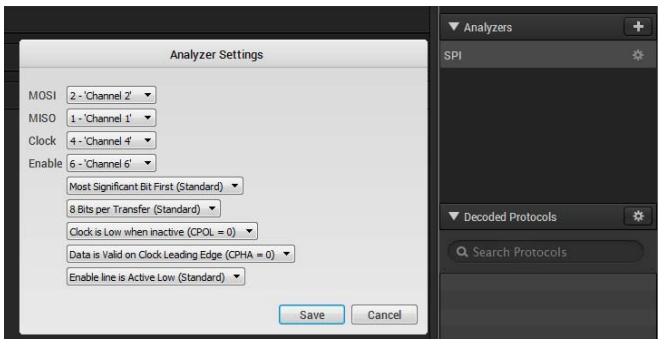


Рис. 2.2.2. Окно установки определения протокола SPI

Для удобства также надо установить вид отображаемых чисел BIN (рис. 2.2.3).

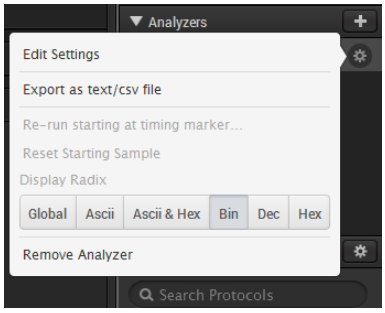


Рис. 2.2.3. Окно установки отображаемых двоичных чисел

Включение процесса записи лучше сделать по спаду на защелке (SS) (рис. 2.2.4).

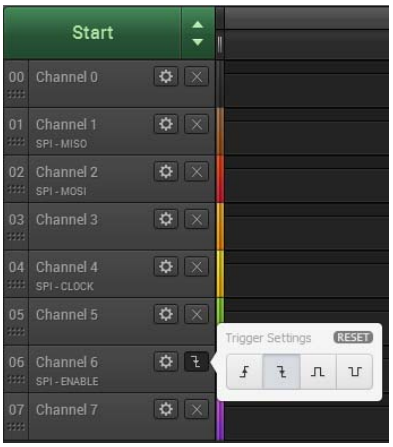


Рис. 2.2.4. Окно установки триггера защелки SS

В этом случае процесс записи начнется только при нажатии кнопки. Длительность записи в 10 миллисекунд будет достаточно.

Задания для самостоятельного решения

1. Необходимо с помощью логического анализатора посмотреть сигналы SPI выходов.
2. Необходимо зажигать новые светодиоды, не гася предыдущие. Посмотреть сигналы на логическом анализаторе.

Практическая работа №3
Сдвиговый регистр 74HC165

Для понимания принципа работы сдвигового регистра 74HC165 необходимо проработать материал из лабораторной работы № 6 касательно этого регистра.

Для изучения принципа работы 74HC165 при формировании логических сигналов на входах используется модуль, схема электрическая принципиальная которого и внешний вид представлены на рис. 2.3.1. Модуль имеет 8 переключателей. В положении «ON» на выходах формируется логический «0», а при отключенном состоянии -«1».

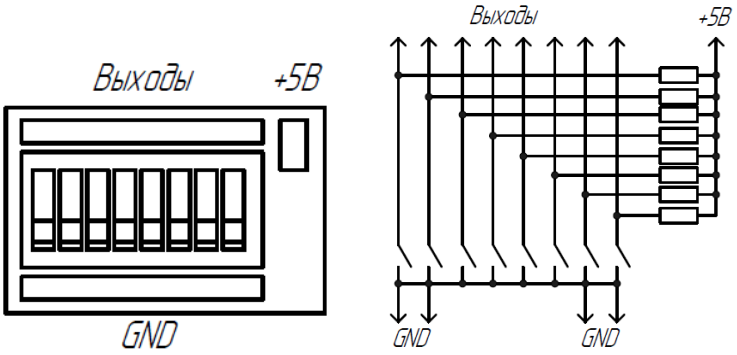


Рис. 2.3.1. Внешний вид и схема модуля с подтягивающими резисторами

Необходимо собрать схему представленную на рис. 2.3.2.

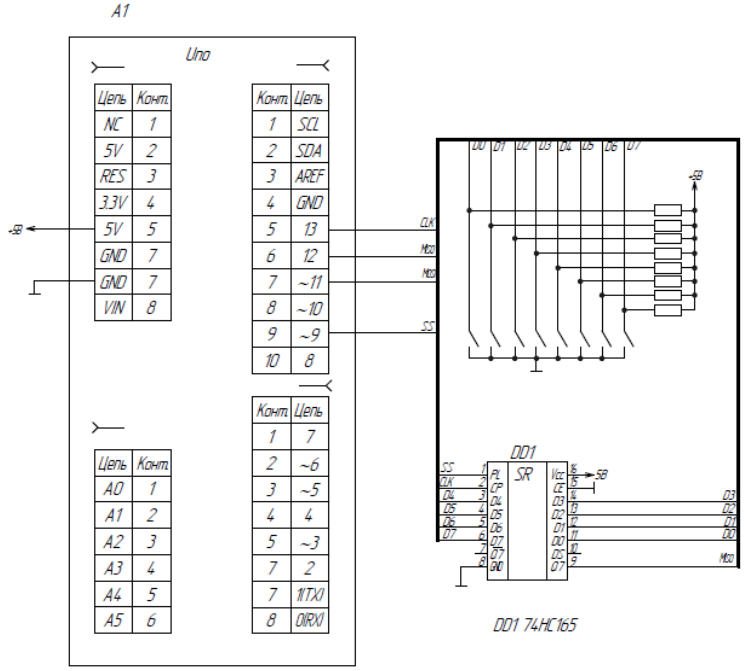


Рис. 2.3.2. Схема подключения модуля с подтягивающими резисторами и сдвигового регистра 74HC165 к платформе

Необходимо записать следующую программу:

```
// Программа тестирования 74HC165
// подключение библиотеки SPI
#include <SPI.h>
int ssin_pin=9; // пин SS in
```

```
// двойной массив, обозначающий кнопку
  byte str;
  void setup(){
SPI.begin();
Serial.begin(9600);
// Сконфигурировать защелки как выход и записать в них 1 (запрещен прием и передача)
pinMode(ssin_pin, OUTPUT);
digitalWrite(ssin_pin, HIGH);
}
void loop(){
digitalWrite(ssin_pin, LOW); // Защелку вход вниз
digitalWrite(ssin_pin, HIGH); // Защелку вход вверх
str=SPI.transfer(0);          // получаем код
Serial.println(str,BIN);
delay(1000);
}
```

Для просмотра логическим анализатором сигналов SPI необходимо назначить каналам выводы интерфейса, как и в прошлой работе. Запуск записи можно сделать по защелке и назначить работу триггера записи по спаду импульса. Длительность записи в 10 миллисекунд будет достаточно.

Задания для самостоятельного решения

1. Необходимо с помощью логического анализатора посмотреть сигналы SPI выходов.
2. Необходимо смоделировать работу охранной системы, у которой при срабатывании одного из ключей, выводится сигнал тревоги на светодиод.

Практическая работа №4
Аппаратные прерывания

Прерывания – очень важный механизм Arduino, позволяющий внешним устройствам взаимодействовать с контроллером при возникновении разных событий. Установив обработчик аппаратных прерываний в в программе, можно реагировать на включение или выключение кнопки, нажатие клавиатуры, мышки, таймера RTC, получение новых данных по UART, I2C или SPI.

Прерывание – это сигнал, который сообщает процессору о наступлении какого-либо события, которое требует незамедлительного внимания. Процессор должен отреагировать на этот сигнал, прервав выполнение текущих инструкций и передав управление обработчику прерывания (ISR, Interrupt Service Routine). Обработчик — это обычная функция, которую пишет программист и помещает туда тот код, который должен отреагировать на событие (рис. 2.4.1) [4].

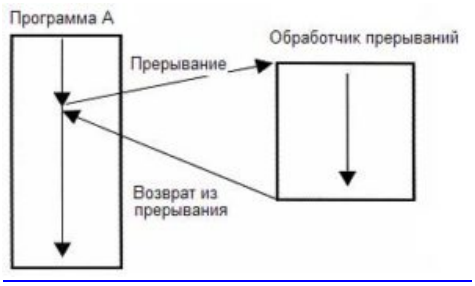


Рис. 2.4.1. Схема организации прерываний [4]

После обслуживания прерывания ISR функция завершает свою работу и процессор возвращается к прерванным занятиям – продолжает выполнять код с того места, в котором остановился. Все это происходит автоматически.

Аппаратные и программные прерывания. Прерывания в Ардуино можно разделить на несколько видов:

Аппаратные прерывания. Прерывание на уровне микропроцессорной архитектуры. Самое событие может произойти в производительный момент от внешнего устройства – например, нажатие кнопки на клавиатуре, движение компьютерной мыши и т.п.

Программные прерывания. Запускаются внутри программы с помощью специальной инструкции. Используются для того, чтобы вызвать обработчик прерываний.

Внутренние (синхронные) прерывания. Внутреннее прерывание возникает в результате изменения или нарушения в исполнении программы (например, при обращении к недопустимому адресу, недопустимый код операции и другие).

Аппаратные прерывания возникают в ответ на внешнее событие и исходят от внешнего аппаратного устройства. В Ардуино представлены 4 типа условий срабатывания аппаратных прерываний. Все они различаются сигналом на контакте прерывания:

LOW – выполняется по низкому уровню сигнала, когда на контакте нулевое значение. Прерывание может циклично повторяться – например, при нажатой кнопке.

CHANGE – прерывание вызывается при смене сигнала от LOW к HIGH и наоборот. Функция выполняется только один раз при любой смене сигнала (рис. 2.4.2) [4].

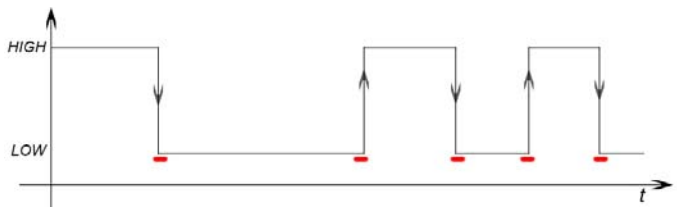


Рис. 2.4.2. Схема условия срабатывания прерывания CHANGE [4]

Вероятнее всего прерывание необходимо будет выполнять только один раз и поэтому два ниже следующих прерывания будут наиболее востребованы.

RISING – прерывание вызывается, когда значение меняется от LOW к HIGH. Разовый вызов функции, на каждый удовлетворяющий вызову переход (рис. 2.4.3) [4].

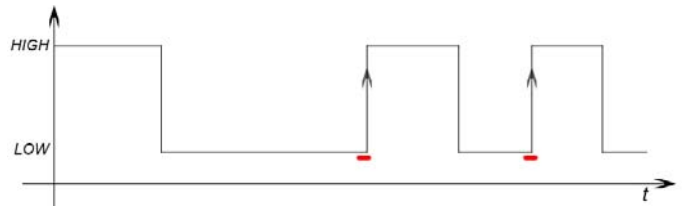


Рис. 2.4.3. Схема условия срабатывания прерывания RISING [4]

FALLING – прерывание вызывается, когда значение меняется от HIGH к LOW. Разовый вызов функции, на каждый удовлетворяющий вызову переход (рис. 2.4.4) [4] .

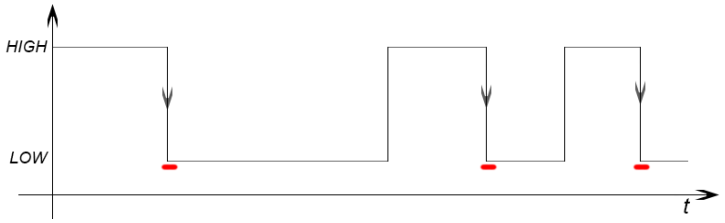


Рис. 2.4.4. Схема условия срабатывания прерывания FALLING [4]

Прерывания полезны в программах Ардуино, так как помогают решать проблемы синхронизации. Например, при работе с UART прерывания позволяют не отслеживать поступление каждого символа. Внешнее аппаратное устройство подает сигнал прерывания, процессор сразу же вызывает обработчик прерывания, который вовремя захватывает символ. Это позволяет экономить процессорное время, которое без прерываний тратилось бы на проверку статуса UART, вместо этого все необходимые действия выполняются обработчиком прерывания, не затрагивая главную программу. Особых возможностей от аппаратного устройства не требуется.

Основными причинами, по которым необходимо вызвать прерывание, являются:

1. Определение изменения состояния вывода;
2. Прерывание по таймеру;
3. Прерывания данных по SPI, I2C, USART;
4. Аналогово-цифровое преобразование;
5. Готовность использовать EEPROM, флеш-память.

При поступлении сигнала прерывания работа в цикле loop() приостанавливается. Начинается выполнение функции, которая объявля-

ется на выполнение при прерывании. Объявленная функция не может принимать входные значения и возвращать значения при завершении работы. На сам код в основном цикле программы прерывание не влияет. Для работы с прерываниями в Ардуино используется стандартная функция attachInterrupt().

В зависимости от аппаратной реализации конкретной модели микроконтроллера есть несколько прерываний. Плата Arduino Uno имеет 2 прерывания на втором и третьем пине, но если требуется более двух выходов, плата поддерживает специальный режим «pin-change». Этот режим работает по изменению входа для всех пинов. Отличие режима прерывания по изменению входа заключается в том, что прерывания могут генерироваться на любом из восьми контактов. Обработка в таком случае будет сложнее и дольше, так как придется отслеживать последнее состояние на каждом из контактов.

Синтаксис *attachInterrupt()*

Функция *attachInterrupt* используется для работы с прерываниями. Она служит для соединения внешнего прерывания с обработчиком.

Синтаксис вызова *attachInterrupt(interrupt, function, mode)*

Аргументы функции:

- *interrupt* – номер вызываемого прерывания (стандартно 0 – для 2-го пина, для платы Ардуино Уно 1 – для 3-го пина),
- *function* – название вызываемой функции при прерывании(важно – функция не должна ни принимать, ни возвращать какие-либо значения),
- *mode* – условие срабатывания прерывания.

Важные замечания.

При работе с прерываниями нужно обязательно учитывать следующие важные ограничения:

Функция – обработчик не должна выполняться слишком долго. Все дело в том, что Ардуино не может обрабатывать несколько прерываний одновременно. Пока выполняется ваша функция-обработчик, все остальные прерывания останутся без внимания и вы можете пропустить важные события. Если надо делать что-то большое – просто передавайте обработку событий в основном цикле loop(). В обработчике вы можете лишь устанавливать флаг события, а в loop – проверять флаг и обрабатывать его.

Нужно быть очень аккуратными с переменными. Интеллектуальный компилятор C++ может «переоптимизировать» вашу программу – убрать не нужные, на его взгляд, переменные. Компилятор просто не увидит, что вы устанавливаете какие-то переменные в одной части, а используете – в другой. Для устранения такой вероятности в случае с базовыми типами данных можно использовать ключевое слово volatile, например так: *volatile boolean state = 0*. Но этот метод не сработает со сложными структурами данных.

Не рекомендуется использовать большое количество прерываний (старайтесь не использовать более 6-8). Большое количество разнообразных событий требует серьезного усложнения кода, а, значит, ведет к ошибкам. К тому же надо понимать, что ни о какой временной точности исполнения в системах с

большим количеством прерываний речи быть не может – вы никогда точно не поймете, каков промежуток между вызовами важных для вас команд.

В обработчиках категорически нельзя использовать delay(). Механизм определения интервала задержки использует таймеры, а они тоже работают на прерываниях, которые заблокирует ваш обработчик. В итоге все будут ждать всех и программа зависнет. По этой же причине нельзя использовать протоколы связи, основанные на прерываниях (например, i2c).

Для подавления дребезга контактов существует два способа: программный и аппаратный. Программный заключается в введении задержки на время затухания дребезга. Аппаратный заключается в введении в схемотехнику входа RC цепочки и логического элемента, например инвертирующего триггера Шмидта. Схема такой реализации представлена на рис. 2.4.5. Там же представлен внешний вид модуля, содержащего 2 кнопки с аппаратным подавлением дребезга. Кнопки выдают при нажатии логическую единицу. В этом случае наиболее подходит режим прерывания RISING.

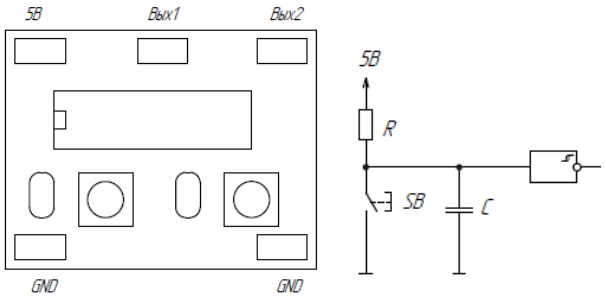


Рис. 2.4.5. Внешний вид и схема одного канала модуля подавления дребезга

Для примера подключим модуль с кнопками к 2 и3 входу.

Первая программа обрабатывает нажатие кнопки и переключает светодиод из включенного состояния в выключенное.

```
//Программа обработки прерываний
int PIN_LED=13;
int buttonINT=0;
volatile boolean actionState = HIGH;
void setup() {
  pinMode(PIN_LED, OUTPUT);// Конфигурация выхода
  attachInterrupt(buttonINT, LedON, RISING); // установка вектора прерывания(подпрограммы) LedON, срабатывание по фронту импульса
}
 void loop() {
  // В функции loop мы ничего не делаем, т.к. весь код обработки событий будет в функции LedON
}
```

```
void LedON() {
 actionState = !actionState; //
 // Выполняем другие действия, например, включаем или выключаем светодиод
 digitalWrite(PIN_LED, actionState);
}
```

Задания для самостоятельного решения

1. Необходимо включать поочередно три светодиода при нажатии на кнопку. Предусмотреть режим выключения всех светодиодов.

2. Необходимо обрабатывать прерываниями нажатие двух кнопок и регулировать яркость светодиода.

Практическая работа №5
Расширитель IIC PCF8574

Для понимания принципа работы расширителя IIC PCF8574 необходимо изучить материал лабораторной работы №7.

Напишем программу, считывающую показания с 4 кнопок (DIP переключатель) и выводящую эти значения в виде горящих или погасших светодиодов на светодиодную линейку (рис. 2.5.1).

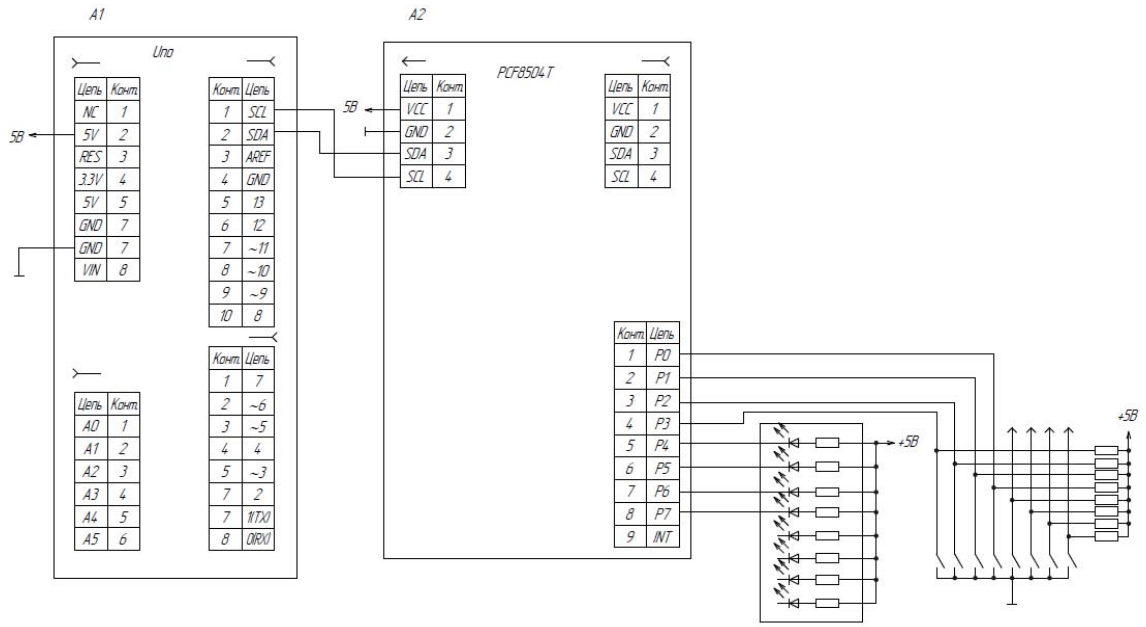


Рис. 2.5.1. Схема подключения модуля переключателей и светодиодной линейки через расширитель IIC PCF8574 к платформе

Программа имеет следующий вид:

```
// Тестирование PCF8574
```

```
#include <Wire.h>                  //  Подключаем библиотеку для работы с шиной I2C
byte str=B11111111;                // Маска
int    address=0x3F;
void setup(){
   Wire.begin();                 // Инициализация библиотеки Wire
   }
 void loop(){
  Wire.beginTransmission(address);   //Начинаем процедуру передачи данных по интерфейсу I2C
  Wire.write(B11111111);           // Записываем 1 во все разряды, гасим светодиоды на DIP подтяжку
  Wire.endTransmission();          // Завершаем процедуру передачи данных
   Wire.requestFrom(address, 1);                  // Получаем 1 байт
   str=Wire.read();                          // получаем маску строки (строки начинаются с 4 разряда)
   for (int i=0; i<=3; i++){ // цикл сканирования маски начиная с 1 разряда (с 4 по 7 подключены светодиоды)
   if(bitRead(str,i)==LOW)      // проверяем, если в полученной маске на месте переключателя 0 включаем светодиод
    bitWrite(str,i+4, 0);
    else
    bitWrite(str,i+4, 1);
  }
    Wire.beginTransmission(address);  //Начинаем процедуру передачи данных по интерфейсу I2C
    Wire.write(str);           // отправляем байт данных
    Wire.endTransmission();                  // Завершаем процедуру передачи данных
      }
```

Задания для самостоятельного решения

1. Необходимо посмотреть логическим анализатором сигнал с выхода.
2. Необходимо подключить кнопку и включать поочередно все 8 светодиодов.

Практическая работа №6
Датчик Холла

Датчик на основе эффекта Холла – это электроприбор, который регистрирует изменение магнитного поля. Внешний вид модуля с датчиком Холла 49E представлен на рис. 2.6.1 [19].

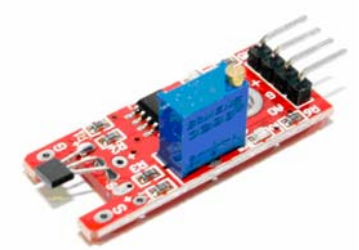


Рис. 2.6.1. Внешний вид модуля датчика Холла [19]

Датчик Холла – это прибор, который регистрирует изменение напряжённости магнитного поля. Датчики на основе эффекта Холла нашли широкое применение в быту и промышленности. Так, к примеру, они используются как:

- датчики скорости вращения – используются в автомобилестроении и везде, где требуется определить скорость вращения колеса или иного вращающегося объекта;
- датчики приближения; типичный пример – раскладной чехол на вашем смартфоне, который включает подсветку экрана при открытии;
- измерение угла поворота;
- измерение величины вибрации;
- измерение величины магнитного поля – цифровые компасы;
- измерение силы тока;
- измерение воздушных зазоров, уровня жидкости и т.д.

Модуль с датчиком Холла содержит следующие компоненты: подстроечный резистор, двухканальный компаратор, несколько согласующих резисторов, пару светодиодов и собственно, сам датчик Холла 49E.

Подстроечный резистор служит для настройки чувствительности датчика Холла. Первый светодиод сигнализирует о наличии напряжения питания на модуле, второй – о превышении магнитным полем установленного порога срабатывания.

Модуль с датчиком имеет 4 вывода. Их подключение к плате Arduino приведено на рис. 2.6.2.

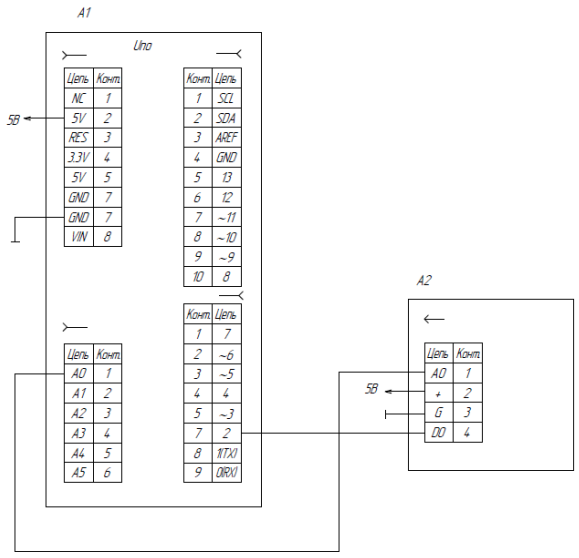


Рис. 2.6.2. Схема подключения датчика Холла

Пример программы для считывания показаний с цифрового и аналогового выходов датчика представлен ниже. В программе реализован каждые 100 мс опрос датчика и вывод значения в последовательный порт.

```
// программа для считывания показаний с цифрового и аналогового выходов датчика Холла
int analogPin = A0; // с аналогового выхода датчика Холла
int digitalPin = 2; // с цифрового выхода датчика Холла
void setup() {
  pinMode(digitalPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  int analogValue = analogRead(analogPin); // считываем аналоговое значение
  int digitalValue = digitalRead(digitalPin); // считываем цифровое значение
  Serial.println((String)digitalValue + "\t" + (String)analogValue);
  delay(100);
}
```

Мы видим два столбца с цифрами. В первом – показания цифрового канала. Если значение «0» – магнитное поле не превышает заданный порог, если «1» – превышает. Я поднёс магнит к датчику, и в нескольких строках пробежали значения «1». Порог устанавливается подстроечным резистором. А во втором столбце – значения с аналогового канала датчика. Чтобы понять, что они означают, необходимо составить таблицу соответствия, отмечая направление магнитных линий (полярность магнита) и удалённость магнита от датчика. На основании этой таблицы можно будет трактовать показания датчика Холла.

Задания для самостоятельного решения

1. Необходимо выводить уровень магнитного поля на сведодиодную линейку.

2. Необходимо организовать подсчет количества изменений магнитного поля за минуту.

Практическая работа №7
Датчик шума

Рассмотрим на этом занятии аналоговый датчик звука для Ардуино. Состоит датчик из платы, на которой смонтированы выходы, усилитель звука, подстроечный резистор и электретный микрофон, чувствительный к звуку, приходящему во всех направлениях. Регулятором чувствительности (переменным резистором) можно регулировать порог срабатывания датчика. Внешний вид датчика представлен на рис. 2.7.1 [20].

Рис. 2.7.1. Датчик звука Arduino для слежения за уровнем шума и обнаружения громких сигналов [20]

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.

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 (A0) подключается к любому аналоговому входу на Arduino Uno, а выход DO к Pin 2., как на рис. 2.7.2.

Включение света по хлопку.

Необходимо вначале отрегулировать чувствительность датчика. С помощью следующей программы. Пороговое значение для датчика необходимо

взять около 512. Чем громче звук, тем больше число на выходе. Пороговое значение регулируется с помощью потенциометра. Схема подключения датчика шума представлена на рис. 2.7.2

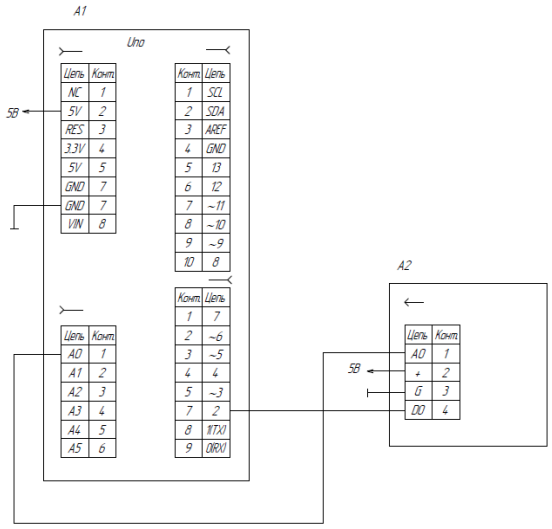


Рис. 2.7.2. Схема подключения датчика шума

```
// программа регулировки чувствительности датчика шума
int micPin = A0; // аналоговый вход A0
int digitalmicPin = 2; // цифровой вход D2
int micValue;
void setup() {
  pinMode(digitalmicPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  Serial.println(analogRead(micPin));
  micValue = digitalRead(digitalmicPin);
  if (micValue == HIGH){
    Serial.println("--------- HIGH SOUND ----------");
  delay(2);
  }
}
```

Чтобы смоделировать работу светильника, который будет включаться по хлопку в ладоши необходимо собрать электрическую схему из следующих элементов: светодиод с резистором, плата Arduino и датчик звука. Светодиод можно подключить к любому выходу, в программе мы использовали Pin 13.

```
// программа включения светодода по хлопку
boolean statuslamp; // состояние лампы: true - включено, false - выключено
```

```
int ledPin=13;
int micPin=A0;
void setup()
{
pinMode(ledPin,OUTPUT); // настройка вывода LED на выход
pinMode(micPin,INPUT); // к аналоговому входу A0 подключаем датчик
statuslamp=false; // начальное состояние – лампа выключена
Serial.begin(9600); // подключаем монитор порта
}
void loop()
{
//Serial.println (analogRead(micPin)); // выводим значение датчика на монитор

if(analogRead(micPin)>480) // регистрация хлопка на датчике звука
{
statuslamp=!statuslamp; // меняем статус лампы при регистрации хлопка
digitalWrite(ledPin,statuslamp); // переключаем светодиод на выходе 13
delay(100); // задержка, "дребезга" хлопков
}
}
```

В этой программе процесс вывода в последовательный порт занимает некоторое время, поэтому задержку можно поставить поменьше. При этом устройство работает не четко, так как время хлопка может попасть на период вывода информации в порт. Если монитор порта отключить и записать программу, приведенную ранее, работа устройства улучшается.

Если нет необходимости определять уровень звука, а нужен лишь порог срабатывания, может быть применен специализированный модуль на операционном усилителе LM393, указанный ранее. В этом случае используется цифровой выход D0. Внешний вид схемы модуля представлен на рис. 2.7.3 [20].

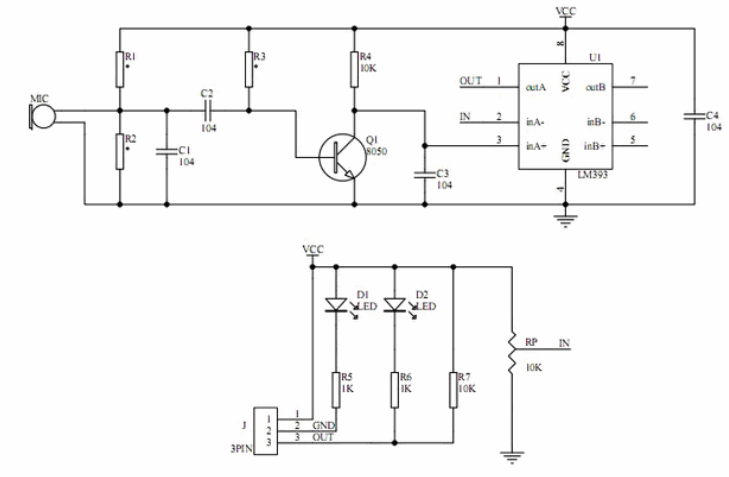


Рис. 2.7.3. Схема модуля датчика шума [20]

Задания для самостоятельного решения

1. Необходимо выводить уровень звука в виде аналоговой светодиодной линейки.

2. Сделать устройство, осуществляющее подсчет хлопков за 1 минуту.

Практическая работа №8
Фоторезистор

Датчики освещенности (освещения), построенные на базе фоторезисторов, довольно часто используются в устройствах автоматики. Они относительно просты, не дороги, их легко найти и купить. Фоторезистор позволяет контролировать уровень освещенности и реагировать на его изменение.

Фоторезистор это электронный прибор, изменяющий свое сопротивление, в зависимости от уровня света, падающего на него. Условное графическое обозначение и внешний вид представлен на рис. 2.8.1 [15].

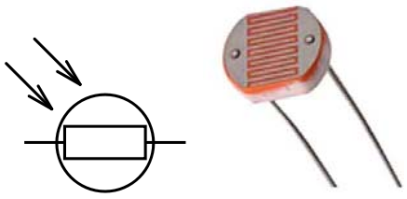


Рис. 2.8.1. Условное графическое обозначение и внешний вид фоторезистора [15]

Фоторезисторы достаточно активно применяются в самых разнообразных системах. Самый распространенный вариант применения – фонари уличного освещения.

Основным недостатком фоторезисторов является чувствительность к спектру. В зависимости от типа падающего света сопротивление может меняется на несколько порядков. К минусам также относится низкая скорость реакции на изменение освещённости. Если свет мигает – датчик не успевает отреагировать. Если же частота изменения довольно велика – резистор вообще перестанет «видеть», что освещённость меняется.

К достоинствам можно отнести простоту и доступность. Прямое изменение сопротивления в зависимости от попадающего на неё света позволяет упростить электрическую схему подключения. Сам фоторезистор очень дешев.

Схема подключения датчика освещенности к платформе Arduino довольна проста. Если используется фоторезистор, то в схеме подключения датчик реализован как делитель напряжения. Одно плечо меняется от уровня освещённости, второе — подаёт напряжение на аналоговый вход. В микросхеме контроллера это напряжение преобразуется в цифровые данные через АЦП. Т.к.

сопротивление датчика при попадании на него света уменьшается, то и значение падающего на нем напряжения будет уменьшаться. Фоторезистор смонтирован на небольшой монтажной плате внешний вид модуля и назначение выводов, а также схема подключения представлены на рис. 2.8.2.

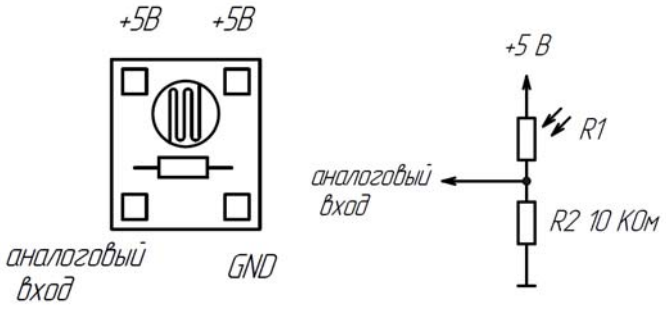


Рис. 2.8.2. Внешний вид модуля и схема подключения фоторезистора

В зависимости от того, в каком плече делителя мы поставили фоторезистор, на аналоговый вход будет подаваться или повышенное или уменьшенное напряжение. В том случае, если одна нога фоторезистора подключена к земле, то максимальное значение напряжения будет соответствовать темноте (сопротивление фоторезистора максимальное, почти все напряжение падает на нем), а минимальное – хорошему освещению (сопротивление близко к нулю, напряжение минимальное). Если мы подключим плечо фоторезистора к питанию, то поведение будет противоположным.

Несколько слов относительно дополнительного резистора на 10 кОм. У него в схеме две функции: ограничивать ток в цепи и формировать нужное напряжение в схеме с делителем. Ограничение тока нужно в ситуации, когда полностью освещенный фоторезистор резко уменьшает свое сопротивление. А формирование напряжения — для предсказуемых значений на аналоговом порту. На самом деле для нормальной работы с фоторезисторами может быть достаточно и сопротивления 1 кОм.

Меняя значение резистора можно «сдвигать» уровень чувствительности в «темную» и «светлую» сторону. Так, 10 кОм даст быстрое переключение наступления света. В случае 1 кОм датчик света будет более точно определять высокий уровень освещенности.

Если нет необходимости определять уровень освещенности, а нужен лишь порог срабатывания, может быть применен специализированный модуль на операционном усилителе LM393. Внешний вид представлен на рис. 2.8.3 [15].

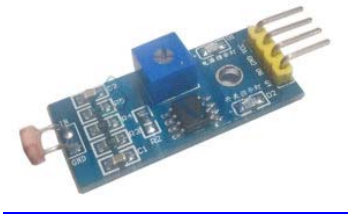


Рис. 2.8.3. Внешний вид модуля фоторезистора [15]

Технические параметры:
- напряжение питания: 3,3 В-5,5 В;
- потребляемый ток: 10 мА;
- цифрового выход: TTL (логическая «1» или логический «0»);
- аналогового выход: 0 В … Vcc;
- диаметр монтажного отверстия: 2.5 мм;
- выходной ток: 15 мА;
- габариты: 42мм х 15мм х 8мм.

Общие сведения.

Существует два модуля, визуально отличие только в количестве выводов (3 pin и 4 pin), дополнительный вывод добавлен, для снятия прямых показаний с фоторезистора (аналоговый выход).

Цифровой вывод DO, устанавливается в лог «0» или лог «1», в зависимости от яркости, чувствительность выхода, можно регулировать с помощью встроенного потенциометра. Выходной ток цифрового выхода способен выдать более 15 мА, что очень упрощает использования модуля и дает возможность использовать его минуя, контроллер Arduino и подключая его напрямую ко входу однокональному реле или одному из входов двухконального реле. Принципиальную схему модуля освещенности на LM393 с 3 pin и 4 pin, показана на рис. 2.8.4 [15].

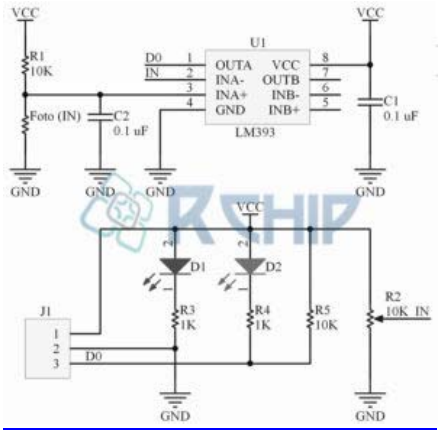


Рис. 2.8.4. Принципиальная схема модуля освещенности [15]

Программа для определения уровней освещенности с выдачей их в последовательный порт следующая:

```
// Определение уровня освещенности
int PIN_LED=10;
int PIN_PHOTO_SENSOR=A0;
 void setup() {
  Serial.begin(9600);
  pinMode(PIN_LED, OUTPUT);}
 void loop() {
  int val = analogRead(PIN_PHOTO_SENSOR);
  Serial.println(val);
  if (val < 300) {
    digitalWrite(PIN_LED, LOW);
  } else {
    digitalWrite(PIN_LED, HIGH);
  }
}
```

Следующая программа включает поочередно 8 светодиодов, в зависимости от уровня освещенности. Схема подключения представлена на рис. 2.8.4.

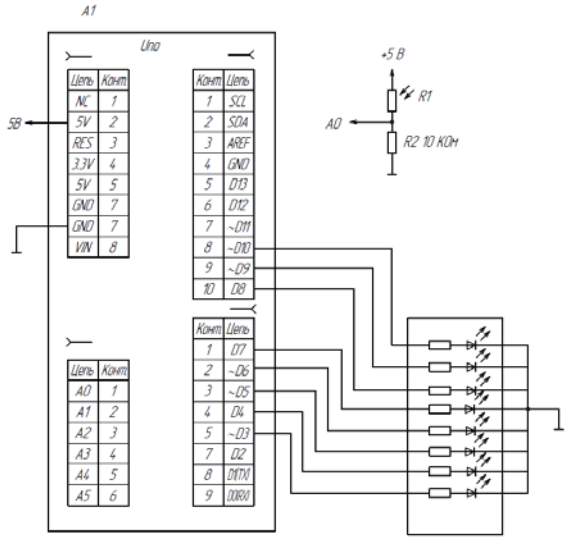


Рис. 2.8.4. Схема подключения фоторезистора и светодиодной линейки

```
// Светодиодная шкала уровень освещенности
// Контакт подключения светодиодов
int leds[8]={3,4,5,6,7,8,9,10};
int PIN_PHOTO_SENSOR=A0; // Контакт A0 для входа фоторезистора
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(PIN_PHOTO_SENSOR); // Чтение показаний фоторезистора
// Применение функции map()
val = map(val, MIN_LIGHT, MAX_LIGHT, 0, 8);
// ограничиваем, чтобы не превысило границ
val = constrain(val, 0, 8);
// зажечь кол-во светодиодов, пропорциональное освещенности
// остальные потушить
for(int i=0;i<8;i++){
if(i<=val) // зажечь светодиоды
digitalWrite(leds[i],HIGH);
else // потушить светодиоды
digitalWrite(leds[i],LOW);}
delay(10); // пауза перед следующим измерением
}
```

Задания для самостоятельного решения

1. Модифицировать текст программы и схему так, чтобы в зависимости от уровня освещенности менялась яркость светодиода.

2. Подключить ЖК индикатор и выводить значение освещенности на этот индикатор в числовом виде и в виде аналоговой шкалы.

Практическая работа №9
Электродвигатель

Двигатели есть в каждом механизированном устройстве. В одних устройствах они приводят в действие колеса, заставляя машину перемещаться в нужном направлении. В других – двигатели крутят пропеллеры, создавая вертикальную тягу для полета. Двигатели позволяют вращаться суставам промышленного робота-манипулятора, и перемещают каретку 3D-принтера.

Существует множество типов двигателей. К самым распространенным в электронике можно отнести двигатели постоянного тока, шаговый двигатель, и бесколлекторный двигатель. У каждого типа есть свои особенности, плюсы и минусы. Одни больше подходят для точных перемещений, другие позволяют легко поднять в небо мультикоптер. Под каждый проект нужно тщательно выбирать нужный тип двигателей.

Малогабаритный двигатель постоянного тока представлен на рис. 2.9.1 [15].

Рис. 2.9.1. Внешний вид двигателя постоянного тока [15]

Потребляемый ток такого двигателя от 200 мА до 1 А, что выше чем максимальный ток портов микроконтроллера, поэтому его напрямую подключать нельзя. Кроме того, большинство мощных двигателей требует напряжение более 5 В. Распространены двигатели на 12, на 24 и на 48 Вольт. Для включения двигателя требуется транзистор. Ниже представлена схема управления двигателем при помощи биполярного NPN и PNP транзистора (рис. 2.9.2).

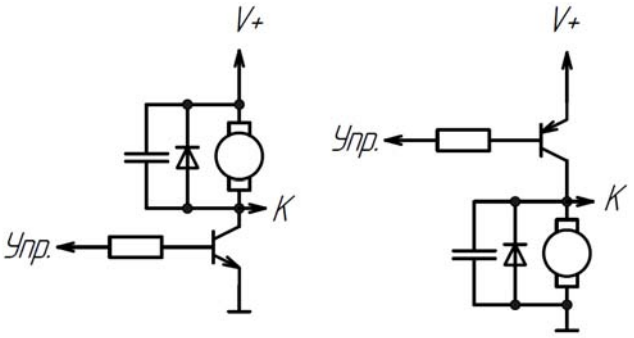


Рис. 2.9.2. Схема управления двигателем при помощи биполярного NPN и PNP транзистора

Как видно, схема очень простая. На базу транзистора подается слабый сигнал от Arduino через резистор 1кОм, вследствие чего транзистор открывает мощный канал, по которому ток проходит от плюса к минусу через двигатель. В цепи обязательно нужно поставить защитный диод, например 1N4001 или 1N4007. Этот диод не даст выйти из строя транзистору и контроллеру в момент остановки двигателя, когда ЭДС самоиндукции создаст на обмотках скачок напряжения.

В этой схеме можем использовать, например, NPN транзистор 2N2222A или BC337. Этот биполярный транзистор может управлять током до 1А и напряжением до 45 В, так что его можно вполне использовать для небольших моторов.

С помощью одного транзистора можно включать и выключать двигатель постоянного тока в одном направлении. Для управления направлением движе-

ния двигателя может использоваться Н-мост на биполярных транзисторах, схема которого представлена на рис. 2.9.3

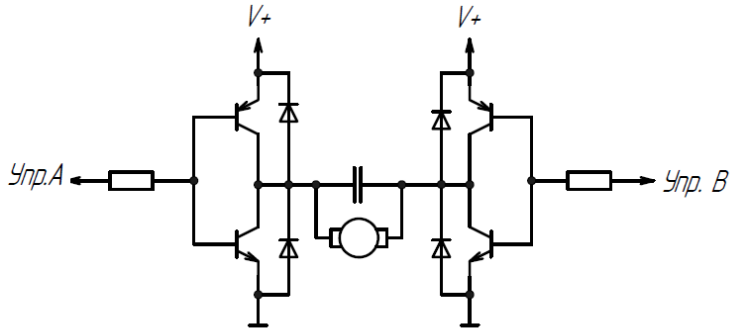


Рис. 2.9.3. Схема Н моста на биполярных транзисторах

Упр.А и Упр.В на рис. 2.9.3 – это вход слабых управляющих сигналов. В случае Arduino, на них необходимо подавать либо 0 (земля) либо +5В. V+ – это питание двигателей, оно может быть во много раз выше напряжения управляющего сигнала.

В зависимости от того, на какой из входов мы подаем положительный сигнал, двигатель будет крутиться в одну или в другу сторону. Как правило, в схему драйвера двигателя постоянного тока помимо самого Н-моста, добавляют защитные диоды, фильтры, опторазвязки и прочие улучшения.

Схема модуля транзистора с назначением выводов представлена на рис 2.9.4.

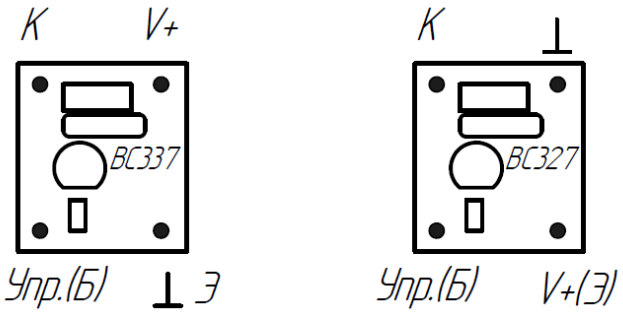


Рис. 2.9.4. Схема модуля транзистора с назначением выводов

Для управления сразу двумя электродвигателями может использоваться Н мост на микросхеме MX1508, имеющей МОП ключи с низким сопротивлением, минимальное тепловыделение, позволяющее использовать драйвер без радиаторов теплоотвода, маленький размер, низкое энергопотребление, благодаря чему он идеально подходит для работы в портативных устройствах при питании от аккумулятора. Внешний вид моста представлен на рис. 2.9.5 [15].

Рис. 2.9.5. Внешний вид модуля H моста на микросхеме MX1508 [15]

Двойной H-мостовой драйвер мотора, может управлять двумя двигателями постоянного тока или 4-проводным двухфазным (биполярным) шаговым двигателем.

Напряжение питания модуля 2-10 В.

Входной сигнал напряжение 1.8-7 В.

Рабочий ток 1.5А на канал, пиковый ток до 2.5А, низкий ток в режиме ожидания (менее 0,1 мкА).

Необходимо подключить электродвигатель к платформе к выводу №11. Программа имеет следующий вид:

```
// Регулировка скорости вращения электродвигателя
int Motor = 11;         //  Номер Pin к которому подключен электродвигатель
int Speed =0;    //  Переменная в которой хранится уровень скорости (От 0 до 254)
void setup()  {
  pinMode(Motor, OUTPUT); // Порт 11 (Motor) будет работать как Выход.
}
void loop()  { // Этот цикл будет выполняться бесконечное количество раз.
  for (int i=0; i<=254; i+=15){
Speed=i;
analogWrite(Motor, Speed);    // Устанавливаем состояние скорости для электродвигателя
  delay(50); // Пауза 50 миллисекунд.
}
 for (int i=254; i>=0; i-=15){
 Speed=i;
 analogWrite(Motor, Speed);    // Устанавливаем состояние скорости для электродвигателя
  delay(50); // Пауза 50 миллисекунд.
 }
}
```

Задания для самостоятельного решения

1. Необходимо осуществить управление скоростью вращения электродвигателя от кнопок или переменного резистора.

2. Подключить двигатель через Н мост и осуществить реверсивное управление электродвигателем.

Практическая работа №10
Дисплей на базе драйвера TM1637

Часто проекты требуют вывода данных с различных датчиков и устройств в монитор порта или же на дисплей, например, для вывода данных с датчика температуры. Для этих целей можно использовать светодиодные дисплеи. С их помощью можно выводить данные в удобном формате за счет большего размера цифр. Рассмотрим модуль семисегментных индикаторов на четыре знакоместа на базе драйвера TM1637, который позволяет управлять всем модулем всего по двум проводам. Данная микросхема разработана китайской электронной промышленностью и была предназначена для управления отображением времени и счетчика в видеоплеерах, спутниковых ресиверах и тому подобных устройств. Внешний вид модуля представлен на следующем рис. 2.10.1 [21].

Рис. 2.10.1. Внешний вид модуля семисегментных индикаторов на базе драйвера TM1637 [21]

Для работы с этим модулем используется библиотека TM1637.h.

В целом эта библиотека для Arduino позволяет:

- выводить шестнадцатеричный знак в любое знакоместо — функция Display(x,y) где x – номер разряда от 0 до 3, y – знак 0..9, A,b,C,d,E,F;
- выводить сразу группу из 4-х шестнадцатеричных знаков- функция Display(x[]) где x[] – массив из 4 элементов, по числу знаков;
- выводить десятичное целое число – функция Display(x) где x – целое от -999 до 9999;
- выводить десятичное число с плавающей точкой – функция Display(x) где x – число с плавающей точкой от -999 до 9999;
- очищать дисплей – функция clearDisplay(x) – дело в том, что информация на индикаторе сохраняется до тех пока не запишешь новую;
- устанавливать яркость индикатора функция set(x) где x – яркость от 0 до 7, по умолчанию яркость равна 2.

Вместо цифры можно написать следующие параметры:

- BRIGHT_TYPICAL = 2 Средний;
- BRIGHT_DARKEST = 0 Тёмный;
- BRIGHTEST = 7 Яркий.

Модуль, подключаются к платам Arduino по двум проводам. Подключать модуль можно к любым выводам на плате Arduino, кроме тех что являются аналоговыми (помеченные как A0..A7).

Рассмотрим несколько программ. Вот эта программа показывает числа от 1-го до 4-х поразрядно, и включение/выключение часового разделителя через каждую секунду

```
#include "TM1637.h"
int8_t DispMSG[] = {1, 2, 3, 4};

//Определяем пины для подключения к плате Arduino
#define CLK 3
#define DIO 2
//Создаём объект класса TM1637, в качестве параметров
//передаём номера пинов подключения
TM1637 tm1637(CLK, DIO);
void setup()
{
//Инициализация модуля
tm1637.init();
//Установка яркости горения сегментов
/*
* BRIGHT_TYPICAL = 2 Средний
* BRIGHT_DARKEST = 0 Тёмный
* BRIGHTEST = 7      Яркий
*/
tm1637.set(BRIGHT_TYPICAL);
}

void loop()
{
//Задание на включение разделителя
tm1637.point(true);
//Выводим массив на дисплей
tm1637.display(DispMSG);
//Задержка
delay(1000);
//Задание на выключение разделителя
tm1637.point(false);
//Выводим массив на дисплей
```

```
tm1637.display(DispMSG);
//Задержка
delay(1000);
}
```

В самом начале скетча, объявляется переменная DispMSG[], представляющая собой массив из 4-х элементов типа uint8_t. Функция display() класса tm1637 выводит каждый элемент массива DispMSG[] на дисплей, при чем вывод значений осуществляется поразрядно начиная с крайнего левого и заканчивая крайним правым разрядом. Чтобы было легче понять как работает функция display(), можно изменить код в функции loop() на следующий:

```
void loop()
{
//Задание на включение разделителя
tm1637.point(true);
//Выводим массива на дисплей порязрядно
tm1637.display(0, DispMSG[0]);
tm1637.display(1, DispMSG[1]);
tm1637.display(2, DispMSG[2]);
tm1637.display(3, DispMSG[3]);
//Задержка
delay(1000);
//Задание на выключение разделителя
tm1637.point(false);
//Выводим массива на дисплей порязрядно
tm1637.display(0, DispMSG[0]);
tm1637.display(1, DispMSG[1]);
tm1637.display(2, DispMSG[2]);
tm1637.display(3, DispMSG[3]);
//Задержка
delay(1000);
}
```

Как видно из листинга программы, такая запись кода программы делает его более доступным для понимания, но в то же время код сильно разрастается. В библиотеке TM1637.h. есть файл TM1637.CPP, в котором содержится массив символов цифр. При этом номер элемента в массиве совпадает с цифрой. Можно также генерировать буквы. Модифицировав библиотеку TM1637.h. можно расширить количество отображаемых символов. Все символы отображены на рис. 2.10.2 [21].

	Символ	Цифра	Формат записи	Индекс элемента
Уже существующие элементы массива	O	0	0x3F	0
	I	1	0x06	1
		2	0x5B	2
		3	0x4F	3
		4	0x66	4
	S	5	0x6D	5
		6	0x7D	6
		7	0x07	7
		8	0x7F	8
		9	0x6F	9
	A		0x77	10
	b		0x7C	11
	C		0x39	12
	d		0x5E	13
	E		0x79	14
	F		0x71	15

Добавленные в массив пользовательские символы	G		0x3d	16
	H		0x76	17
	h		0x74	18
	i		0x04	19
	J		0x1E	20
	L		0x38	21
	l		0x18	22
	M		0x37	23
	n		0x54	24
	P		0x73	25
	r		0x50	26
	t		0x78	27
	U		0x3E	28
	u		0x1C	29
	Y		0x6E	30

Рис. 2.10.2. Коды символов драйвера TM1637 [21]

31 элементом массива добавлен код гашения сегментов в индикаторе B00000000, что соответствует 0x0 в шестнадцатеричной форме.

Задания для самостоятельного решения

1. Вывести фразы «LEGO», «Err».
2. Необходимо осуществить вывод на индикатор оцифрованные значения освещенности с фоторезистора.
3. Необходимо обеспечить попеременный вывод информации двух типов- с мигающим двоеточием и с постоянно горящим двоеточием. Данные выводятся длительностью по 5 секунд.

Практическая работа №11
Светодиодная матрица с драйвером MAX7219

Светодиодная матрица – это графический индикатор, который можно использовать для вывода простых изображений, букв и цифр. Более подробно о светодиодной матрице можно узнать в лабораторной работе №5. Электронной промышленностью выпускается микросхема MAX7219, представляющая собой контроллер семисегментных индикаторов и матриц.

Модуль светодиодной матрицы с микросхемой MAX7219. Модуль представляет из себя плату с микросхемой, необходимой для неё обвязкой и, собственно, матричным индикатором. Обычно индикатор не впаивают в плату, а вставляют в разъем. Это сделано для того, чтобы группу модулей можно было сначала закрепить на какой то поверхности винтами, а затем вставить в них матрицы. Внешний вид модуля представлен на рис. 2.11.1 [15].

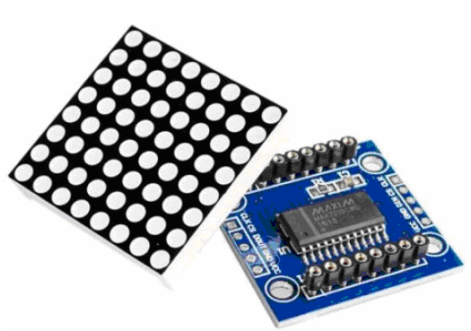


Рис. 2.11.1. Внешний вид модуля светодиодной матрицы
с драйвером MAX7219 [15]

У модуля есть пять выводов на каждой стороне. С одной стороны данные входят в модуль, с другой стороны данные выходят из модуля и передаются в следующий. Это позволяет соединять матрицы у цепочку.

Входной разъем:
- VCC, GND — питание;
- DIN — вход данных;
- CS — выбор модуля (chip select);
- CLK — синхроимпульс.

Выходной разъем:
- VCC, GND — питание;
- DOUT — выход данных;
- CS — выбор модуля (chip select);
- CLK — синхроимпульс.

Работает модуль от напряжения 5 В.

Подключение

Подключается матричный модуль к контроллеру Arduino по следующей табл. 2.11.1

Таблица 2.11.1

Данные для подключения светодиодной матрицы

Светодиодная матрица 8×8 с MAX7219	VCC	GND	CIN	CS	CLK
Ардуино Уно	+5V	GND	11	9	13

Для управления микросхемой MAX7219 воспользуемся библиотекой Max72xxPanel. Установим библиотеку и напишем небольшую программу, которая будет выводить на дисплей всего одну точку с координатами x=3 и y=4. Точка будет мигать с периодом 600 миллисекунд.

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

```
#include <Max72xxPanel.h>

int pinCS = 9;
int numberOfHorizontalDisplays = 1; // количество матриц по-горизонтали
int numberOfVerticalDisplays = 1; // количество матриц по-вертикали

Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontalDisplays, num-
berOfVerticalDisplays);

void setup() {
  matrix.setIntensity(4); // яркость от 0 до 15
}

void loop() {
  matrix.drawPixel(3, 4, HIGH); // зажигаем пиксель с координатами {3,4}
  matrix.write(); // вывод всех пикселей на матрицу
  delay(300);
  matrix.drawPixel(3, 4, LOW); // гасим пиксель
  matrix.write();
  delay(300);
}
```

Как уже говорилось ранее, матричные модули с микросхемой MAX7219 можно легко объединять. Именно для этой цели в начале программы мы задаем количество матриц по горизонтали и по вертикали. В данном случае используется одна матрица, так что оба этих параметра будут равны 1.

Важно отметить, что после включения и выключения пикселей с помощью функции drawPixel, необходимо вызвать функцию write. Без функции write пиксели не высветятся на матрице!

Теперь напишем программу, которая отобразит на матрице смайл. Смайл зашифруем с помощью массива из восьми байт. Каждый байт массива будет отвечать за строку матрицы, а каждый бит в байте за точку в строке.

В библиотеке Max72xxPanel есть функция setRotation, которая задает ориентацию изображения на матрице. Например, если мы захотим повернуть смайл на 90 градусов, нужно будет сразу после вызова функции setIntensity вызвать setRotation с соответствующими аргументами:

```
matrix.setRotation( 0, 1 );
```

Первый параметр – это индекс матрицы, в нашем случае он равен нулю; второй параметр – количество поворотов на 90 градусов.

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>
```

```
int pinCS = 9;
int numberOfHorizontalDisplays = 1; // количество матриц по-горизонтали
int numberOfVerticalDisplays = 1; // количество матриц по-вертикали
Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontalDisplays, numberOfVerticalDisplays);

const byte data[8] = {
  0b00111100,
  0b01000010,
  0b10100101,
  0b10000001,
  0b10100101,
  0b10011001,
  0b01000010,
  0b00111100
};
void setup() {
  matrix.setIntensity(7); // яркость от 0 до 15
   matrix.setRotation( 0, 1 );
  matrix.fillScreen(LOW); // очистка матрицы
  for ( int y = 0; y < 8; y++ ) {
    for ( int x = 0; x < 8; x++ ) {
      // зажигаем x-й пиксель в y-й строке
      matrix.drawPixel(x, y, data[y] & (B00000001<<x));
    }
  }
  matrix.write(); // вывод всех пикселей на матрицу
}
void loop() {
}
```

Вывод текста с помощью библиотеки Adafruit-GFX-Library.

Подобным же образом можно выводить на матрицу и любой другой символ, например, букву. Но чтобы иметь возможность отображать любую букву английского алфавита, нам необходимо будет определить в программе целых 26 восьмибайтных массива.

В популярной библиотеке Adafruit-GFX-Library помимо функций для работы с графикой и текстом, имеется и база латинских букв в верхнем и нижнем регистрах, а также все знаки препинания и прочие служебные символы. Ссылка на библиотеку есть в конце урока.

Отобразить символ на матрице можно с помощью функции drawChar.

drawChar(x, y, символ, цвет, фон, размер);

Первые два параметра функции отвечают за координаты верхнего левого угла символа. Третий параметр – это сам символ. Цвет символа в нашем случае будет равен 1 или HIGH, так как матрица двухцветная. Фон равен 0 или LOW. Последний параметр «размер» сделаем равным 1.

Напишем программу, которая будет по очереди выводить на матрицу все буквы фразы: «HELLO WORLD!».

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 9;
int numberOfHorizontalDisplays = 1;
int numberOfVerticalDisplays = 1;

Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontalDisplays, numberOfVerticalDisplays);

String tape = "HELLO WORLD!";
int wait = 500;

void setup() {
  matrix.setIntensity(7); // яркость от 0 до 15
}

void loop() {
  for ( int i = 0 ; i < tape.length(); i++ ) {
    matrix.fillScreen(LOW);
    matrix.drawChar(0, 0, tape[i], HIGH, LOW, 1);
    matrix.write();
  delay(wait);
  }
 }
```

В библиотеке Adafruit_GFX имеется множество функций для работы с графикой. Например, drawCircle(3, 3, 2, HIGH) начертит окружность с центром {3,3} и радиусом 2. Последний параметр – цвет, но в случае монохромной матрицы он равен 1 или HIGH. Функция drawLine(0, 0, 3, 6, HIGH) начертит отрезок между точками {0,0} и {3,6}.

Задания для самостоятельного решения

1. Сделать на матрице сменяющуюся картинку в виде двух смайликов по 3 секунды каждая.
2. Сделать так, чтобы смайлик двигался за пределы матрицы.
3. Сделать бегущую змейку, двигающуюся по экрану.

Практическая работа №12
Шаговый двигатель 28BYJ-48

На данном занятии пойдет речь о четырехфазном шаговом двигателе 28BYJ-48, работающим от постоянного напряжения 5В (существует модификация на 12В). Так как двигатель потребляет значительный ток, то нет возможности подключить его напрямую к выводам Arduino UNO, для этого используется модуль драйвера шагового двигателя, основанном на микросхеме ULN2003. Внешний вид двигателя с контроллером представлен на рис. 2.12.1 [15].

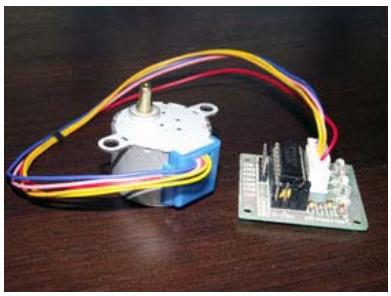


Рис. 2.12.1. Внешний вид шагового двигателя 28BYJ-48
с драйвером на ULN2003 [15]

Технические параметры двигателя 28BYJ-48 следующие:
- модель 28BYJ-48;
- тип шагового двигателя: униполярный;
- напряжение питания: 5 В, постоянный ток;
- количество фазы: 4;
- частота: 100 Гц;
- сопротивление постоянного тока: 50 Ом ± 7%.

Четырехфазный шаговый двигатель (28BYJ-48) – это бесколлекторный двигатель, вращение вала осуществляется шагами (дискретное перемещение). На роторе (валу), расположен магнит, а вокруг него расположены катушки, если поочередно подавать ток на эти катушки, создается магнитное поле, которое отталкивает или притягивает магнитный вал, тем самым заставляя двигатель вращаться. Такая конструкция позволяет с большой точностью управлять валом, относительно катушек. Принципиальная схема четырехфазного шагового двигателя 28BYJ-48 приведена на рис. 2.12.2 [15].

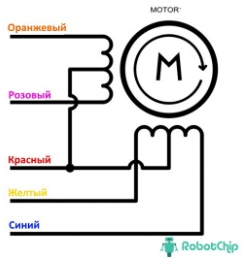


Рис. 2.12.2. Принципиальная схема четырехфазного шагового двигателя 28BYJ-48 [15]

Из схемы видно, что в двигателе содержится две обмотки, которые в свою очередь разделены на четыре, из-за этого и название четырехфазный. Центральные отводы катушек подключены вместе и служат для питания двигателя, так как каждая обмотка подключена к питанию, такие двигатели называют униполярный. На валу 28BYJ-48 расположено 8 магнитов, с чередующими полюсами (то есть, четыре магнита с двумя полюсами). Двигатель в разобранном состоянии представлен на рис. 2.12.3 [15].

Рис. 2.12.3. Двигатель 28BYJ-48 в разобранном состоянии [15]

Из рисунка видно, что внутри расположен редуктор, с примерным передаточным числом в 1:64, если быть точнее 1:63,68395. Это означает, что двигатель за один оборот осуществляет 4075.7728395 шага. Данный двигатель поддерживает полушаговый режим и за один полный оборот может совершать 4076 шага, а точнее за 1 делает примерно 11,32 шага. (4076 / 360 = 11,32).

Шаговые двигатели бывают униполярные и биполярные. В первом случае электромагнитные катушки двигателя соединены таким образом, что мы можем генерировать на них поле только одного направления. Схема работы униполярного двигателя выглядит следующим образом (рис. 2.12.4) [15].

На схеме изображен так называемый волновой режим работы. На каждом шаге работы двигателя мы подаем напряжение только на одну катушку, в которой возникает магнитное поле. Часть сердечника, ближайшая к ротору начинает действовать как южный полюс магнита, вследствие чего ротор поворачивается к ней своим северным полюсом.

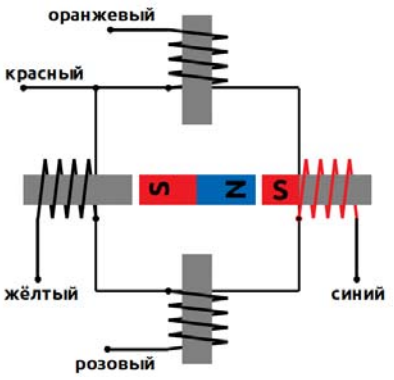


Рис. 2.12.4. Схема работы униполярного двигателя [15]

Полношаговый режим. Очевидно, что если мы будем подключать одновременно сразу две соседние катушки, то магнитное поле, действующее на ротор будет сильнее, тем самым повысится и крутящий момент двигателя. Такой режим работы униполярного двигателя называется полношаговым (рис. 2.12.5 [15]).

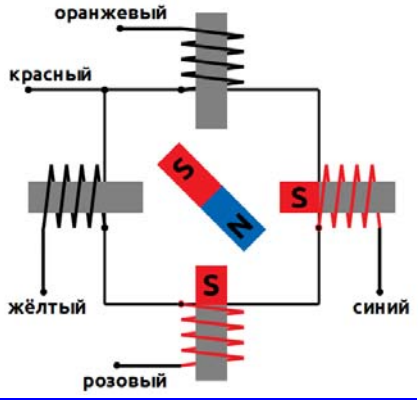


Рис. 2.12.5. Схема работы полношагового режима [15]

Полушаговый режим. Наконец, мы можем комбинировать волновой и полношаговый режим, получив полушаговый режимработы. В таком режиме за один оборот ротора двигатель делает в два раза больше шагов, тем самым, увеличивая точность позиционирования (рис. 2.12.6 [15]). Однако, в таком режиме двигатель каждый второй шаг имеет сниженный крутящий момент, о чём не стоит забывать.

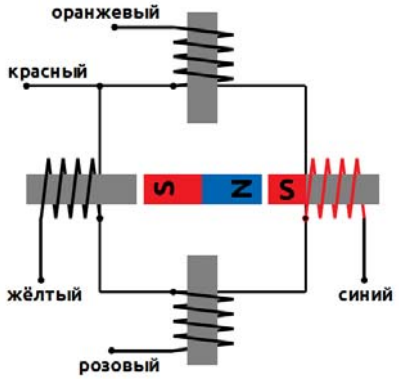


Рис. 2.12.6. Схема работы полушагового режима [15]

Надо заметить, что в реальном шаговом двигателе ротор, как правило имеет больше 2-х полюсов, а катушки могут иметь различную конфигурацию. Например, в нашем моторе 28BYJ-48 электромагниты расположены не перпендикулярно оси ротора, а вдоль него.

Чаще всего, при использовании шагового двигателя 28BYJ-48, используют два режима подключения:

- полношаговый режим – за 1 такт, ротор делает 1 шаг;
- полушаговый режим – за 1 такт, ротор делает ½ шага.

Ниже на рис. 2.12.7 [15] переставлена таблица последовательности тактов.

Контакт мотора	Полушаговой режим							
	1	2	3	4	5	6	7	8
4 - Оранжевый	1	1	0	0	0	0	0	1
3 - Желтый	0	1	1	1	0	0	0	0
2 - Розовый	0	0	0	1	1	1	0	0
1 - Синий	0	0	0	0	0	1	1	1

Контакт мотора	Полношаговый режим			
	1	2	3	4
4 - Оранжевый	1	1	0	0
3 - Желтый	0	1	1	0
2 - Розовый	0	0	1	1
1 - Синий	1	0	0	1

Рис. 2.12.7. Последовательности тактов шагового двигателя [15]

Модуль шагового двигателя ULN2003: цифровой вывод микроконтроллера может выдать ток ~40 мА, а одна обмотка 28BYJ-48 в пике потребляем ~320 мА, следовательно если подключить двигатель напрямую, микроконтроллер сгорит. Для защиты был разработан «Модуль шагового двигателя ULN2003», в котором используется микросхема ULN2003A (по сути, состоящая из 7 ключей), позволяющая управлять нагрузкой до 500 мА (один ключ). Данный модуль может работать с 5 В и 12 В двигателем 28BYJ-48, для переключения необходимо установить или убрать перемычку (по умолчанию перемычка установлена, питание 5 В). Внешний вид модуля со схемой назначения выводов и принципиальной схемой представлен на рис. 2.12.8 [15].

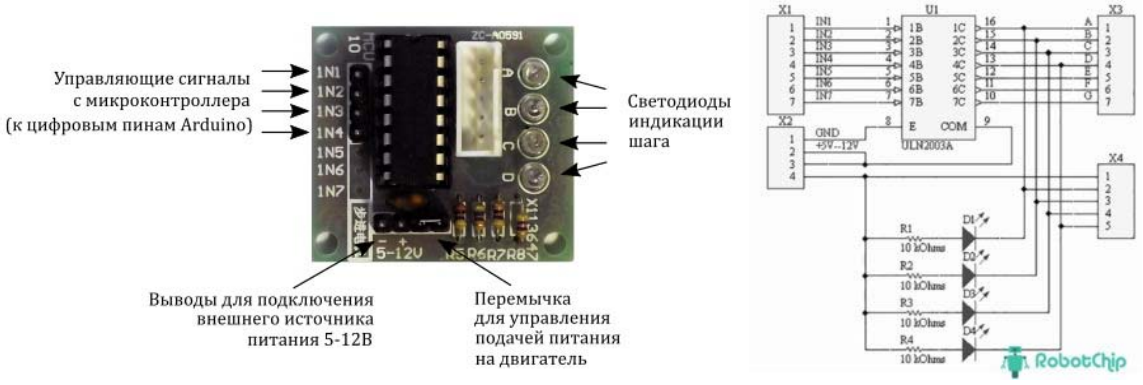


Рис. 2.12.7. Внешний вид модуля со схемой назначения выводов и принципиальной схемой [15]

Назначение X1:
- IN1 . . . IN7: Вход 1 … 7
Назначение X2:
- 1 – GND: «-» питание модуля
- 2 – Vcc: «+» питание модуля (5В или 12В)
- 3 – Vcc: «+» питание модуля (перемычка, только при 5В)
- 4 – Vcc: «+» питание модуля (перемычка, только при 5В)
Назначение X3:
- A. . . G: Выход 1 … 7
Назначение X3
- 1 – Питание;
- 2 – A;
- 3 – B;
- 4 – C;
- 5 – D.

Задание. Необходимо подключить шаговый двигатель к платформе по схеме IN1-ШТ4: 8-11 выводы. Загрузить следующую программу.

```
#include <Stepper_28BYJ.h>
// изменить количество шагов для вашего мотора
#define STEPS 4078
Stepper_28BYJ stepper(STEPS, 8, 9, 10, 11);
void setup()
{
  // установим скорость вращения 3 об/мин
  stepper.setSpeed(14);
}
void loop()
{
  stepper.step(4076); // Делаем 4076 шагов в одну строну (полный оборот)
   stepper.step(-4076);// Делаем 4076 шагов в другую строну
}
```

Задания для самостоятельного решения

1. Сделать непрерывное вращение двигателя.
2. Сделать управление направлением вращения от двух кнопок.
3. Сделать регулировку оборотов от переменного резистора.

Практическая работа №13
Сервопривод

Сервопривод (сервомотор) является важным элементом при конструировании различных роботов и механизмов. Это точный исполнитель, который имеет обратную связь, позволяющую точно управлять движениями механизмов. Другими словами, получая на входе значение управляющего сигнала, сервомотор стремится поддерживать это значение на выходе своего исполнительного элемента. Схема устройства сервопривода представлена на рис. 2.13.1 [20].

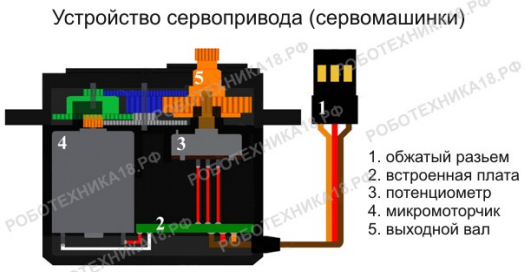


Рис. 2.13.1. Схема устройства сервопривода [20]

Сервоприводы широко используются для моделирования механических движений роботов. Сервопривод состоит из датчика (скорости, положения и т.п.), блока управления приводом из механической системы и электронной схемы. Редукторы (шестерни) устройства выполняют из металла, карбона или пластика. Пластиковые шестерни сервомотора не выдерживают сильные нагрузки и удары.

Сервомотор имеет встроенный потенциометр, который соединен с выходным валом. Поворотом вала, сервопривод меняет значение напряжения на потенциометре. Плата анализирует напряжение входного сигнала и сравнивает его с напряжением на потенциометре, исходя из полученной разницы, мотор будет вращаться до тех пор, пока не выровняет напряжение на выходе и на потенциометре.

Управление сервоприводом с помощью широтно импульсной модуляции.

Для управления сервоприводом написана библиотека Servo.h. Вот основные команды библиотеки.

Servo.attach()

Подключает Servo к указанному выходу, с которого осуществляется управление приводом. На ранних версиях Arduino-0016 и более ранних, библиотека Servo поддерживала управления только через порты 9 и 10.

Синтаксис

servo.attach(pin)

servo.attach(pin, min, max)

Параметры

servo: переменная типа Servo

pin: номер выхода, к которому подключаем servo и с которого осуществляется управление приводом

min (опциональный): ширина импульса в микросекундах, соответствующий минимальному (угол 0 градусов) положению сервопривода (по умолчанию 544);

max (optional): ширина импульса в микросекундах, соответствующий максимальному (угол 180 градусов) положению сервопривода.

Servo.write()

Передает значения для управления приводом. Для стандартного сервопривода это угол поворота. Для привод постоянного вращения, функция задает скорость вращения (0 – для максимальной скорости вращения в одну сторону, 180 – для максимальной скорости в другую сторону и около 90 для неподвижного состояния).

Синтаксис

servo.write(angle)

Параметры

servo: переменная типа Servo

angle: значение записываемое в servo, от 0 до 180

Servo.writeMicroseconds()

Передает значение для управления сревоприводом в микросекундах (uS), устанавливая угол поворота на это значение. Для стандартного привода значение 1000 максимальный поворот против часовой стрелки, 2000 максимальный поворот по часовой стрелке, 1500 посередине.

Замечание: некоторые производители не придерживаются стандартных значение и такие приводы могут управляться значениями от 700 до 2300. Поэкспериментируйте со значениями, до момента пока привод не повернется и остановится в крайнем положение. Тем не менее следует избегать постоянного использования привода на значениях больше допустимых.

Приводы постоянного вращения реагируют на данную команду подобно реакции на функцию write().

Синтаксис

servo.writeMicroseconds(uS)

Параметры

servo: переменная типа Servo

uS: значение в микросекундах (int)

Servo.read()

Считывает значение текущего положения сервопривода (значение записанное последним вызовом функции write()).

Синтаксис

servo.read()

Параметры

servo: a variable of type Servo

Возвращаемое значение

Положение (угол) сервопривода от 0 до 180.

Servo.attached()

Проверяет, если переменная Servo подключена к выходу.

Синтаксис

servo.attached()

Параметры

servo: переменная типа Servo

Возвращаемое значение

true, если подключена; false – в противном случае

Servo.detach()

Отсоединяет переменную Servo от указанного выхода. Если все Servo переменные отсоединены, то выходы 9 и 10 могут быть использованы в режиме ШИМ с помощью analogWrite().

Синтаксис

servo.detach()

Параметры

servo: переменная типа Servo

Подключается сервопривод следующим образом. У сервопривода SG90 имеется три контакта:

- коричневый – земля;
- красный – питание +5 В;
- оранжевый (или желтый) – сигнальный.

Задание 1. Необходимо записать следующую программу.

```
// Подключение сервопривода
#include <Servo.h> // подключаем библиотеку для работы с сервоприводом
Servo servo1; // объявляем переменную servo типа «servo1»
void setup() // процедура setup
{
servo1.attach(11); // привязываем сервопривод к аналоговому выходу 11
}
void loop() // процедура loop
{
servo1.write(0); // ставим угол поворота под 0
delay(2000); // ждем 2 секунды
servo1.write(90); // ставим угол поворота под 90
delay(2000); // ждем 2 секунды
servo1.write(180); // ставим угол поворота под 180
delay(2000); // ждем 2 секунды
}
```

Задания для самостоятельного решения

1. Сделать управление углом поворота от данных, поступающих с переменного резистора.
2. Сделать указатель уровня освещенности с помощью сервопривода.
3. Сделать указатель температуры термопары с помощью сервопривода.

Практическая работа №14
Светодиоды с пиксельной адресацией WS2812B

На этом занятии рассмотрим адресную RGB – светодиодную ленту WS2812B. Лента основана на светодиодах WS2812B в корпусе LED 5050, куда в корпус производители поместили не только три встроенных светодиода (Красный, Зеленый, Синий), но и управляемый ШИМ драйвер, управляющий их яркостью. Благодаря этому мы можем получить произвольный цвет, изменяя яркость встроенных светодиодов, а так же управлять отдельно взятым пикселем на ленте. Собственно, три встроенных разноцветных светодиода вместе с ШИМ драйвером и образуют светодиод WS2812B.

Эти светодиоды с драйвером образуют пиксель WS2812B, внешний вид которого представлен на рис. 2.14.1 [21].

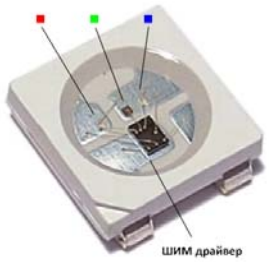


Рис. 2.14.1. Светодиод с пиксельной адресацией WS2812B [21]

Технические характеристики.

Светодиод WS2812B работает от напряжения 5 В (±0.5).

Ток ~20 мА на один встроенный светодиод, то есть ~60 мА на пиксель в целом.

Рабочая температура от -20 до +80 %.

Подключается светодиодная лента довольно-таки просто, необходимо подать на +5 В и GND, плюс (+) и минус (-) от 5 В блока питания, а контакт DIN соединить с портом микроконтроллера, как правило, по умолчанию используется 6-й порт Arduino, но вы вправе выбрать и любой другой свободный порт. Так же рекомендуется соединить земли Arduino и блока питания, как нарисовано на рисунке ниже.

Будьте внимательны, лента на светодиодах WS2812B имеет направление, с одной стороны она имеет контакты DIN (вход), +5V, GND, а с другой стороны DO (выход), +5V, GND, подключать необходимо именно вход, то есть DIN,

иначе лента не будет работать. Так же на ленте нарисованы стрелки, указывающие на направление.

Протокол управления. Каждый светодиод WS2812B имеет один вход (DIN) и один выход (DO). Выход каждого светодиода подключается к входу следующего. Подавать сигналы же надо на вход самого первого светодиода, таким образом, он запустит цепь, и данные будут поступать от первого ко второму, от второго к третьему и т. д.

Команды светодиодам передаются пачками по 24 бита (3 байта, один байт на каждый цвет, первым передается байт для зеленого, потом для красного, и заканчивает байт для синего светодиода. Порядок бит – от старшего к младшему). Перед каждой пачкой идет пауза в 50 мкс. Пауза больше 100 мкс воспринимается как окончание передачи. Все биты, будь то 0 или 1, имеют фиксированное время 1.25 мкс. Бит 1 кодируется импульсом в 0.8 мкс, после чего идет пауза в 0.45 мкс. Бит 0 кодируется импульсом в 0.4 мкс, после чего идет пауза в 0.85 мкс. Собственно, наглядная диаграмма на фото ниже. Так же допускаются небольшие погрешности в 0-150 нс на каждый фронт. Ну и следует учесть, что подобное необходимо повторить для каждого светодиода на ленте, после чего сделать паузу минимум в 100 мкс. Потом можно повторить передачу. Схема этого процесса представлена на рис. 2.14.2. [21].

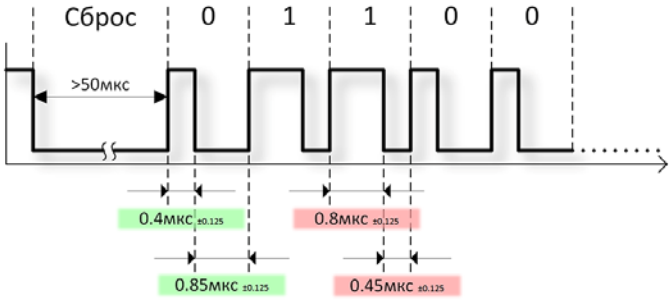


Рис. 2.14.2. Схема сигнала WS2812B [21]

Глядя на все эти цифры, становится ясно, что сделать все это, используя стандартные функции digitalWrite, delay и тому подобные – попросту невозможно, ввиду их долгой работы и неточности. Реализовать подобный протокол можно только использовав специальные библиотеки вроде CyberLib или написав собственную на чистом Си или, на Ассемблере.

Библиотека для работы с WS2812B. Библиотека Adafruit NeoPixel, разрабатывается компанией Adafruit Industries. Предназначена для работы со светодиодными лентами и неопиксельными кольцами. Библиотека написана на Си и Ассемблере с небольшим использованием Wiring. Поддерживает все виды Arduino. Содержит меньший функционал по сравнению с FastLED, немного медленней, но имеет более компактный вид, только основное для работы.

Подключение. Для того чтобы подключить библиотеку к своему скетчу необходимо в самом его начале прописать следующее:

#include "Adafruit_NeoPixel.h"

Эта строчка подключит заголовочный файл библиотеки и даст возможность использовать ее в своем проекте.

Далее необходимо создать основной объект библиотеки, с которым мы и будем работать.

```
// Параметр 1 – Количество пикселей в ленте.
// Параметр 2 – Порт, к которому подключена лента.
// Параметр 3 – Дополнительные флаги (по мере необходимости):
//  - NEO_KHZ800 – Передача данных на частоте 800 кГц (для продуктов и лент на светодиодах WS2812).
//  - NEO_KHZ400 – Передача данных на частоте 400 кГц (для продуктов и лент на светодиодах WS2811)
//  - NEO_GRB – Последовательность цветов при передаче данных (Зеленый, Красный, Синий).
//  - NEO_RGB – Последовательность цветов при передаче данных (Красный, Зеленый, Синий).
//  - и т.д.
```

Adafruit_NeoPixel strip = Adafruit_NeoPixel(60, 6, NEO_GRB + NEO_KHZ800);

Здесь мы создаем объект strip для управления светодиодной лентой состоящей из 60 пикселей типа WS2812B. И указываем, что вход управления лентой подключен к 6 порту Arduino.

1. По умолчанию библиотека настраивается на светодиоды WS2812, то есть выставляет флаги NEO_GRB + NEO_KHZ800, по этому, если вы используете ленту WS2812 или WS2812B данный параметр можно игнорировать и указать только порт и количество пикселей в ленте.
2. Если вы подключаете светодиодную ленту к 6 порту, то и порт можно не указывать, таким образом, обязательным параметром является только количество пикселей в ленте.

Инициализация. Для инициализации нашей ленты и дальнейшей работы с ней, необходимо в секции setup() вызвать две команды ранее созданного объекта ленты.

```
void setup()
{
  strip.begin(); // Функция begin() настроит порт Arduino и выставит значения по умолчанию.
```

```
  strip.show();  // Функция show() передаст команду на выключение всем пикселям.
        }
```

Функция show() в секции setup() не обязательная. Так как после инициализации все данные сброшены по умолчанию, функция по сути просто передает команды на выключение ленты.

Команды управления.

1. ***setPixelColor()***

Для того, чтобы задать любому пикселю ленты определенный цвет, существует переопределяемая функция setPixelColor(), переопределяемая потому, что имеет три варианта входящих параметров.

Первый вариант это:

strip.setPixelColor(n, red, green, blue)

Где параметр:

n – номер пикселя в ленте (отсчет идет с нуля, то есть если в вашей ленте 60 пикселей, то первый из них будет 0, а последний 59).

red – число от 0 до 255 определяющее яркость красного светодиода в пикселе.

green – число от 0 до 255 определяющее яркость зеленого светодиода в пикселе.

blue – число от 0 до 255 определяющее яркость синего светодиода в пикселе.

Таким образом, устанавливая разную яркость трем разноцветным светодиодам в пикселе, мы как бы смешиваем три цвета – красный, зеленый и синий, получая при этом недостающие цвета и их оттенки. Например, для того, чтобы получить желтый цвет, нам надо смешать зеленый и красный, а чтобы получить пурпурный цвет, надо смешать синий и красный. Белый цвет достигается смешиванием всех трех цветов. Данная технология смешивания цветов широко распространена в современных телевизорах, мониторах и в цветных принтерах.

Следующий вариант функции аналогичен первому, с тем лишь исключением, что имеет дополнительный пятый параметр «white».

strip.setPixelColor(n, red, green, blue, white)

Данный параметр используется в лентах, в которых помимо трехцветных пикселей, установлены еще и белые светодиоды, и данным параметром устанавливается их яркость.

Для того, чтобы использовать данную функцию, необходима как сама RGBW лента, так и при создании объекта «strip» в параметре №3 необходимо указать NEO_GRBW или NEO_RGBW.

strip.setPixelColor(n, color);

Последний вариант функции задает цвет пикселю в виде 32-битного числа. Данный вариант удобен для большинства людей и различных эффектов, так как позволяет работать с одним, а не тремя - четырьмя числам характеризующими один цвет.

2. ***Color()***

Функция Color() объединяет цвета в одно 32-битное число.

```
uint32_t magenta = strip.Color(255, 0, 255); // Пурпурный.
uint32_t greenishwhite = strip.Color(0, 64, 0, 64); // Светло зеленый.
```

3. ***show()***

Функция show() передает данные в ленту. Данную функцию необходимо вызывать каждый раз после того, как вы изменили цвета пикселей или настройки ленты.

```
void loop()
{
  strip.setPixelColor(0, random(255), random(255), random(255)); // Случайный цвет и оттенок.
  strip.show(); // Передаем в ленту.
  delay(1000); // Ждем одну секунду.
}
```

4. ***getPixelColor()***

Функция getPixelColor() возвращает цвет пикселя в виде 32-битного числа. В параметре n задается номер пикселя, цвет которого необходимо вернуть.

```
uint32_t color = strip.getPixelColor(n);
```

5. ***numPixels()***

Функция возвращает количество пикселей в ленте. Используется как правило в циклах или при работе с несколькими светодиодными лентами.

```
uint16_t n = strip.numPixels();
```

6. ***setBrightness()***

Функция задает общую яркость светодиодной ленты. Например, в примере ниже задается 1/4 яркость ленты (всего 255 градаций).

```
strip.setBrightness(64);
```

Как и в случае с setPixelColor(), изменения вступают в силу, только после вызова функции show().

Авторы библиотеки не рекомендуют использовать данную функцию при создании световых эффектов, и рекомендуют вызывать ее разово в секции setup().

Выделяемая память. Библиотека NeoPixel резервирует 3 байта оперативной памяти на каждый пиксель. Учитывая, что в Arduino UNO всего два килобайта оперативной памяти, плюс она нужна для реализации ваших спецэффектов и скетча, ее может не хватить для подключения больших лент (от 200 пикселей и больше). Поэтому, если вы хотите использовать ленты с несколькими сотнями пикселей, посмотрите в сторону Arduino Mega или Arduino Duo.

Задание. Необходимо подключить линейку из 6 светодиодов к платформе и загрузить следующую программу:

```
#include <Adafruit_NeoPixel.h>
```

```
int PIN=6; // номер порта к которому подключен модуль
int count_led=6; // количество светодиодов
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(count_led, PIN, NEO_GRB +
NEO_KHZ800); //first number change does distance between colors
void setup() {
  pixels.begin();
  pixels.setBrightness(1);
  pixels.show(); // Устанавливаем все светодиоды в состояние "Выключено"
}
void loop() {
pixels.setPixelColor(0, pixels.Color(255,0,0)); // Назначаем для первого светодио-
да цвет "Красный"
pixels.setPixelColor(1, pixels.Color(0,255,0)); // Назначаем для первого светодио-
да цвет "Зеленый"
pixels.setPixelColor(2, pixels.Color(250,255,0)); // Назначаем для первого свето-
диода цвет "Желтый"
pixels.setPixelColor(3, pixels.Color(0,0,255)); // Назначаем для первого светодио-
да цвет "Синий"
 pixels.show();
}
```

Задания для самостоятельного решения

1. Сделать автоматически меняющуюся интенсивность свечения. Сначала в большую сторону, затем в меньшую.

2. Сделать бегущую волну из светодиодов разного цвета.

3. Сделать управление интенсивностью цвета свечения всех светодиодов от переменного резистора и кнопки, задающей изменяемый цвет.

ЗАКЛЮЧЕНИЕ

Данное учебное пособие содержит основные сведения о принципах действия большого числа электронных узлов, выдающих измерительный сигнал различной формы. Широко также представлено описание компонентов, реализующих различные способы отображения как текстовой (числовой), так и графической информации.

Материал собран из различных источников и дан в виде набора лабораторных и практических работ. В каждой работе имеется как теоретические сведения, так и практическое задание, включающее в себя схему подключения и отлаженную программу для проверки работоспособности электронных блоков. Студентам также предоставляется возможность проявить самостоятельность и творчество, заключающееся в модифицировании предложенной программы для расширения ее функциональных возможностей.

Данное пособие необходимо студентам направлений 11.03.03 «Конструирование и технология ЭС» и 12.03.01 «Приборостроение», может быть использовано в самостоятельной работе, при подготовке к лабораторным и практическим занятиям, а также в курсовом и дипломном проектировании, как справочное пособие. Кроме того, оно подводит студента к более глубокому изучению материала.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Общие ресурсы по Arduino. — 2018 [Электронный ресурс]. Дата обновления: 05.10.2018. — http://arduino.ru/Hardware/ArduinoBoardUno (дата обращения: 10.11.2018).
2. Подключение LCD 1602 (HD44780) к Arduino. — 2018 [Электронный ресурс]. Дата обновления: 18.03.2014. — http://zelectro.cc (дата обращения: 10.11.2018).
3. Подключение дисплея LCD 1602 к arduino по i2c / IIC Подключение LCD 1602 (HD44780) к Arduino. — 2018 [Электронный ресурс]. Дата обновления: 18.03.2014. https://arduinomaster.ru/datchiki-arduino/lcd-i2c-arduino-displey-ekran/ (дата обращения: 10.11.2018).
4. Датчик температуры и влажности ардуино DHT11 и DHT22. — 2018 [Электронный ресурс]. Дата обновления: 18.03.2014. https://arduinomaster.ru/ (дата обращения: 10.11.2018).
5. Робототехника и автоматизация: блог Гридина Семена — 2018 [Электронный ресурс]. Дата обновления: 18.03.2014. http://kip-world.ru/ (дата обращения: 10.11.2018).
6. Семисегментный индикатор. Динамическая индикация— 2018 [Электронный ресурс]. Дата обновления: 18.03.2014 http://www.rotr.info (дата обращения: 10.11.2018).
7. SNx4HC595 8-Bit Shift Registers With 3-State Output Registers 2018, Texas Instruments Incorporated
8. A-1588AS Ф3.7MM / 8×8 / 1.3 INCH (33.3MM) SUPER RED DOT MATRIX 2012-5-27 TOPLIGHT
9. SPI и Arduino: теория— 2018 [Электронный ресурс]. Дата обновления: 18.03.2014 http://robocraft.ru/blog/arduino/518.html (дата обращения: 10.11.2018).
10. Робототехника— 2018 [Электронный ресурс]. Дата обновления: 18.03.2014 http://robots4life.ru (дата обращения: 10.11.2018).
11. Сообщество EasyElectronics.ru— 2018 [Электронный ресурс]. Дата обновления: 18.03.2014 http://we.easyelectronics.ru (дата обращения: 10.11.2018).
12. SNx4HC165 8-Bit Parallel-Load Shift Registers © 2018, Texas Instruments Incorporated
13. PCF8574; PCF8574A Remote 8-bit I/O expander for I2C-bus with interrupt 2013 NXP B.V
14. VS1838B Infrared Receiver module 2012 LFN
15. Robotclass— 2018 [Электронный ресурс]. Дата обновления: 18.03.2014 http://robotclass.ru (дата обращения: 10.11.2018).
16. 3DiY - Скорая помощь для Ваших проектов! — 2018 [Электронный ресурс]. Дата обновления: 18.03.2014 http://3d-diy.ru (дата обращения: 10.11.2018).

17. MAX6675 Cold-Junction-Compensated K-Thermocouple-to-Digital Converter (0°C to +1024°C) 2002 Maxim Integrated Products

18. DS18B20 Programmable Resolution 1-Wire Digital Thermometer 2018 Maxim Integrated Products, Inc.

19.Блог об Ардуино — 2018 [Электронный ресурс]. Дата обновления: 30.12.2017 http://soltau.ru (дата обращения: 10.11.2018).

20.RoboТехника — 2018 [Электронный ресурс]. Дата обновления: 18.03.2014http://роботехника18.рф (дата обращения: 10.11.2018).

21. Ардуино в Кыргызстане — 2018 [Электронный ресурс]. Дата обновления: 18.03.2014 http://arduino.on.kg (дата обращения: 10.11.2018).

ОГЛАВЛЕНИЕ

Учебное издание

Макаров Олег Юрьевич
Турецкий Андрей Владимирович
Хорошайлова Марина Владимировна

ЭЛЕКТРОНИКА И МИКРОПРОЦЕССОРНАЯ ТЕХНИКА

Практикум

Компьютерный набор А. В. Турецкого

Редактор Кулакова Н.В.

Подписано к изданию _____2018
Объем файла _____ МБ

ФГБОУ ВПО «Воронежский государственный
технический университет»
394026 Воронеж, Московский просп., 14