Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Локтионова Оксана Геннадьевна
Должность: проректор по учебной работе
Дата подписания: 26.07.2022 10:13:58
Уникальный программный ключ:
0b817ca911e6668abb13a5d426d39532f60fe4fe

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ ДЕТЕЙ
КАФЕДРЫ «СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ И ПОИСКОВОГО КОНСТРУИРОВАНИЯ»
РОБОФАБРИКА

Основы работы с Arduino

часть 2

Методические указания к лабораторной работе

Волгоград
2021

УДК 621.1.016 (075)

Р е ц е н з е н т

К.В. Приходьков

Печатается по решению редакционно-издательского совета
Волгоградского государственного технического университета

Основы работы с Arduino часть 2: метод. указания к лабораторной работе / сост.: И.Д. Казаков; ВолгГТУ. – Волгоград, 2021. –23 с.

Излагаются инструкции по сборке и программированию конструктора промышленного робота на базе Arduino. Даются рекомендации по изучению теоретического материала к лабораторной работе и ее выполнению.

Предназначены в помощь слушателям центров дополнительного образования, изучающим программы, связанные с проектированием, сборкой и программированием роботов.

Ил. 24. Табл. 3. Библиогр.: 0 назв.

© Волгоградский государственный
технический университет, 2021

Оглавление

1. Введение

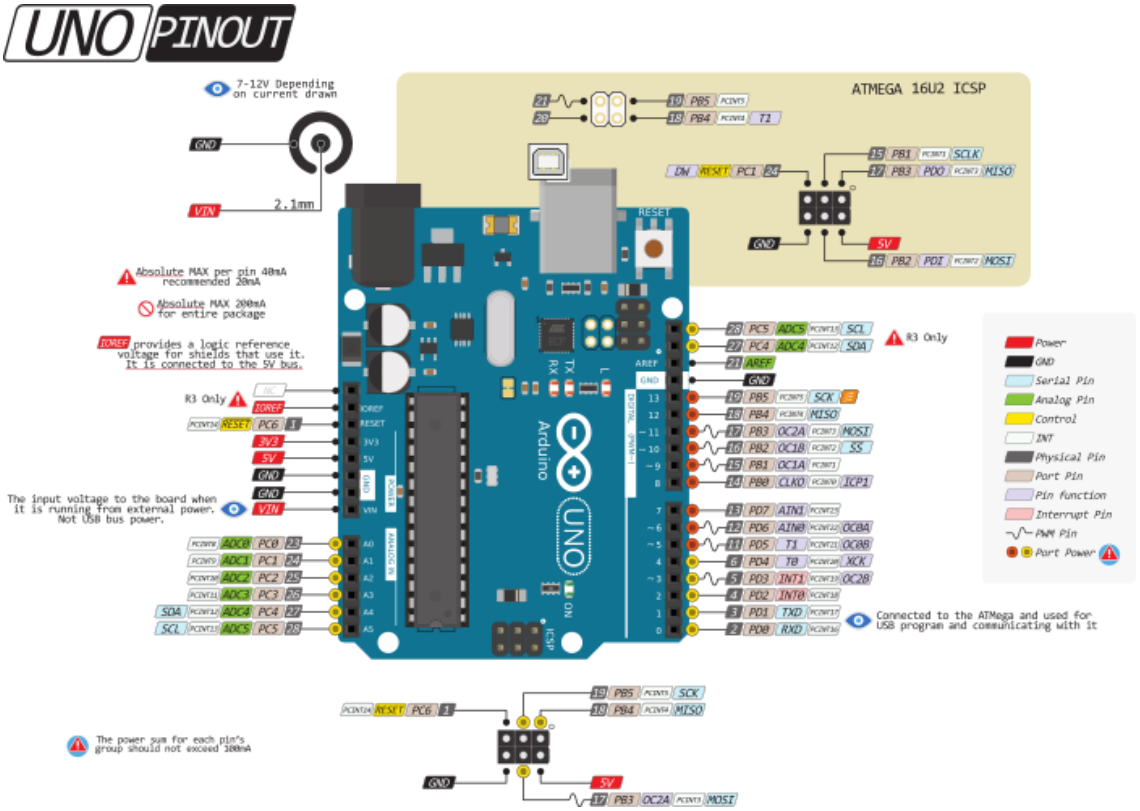


Рисунок 1 - Arduino Uno R3

Плата Arduino (рис. 1) предназначена для подключения и программирования различных датчиков, модулей и сенсоров. С ее могут быть разработаны многие элементы умного дома, такие как:

- Умный свет
- Умный контроль климата дома
- Сигнализация
- И другие элементы

Давайте повторим основные особенности платы и ее программирования. Начнем с определения разновидностей пинов (контактов) платы. Существует несколько основных разновидностей контактов на плате Arduino:

- Аналоговые пины: A0, A1, A2...
- Цифровые пины 0, 1, 2...
- Пины питания VCC, VIN, GND, 5V, 3V3
- Пины приема-передачи данных RX, TX
- Пины для обновления программы платы

Теперь вспомним программные особенности. Плата Arduino программируется на языке C++ с дополнительными библиотеками.

Простейшая программа контроллера состоит из двух пустых функций (рис. 2):

```
void setup() {
  // выполняется один раз при запуске платы

}

void loop() {
  // бесконечный цикл. выполняется после setup()

}
```

Рисунок 2 - Самая простая работоспособная программа

Функция - это ограниченный фрагмент программного кода, который выполняет определенную задачу и может быть вызван множество раз из разных участков программы.

Синтаксис функций:

[тип возвращаемого значения] [наименование функции] ([аргументы функции]) {[тело функции]}

Несколько примеров функций (рис. 3):

```
1  int function(int a, int b)
2  {
3    return a*b;
4  }
5
6  int function2(int a, int b)
7  {
8    Serial.print(function(2, 5));
9    float c = function(54, 9);
10 }
```

Функция возвращает результат умножения значения переменной **a** на значение переменной **b**

Функция передает значение умножения **2** на **5** в монитор порта, а также записывает в переменную **c** значение умножения **54** на **9**

Функция может принимать параметры и должна возвращать некоторое значение, возможно пустое. За это действие отвечает команда **return**, которую обязательно необходимо прописывать в конце каждой функции. После команды необходимо записать название переменной, арифметическое выражение или какое-либо значение одного типа данных с типом возвращаемого функцией значения.

Исключение: при указании типа возвращаемых данных **void использование команды return не является обязательным*

Существует множество различных типов данных:

bool - логический тип данных, хранит одно из двух значений: **true** или **false**

int - целочисленный тип данных, хранит целое значение, например: **0**, **15**, **-1432**

float - тип данных с плавающей запятой (дробные числа), например: **0.005**, **-325.0**

char - символ, например: ‘**э**’, ‘**Q**’, ‘**Λ**’

string - строка, например: “**ПРИМЕР**”, “**большой пример**”, “**самый большой пример, который содержит и символы, и знаки препинания, и даже растянут на несколько строк!**”

void - пустое множество

Комментарии являются необязательной частью программы, так как они не учитываются при компиляции, но крайне полезны для программистов. В комментариях указываются выходные и входные данные функций или определенного участка программы. Комментарии бывают двух типов:

- Однострочные. Записываются после двух символов слэш **/**, например:

  // Example comment

  Компилятор не будет учитывать часть строки после **//**

- Многострочные. Указываются после символов слэш со звездочкой **/*** и до символов звездочка и слэш ***/**, например:

  /* Example
  comment */

  Компилятор будет игнорировать все строки, которые находятся между **/*** и ***/**

2. Решение простых задач программирования

Задачи:

1. Реализовать программу для суммирования 2-х чисел.
2. Реализовать программу мигания светодиодом с интервалом в 1 секунду.

Для реализации данных программ нам потребуется вспомнить информацию, изученную на предыдущей ступени обучения.

Написание программы начинается с написания псевдокода - словесного описания алгоритма. Для суммы двух чисел псевдокод будет выглядеть следующим образом:

- Ввести число A
- Ввести число B
- Сложить число A с числом B и присвоить результат числу C
- Вывести число C

Следующим шагом будет замена каждой строки псевдокода на функции:

1. Для того, чтобы ввести число в программу, необходимо объявить переменную.

Для этого укажем в начале строки тип данных, который будет использоваться. В нашем случае числа будут целыми, следовательно использует int. Далее назовем переменную по смыслу ее назначения, например var_A (от анг. variable - переменная) будет хранить значение переменной A.

```
int var_A;
```

Переменную объявили, теперь нужно записать в нее значение. Используем операцию присваивания “=”:

```
var_A = 3;
```

2. Точно также объявим переменную B и C. Переменной B присвоим значение 2. Присваивать значение можно сразу при ее объявлении:

```
int var_A = 3;
```

Также можно объявить несколько переменных подряд:

```
int var_B, res_C;
var_B = 2;
```

3. Присвоим переменной res_C значение суммы переменных var_A и и var_B:

```
res_C = var_A + var_C;
```

4. Для отображения значения переменной res_C напишите две дополнительные функции:

```
Serial.begin( 9600 ); // Инициализация монитора порта
Serial.println( res_C ); // Вывод переменной rec_C в монитор порта
```

Таким образом собрав все эти знания вместе должно получится следующее (рис. 4):

```
void setup() {
  int var_A = 3; //объявляем переменную A со значением 3
  int var_B, res_C; //Объявляем переменные B и C

  var_B = 2; //присваиваем переменной B значение 2

  res_C = var_A + var_B; //присваиваем переменной C значение суммы A и B

  Serial.begin(9600); //подключаем монитор порта
  Serial.println( res_C ); //выводим значение переменной C в порт
}

void loop() {}
```

Рисунок 4 - Программа суммирования двух чисел

Теперь разберем задачу со светодиодом. Светодиод является отдельным от платы Ардуино полупроводниковым прибором, используемым для излучения света. Плата хоть и является “умной”, но все же самостоятельно не может определить что к ней подключено и подключено ли вообще. Специально для ответа на вопросы что и к каким пинам Ардуино подключено, существует функция pinMode.

void pinMode (pin, mode) - функция, которая указывает плате тип устройства, подключенного к определенному пину платы. Это относится, в основном, к простым устройствам у которых сигнальным является только один пин (светодиод, датчик линии и т.д.), либо к сложным устройствам (ультразвуковой дальномер, драйвер двигателя и т.д.).

Функция принимает на вход два параметра:

1) pin - числовой или буквенно-числовой номер пина, к которому подключено устройство
2) mode - тип устройства, OUTPUT или INPUT (устройство вывода или ввода информации, соответственно).

Также потребуется функция, с помощью которой мы сможем подать необходимое напряжение на светодиод, чтобы его “зажечь”

или “потушить”. Одна из таких функций записывается следующим образом:

void digitalWrite(pin, value)

Функция принимает на вход два параметра:

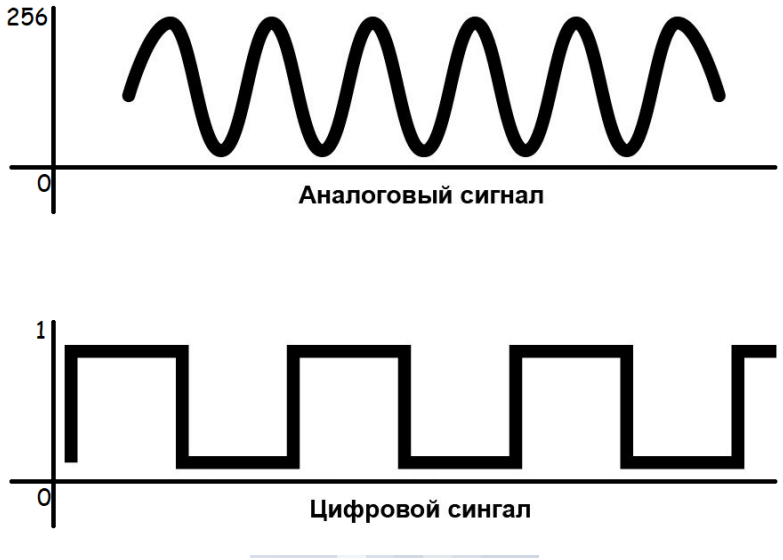
1) pin - номер пина, к которому подключено устройство
2) value - значение (HIGH или LOW) уровня сигнала на данном pin.

Задание. Дополнить программу (рис. 5) таким образом, чтобы напряжение на светодиоде менялось каждую секунду.

```
1  void setup() {
2    pinMode( 13, OUTPUT );
3  }
4
5  void loop() {
6    d       W    (    , HIGH );
7    delay( 1000 );
8    d            (    ,      );
9    delay( 1000 );
10 }
```

Рисунок 5 - Задание для самостоятельного выполнения

3. Изучение функций для работы с цифровыми и аналоговыми пинами.



Различие между аналоговым и цифровым сигналом (рис. 6) заключается в различной структуре передаваемого сигнала. Аналоговые сигналы – непрерывный поток колебаний с изменяющимися амплитудой и частотой. Цифровой сигнал – дискретные колебания, т.е. сигнал может принимать одно из двух значений LOW (низкий уровень сигнала, соответствует значению напряжения 0 вольт) или HIGH (высокий уровень сигнала, соответствует максимальному напряжению сигнала).

Цифровые устройства могут быть либо включены (замкнуты), либо выключены (разомкнуты), например: диоды (светодиоды и фотодиоды), выключатели, реле и так далее.

Аналоговые устройства меняют сигналы, в зависимости от входного напряжения, либо от внешних факторов. Переменные резисторы меняют сопротивление при перемещении ползунка, датчик влажности - в зависимости от концентрации воды в воздухе или почве и так далее.

Так как цифровые сигналы от аналоговых отличаются, по сути, диапазоном значений сигнала, то и функции мало чем отличаются, а именно названием и тем же самым диапазоном сигналов!

Итак, перейдем к самим функциям:

int digitalRead(int pin); /*Считывает значение сигнала (0 или 1) на цифровом входном пине.*/

int analogRead(int pin); /*Считывает значение сигнала от 0 до 1023 на аналоговом входном пине.*/

void digitalWrite(int pin, int value); /*Записывает значение сигнала от 0 до 1 на цифровой выходной пин.*/

void analogWrite(int pin, int value); /*Записывает значение сигнала от 0 до 255 на аналоговый выходной пин.*/

Проверим эти функции на джойстике, так как он включает в себя как аналоговые, так и цифровые элементы. Сначала соберем электрическую схему (рис. 7), затем запрограммируем.

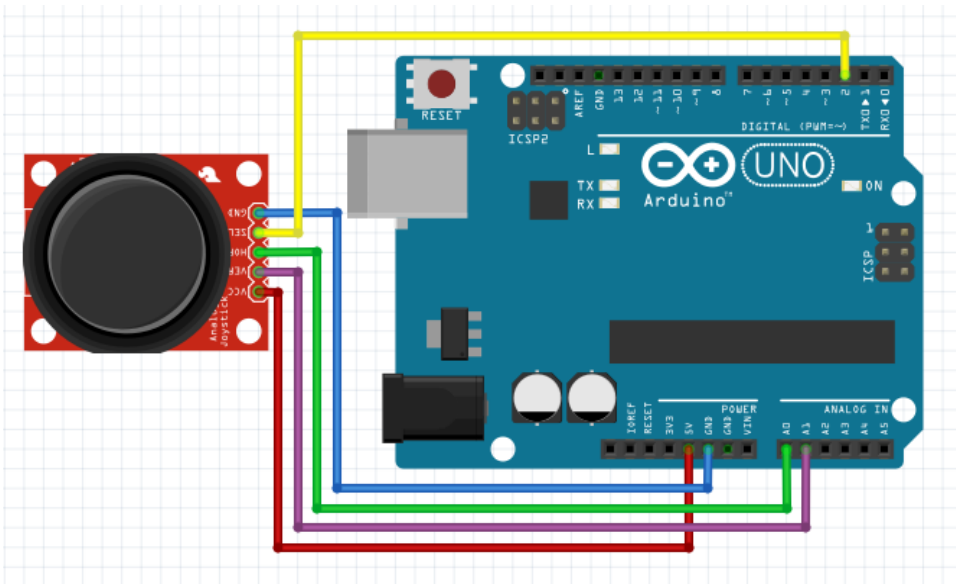


Рисунок 7 - Схема подключения джойстика

Данная программа (рис. 8) получает все данные с джойстика и выводит их в монитор порта.

```
int valueOX;
int pinOX = A0;

int valueOY;
int pinOY = A1;

int valueButton;
int pinButton = 2;

void setup() {
  Serial.begin(9600);
  pinMode(pinOX, INPUT);
  pinMode(pinOY, INPUT);
  pinMode(pinButton, INPUT);
}

void loop() {
  valueOX = analogRead( pinOX );
  valueOY = analogRead( pinOY );
  valueButton = analogRead( pinButton );

  Serial.println("OX = " + valueOX);
  Serial.println("\nOY = " + valueOY);
  Serial.println("\nButton = " + valueButton);
}
```

Рисунок 8 - Программа джойстика

4. Основы работы с LCD1602

Выводить данные с платы можно многими способами. Наиболее простым является монитор порта (функции Serial), но при его использовании требуется компьютер. Если требуется создать автономное устройство, для работы которого не будут использоваться компьютер, телефон или планшет, можно использовать другие средства вывода информации. Например, можно использовать множество светодиодов. Установив их в особом порядке можно получить своеобразный дисплей, где каждый светодиод будет являться пикселем. А можно использовать уже готовый дисплейный модуль. Его сейчас и разберем.

Для работы потребуется установить дополнительные библиотеки:

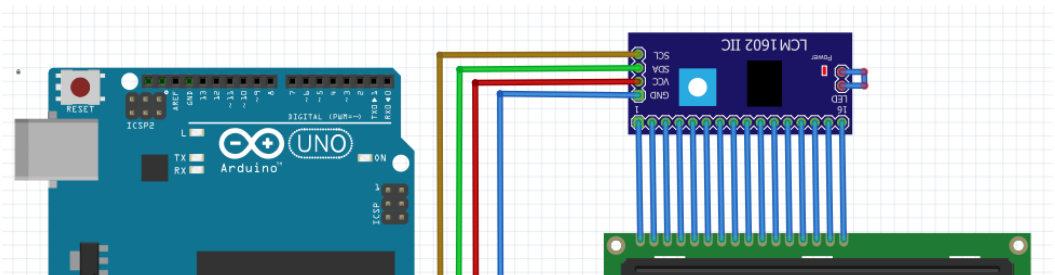
- Wire.h
- LiquidCrystal_I2C.h

Дисплейный модуль LCD1602, представленный в наборе, является двухстрочным дисплеем с шестнадцатью ячейками в каждой (отсюда и “1602” в названии). Модуль спаян с платой передачи данных по протоколу I2C, которая упрощает подключение с 16 контактов до 4-х. На обратной стороне I2C модуля есть 3 перемычки, замыкая которые, можно изменять адрес обращения к модулю. Таким образом, на одной плате может быть подключено до восьми дисплейных модулей с разными адресами.

Подключение по протоколу I2C следует производить в соответствии со следующей таблицей (таблица 1) и схемой (рис. 9):

Таблица 1 - Подключение LCD1602 к Arduino Uno

LCD I2C	**Arduino Uno R3**
VCC	5V
GND	GND
SDA	A4
SCL	A5



Для вывода текста на дисплей используется следующая программа (рис. 10):

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  lcd.init();
  lcd.backlight();
  lcd.print("LCD 1602");
  lcd.setCursor(8, 1);
  lcd.print("TESTING");
}

void loop() {
  lcd.print(0, 1);
  lcd.print(millis() / 1000);
}
```

Рисунок 10 - Программа LCD1602

Данная программа будет выводить текст:

LCD 1602
X TESTING, где **X** - время работы программы, в секундах

Объявление объекта **LiquidCrystal_I2C**:

LiquidCrystal_I2C lcd (0x27, 16, 2);

lcd - название одного из подключенных дисплейных модулей (или единственного дисплейного модуля)

0x27 - адрес модуля, к которому будет происходить обращение. Стандартный адрес 0x27

16 - количество столбцов на дисплее

2 - количество строк на дисплее

Функции:

void lcd.init(); /* Запускает инициализацию дисплея. Своеобразный pinMode для дисплейного модуля */

void lcd.backlight(); /* Включение подсветки модуля */

void lcd.print(“TEXT”); /* Функция для вывода строки. Вывод начинается с позиции курсора */

void lcd.print(val); // Выводить можно не только строки или символы, но также и числа*/

void lcd.setCursor(10, 1); /* Перемещение курсора в конкретную позицию на экране. Первый аргумент указывает столбец, второй - строку. В начале работы курсор установлен в позиции 0,0. Если было выведено сообщение, курсор автоматически перемещается */

void lcd.clear(); /* Очистка экрана. Удаление ранее выведенного сообщения */

void lcd.home(); /* Возвращает курсор в позицию 0,0. Аналогичная запись lcd.setCursor(0, 0); */

5. Основы работы с датчиком DHT11

Для работы потребуется установить дополнительные библиотеки:

- DHT.h
- adafruit_sensor.h

DHT11 - цифровой датчик температуры и влажности. Используется при необходимости поддержки определенного диапазона температуры и/или влажности. Подключение происходит по следующей схеме (рис. 11):

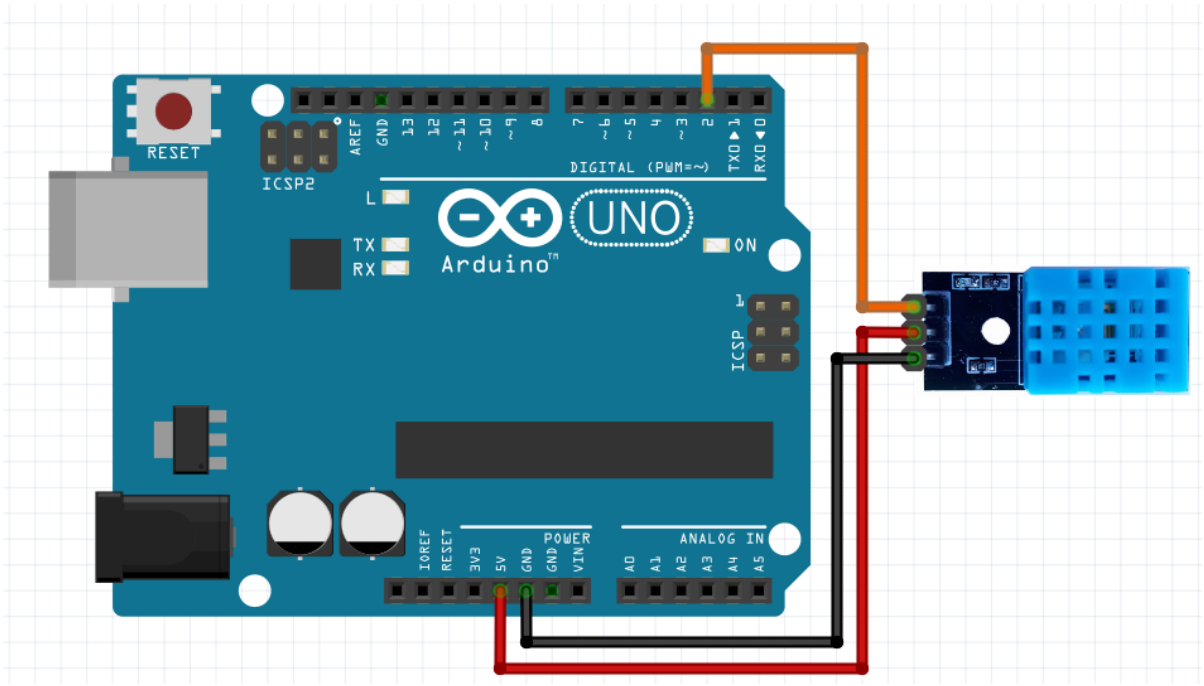


Рисунок 11 - Схема подключения DHT11

Следующая программа (рис. 12) отправляет в монитор потра показания с датчика, а именно влажность (humidity) и температуру (temperature).

```
#include <DHT.h>
DHT dht(2, DHT11);
void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print("; Temperature: ");
  Serial.println(t);
}
```

Рисунок 12 - Программа DHT11

Объявление объекта **DHT**:

DHT dht (2, DHT11);

2 - логический (сигнальный) пин, к которому подключен датчик температуры

DHT11 - модель датчика. DHT11 или DHT22

Функции:

void dht.begin(); /* Запускает инициализацию датчика DHT */

float dht.readHumidity(); /* Считываем показания влажности с датчика*/

float dht.readTemperature(); /* Считываем показания температуры с датчика */

6. Основы работы с RFID

Для работы потребуется установить дополнительные библиотеки:

- SPI.h
- MFRC522.h

Модуль RFID-RC522 является устройством для считывания и записи информации с электромагнитных ключей (RFID меток). На основе модуля можно построить прототип двери с электромагнитным замком. Подключение модуля производится по схеме, представленной на рисунке 13.

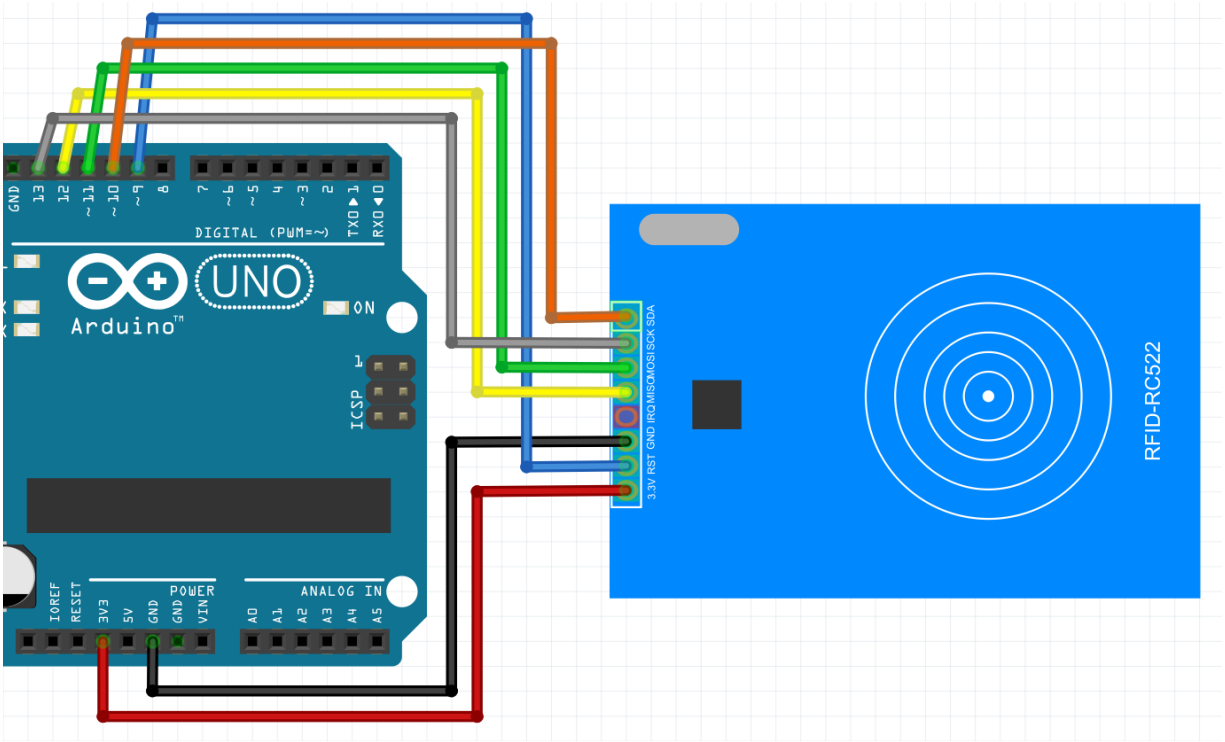


Рисунок 13 - Схема подключения RFID

Для проверки работоспособности необходимо использовать следующую программу (рис. 14):

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

#define RST_PIN         9
#define SS_PIN          10
const uint8_t valid_uid[] = {0xEF, 0xFB, 0xF6, 0x01};

MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
}

void loop() {
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }

  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  if ((mfrc522.uid.size == sizeof(valid_uid)) &&
     (memcmp(&mfrc522.uid.uidByte, &valid_uid,
     sizeof(valid_uid))==0)) {
    Serial.println("Доступ разрешен!");
  } else {
    Serial.println("Доступ запрещен!");
  }
}
```

Рисунок 14 - Программа RFID модуля

*Необходимо заменить **valid_uid** на ID карты из набора. Для этого необходимо использовать программу из примера DumpInfo. В монитор порта при этом будет выводится вся информация о карте, включая ее ID. В данном примере выводится следующая информация:

Card UID: EF FB F6 01
Card SAK: 08
PICC type: MIFARE 1KB

7. Основы работы с электромагнитным реле

Модуль реле необходим для включения устройств, которым необходим внешний источник энергии. Например: лампа накаливания, вентилятор, электрочайник и т.д.

Большое количество данных модулей используется при создании умных вещей.

Подключение реле производится в соответствии со следующей схемой:

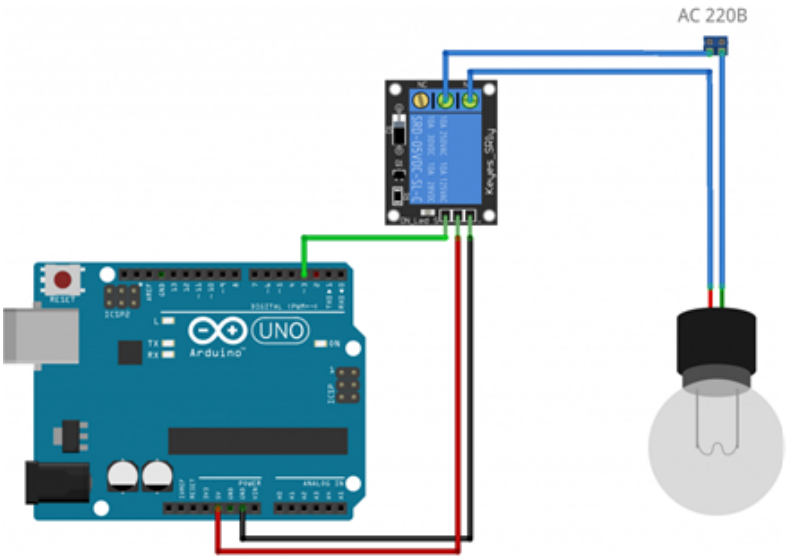


Рисунок 15 - Схема подключения элуктромагнитного реле

Программа для управления реле, представлена на рисунке 16.

```
int relayPin = 3;
void setup() {
  Serial.begin(9600);
  pinMode( relayPin, OUTPUT );
}

void loop() {
  digitalWrite( relayPin, HIGH );
  Serial.println( "Включили свет!" );
  delay( 1000 );

  digitalWrite( relayPin, LOW );
  Serial.println( "Отключили свет!" );
  delay( 1000 );
}
```

Рисунок 16 - Программа для работы с электромагнитным реле

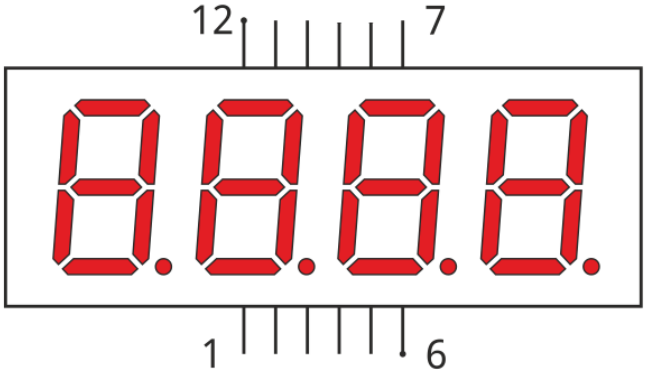
8. Основы работы с семисегментным четырехразрядным индикатором

Еще одним устройством для вывода информации являются семисегментные индикаторы. Они применяются для вывода определенных символов и цифр. На четырехразрядном индикаторе можно выводить дробные числа с точностью до тысячных, если число меньше 10 (например **9.154**). Также можно вывести строку, содержащую некоторые буквы и символы (например **E_1.4**).

Подключение индикатора производится в соответствии с таблицей 2, порядок номеров контактов индикатора представлен на рисунке 17.

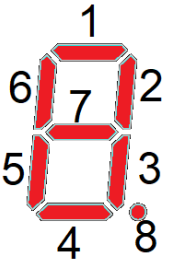
Таблица 2 - Подключение дисплея

Индикатор	Arduino
1	13
2	12
3	11
4	10
5	9
6	A4
7	6
8	A3
9	A2
10	7
11	8
12	A1



**Рекомендуется подключить контакты, выделенные в таблице серым цветом, через резисторы 300 - 1000 Ом.*

Сегменты индикатора (рисунок 18) представляют собой светодиоды, включая и выключая которые можно получить необходимые символы. При программировании символа, будут использоваться массивы. На рисунке справа изображен порядок расстановки светодиодов. Для вывода цифры **3**, например, необходимо использовать следующий массив:



int seg[] = {1, 1, 1, 1, 0, 0, 1, 0};

Для программирования рекомендуется создать массивы с используемыми символами и цифрами заранее. Например, для цифр можно воспользоваться двумерным массивом:

```
int d[10][8] = {      {1, 1, 1, 1, 1, 1, 0, 0},   //0
                      {0, 1, 1, 0, 0, 0, 0, 0},   //1
                      {1, 1, 0, 1, 1, 0, 1, 0},   //2
                      {1, 1, 1, 1, 0, 0, 1, 0},   //3
                      {0, 1, 1, 0, 0, 1, 1, 0},   //4
                      {1, 0, 1, 1, 0, 1, 1, 0},   //5
                      {1, 0, 1, 1, 1, 1, 1, 0},   //6
                      {1, 1, 1, 0, 0, 0, 0, 0},   //7
                      {1, 1, 1, 1, 1, 1, 1, 0},   //8
                      {1, 1, 1, 1, 0, 1, 1, 0}    //9
                };
```

При этом алгоритм вывод массива **d[0][ ]** будет соответствовать цифре **0**, **d[1][ ]** - **1**, **d[2][ ]** - **2** и т.д.

Для вывода данных символов необходимо воспользоваться программой, представленной на рисунках 19 и 20.

```
int anodPins[] = {A1, A2, A3, A4};
int segmentsPins[] = {8, 6, 10, 12, 13, 7, 9, 11};

void setup() {
  for (int i = 0; i < 4; i++) {
    pinMode(anodPins[i], OUTPUT);
  }
  for (int i = 0; i < 8; i++) {
    pinMode(segmentsPins[i], OUTPUT);
  }
}

int d[10][8] = {{1, 1, 1, 1, 1, 1, 0, 0},   //0
                {0, 1, 1, 0, 0, 0, 0, 0},   //1
                {1, 1, 0, 1, 1, 0, 1, 0},   //2
                {1, 1, 1, 1, 0, 0, 1, 0},   //3
                {0, 1, 1, 0, 0, 1, 1, 0},   //4
                {1, 0, 1, 1, 0, 1, 1, 0},   //5
                {1, 0, 1, 1, 1, 1, 1, 0},   //6
                {1, 1, 1, 0, 0, 0, 0, 0},   //7
                {1, 1, 1, 1, 1, 1, 1, 0},   //8
                {1, 1, 1, 1, 0, 1, 1, 0}    //9
              };
```

```
void loop() {
    for (int k = 0; k < 8; k++) {
      digitalWrite(segmentsPins[k], ((d[1][k] == 1) ? LOW : HIGH));
    }
    digitalWrite(anodPins[1], HIGH);
    delay(1);
    digitalWrite(anodPins[1], LOW);
}
```

Рисунок 20 - Продолжение программы управления индикатором

Изменение индекса массива **anodPins[ ]** будет менять разряд, отображаемого значения.

Строка программы **digitalWrite(segmentsPins[k], ((d[1][k] == 1) ? LOW : HIGH));** аналогична следующей записи:

```
if(d[1][k] == 1)
  digitalWrite(segmentsPins[k], LOW);
else
  digitalWrite(segmentsPins[k], HIGH);
```

9. Основы работы со светодиодной матрицей

Этот модуль не просто так называют матрицей. По сути, он представляет собой таблицу из светодиодов, состоящую из 8 строк и 8 столбцов. Изображение на матрице получается чередованием низкого и высокого сигнала, подаваемого на строки и столбцы матрицы, например:

LOW	HIGH	HIGH	HIGH	HIGH	HIGH	HIGH	LOW
HIGH	LOW	HIGH	HIGH	HIGH	HIGH	LOW	HIGH
HIGH	HIGH	LOW	HIGH	HIGH	LOW	HIGH	HIGH
HIGH	HIGH	HIGH	LOW	LOW	HIGH	HIGH	HIGH
LOW	HIGH	HIGH	HIGH	HIGH	HIGH	HIGH	LOW
HIGH	LOW	HIGH	HIGH	HIGH	HIGH	LOW	HIGH
HIGH	HIGH	LOW	HIGH	HIGH	LOW	HIGH	HIGH
HIGH	HIGH	HIGH	LOW	LOW	HIGH	HIGH	HIGH

Для использования изображения в программе рекомендуется использовать двумерный массив:

```
int picture[8][8] = {  {0, 1, 1, 1, 1, 1, 1, 0},
                       {1, 0, 1, 1, 1, 1, 0, 1},
                       {1, 1, 0, 1, 1, 0, 1, 1},
                       {1, 1, 1, 0, 0, 1, 1, 1},
                       {0, 1, 1, 1, 1, 1, 1, 0},
                       {1, 0, 1, 1, 1, 1, 0, 1},
                       {1, 1, 0, 1, 1, 0, 1, 1},
                       {1, 1, 1, 0, 0, 1, 1, 1}  };
```

В программе проще всего реализовать управление матрицей двумя вложенными циклами и одним ветвлением, рисунок 21.

```
for (int r = 0; r < 8; ++r) {
  for (int c = 0; c < 8; ++c) {
    digitalWrite(colPins[c], ((picture[r][c] == 1) ? HIGH : LOW));
  }
  digitalWrite(rowPins[r], LOW);
  delay(1);
  digitalWrite(rowPins[r], HIGH);
}
```

Рисунок 21 - Цикл для управления матрицей

Таким образом, вся программа представлена на рисунке 22.:

```
// Указываем пины, к которым подключены строки матрицы
int rowPins[] = {13, 8, A3, 10, 5, A2, 4, A0};


// Указываем пины, к которым подключены столбцы матрицы
int colPins[] = {9, 3, 2, 12, A1, 11, 7, 6};

int picture[8][8] = { {0, 1, 1, 1, 1, 1, 1, 0},
                      {1, 0, 1, 1, 1, 1, 0, 1},
                      {1, 1, 0, 1, 1, 0, 1, 1},
                      {1, 1, 1, 0, 0, 1, 1, 1},
                      {0, 1, 1, 1, 1, 1, 1, 0},
                      {1, 0, 1, 1, 1, 1, 0, 1},
                      {1, 1, 0, 1, 1, 0, 1, 1},
                      {1, 1, 1, 0, 0, 1, 1, 1}  };


void setup()
{
  for (int i = 0; i < 8; ++i) {
    pinMode(rowPins[i], OUTPUT);
    pinMode(colPins[i], OUTPUT);
  }
}

void loop()
{
  for (int r = 0; r < 8; ++r) {
    for (int c = 0; c < 8; ++c) {
      digitalWrite(colPins[c], ((picture[r][c] == 1) ? HIGH : LOW));
    }
    digitalWrite(rowPins[r], LOW);
    delay(1);
    digitalWrite(rowPins[r], HIGH);
  }
}
```

Рисунок 22 - Программа для управления светодиодной матрицей

Строка программы **digitalWrite(colPins[c], ((picture[r][c] == 1) ? HIGH : LOW));** аналогична следующей записи:

```
if(picture[r][c] == 1)
  digitalWrite(colPins[c], HIGH);
else
  digitalWrite(colPins[c], LOW);
```

Подключение матрицы производится в соответствии с таблицей 3.

Таблица 3 - Подключение матрицы светодиодов с разделением на контакты строк и столбцов

Матрица	Arduino
1	5
2	4
3	3
4	2
5	A0
6	A1
7	A2
8	A3
9	13
10	12
11	11
12	10
13	9
14	8
15	7
16	6

Номер строки	Pin
1	13
2	8
3	A3
4	10
5	5
6	A2
7	4
8	A0

Номер столбца	Pin
1	9
2	3
3	2
4	12
5	A1
6	11
7	7
8	6

На части матрицы с контактами найдите цифру 1. Это порядковый номер ближайшего пина. Дальнейшая нумерация контактов идет по часовой стрелке до контакта под номером 16.

**Рекомендуется подключить контакты, выделенные в таблице серым цветом, через резисторы, номиналом от 300 Ом до 1 КОм.*

10. Основы работы с кнопочной матрицей 4*4

Для работы потребуется установить дополнительную библиотеку:

- Keypad.h

Кнопочную матрицу, состоящую из простых кнопок, можно использовать в качестве клавиатуры для ввода информации в контроллер.

Подключение матрицы производится в соответствии с схемой, представленной на рисунке 23.

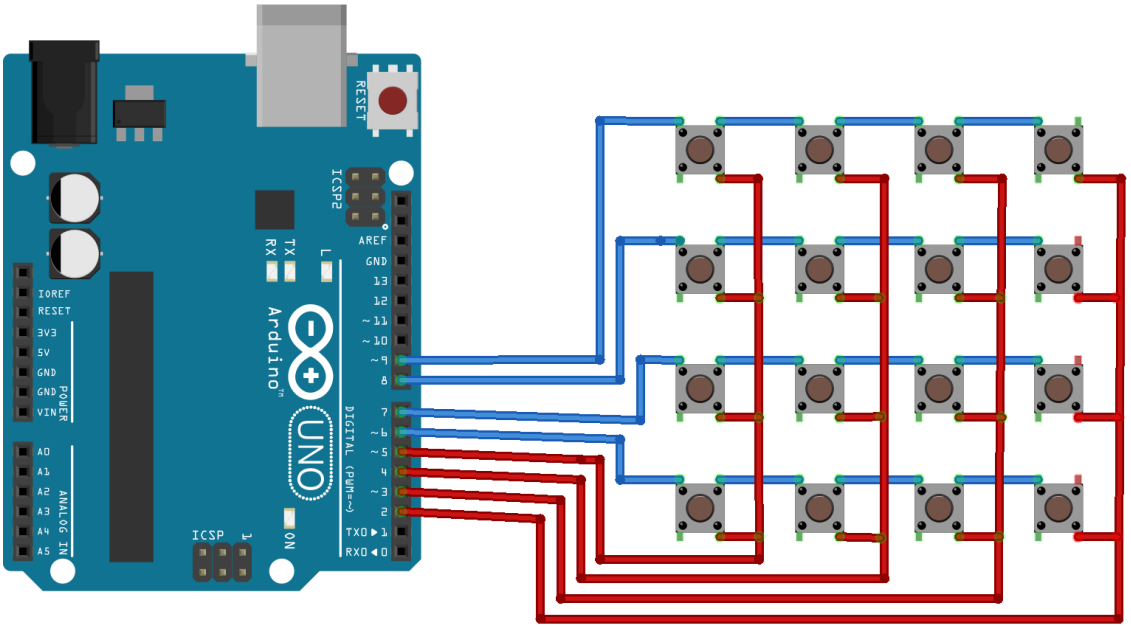


Рисунок 23 - Схема подключения матрицы кнопок

Для взаимодействия с контроллером посредством матричной клавиатуры используется программа, представленная на рисунке 24.

```
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {5, 4, 3, 2};
byte colPins[COLS] = {6, 7, 8, 9};

Keypad myKeypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup(){
  Serial.begin(9600);
}

void loop(){
  char key = myKeypad.getKey();

  if (key != NO_KEY){
    Serial.println(key);
  }
}
```

*Рисунок 24 - Программа для работы с кнопочной матрицей 8*8*

Объявление объекта **Keypad**:

Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

makeKeymap(keys) - связывание кнопок с массивом символов

keys - массив символов

rowPins - контакты строк клавиатуры

colPins - контакты столбцов клавиатуры

ROWS - количество строк клавиатуры

COLS - количество столбцов клавиатуры

Функции:

myKeypad.getKey(); /* результатом функции является символ, соответствующий данной кнопке. NO_KEY - ни одна кнопка не нажата */

Учебное издание

Составители:

Казаков Игорь Дмитриевич

ОСНОВЫ РАБОТЫ С ARDUINO ЧАСТЬ 2.

Методические указания к лабораторной работе

Темплан 2021 г. (учебно-методическая литература). Поз. №
Подписано в печать 00.00.2020. Формат 60x84 1/16. Бумага офсетная.
Гарнитура Times. Печать офсетная. Усл. печ. л. 1,4.
Тираж 10 экз. Заказ

Волгоградский государственный технический университет.
400005, г. Волгоград, просп. В. И. Ленина, 28, корп. 1.

Отпечатано в типографии ИУНЛ ВолгГТУ.
400005, г. Волгоград, просп. В. И. Ленина, 28, корп. 7.