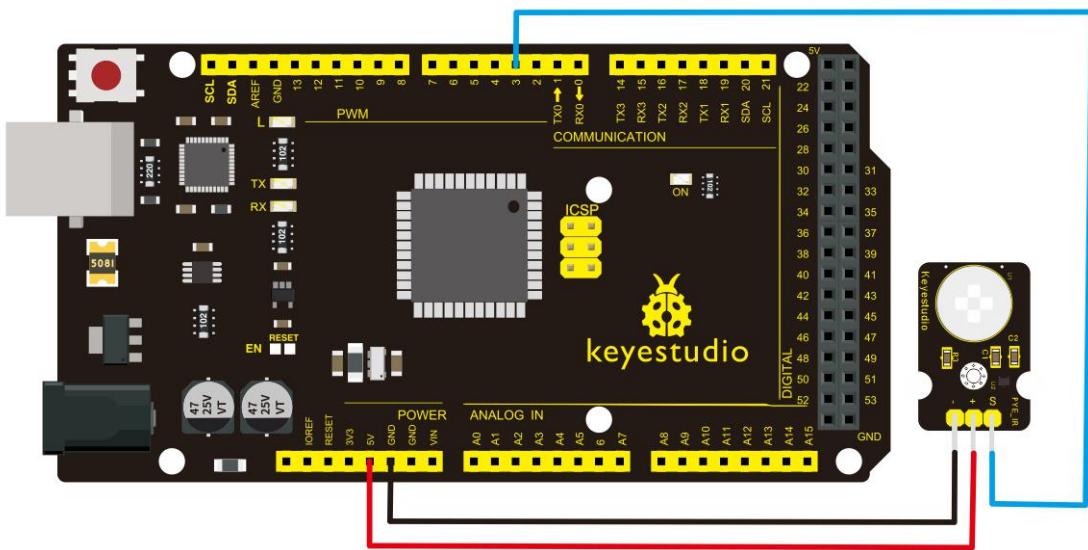


Подключение для MEGA 2560 R3



5. образец кода

```
/*
```

супер обучающий комплект keyestudio

Project 23

PIR

<http://www.keyestudio.com>

* /

byte sensorPin = 3;

байт-индикатор = 13;

установка void ()

{

pinMode (sensorPin, ВХОД);

pinMode (индикатор, ВЫХОД);

Serial.begin (9600);

}

пустой цикл ()

{

состояние байта = digitalRead (sensorPin); digitalWrite

(индикатор, состояние);

if (state == 1) Serial.println («Кто-то находится в этой области!»); иначе, если

(состояние == 0) Serial.println («Никто!»);

задержка (500);

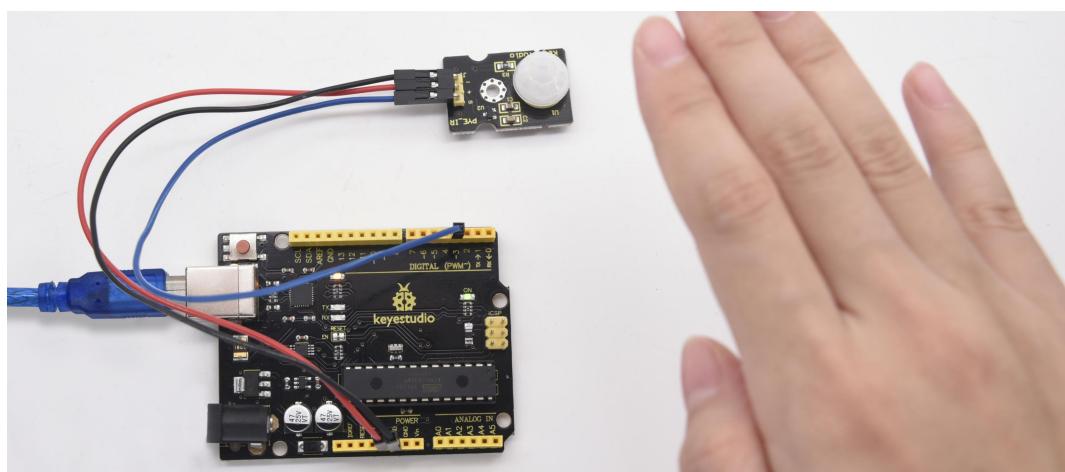
}

||||||||||||||||||||||||||||||||||||||

6. результат теста

Если датчик обнаружит, что кто-то движется поблизости, загорится индикатор D13 на плате V4.0, и "Кто-то находится в этой области!" отображается на последовательном мониторе.

Если движение не обнаружено, индикатор D13 на плате V4.0 не горит, и "Никто!" отображается на последовательном мониторе.



```
Somebody is in this area!
No one!
No one!
No one!
No one!
No one!
No one!
Somebody is in this area!
No one!
```

Autoscroll Show timestamp Newline 9600 baud Clear output

Проект 24: Аналоговый датчик газа



1. Введение

Этот аналоговый датчик газа - MQ2 используется в оборудовании для обнаружения утечки газа на рынках бытовой электроники и на промышленных рынках.

Этот датчик подходит для обнаружения сжиженного нефтяного газа, бутана, пропана, метана, спирта, водорода и дыма. Обладает высокой чувствительностью и быстрым откликом. Кроме того, чувствительность можно регулировать потенциометром.

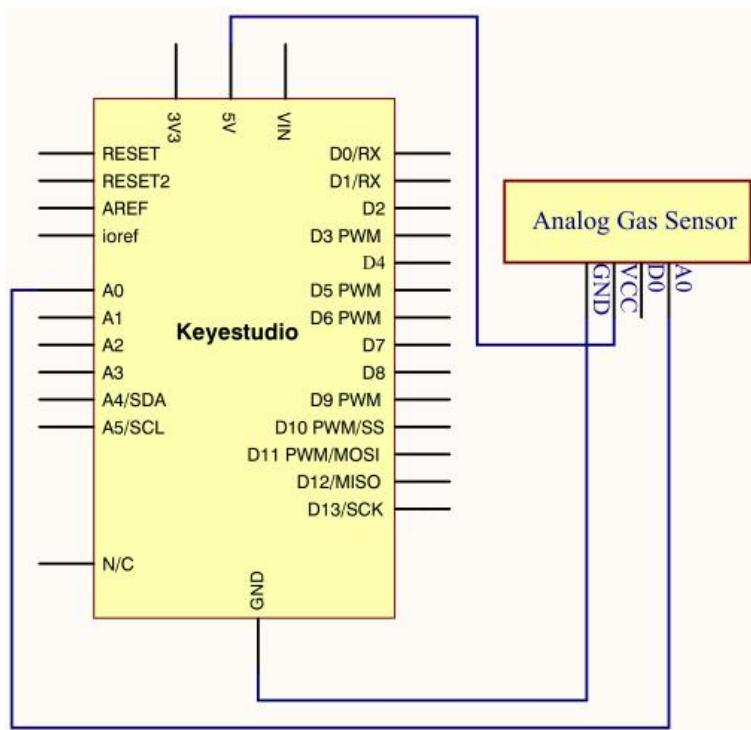
2. Требуется оборудование

- Плата V4.0 или плата MEGA 2650 * 1
- Аналоговый датчик газа * 1
- Перемычка * 1
- USB-кабель * 1

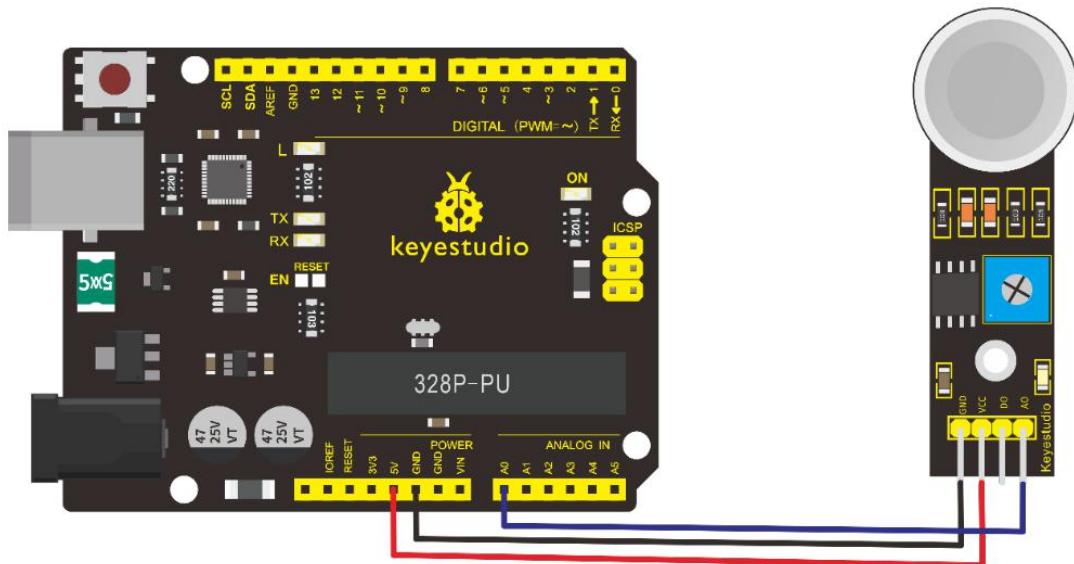
3. Спецификация

- Источник питания 5В
- Тип интерфейса Аналоговый
- Широкий диапазон обнаружения
- Быстрый отклик и высокая чувствительность Простая
- схема привода
- Стабильный и долгий срок службы

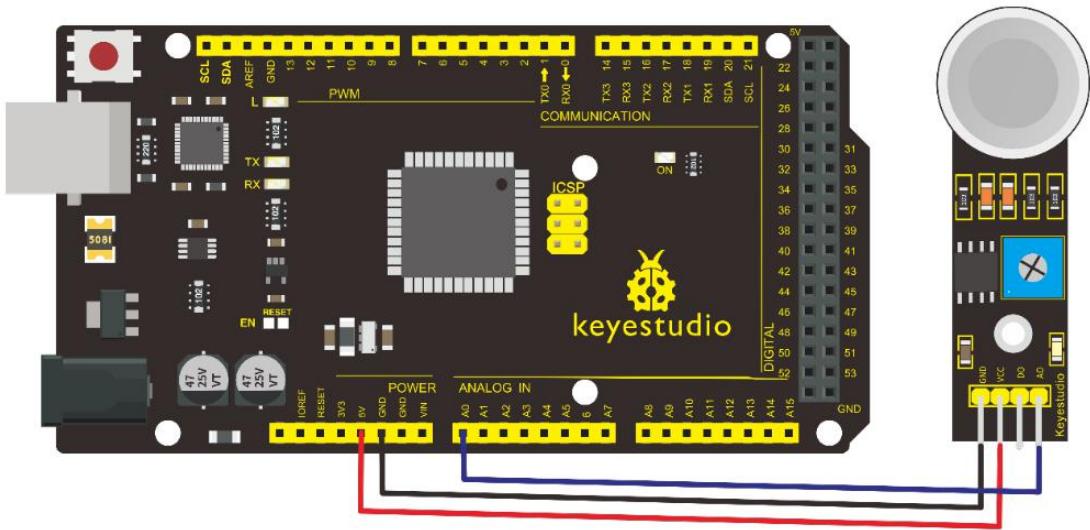
4. схема подключения



Подключение для V4.0



Подключение для MEGA 2560 R3



5. образец кода

```
/*
```

супер обучающий комплект keyestudio

Project 24

Газ

<http://www.keyestudio.com>

```
*/
```

установка void ()

```
{
```

Serial.begin (9600); // Установить скорость последовательной передачи 9600 бит / с

```
}
```

пустой цикл ()

```
{int val;
```

val = analogRead (0); // Считываем значение газа из аналого 0 Serial.println (val, DEC);

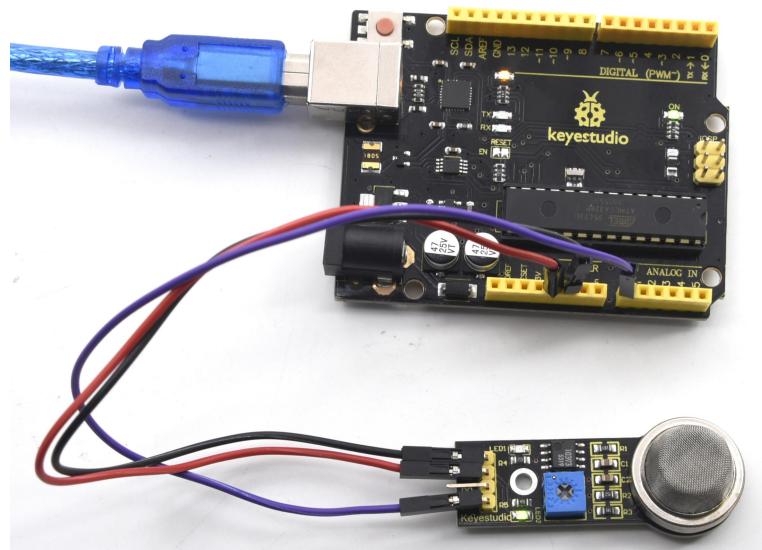
// Выводим значение на последовательный порт

задержка (100);

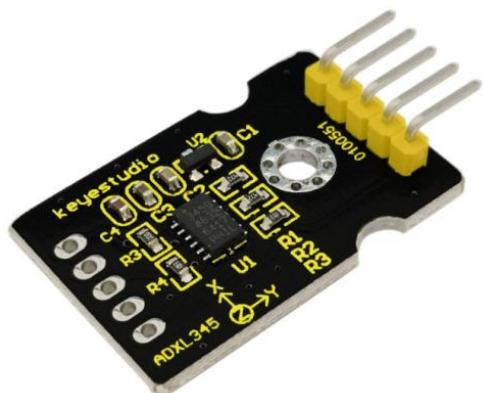
}

||||||||||||||||||||||||||||||||||

6. результат теста



Проект 25: трехосное ускорение ADXL345



1. Введение

ADXL345 - это небольшой, тонкий, маломощный 3-осевой МЭМС-акселерометр с высоким разрешением (13 бит) для измерения при

до + -16 г. Данные цифрового вывода форматируются как 16-битное дополнение до двоек и доступны через цифровой интерфейс SPI (3- или 4-проводный) или I2C.

ADXL345 хорошо подходит для измерения статического ускорения силы тяжести в приложениях для измерения наклона, а также динамического ускорения в результате движения или удара. Его высокое разрешение (4 мг / младший бит) позволяет измерять изменение наклона менее 1,0 градуса.

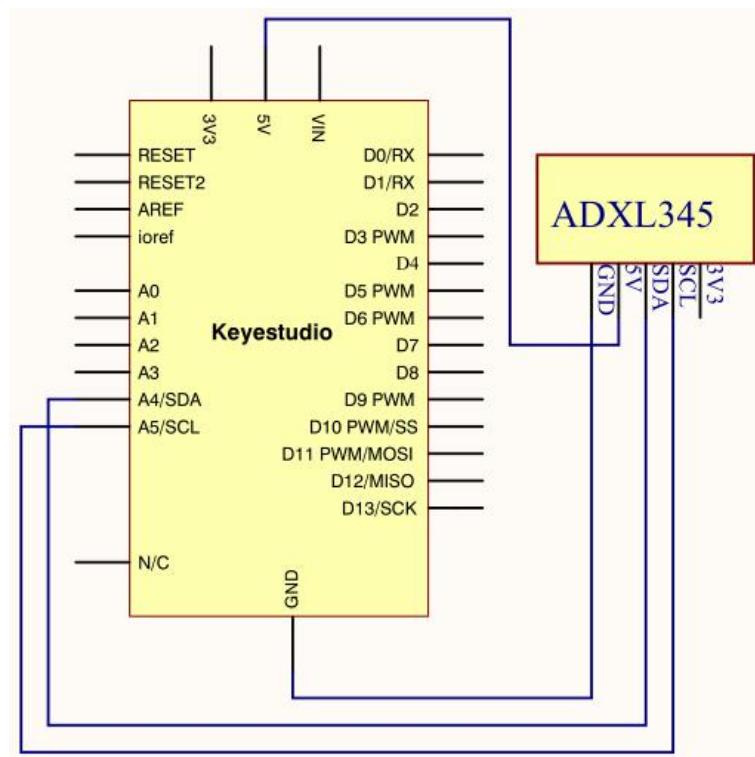
2. Требуется оборудование

- Плата V4.0 или плата MEGA 2650 * 1
- Датчик ADXL345 * 1
- Перемычка * 1
- USB-кабель * 1

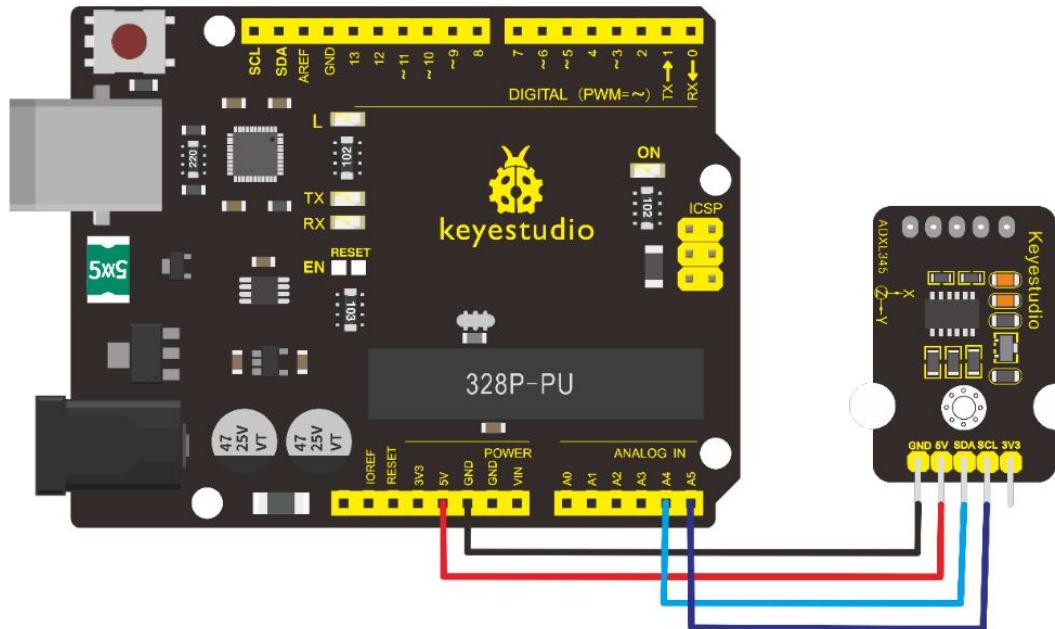
3. Спецификация

- Напряжение питания 2,0-3,6 В постоянного тока
- Сверхнизкое энергопотребление 40 мкА в режиме измерения, 0,1 мкА в режиме ожидания при 2,5 В
- Обнаружение касания / двойного касания
- Обнаружение свободного падения
- Интерфейсы SPI и I2C

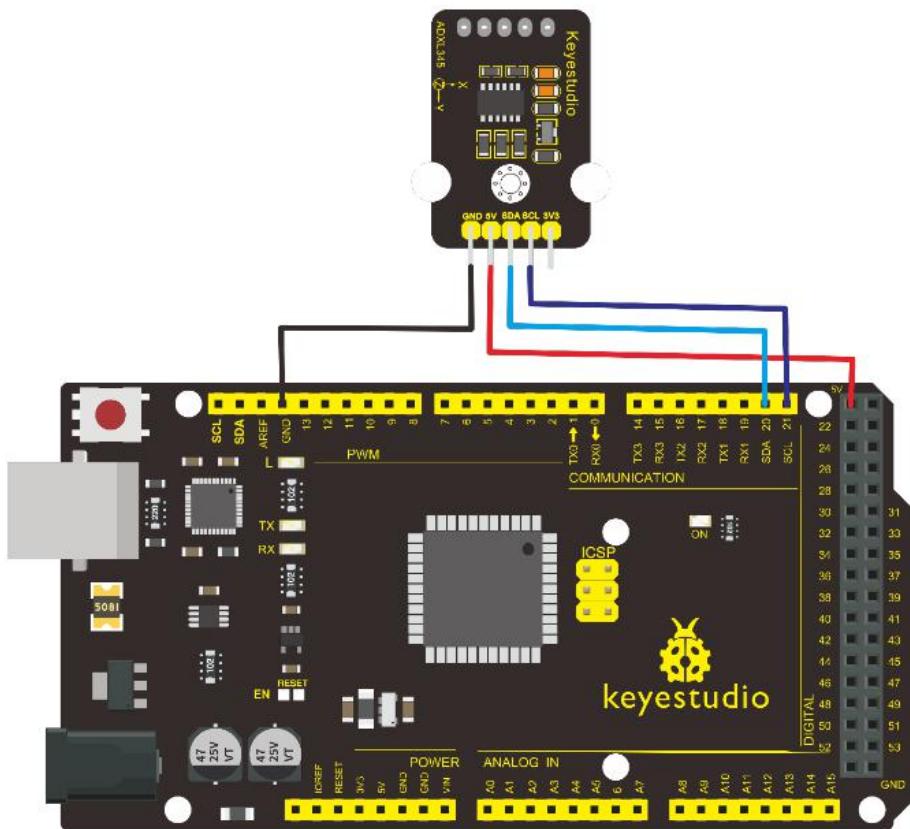
4. схема подключения



Подключение для V4.0



Подключение для MEGA 2560 R3



5. образец кода

```
/*
```

супер обучающий комплект keyestudio

Project 25

ADXL345

<http://www.keyestudio.com>

```
*/
```

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

```
// Регистры для ADXL345
```

```
#определить ADXL345_ADDRESS (0xA6 >> 1) // адрес для
```

устройство 8 бит, но переключитесь на

// вправо на 1 бит до

сделайте это 7 бит, потому что

// только библиотека проводов

принимает 7-битные адреса

определить ADXL345_REGISTER_XLSB (0x32)

int accelerometer_data [3];

// void, потому что это только сообщаетシリアル о необходимости отправки данных в его выходной регистр

// записывает данные в буфер подчиненного устройства

void i2c_write (int адрес, байтовый регистр, байтовые данные) {

// Отправляем адрес выходного регистра

Wire.beginTransmission (address);

// Подключение к устройству

Wire.write (reg);

// Отправляем данные

Wire.write (данные); // младший байт

Wire.endTransmission ();

}

// недействительно, потому что используются указатели

```
// микроконтроллер считывает данные из входного регистра датчика void
i2c_read (int address, byte reg, int count, byte * data) {
    // Используется для чтения количества полученных данных int i = 0;

    // Отправляем адрес входного регистра
    Wire.beginTransmission (address);

    // Подключение к устройству
    Wire.write (reg);

    Wire.endTransmission ();

    // Подключаемся к устройству
    Wire.beginTransmission (адрес);

    // Запрос данных у ведомого
    // Счетчик означает количество байтов для запроса
    Wire.requestFrom (address, count);

    while (Wire.available ()) // ведомое устройство может отправить меньше запрошенного

    {
        char c = Wire.read (); // получаем байт как символьные данные [i] = c;
        i++;
    }
}
```

```

Wire.endTransmission ();

}

void init_adxl345 () {

    байтовые данные = 0;

    i2c_write (ADXL345_ADDRESS, 0x31, 0x0B); // 13-битный

    режим + _ 16g

    i2c_write (ADXL345_ADDRESS, 0x2D, 0x08); // Мощность

    регистр

    i2c_write (ADXL345_ADDRESS, 0x1E, 0x00); // Икс

    i2c_write (ADXL345_ADDRESS, 0x1F, 0x00); // Y

    i2c_write (ADXL345_ADDRESS, 0x20, 0x05); // Z

    // Проверяем, сработало ли это!

    i2c_read (ADXL345_ADDRESS, 0X00, 1 и данные); если (данные ==

    0xE5)

        Serial.println («Успешная работа»);

    еще

        Serial.println («не работает»);

}

```

```
void read_adxl345 () {  
    byte bytes [6];  
  
    memset (байты, 0,6);  
  
    // Считываем 6 байтов из ADXL345  
  
    i2c_read (ADXL345_ADDRESS, ADXL345_REGISTER_XLSB,  
    6, байты);  
  
    // Распаковываем данные  
  
    for (int i = 0; i <3; ++ i) {  
  
        Accelerometer_data [i] = (int) bytes [2 * i] + (((int) bytes [2  
        * i + 1]) << 8);  
  
    }  
}  
  
// инициализируем и запускаем все void  
  
setup () {  
  
    Wire.begin ();  
  
    Serial.begin (9600);  
  
    для (int я = 0; я <3; ++ я)  
  
    {Accelerometer_data [я] = 0;  
  
    }  
  
    init_adxl345 ();
```

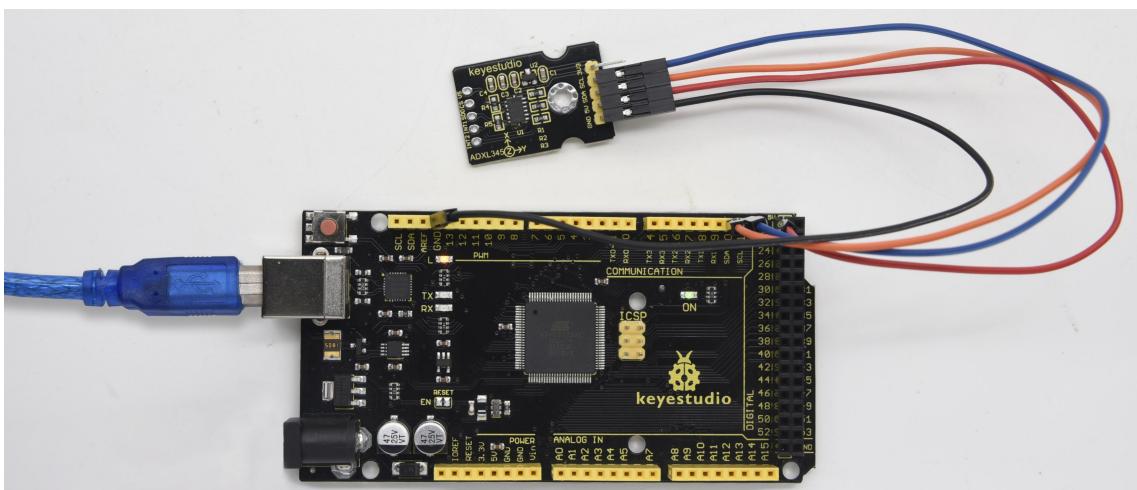
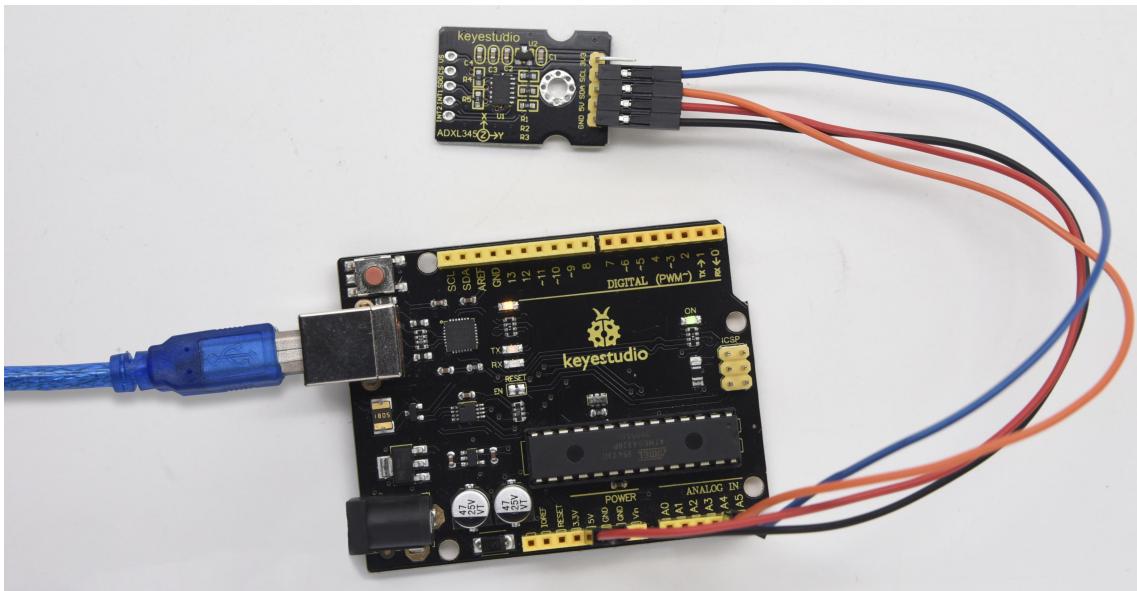
```
}
```

```
void loop () {  
  
    read_adxl345 ();  
  
    Serial.print («ACCEL»);  
  
    Serial.print (float (accelerometer_data [0]) * 3.9 / 1000); // масштабный коэффициент 3,9 мг / младший  
    бит в 13-битном режиме  
  
    Serial.print ("\t");  
  
    Serial.print (float (Accelerometer_data [1]) * 3.9 / 1000);  
  
    Serial.print ("\t");  
  
    Serial.print (float (Accelerometer_data [2]) * 3.9 / 1000);  
  
    Serial.print ("\n");  
  
    задержка (100);  
  
}
```

6. результат теста

Выполните подключение, как показано на схеме выше, и включите питание, затем загрузите код и откройте монитор последовательного порта, он отобразит трехосное ускорение датчика и его состояние, как показано на графике.

ниже.



COM7

```
ACCEL -0.26    1.52   -0.47
ACCEL -0.12    0.87   -1.10
ACCEL -1.13    0.41   1.80
ACCEL -0.51    0.68   0.47
ACCEL -0.39    -0.91  0.39
ACCEL 0.15     0.85   -0.39
ACCEL 0.85     0.77   -1.61
ACCEL 0.12     0.13   -0.46
ACCEL -0.17    -0.11  0.19
ACCEL 0.78     0.93   0.37
ACCEL 0.47     0.51   0.15
ACCEL 0.68     0.88   0.07
ACCEL 0.45     -0.69  -0.21
ACCEL 0.38     0.62   0.18
ACCEL 0.50     2.75   0.04
ACCEL 0.21     1.64   -0.30
ACCEL -0.30    -0.31  0.53
ACCEL -0.79    0.29   0.98
ACCEL 0.07     0.95   -0.01
ACCEL 0.57     1.50   -0.83
ACCEL -0.14    1.42   -0.07
ACCEL -0.74    0.90   0.51
ACCEL -0.53    0.74   0.23
ACCEL -0.29    0.92   -0.01
```

Autoscroll Show timestamp Newline 9600 baud Clear output

Проект 26: Ультразвуковой датчик HC-SR04



1. Введение

Ультразвуковой датчик HC-SR04 - это очень доступный датчик приближения / расстояния, который в основном используется для предотвращения объектов в различных проектах робототехники.

Это, по сути, дает вашим глазам Arduino / пространственную осведомленность и может предотвратить сбой или падение вашего робота со стола. Он также использовался в турелях, датчиках уровня воды и даже в качестве датчика парковки.

В этом простом проекте будет использоваться датчик HC-SR04 с Arduino и эскизом обработки, чтобы обеспечить более интерактивное отображение на экране вашего компьютера.

2. Требуется оборудование

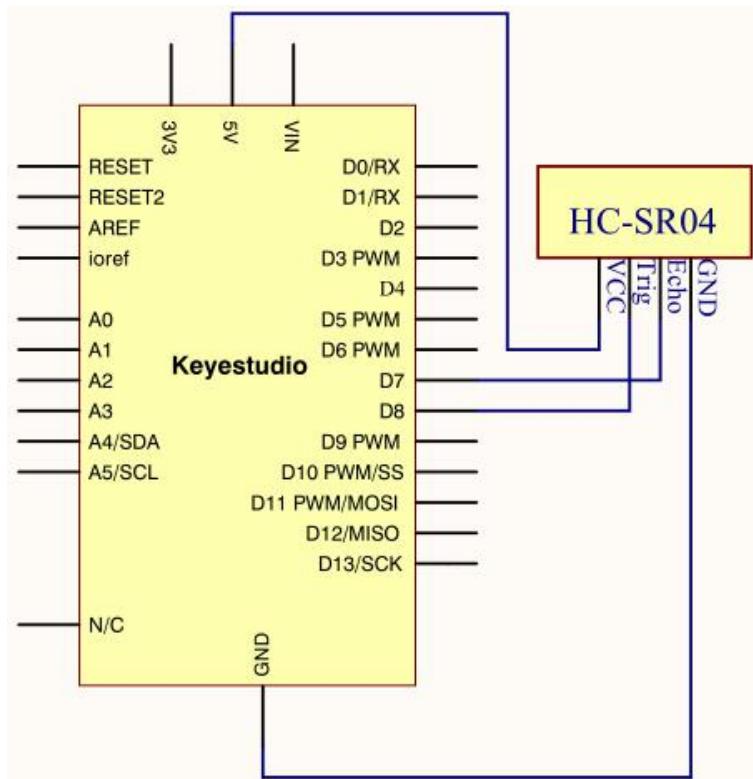
- Плата V4.0 или плата MEGA 2650 * 1

- Ультразвуковой датчик * 1
- Перемычка * 1
- USB-кабель * 1

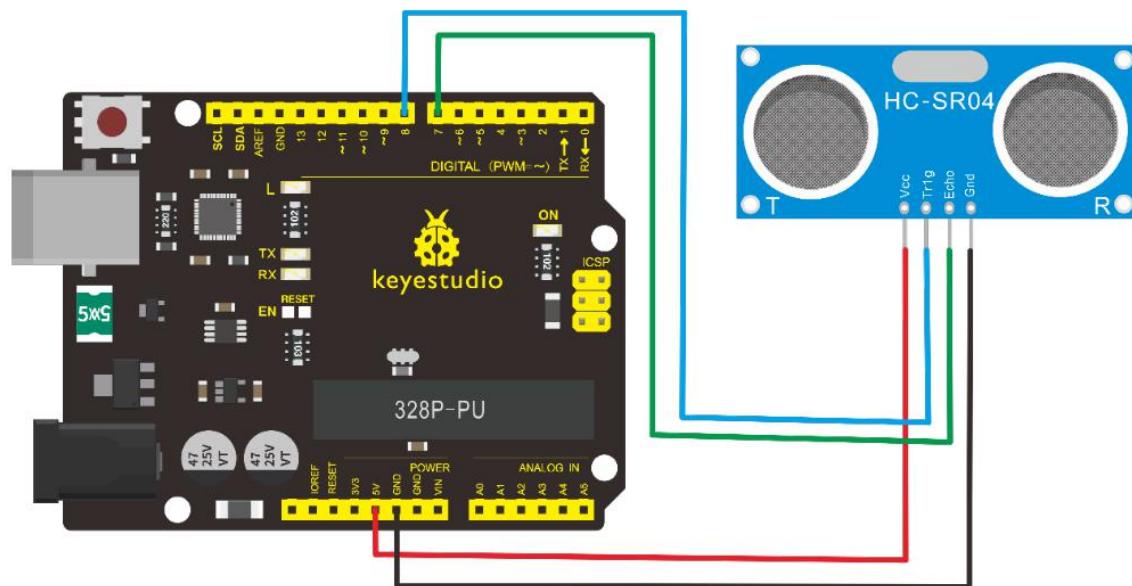
3. Спецификация

- Рабочее напряжение 5 В постоянного тока Рабочий ток 15 мА
- Рабочая частота 40 кГц
- Максимальная дальность 4 м
- Минимальный диапазон 2 см
- Угол измерения 15 градусов
- Входной сигнал триггера Импульс TTL 10 мкс
- Выходной сигнал эхо Входной сигнал рычага TTL и диапазон пропорционально

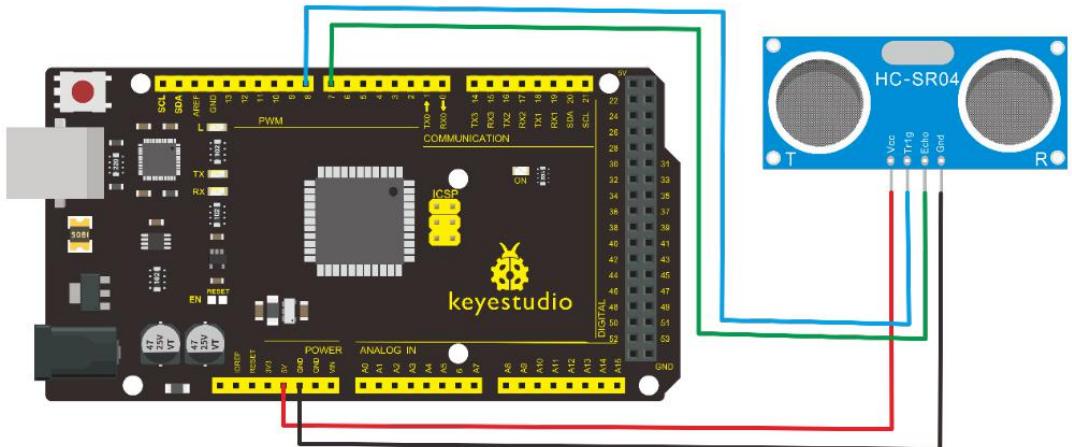
4. схема подключения



Подключение для V4.0



Подключение для MEGA 2560 R3



5. образец кода

```
/*
```

супер обучающий комплект keyestudio

Project 26

Ультразвуковой

<http://www.keyestudio.com>

```
*/
```

```
# define echoPin 7 // Echo Pin
```

```
# define trigPin 8 // Триггерный контакт
```

```
# define LEDPin 13 // Встроенный светодиод
```

```
int maximumRange = 200; // Требуемый максимальный диапазон int minimumRange
```

```
= 0; // Минимальный необходимый диапазон
```

большая продолжительность, расстояние; // Длительность, используемая для расчета расстояния void

```
setup () {
```

```
Serial.begin (9600);
```

```
pinMode (trigPin, ВЫХОД);
```

```
pinMode (echoPin, ВХОД);

pinMode (LEDPin, ВЫХОД); // Используем светодиодный индикатор (при необходимости)

}

void loop () {

/* Следующий цикл trigPin / echoPin используется для определения

расстояние до ближайшего объекта путем отражения от него звуковых волн. */

digitalWrite (trigPin, LOW);

delayMicroseconds (2);

digitalWrite (trigPin, HIGH);

delayMicroseconds (10);

digitalWrite (trigPin, LOW);

duration = pulseIn (echoPin, HIGH);

// Вычислить расстояние (в см) на основе скорости звука.

расстояние = продолжительность / 58,2;

if (distance> = maximumRange || distance <= minimumRange) {

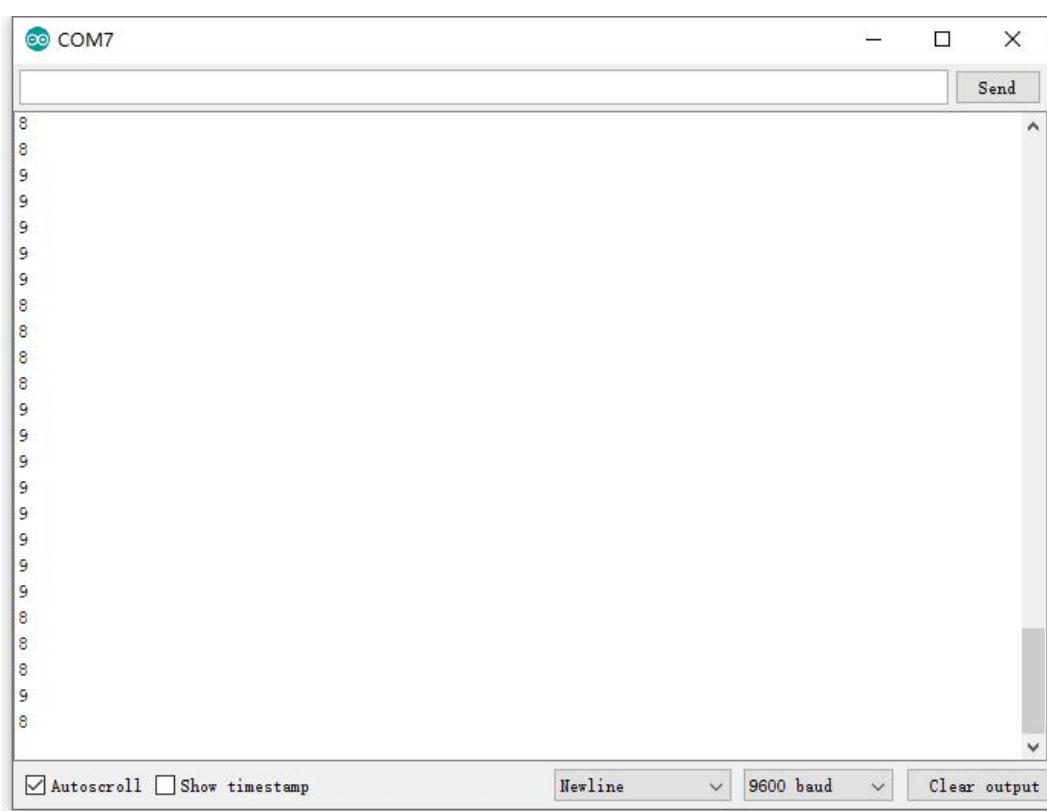
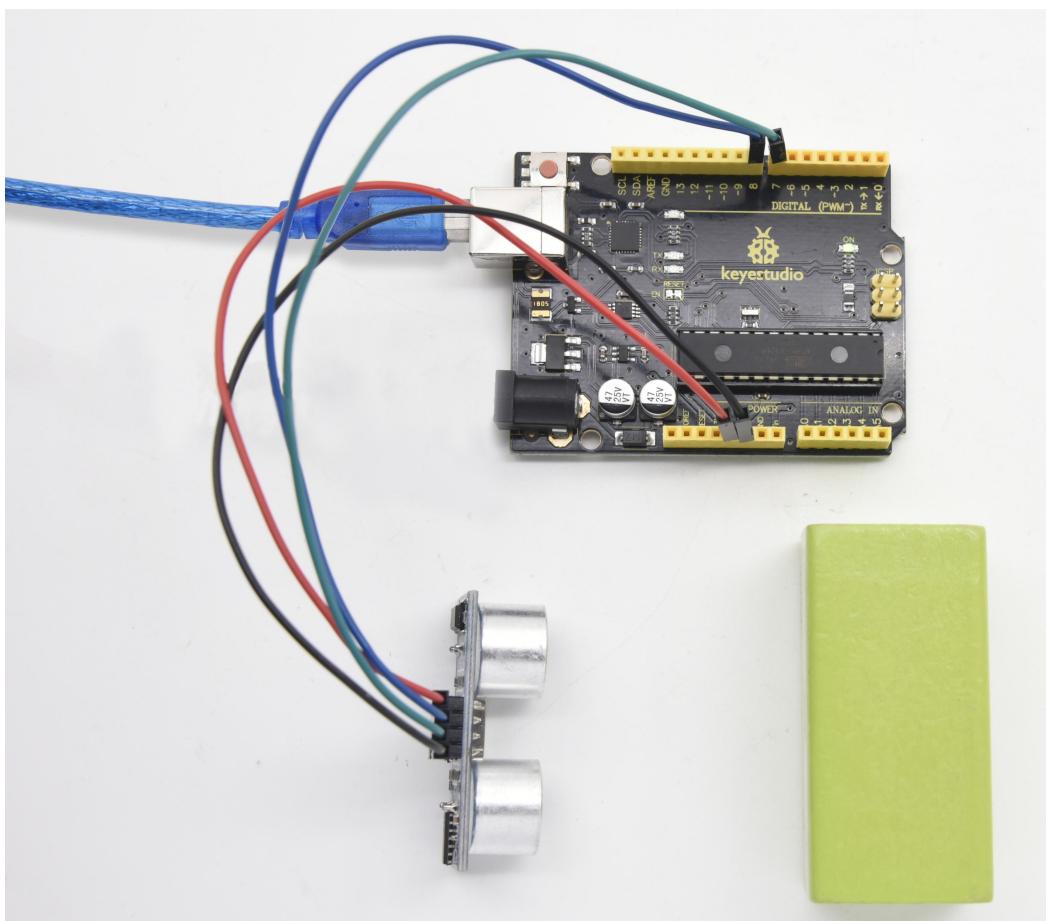
/* Отправить отрицательное число на компьютер и включить светодиод
```

```
для обозначения "вне допустимого диапазона" *  
  
/ Serial.println ("- 1");  
  
digitalWrite (LEDPin, HIGH);  
  
}  
  
else {  
  
/* Отправляем расстояние на компьютер по последовательному протоколу и  
  
выключите светодиод, чтобы указать успешное чтение. */ Serial.println  
(расстояние);  
  
digitalWrite (LEDPin, LOW);  
  
}  
  
// Задержка 50 мс перед следующим чтением. задержка  
(50);  
  
}  
  
||||||||||||||||||||||||||||||||||||
```

6. результат теста

После загрузки кода на плату V4.0 откройте монитор последовательного порта.

Когда объект помещается перед ультразвуковым датчиком (близко и далеко), он определяет расстояние до объекта. Значение будет отображаться на мониторе.



Проект 27: модуль джойстика



1. Введение

Для многих проектов роботов нужен джойстик. Этот модуль обеспечивает доступное решение.

Просто подключившись к двум аналоговым входам, робот будет выполнять ваши команды с

помощью управления X, Y. Он также имеет переключатель, который подключен к цифровому выводу.

Этот модуль джойстика можно легко подключить к Arduino с помощью IO Shield. Этот модуль предназначен для Arduino (V5) с кабелем в комплекте.

2. Требуется оборудование

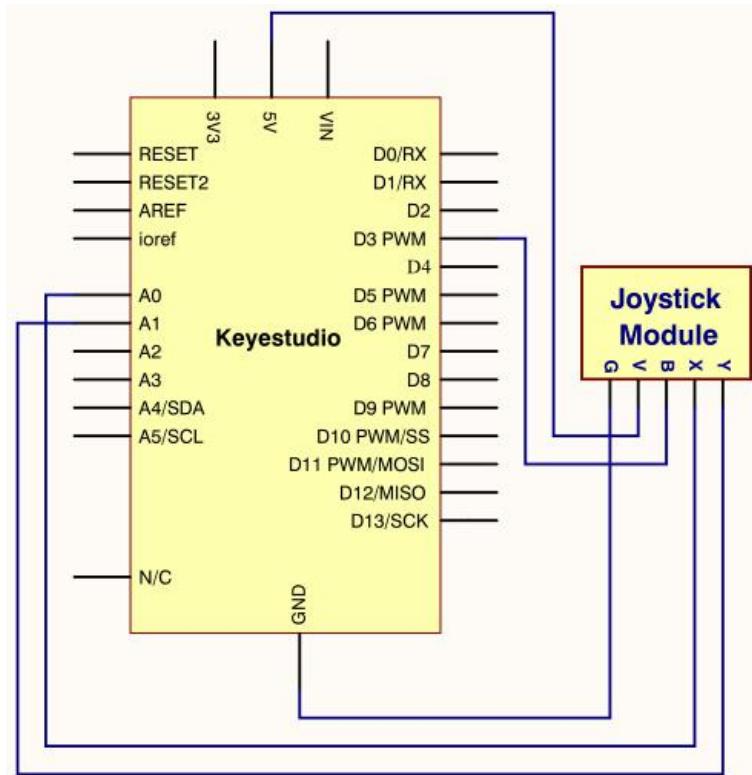
- Плата V4.0 или плата MEGA 2650 * 1
- Модуль джойстика * 1
- Перемычка * 1
- USB-кабель * 1

3. Спецификация

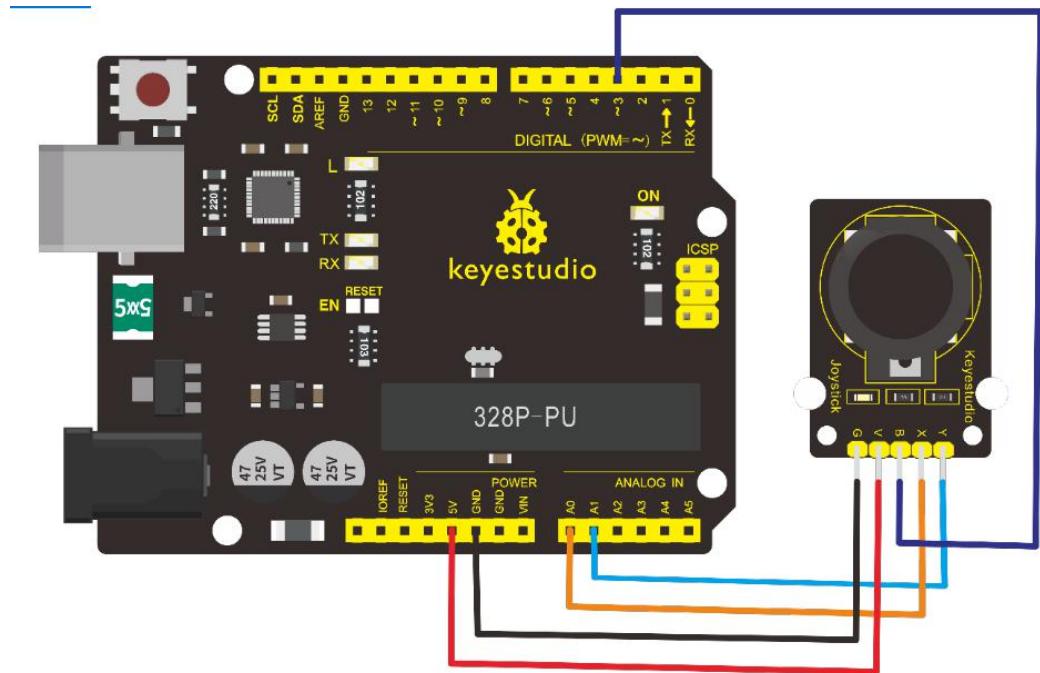
- Напряжение питания от 3,3 В до 5 В

- Интерфейс Аналоговый x2, Цифровой x1

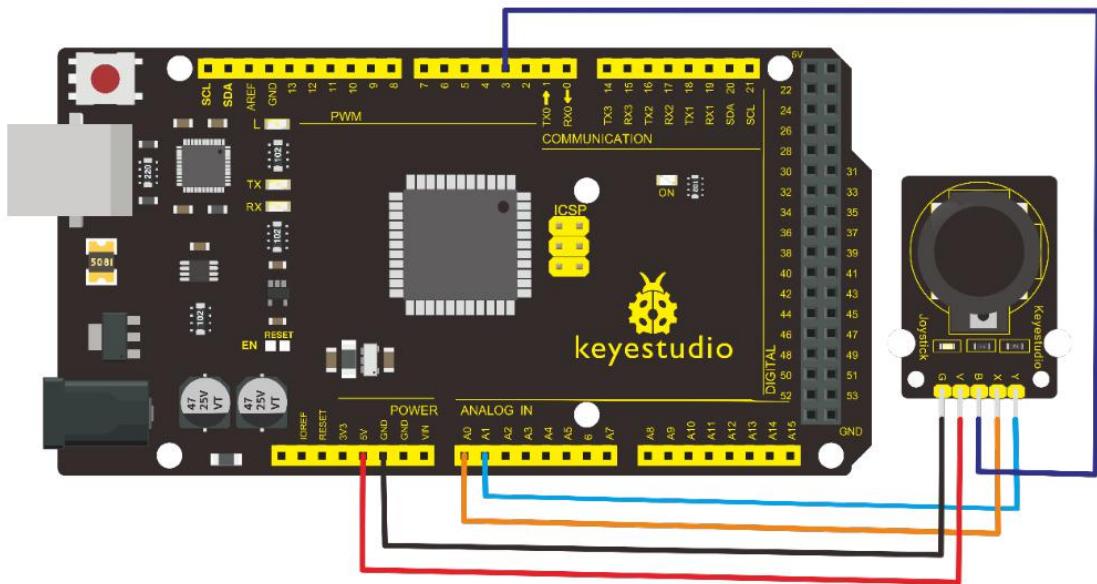
4. схема подключения



Подключение для V4.0



Подключение для MEGA 2560 R3



5. образец кода

```
/*
```

```
keyestudio супер обучающий комплект
```

```
Project 27
```

```
джойстик
```

```
http://www.keyestudio.com
```

```
*/
```

```
int JoyStick_X = 0; // x int
```

```
JoyStick_Y = 1; // y int JoyStick_Z
```

```
= 3; // ключ void setup ()
```

```
{
```

```
pinMode (JoyStick_Z, INPUT);

Serial.begin (9600); // 9600 бит / с

}

пустой цикл ()

{

int x, y, z;

x = analogRead (JoyStick_X);

y = analogRead (JoyStick_Y);

z = digitalRead (JoyStick_Z);

Serial.print (x, DEC);

Serial.print (",");

Serial.print (y, DEC);

Serial.print (",");

Serial.println (z, DEC);

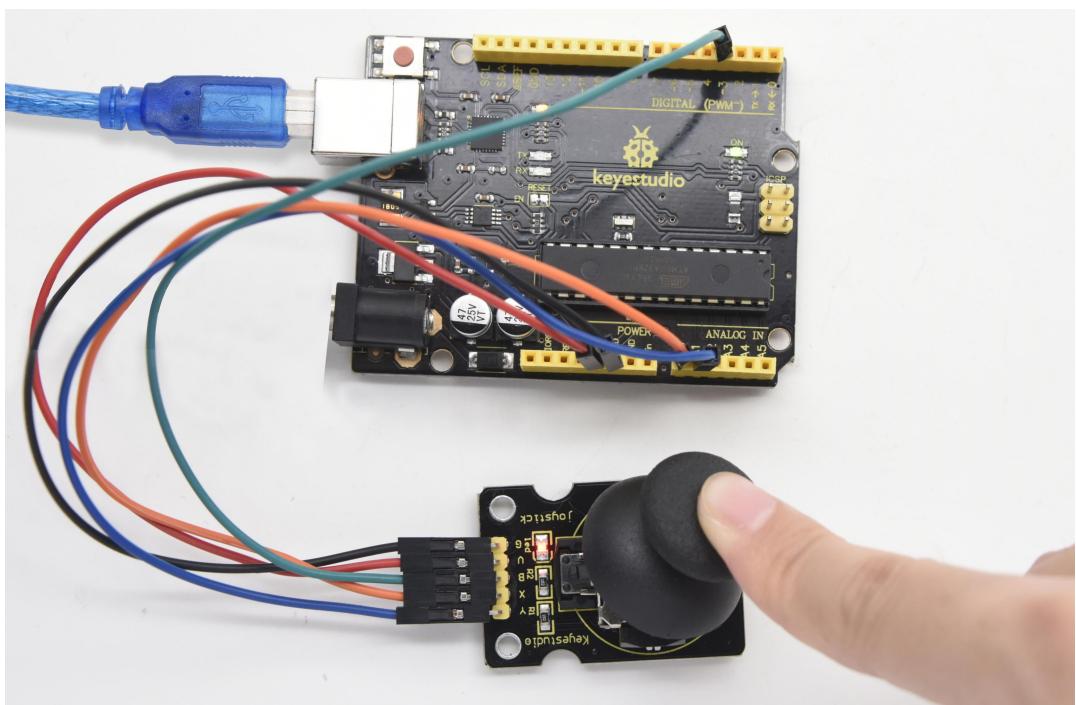
задержка (100);

}

///////////////////////////////
```

6. результат теста

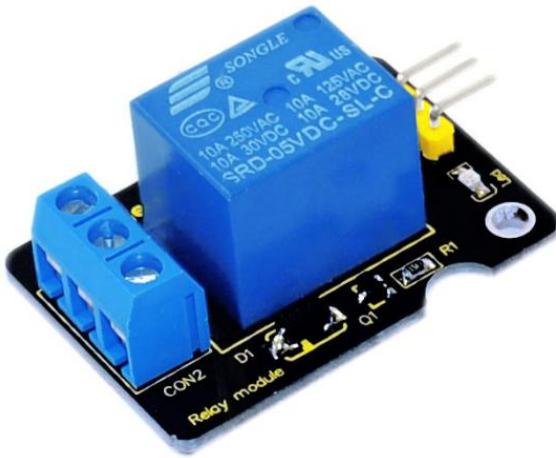
Правильно подключив проводку и загрузив код, откройте последовательный монитор и установите скорость передачи на 9600, нажмите джойстик, вы увидите значение, показанное ниже.



```
COM7
Send
517,1023,0
1023,503,1
633,0,0
517,0,0
517,503,1
518,1022,1
518,0,1
736,0,0
517,1023,0
726,1022,0
912,502,0
643,0,0
517,1023,0
517,857,1
673,0,0
518,0,0
517,503,1
517,1023,0
275,1022,0
0,502,0
667,0,0
517,835,1
887,1023,0
1023,1023,0
```

Autoscroll Show timestamp Newline 9600 baud Clear output

Проект 28: Модуль реле 5В



1. Введение

Этот модуль с одним реле можно использовать в интерактивных проектах. Этот модуль

использует высококачественное реле SONGLE 5v.

Его также можно использовать для управления освещением, электрическим и другим

оборудованием. Модульная конструкция позволяет легко расширить с помощью платы Arduino (не

входит в комплект).

Выход реле - светодиод. Им можно управлять через цифровой порт

ввода-вывода, например, соленоидные клапаны, лампы, двигатели и другие

устройства с высоким током или напряжением.

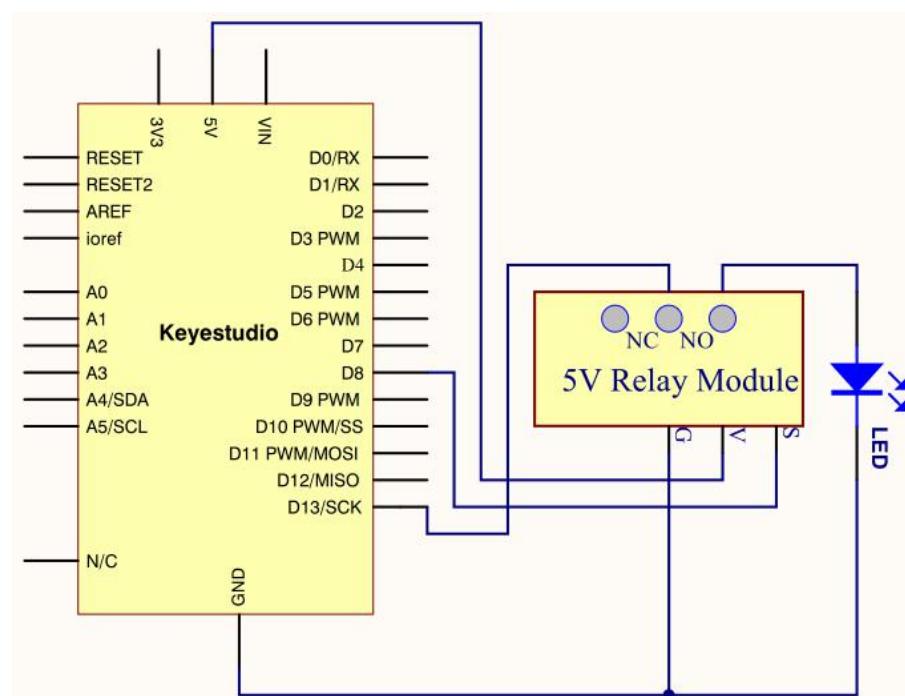
2. Требуется оборудование

- Плата V4.0 или плата MEGA 2650 * 1
- Модуль реле * 1
- Перемычка * 1
- USB-кабель * 1

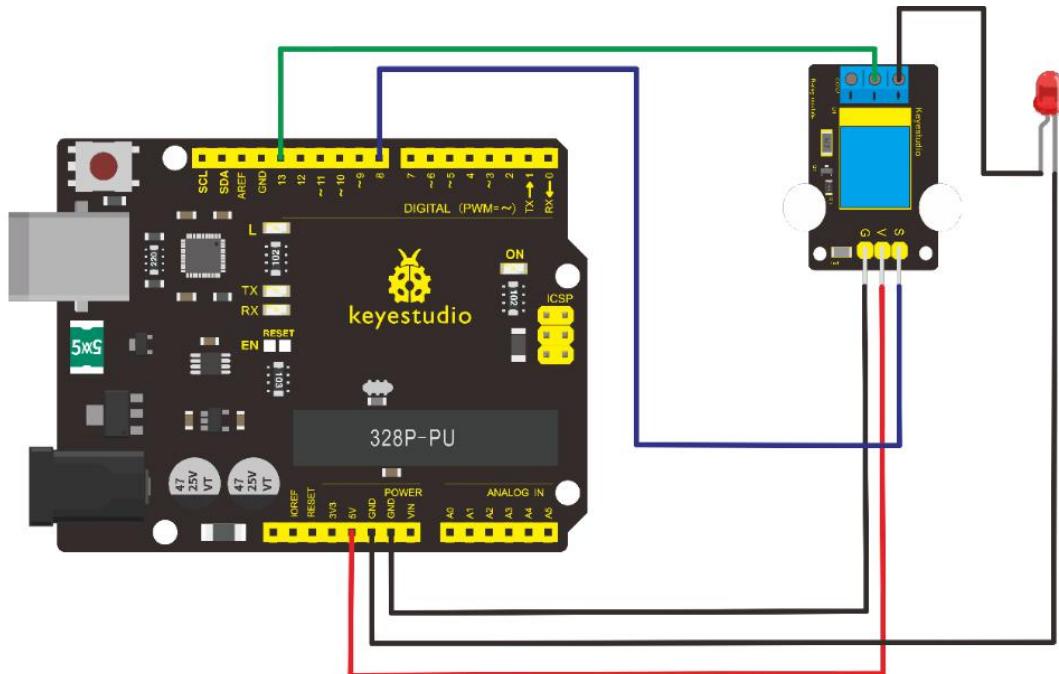
3. Спецификация

- Тип Цифровой
- Номинальный ток 10A (NO) 5A (NC)
- Максимальное коммутируемое напряжение 150VAC 24VDC Цифровой
- интерфейс
- Уровень сигнала управления TTL
- Номинальная нагрузка 8A 150VAC (NO) 10A 24VDC (NO), 5A 250VAC (NO / NC) 5A 24VDC (NO / NC)
- Максимальная коммутируемая мощность AC1200VA DC240W (NO) AC625VA DC120W (NC)
- Время действия контакта 10 мс

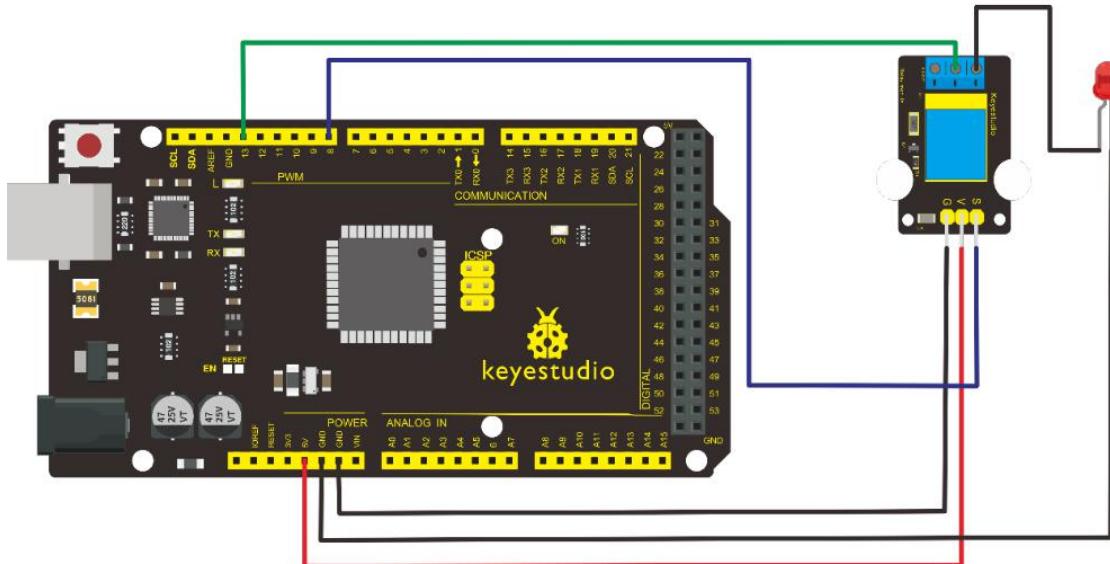
4. схема подключения



Подключение для V4.0



Подключение для MEGA 2560 R3



5. образец кода

/*

супер обучающий комплект keyestudio

Project 28

Реле

<http://www.keyestudio.com>

* /

int Relay = 8;

установка void ()

{

pinMode (13, ВЫХОД); // Установить вывод 13 как выход //

digitalWrite (13, ВЫСОКИЙ); Установить высокий вывод 13

pinMode (Реле, ВЫХОД); // Устанавливаем Pin8 как выход

}

пустой цикл ()

{

digitalWrite (реле, ВЫСОКИЙ); // Выключаем реле задержки

(2000);

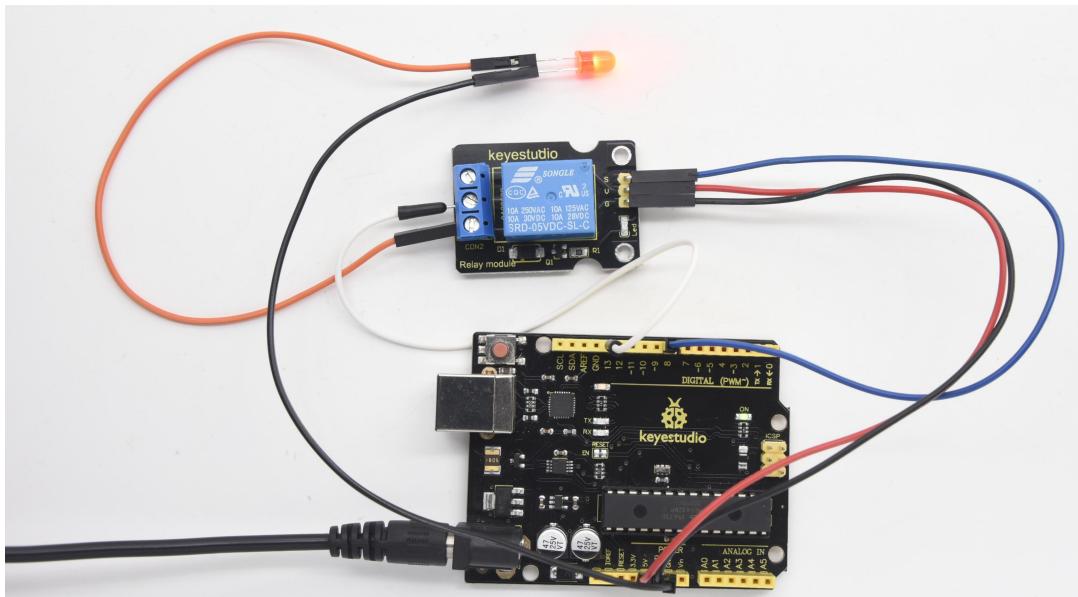
digitalWrite (реле, LOW); // Включаем реле

задержка (2000);

}

||||||||||||||||||||||||||||||||||

6. результат теста



Проект 29: Модуль часов DS3231



1. Введение

DS3231 оснащен встроенным TCXO и кристаллом, что делает его

экономичными часами реального времени I2C с высокой точностью.

Устройство имеет вход батареи, поэтому даже если вы отключите основной источник

питания, оно все равно сможет поддерживать точность

сроки. В интегрированный осциллятор обеспечивает то

долговременная точность устройства и уменьшение количества компонентов.

DS3231 обеспечивает и то и другое коммерческий и

промышленный диапазон температур и поддерживает 16-контактный

мелкоконтрастный корпус (300 мил).

Сам модуль может адаптироваться к системе 3,3 В и 5 В без переключателя уровня, что довольно удобно!

2. Требуется оборудование

- Плата V4.0 или плата MEGA 2650 * 1
- Модуль часов DS3231 * 1
- Перемычка * 1
- USB-кабель * 1

3. Спецификация

- Температурный диапазон от -40 до +85;
- Точность синхронизации $\pm 5 \text{ ppm}$ ($\pm 0,432$ секунды / день) Обеспечивает резервное
- питание от батареи для непрерывного отсчета времени Низкое энергопотребление
-
- Пакет и функции устройства, совместимые с DS3231 Полная функция
- календаря часов содержит секунды и минуты, час, неделю, дату, месяц и год, а также

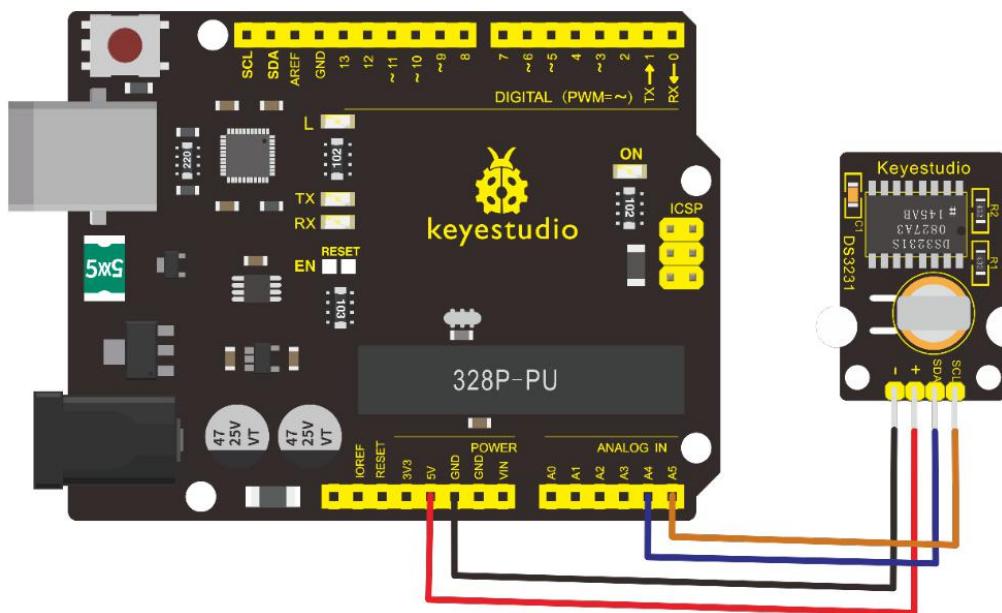
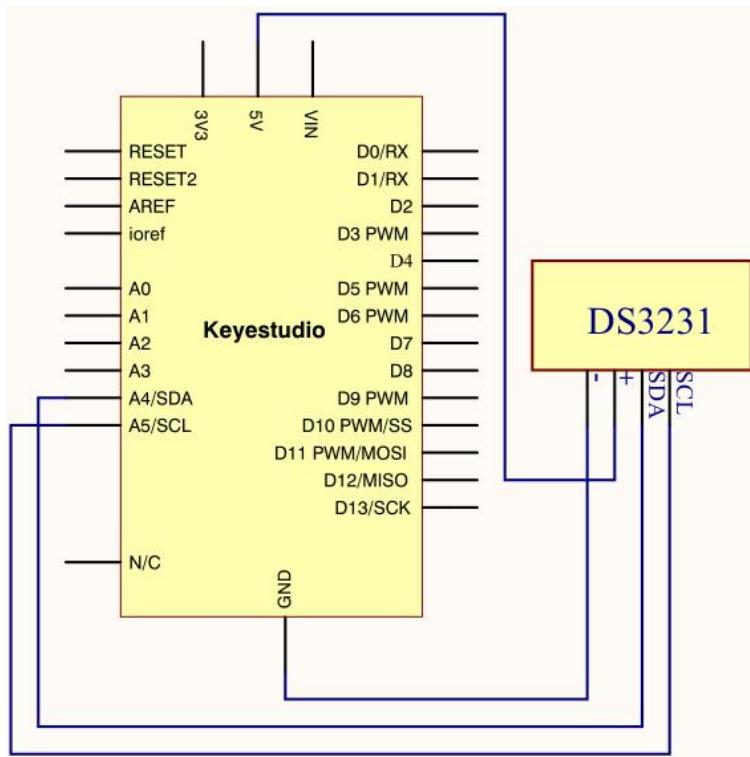
обеспечивает компенсацию високосного года до 2100 года. Два

- календарных часа
- Выход 1 Гц и 32,768 кГц
- Сброс выхода и входной дребезг кнопки Высокая скорость (400 кГц),
- последовательная шина I2C
- Напряжение питания от + 3,3 В до + 5,5 В
- Цифровой датчик температуры с точностью до ± 3 °C Рабочая
- температура от -40 ~ C до +85 ~ C 16 контактов Small Outline
- Package (300mil)

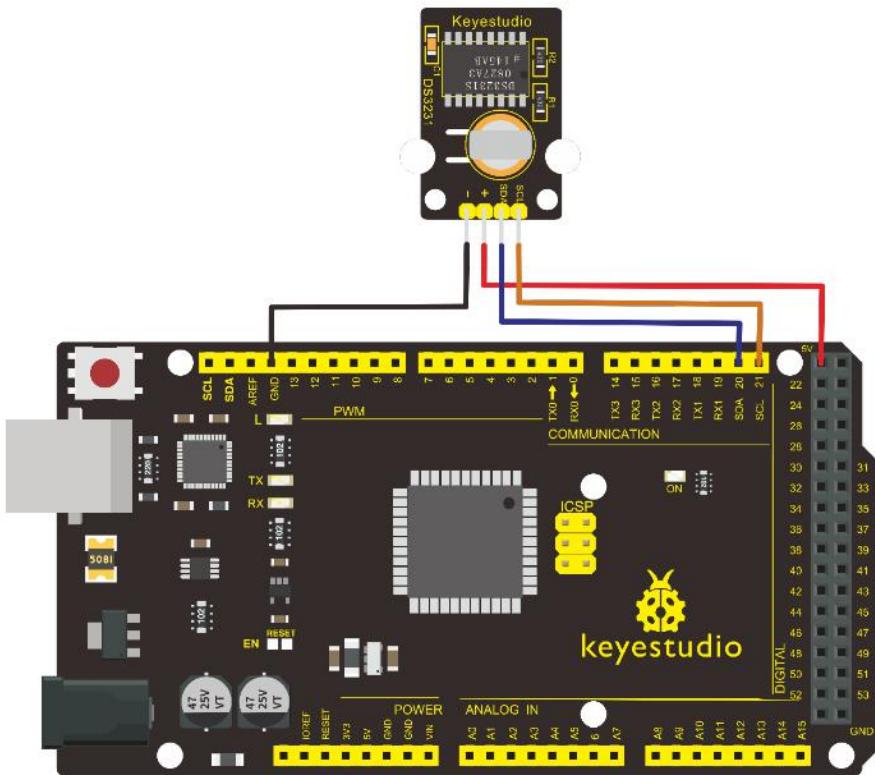
4. схема подключения

Этот модуль использует метод тестирования IIC, поэтому нужно только подключить SDA к Arduino A4, SCL к A5, + к VCC и - к GND следующим образом

Подключение для V4.0



Подключение для MEGA 2560 R3



5. образец кода

Перед компиляцией кода лучше поставить Библиотека DS3231

— под файлом в каталог Arduino.

/*

супер обучающий комплект keyestudio

Project 29

Часы

<http://www.keyestudio.com>

*/

include <Wire.h>

включить "DS3231.h"

DS3231 RTC; // Создаем объект DS3231

```
char weekDay [] [4] = {«Вс», «Пн», «Вт», «Ср», «Чт», «Пт», «Сб»};  
  
// год, месяц, число, час, мин, секунда и день недели (начинается с 0 и продолжается до  
6)  
  
// запись любых несуществующих временных данных может помешать  
нормальной работе RTC.  
  
// Позаботьтесь также о рабочем дне.  
  
DateTime dt (2011, 11, 10, 15, 18, 0, 5); // открываем последовательный порт, и вы  
можете проверить здесь время или изменить время по мере необходимости.
```

```
установка void ()  
  
{      Serial.begin (57600); // устанавливаем скорость передачи 57600 Wire.begin  
       ();  
  
       RTC.begin ();  
  
       RTC.adjust (dt); // Настраиваем дату и время, как указано выше 'dt'  
}  
  
пустой цикл ()  
  
{  
  
       DateTime now = RTC.now (); // получаем текущую дату и время  
  
       Serial.print (now.year (), DEC);  
  
       Serial.print ("/");  
  
       Serial.print (now.month (), DEC);
```

```
Serial.print ("\");

Serial.print (now.date (), DEC);

Serial.print ("\");

Serial.print (now.hour (), DEC);

Serial.print (":");

Serial.print (now.minute (), DEC);

Serial.print (":");

Serial.print (now.second (), DEC);

Serial.println ();

Serial.print (weekDay [now.dayOfWeek ()]);

Serial.println ();

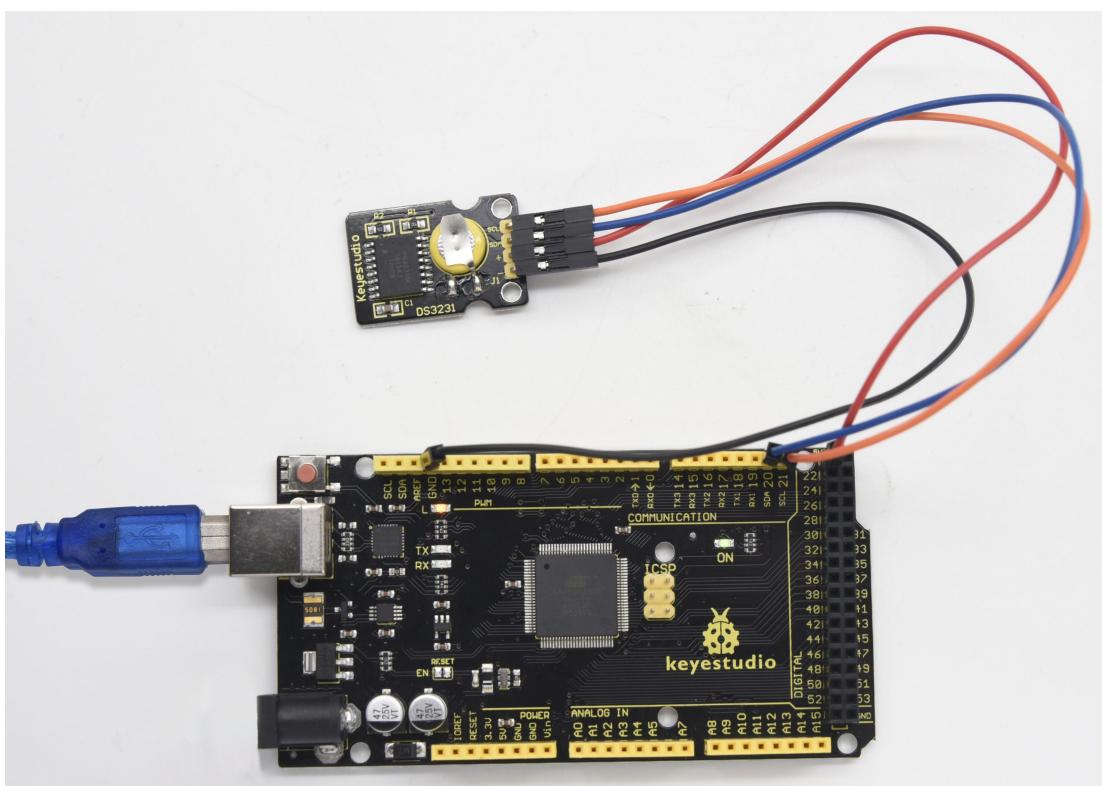
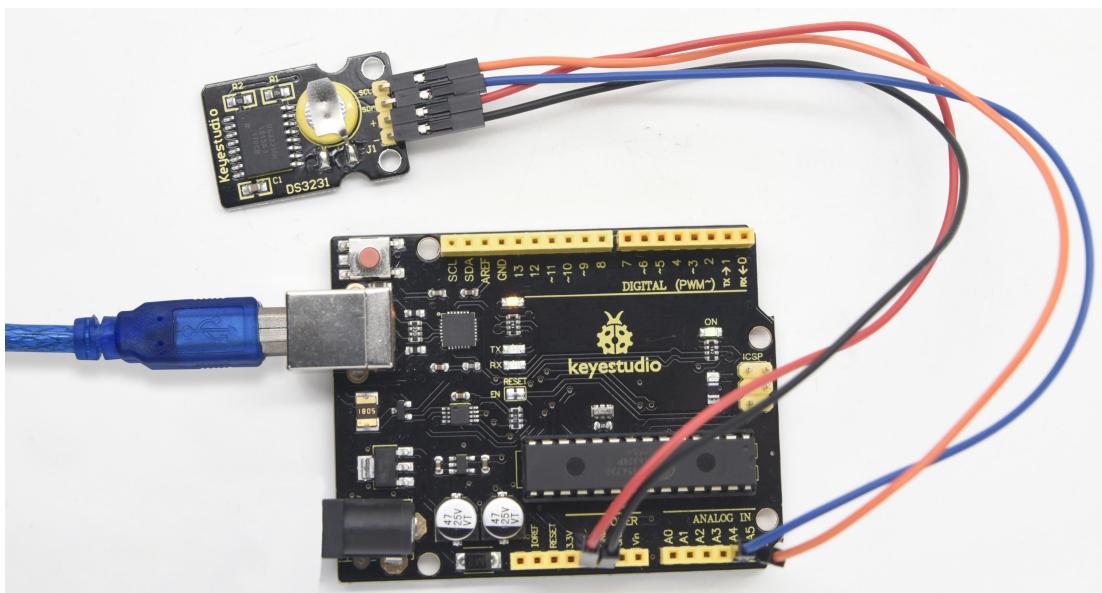
задержка (1000);

}

//////////
```

6. результат теста

Готово загрузить код в Arduino, открыть последовательный монитор и получить следующие результаты

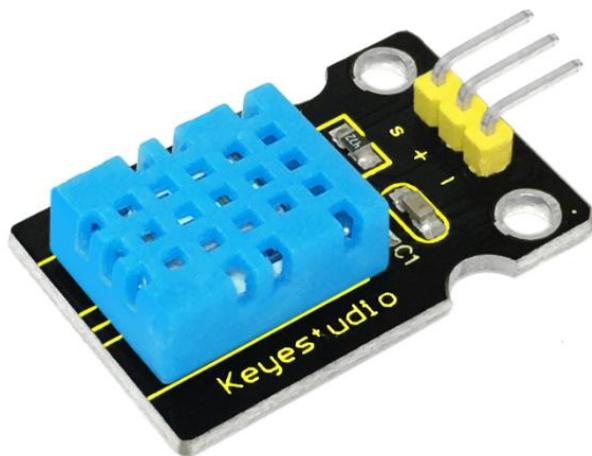


The screenshot shows a terminal window titled "COM7" with the following content:

```
2011/11/10 15:18:0
Fri
2011/11/10 15:18:1
Fri
2011/11/10 15:18:2
Fri
2011/11/10 15:18:3
Fri
2011/11/10 15:18:4
Fri
2011/11/10 15:18:5
Fri
2011/11/10 15:18:6
Fri
2011/11/10 15:18:7
Fri
2011/11/10 15:18:8
Fri
2011/11/10 15:18:9
Fri
2011/11/10 15:18:10
Fri
2011/11/10 15:18:11
Fri
```

At the bottom of the window, there are several control buttons: "Autoscroll" (checked), "Show timestamp" (unchecked), "Newline" (dropdown menu), "57600 baud" (dropdown menu), and "Clear output".

Проект 30: Датчик температуры и влажности DHT11



1. Введение

Этот датчик температуры и влажности DHT11 имеет

откалиброванный цифровой сигнальный выход с комплексом датчиков температуры и

влажности. Его технология обеспечивает высокую надежность

и превосходно долгосрочный стабильность. А

подключен высокопроизводительный 8-битный микроконтроллер.

Этот датчик включает в себя резистивный элемент и чувствительные устройства для

измерения температуры NTC. Он обладает такими преимуществами, как отличное

качество, быстрая реакция, помехоустойчивость и высокая стоимость.

Каждый датчик DHT11 имеет чрезвычайно точные данные калибровки камеры

калибровки влажности. Коэффициенты калибровки хранятся в памяти

программы OTP, внутренние датчики обнаруживают сигналы в процессе, и мы

должны называть эти коэффициенты калибровки.

Система однопроводного последовательного интерфейса интегрирована, чтобы сделать

это быстро и легко. Небольшие размеры, низкое энергопотребление и дальность передачи

сигнала 20 метров делают его широко применяемым и даже наиболее требовательным

приложением. Удобное подключение и специальный пакет могут быть предоставлены в

соответствии с вашими потребностями.

2. Требуется оборудование

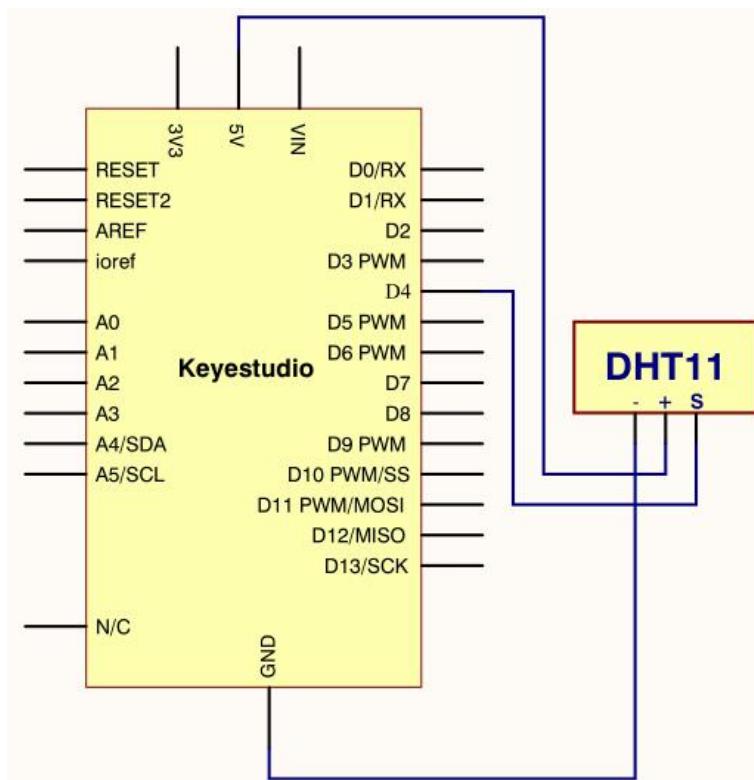
- Плата V4.0 или плата MEGA 2650 * 1
- Датчик DHT11 * 1

- Перемычка * 1
- USB-кабель * 1

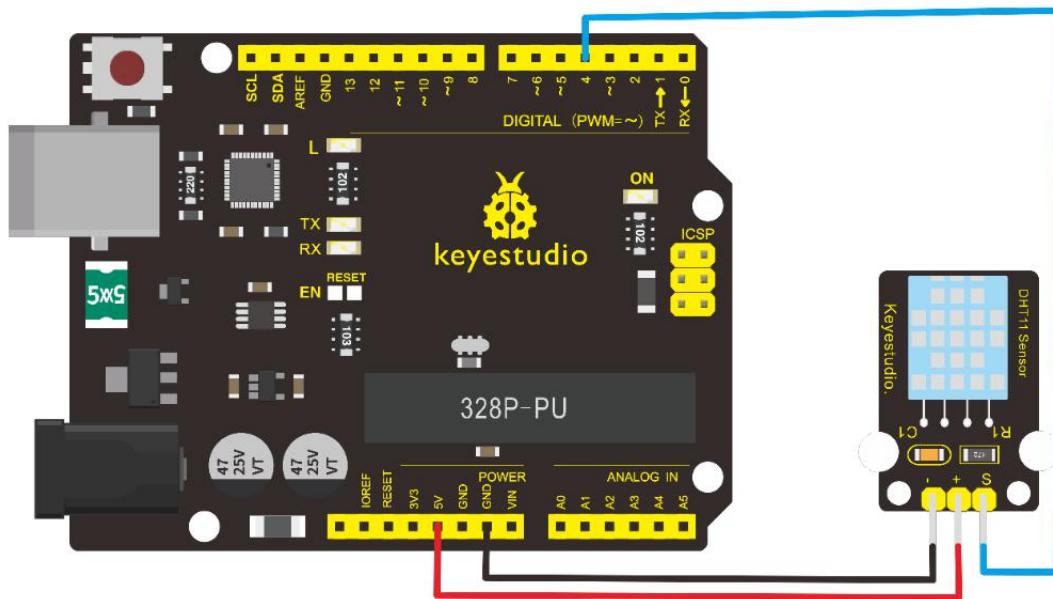
3. Спецификация

- Напряжение питания +5 В
- Диапазон температур 0-50 ° С погрешность ± 2 ° С
- Влажность 20-90% RH ± 5% RH ошибка
- Цифровой интерфейс

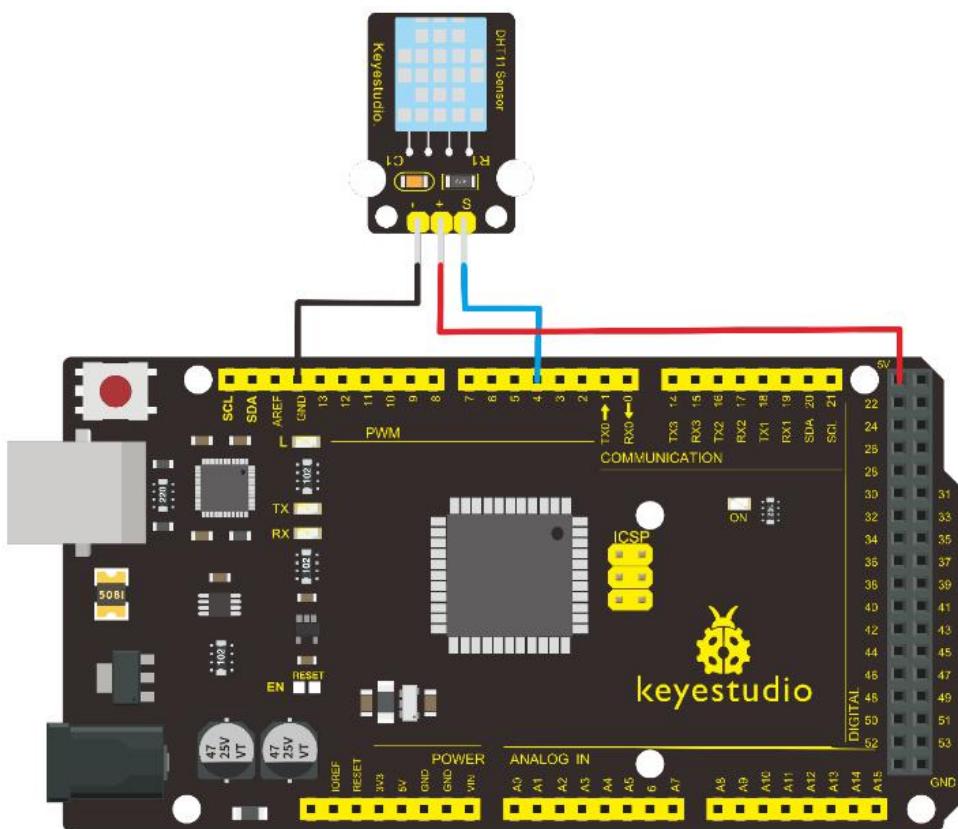
4. схема подключения



Подключение для V4.0



Подключение для MEGA 2560 R3



5. образец кода

Пожалуйста, скачайте [DHT11Lib](#) во-первых. Или же [см. сайт](#)

```
/*
```

```
супер обучающий комплект keyestudio
```

```
Project 30
```

```
DHT11
```

```
http // www.keyestudio.com
```

```
* /
```

```
# включить <dht11.h>
```

```
dht11 DHT;
```

```
# определить DHT11_PIN 4
```

```
void setup () {
```

```
Serial.begin (9600);
```

```
Serial.println («ТЕСТОВАЯ ПРОГРАММА DHT»);
```

```
Serial.print ("ВЕРСИЯ БИБЛИОТЕКИ:");
```

```
Serial.println (DHT11LIB_VERSION);
```

```
Serial.println ();
```

```
Serial.println ("Тип, \tstatus, \tHumidity  
(%), \tTemperatura (C ");
```

```
}
```

```
void loop () {
```

```
int chk;
```

```
Serial.print ("DHT11, \t");

chk = DHT.read (DHT11_PIN);           // ЧИТАТЬ ДАННЫЕ

switch (chk) {

    case DHTLIB_OK:

        Serial.print ("OK, \t");

        сломать;

    case DHTLIB_ERROR_CHECKSUM:

        Serial.print ("Ошибка контрольной суммы, \t");

        сломать;

    case DHTLIB_ERROR_TIMEOUT:

        Serial.print ("Ошибка тайм-аута, \t");

        сломать;

    по умолчанию:

        Serial.print ("Неизвестная ошибка, \t");

        сломать;

}

// ОТОБРАЖЕНИЕ ДАННЫХ

Серийный отпечаток (DHT.влажность, 1);

Serial.print (", \t");

Serial.println (DHT.температура, 1);

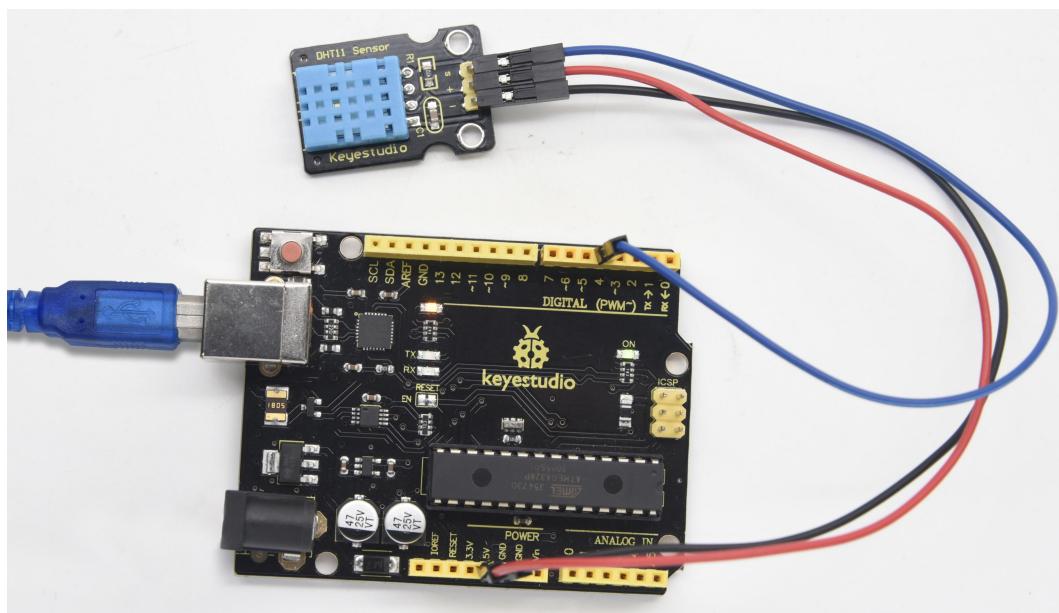
задержка (1000);
```

}

||||||||||||||||||||||||||||||

6. результат теста

Хорошо подключите его и загрузите приведенный выше код на плату V4.0.



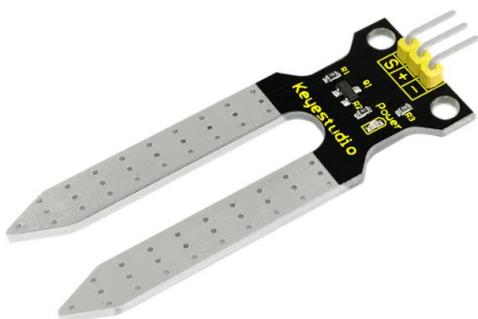
Затем откройте монитор последовательного порта и установите скорость передачи на 9600, вы увидите текущее значение температуры и влажности.

```
LIBRARY VERSION: 0.4.1
Type, status, Humidity (%), Temperature (C)
DHT11, Checksum error, 68, 33
DHT11, Time out error, 68, 33
DHT11, Checksum error, 68, 33
DHT11, Time out error, 68, 33
DHT11, Checksum error, 68, 33
DHT11, Time out error, 68, 33
DHT11, Checksum error, 67, 33
DHT11, Time out error, 67, 33
DHT11, Checksum error, 66, 33
DHT11, Time out error, 66, 33
DHT11, Checksum error, 65, 33
DHT11, Time out error, 65, 33
DHT11, Checksum error, 65, 33
DHT11, Time out error, 65, 33
DHT11, Checksum error, 64, 33
DHT11, Time out error, 64, 33
DHT11, Checksum error, 64, 33
DHT11, Time out error, 64, 33
DHT11, Checksum error, 64, 33
DHT11, Time out error, 64, 33
DHT11, Checksum error, 64, 33
DHT11, Time out error, 64, 33
```

< >

Autoscroll Show timestamp Newline 9600 baud Clear output

Проект 31: Датчик влажности почвы



1. Введение

Это простой датчик влажности почвы, предназначенный для определения влажности почвы.

Если в почве нет воды, аналоговое значение выводится

по датчику уменьшится; в противном случае он увеличится.

Если вы используете этот датчик для автоматического полива, он может определить, хочет ли ваша ботаника пить, чтобы предотвратить ее увядание, когда вы выходите на улицу. Использование датчика с контроллером Arduino делает ваше растение более комфортным, а ваш сад умнее.

Модуль датчика влажности почвы не так сложен, как вы думаете, и если вам нужно обнаружить почву в вашем проекте, это будет ваш лучший выбор.

Датчик устанавливается с двумя датчиками, вставленными в почву, затем, когда ток проходит через почву, датчик получает значение сопротивления, считывая изменения тока между двумя датчиками, и преобразует это значение сопротивления в содержание влаги.

Чем выше влажность (меньше сопротивление), тем выше проводимость почвы.

Поверхность сенсора подверглась металлизации для продления срока службы. Вставьте его в почву и затем используйте преобразователь AD, чтобы прочитать его. С помощью этого датчика растение может напоминать вам, что мне нужна вода.

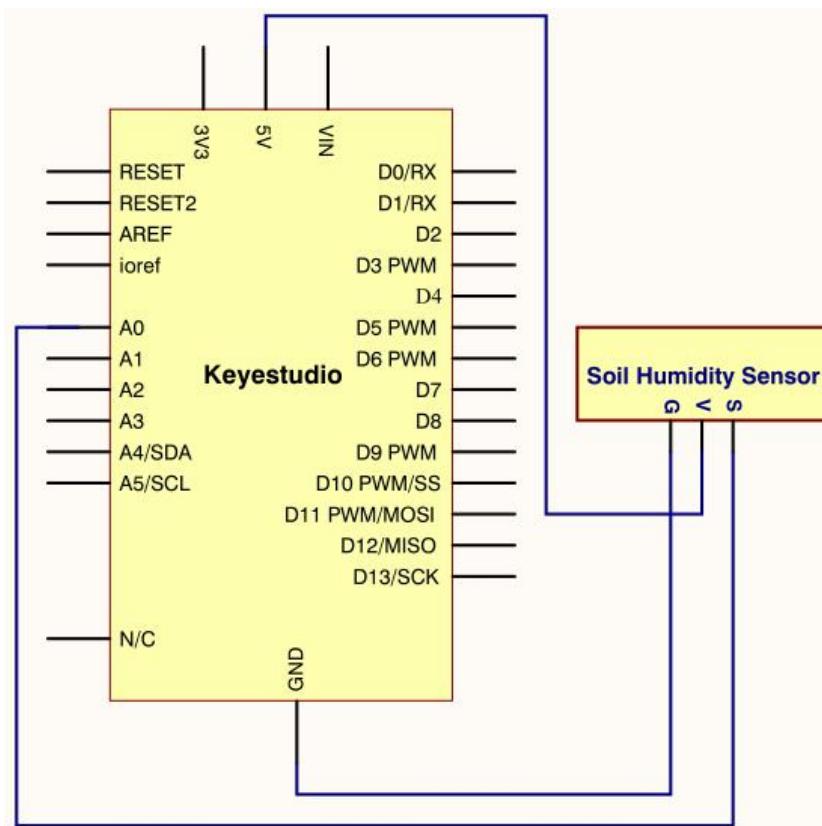
2. Требуется оборудование

- Плата V4.0 или плата MEGA 2650 * 1
- Датчик влажности почвы * 1
- Перемычка * 1
- USB-кабель * 1

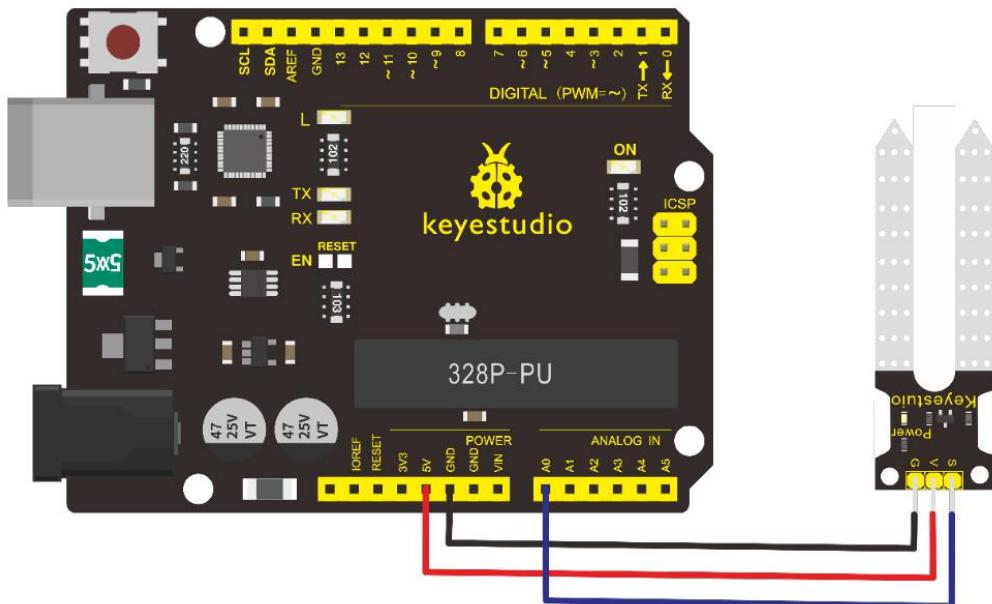
3. Спецификация

- Напряжение источника питания 3,3 В или 5 В
- Рабочий ток ≤ 20 мА
- Выходное напряжение 0-2,3 В (когда датчик полностью погружен в воду, напряжение будет 2,3 В)
- Источник питания 5 В (чем выше влажность, тем выше выходное напряжение)
- Тип датчика Аналоговый выход
- Определение интерфейса Pin1- сигнал, pin2- GND, pin3 - VCC

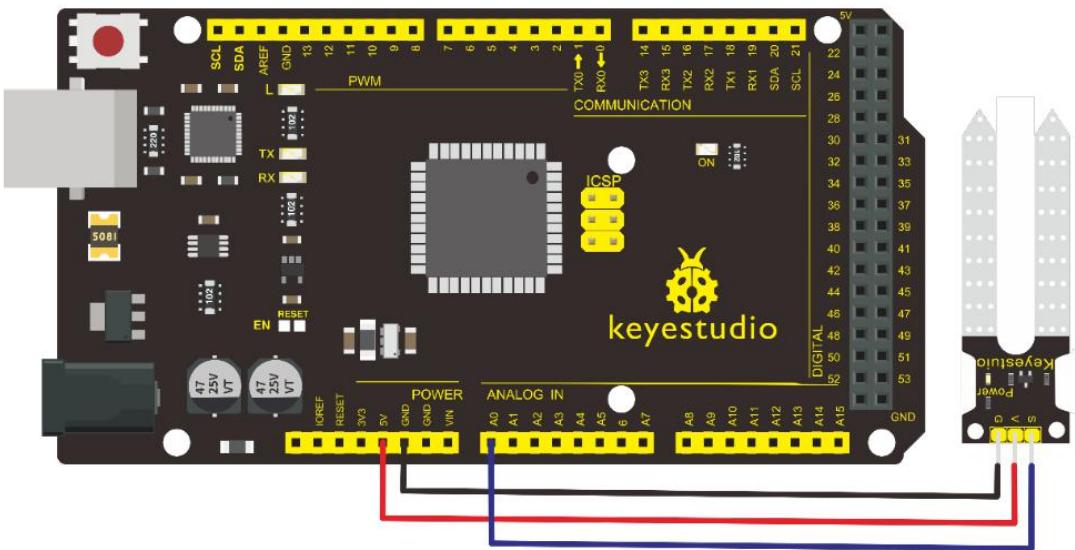
4. схема подключения



Подключение для V4.0



Подключение для MEGA 2560 R3



5. образец кода

```
/*
```

супер обучающий комплект keyestudio

Project 31

датчик влажности почвы

<http://www.keyestudio.com>

```
*/
```

```
void setup () {
```

```
Serial.begin (57600);
```

```
}
```

```
void loop () {
```

```
Serial.print («Значение датчика влажности»);
```

```
Serial.println (analogRead (0));
```

```
задержка (100);
```

}

||||||||||||||||||||||||||||||||

6. результат теста



Проект 32: RFID-модуль RC522



1. Введение

В модуле MF522-AN используется оригинальная конструкция микросхемы считывателя Philips

MFRC522, проста в использовании, низкая стоимость, подходит для

оборудование развитие, развитие из продвинутый

приложения, необходимость для пользователя дизайн / производство терминала RF карты. Его можно загружать непосредственно в различные формы для считывания.

Модуль использует напряжение 3,3 В через интерфейс SPI, используя несколько простых линий, он может быть напрямую подключен к любым модулям связи платы ЦП, чтобы гарантировать стабильную и надежную работу и расстояние до считывателя.

2. Требуется оборудование

- Плата V4.0 или плата MEGA 2650 * 1
- Модуль RFID * 1
- Перемычка * 1
- USB-кабель * 1

3. Спецификация

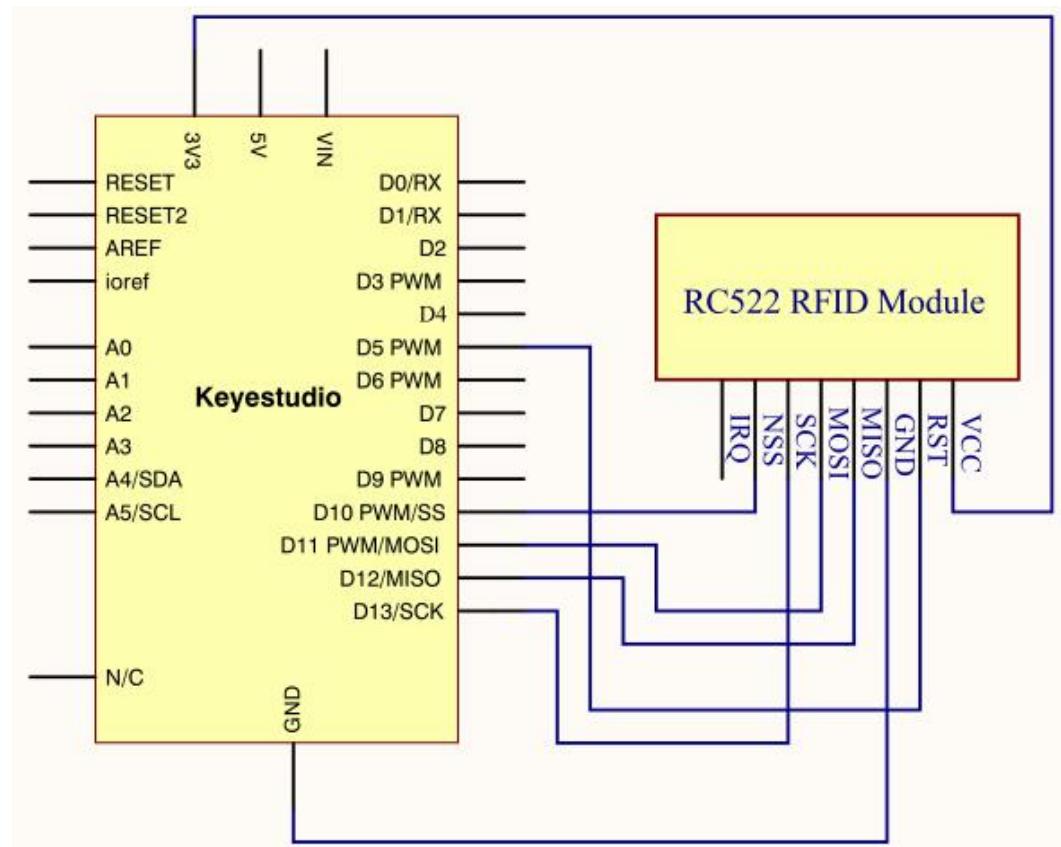
1. Ток 13-26 мА / 3,3 В постоянного тока
2. Ток холостого хода 10-13 мА / постоянный ток 3,3 В
3. Ток сна <80uA
4. Пиковый ток <30 мА
5. Рабочая частота 13,56 МГц.
6. Поддерживаемые типы карт mifare1 S50, mifare1 S70, mifare UltraLight, mifare Pro, mifare Desfire

7. Рабочая температура окружающей среды -20-80 градусов Цельсия.

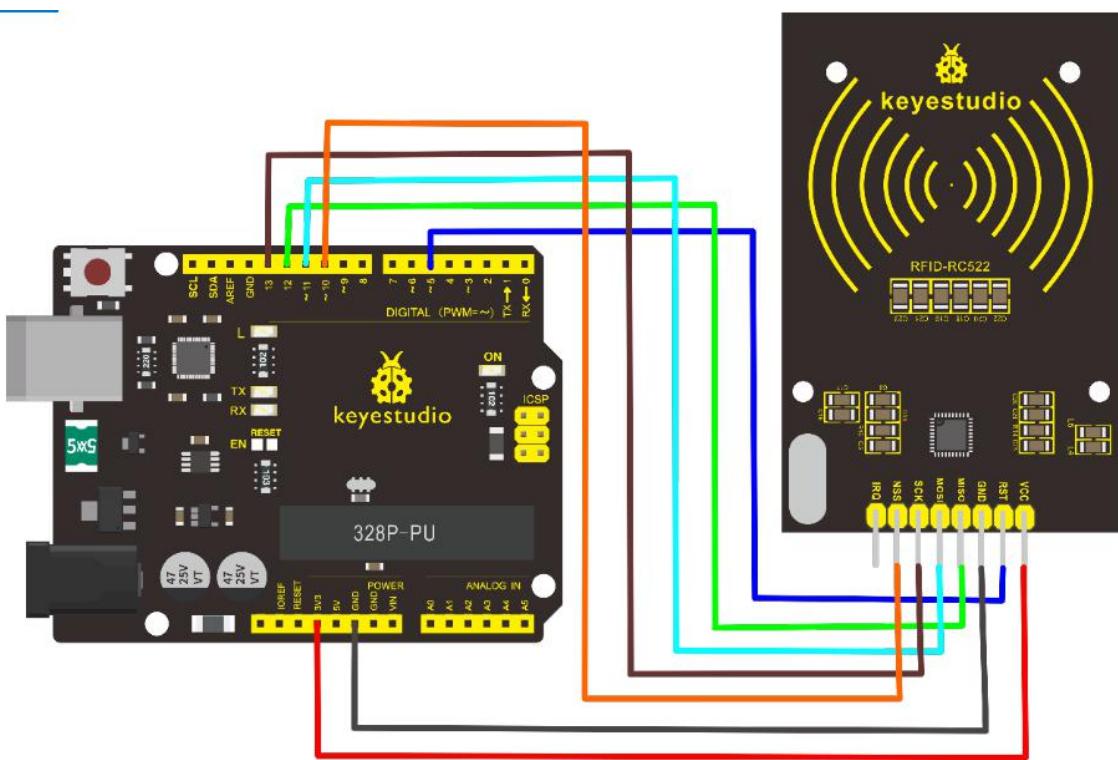
8. Температура окружающей среды при хранении -40-85 градусов Цельсия.

9. Относительная влажность 5% -95%

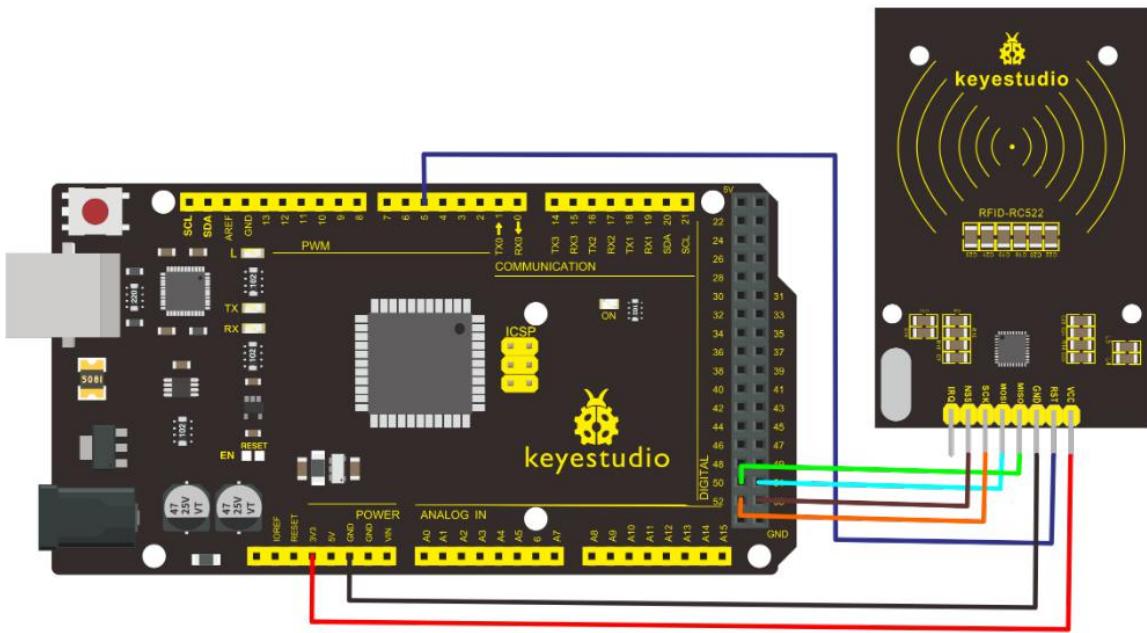
4. схема подключения



Подключение для V4.0



Подключение для 2560 R3



5. образец кода

```
/*
```

Project 32

RFID

<http://www.keyestudio.com>

* /

включить <SPI.h>

определить uchar unsigned char

определить uint unsigned int

define MAX_LEN 16

const int chipSelectPin = 10; // если контроллер UNO, 328,168 const int NRSTPD = 5;

// Командное слово MF522

определить PCD_IDLE 0x00 // НЕТ действия, сдерживать ток

команда

определить PCD_AUTHENT 0x0E // проверяем ключ

определить PCD_RECEIVE 0x08 // получаем данные

определить PCD_TRANSMIT 0x04 // отправляем данные

определить PCD_TRANSCEIVE 0x0C // получение и отправка данных //

определить PCD_RESETPHASE 0x0F сброс

определить PCD_CALCCRC 0x03 // Расчет CRC

```

// Командное слово Mifare_One Card

#define PICC_REQIDL 0x26 // область отслеживания линии

#define PICC_REQALL 0x52 // область отслеживания линии нарушена

#define PICC_ANTICOLL 0x93 //Анти столкновение

#define PICC_SELECTTAG 0x93 // выбираем карты

#define PICC_AUTHENT1A 0x60 // Проверяем ключ

#define PICC_AUTHENT1B 0x61 // Проверяем ключ В

#define PICC_READ 0x30 // Модуль чтения

#define PICC_WRITE 0xA0 // буквенный блок

#define PICC_DECREMENT 0xC0

#define PICC_INCREMENT 0xC1

#define PICC_RESTORE 0xC2 // Переносим данные в буфер //

#define PICC_TRANSFER 0xB0 // Сохраняем данные буфера

#define PICC_HALT 0x50 // Покой

// MF522 Код ошибки возвращается при связи

#define MI_OK 0

#define MI_NOTAGERR 1

#define MI_ERR 2

```

// ----- MFRC522 Регистр -----

// Страница 0: команда и статус

# определить	Зарезервировано00	0x00
# определить	CommandReg	0x01
# определить	CommIEnReg	0x02
# определить	DivIEnReg	0x03
# определить	CommIrqReg	0x04
# определить	DivIrqReg	0x05
# определить	ErrorReg	0x06
# определить	Status1Reg	0x07
# определить	Status2Reg	0x08
# определить	FIFODataReg	0x09
# определить	FIFOLevelReg	0x0A
# определить	WaterLevelReg	0x0B
# определить	ControlReg	0x0C
# определить	BitFramingReg	0x0D
# определить	CollReg	0x0E
# определить	Зарезервировано01	0x0F

// Страница 1: Команда

# определить	Зарезервировано10	0x10
# определить	ModeReg	0x11

# определить	TxModeReg	0x12
# определить	RxModeReg	0x13
# определить	TxControlReg	0x14
# определить	TxAutoReg	0x15
# определить	TxSelReg	0x16
# определить	RxSelReg	0x17
# определить	RxThresholdReg	0x18
# определить	DemodReg	0x19
# определить	Зарезервировано11	0x1A
# определить	Зарезервировано12	0x1B
# определить	MifareReg	0x1C
# определить	Зарезервировано13	0x1D
# определить	Зарезервировано14	0x1E
# определить	SerialSpeedReg	0x1F

// Страница 2: CFG

# определить	Зарезервировано20	0x20
# определить	CRCResultRegM	0x21
# определить	CRCResultRegL	0x22
# определить	Зарезервировано21	0x23
# определить	ModWidthReg	0x24
# определить	Зарезервировано22	0x25
# определить	RFCfgReg	0x26

# определить	GsNReg	0x27
# определить	CWGPsPReg	0x28
# определить	ModGsPReg	0x29
# определить	TModeReg	0x2A
# определить	TPrescalerReg	0x2B
# определить	TReloadRegH	0x2C
# определить	TReloadRegL	0x2D
# определить	TCounterValueRegH	0x2E
# определить	TCounterValueRegL	0x2F

// Страница 3: TestRegister

# определить	Зарезервировано30	0x30
# определить	TestSel1Reg	0x31
# определить	TestSel2Reg	0x32
# определить	TestPinEnReg	0x33
# определить	TestPinValueReg	0x34
# определить	TestBusReg	0x35
# определить	AutoTestReg	0x36
# определить	VersionReg	0x37
# определить	AnalogTestReg	0x38
# определить	TestDAC1Reg	0x39
# определить	TestDAC2Reg	0x3A
# определить	TestADCReg	0x3B

```

#define Зарезервировано31      0x3C
#define Зарезервировано32      0x3D
#define Зарезервировано33      0x3E
#define Зарезервировано34      0x3F

uchar serNum [5];

uchar writeDate [16] = {'T', 'e', 'n', 'g', ' ', 'B', 'o', 0, 0, 0, 0, 0, 0, 0, 0}; uchar секторKeyA [16] [16] =
{{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}, {0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF},};

uchar секторNewKeyA [16] [16] = {{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}, {0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF},};

void setup () {
    Serial.begin (9600);                                // Вывод SOUT считывателя RFID подключен
                                                       // к последовательному контакту RX на 2400 бит /
    c // запуск библиотеки SPI: SPI.begin ();
}

```

```

pinMode (chipSelectPin, ВЫХОД); // Установить цифровой контакт 10 как ВЫХОД

подключить его к контакту RFID / ENABLE

digitalWrite (chipSelectPin, LOW); // Активируем считыватель RFID

pinMode (NRSTPD, ВЫХОД); // Установить цифровой контакт 10, не сбрасывать и

Выключить

digitalWrite (NRSTPD, HIGH);

MFRC522_Init ();

}

пустой цикл ()

{

uchar i, tmp;

статус uchar;

uchar str [MAX_LEN];

uchar RC_size;

uchar blockAddr; // Выбираем адрес операции 0 ~ 63

// поиск карты, возврат типа карты

статус = MFRC522_Request (PICC_REQIDL, str);

```

```

if (status == MI_OK) {

}

status = MFRC522_Anticoll (str);

memcpy (serNum, str, 5);

if (status == MI_OK) {

    Serial.println ("Номер карты : ");

    Serial.print (serNum [0], BIN);

    Serial.print (serNum [1], BIN);

    Serial.print (serNum [2], BIN);

    Serial.print (serNum [3], BIN);

    Serial.print (serNum [4], BIN);

    Serial.println ("");

}

// выбрать карту, вернуть емкость карты RC_size =

MFRC522_SelectTag (serNum);

если (RC_size! = 0) {}

```

```

// записываем данные карты

blockAddr = 7;           // блок данных 7

статус = MFRC522_Auth (PICC_AUTHENT1A, blockAddr, секторKeyA

[blockAddr / 4], serNum); // аутентификация

if (status == MI_OK) {

    // записываем данные

    статус = MFRC522_Write (blockAddr, секторNewKeyA [blockAddr / 4]);

    Serial.print ("установить новый пароль карты, можно изменить

данные Сектора: ");

    Serial.print (blockAddr / 4, DEC);

    // записываем данные

    blockAddr = blockAddr - 3;

    статус = MFRC522_Write (blockAddr, writeDate); если (статус ==

MI_OK)

    {

        Serial.println («OK!»);

    }

}

```

```

// читать карту

blockAddr = 7;           // блок данных 7

статус = MFRC522_Auth (PICC_AUTHENT1A, blockAddr, секторNewKeyA

[blockAddr / 4], serNum); // аутентификация

if (status == MI_OK) {

    // читаем данные

    blockAddr = blockAddr - 3;

    status = MFRC522_Read (blockAddr, str);

    if (status == MI_OK) {

        Serial.println ("Прочитано с карты, данные:");

        ");

        for (i = 0; i <16; i++) {

            Serial.print (str [i]);

        }

        Serial.println ("");

    }

    Serial.println ("");

    MFRC522_Halt ();           // командная карта для перехода в режим ожидания

```

```
}

void Write_MFRC522 (uchar addr, uchar val) {

    digitalWrite (chipSelectPin, LOW);

    SPI.transfer ((адрес << 1) & 0x7E);

    SPI.transfer (val);

    digitalWrite (chipSelectPin, HIGH);

}

uchar Read_MFRC522 (адрес uchar)

{

    uchar val;

    digitalWrite (chipSelectPin, LOW);

    // формат адреса: 1XXXXXXX0

    SPI.transfer (((адрес << 1) & 0x7E) | 0x80);
```

```
val = SPI.transfer (0x00);

digitalWrite (chipSelectPin, HIGH);

return val;

}

void SetBitMask (uchar reg, uchar mask) {

uchar tmp;

tmp = Read_MFRC522 (per);

Write_MFRC522 (reg, tmp | маска); // устанавливаем битовую маску

}

void ClearBitMask (uchar reg, uchar mask) {

uchar tmp;

tmp = Read_MFRC522 (per);

Write_MFRC522 (reg, tmp & (~ маска)); // очистить битовую маску
```

```
    }
```

```
void AntennaOn (недействительно)
```

```
{
```

```
    uchar temp;
```

```
    temp = Read_MFRC522 (TxControlReg);
```

```
    if (! (temp & 0x03)) {
```

```
        SetBitMask (TxControlReg, 0x03);
```

```
}
```

```
}
```

```
void AntennaOff (недействительно)
```

```
{
```

```
    ClearBitMask (TxControlReg, 0x03);
```

```
}
```

```
недействительным MFRC522_Reset (недействительным)
```

```
{
```

```

        Write_MFRC522 (CommandReg, PCD_RESETPHASE);

    }

недействительным MFRC522_Init (недействительным)

{

digitalWrite (NRSTPD, HIGH);

MFRC522_Reset ();

// Таймер: TPrescaler * TreloadVal / 6,78 МГц = 24 мс

        Write_MFRC522 (TModeReg, 0x8D);           // Tayto = 1; f (Таймер) =
6,78 МГц / TPreScaler

        Write_MFRC522 (TPrescalerReg, 0x3E); //TModeReg[3..0] + TPrescalerReg
        Write_MFRC522 (TReloadRegL, 30);

        Write_MFRC522 (TReloadRegH, 0);

        Write_MFRC522 (TxAutoReg, 0x40);          // 100% СПРОСИТЕ

        Write_MFRC522 (ModeReg, 0x3D);           // начальное значение CRC

AntennaOn ();           // открытая антenna

}

```

```

uchar MFRC522_Request (uchar reqMode, uchar * TagType) {

    статус uchar;

    uint backBits;           // полученные биты данных

    Write_MFRC522 (BitFramingReg, 0x07);           // TxLastBists =
                                                // BitFramingReg [2..0] ???

    TagType [0] = reqMode;

    статус = MFRC522_ToCard (PCD_TRANSCEIVE, TagType, 1, TagType, & backBits);

    if ((статус! = MI_OK) || (backBits! = 0x10)) {

        статус = MI_ERR;
    }

    статус возврата;

}

uchar MFRC522_ToCard (команда uchar, uchar * sendData, uchar sendLen, uchar * backData, uint *
backLen)

```

```

{

    статус uchar = MI_ERR; uchar
    irqEn = 0x00; uchar waitIRq =
    0x00; uchar lastBits;

    uchar n;
    uint i;

    переключатель (команда)

    {

        case PCD_AUTHENT:           // аутентификация ключа карты

        {

            irqEn = 0x12;
            waitIRq = 0x10;

            сломать;

        }

        case PCD_TRANSCEIVE: // отправляем данные в FIFO {

            irqEn = 0x77;
            waitIRq = 0x30;

            сломать;

        }
}

```

по умолчанию:

сломать;

}

Write_MFRC522 (CommlEnReg, irqEn | 0x80); // Разрешить запрос прерывания ClearBitMask

(CommlrqReg, 0x80); // очищаем все биты запроса прерывания

SetBitMask (FIFOLevelReg, 0x80); // FlushBuffer = 1, инициализация FIFO

Write_MFRC522 (CommandReg, PCD_IDLE); //Бездействие; отменить текущую команду ???

// записываем данные в FIFO

for (i = 0; i < sendLen; i++) {

Write_MFRC522 (FIFODataReg, sendData [i]);

}

// выполняем команду

Write_MFRC522 (CommandReg, команда);

if (command == PCD_TRANSCEIVE) {

```

SetBitMask (BitFramingReg, 0x80); // StartSend = 1, передача данных

начинается

}

// ждем завершения приема данных

i = 2000; // настроить i в соответствии с тактовой частотой, максимальное время ожидания работы M1 составляет
25 мс ????

делать

{

//CommIrqReg[7..0]

// Set1 TxIRq RxIRq IdleIRq HiAlerIRq LoAlertIRq ErrIRq TimerIRq

n = Read_MFRC522 (CommIrqReg);

я--;

}

while ((i != 0) && (n & 0x01) && (n & waitIRq));




ClearBitMask (BitFramingReg, 0x80); // StartSend = 0

если (i != 0)

{

if (! (Read_MFRC522 (ErrorReg) & 0x1B)) // BufferOvfl Collerr CRCErr ProtocolErr

```

```

{

    статус = MI_OK;

    if (n & irqEn & 0x01) {

        // ???

    }

    if (command == PCD_TRANSCEIVE) {

        n = Read_MFRC522 (FIFOLevelReg);

        lastBits = Read_MFRC522 (ControlReg) & 0x07; если (lastBits)

        {

            * backLen = (n-1) * 8 + lastBits;

        }

        еще

        {

            * backLen = n * 8;

        }

        если (n == 0)

        {

```

```

n = 1;

}

if (n > MAX_LEN) {

    n = MAX_LEN;

}

// читаем данные, полученные FIFO

for (i = 0; i < n; i++) {

    backData [i] = Read_MFRC522 (FIFODataReg);

}

}

}

еще

{

    статус = MI_ERR;

}

}

// SetBitMask (ControlReg, 0x80); // таймер останавливается

```

```

// Write_MFRC522 (CommandReg, PCD_IDLE);

статус возврата;

}

uchar MFRC522_Anticoll (uchar * serNum)

{

    статус uchar;

    uchar я;

    uchar serNumCheck = 0;

    uint unLen;

    Write_MFRC522 (BitFramingReg, 0x00); // TxLastBists = BitFramingReg [2..0]

    serNum [0] = PICC_ANTICOLL;

    serNum [1] = 0x20;

    статус = MFRC522_ToCard (PCD_TRANSCEIVE, serNum, 2, serNum, & unLen);

    если (статус == MI_OK)

    {

        // проверяем порядковый номер карты для (i

        = 0; i <4; i ++)


```

```

{

    serNumCheck ^= serNum [i];

}

if (serNumCheck != serNum [i]) {

    статус = MI_ERR;

}

// SetBitMask (CollReg, 0x80);           // ValuesAfterColl = 1

статус возврата;

}

void CalulateCRC (uchar * plndata, uchar len, uchar * pOutData) {

    uchar я, н;

    ClearBitMask (DivIrqReg, 0x04);          // CRCIrq = 0

    SetBitMask (FIFOLevelReg, 0x80);         // очищаем указатель FIFO

    // Write_MFRC522 (CommandReg, PCD_IDLE);
}

```

```

// записываем данные в FIFO

for (i = 0; i <len; i++) {

    Write_MFRC522 (FIFODataReg, * (pInData + i));

}

Write_MFRC522 (CommandReg, PCD_CALCCRC);

// ждем завершения вычисления CRC

я = 0xFF;

делать

{

    n = Read_MFRC522 (DivIrqReg);

    я--;

}

пока ((и! = 0) && (n & 0x04));                                // CRCIrq = 1

// читаем результат вычисления CRC

pOutData [0] = Read_MFRC522 (CRCResultRegL);

pOutData [1] = Read_MFRC522 (CRCResultRegM);

}

учар MFRC522_SelectTag (учар * serNum)

```

```

{

    учар я;

    статус учар;

    размер учара;

    uint recvBits;

    учар буфер [9];

// ClearBitMask (Status2Reg, 0x08);           // MFCrypto1On = 0

буфер [0] = PICC_SEIECTTAG;

буфер [1] = 0x70;

for (i = 0; i <5; i++) {

    буфер [i + 2] = * (serNum + i);

}

CalulateCRC (буфер, 7 и буфер [7]);          // ??

статус = MFRC522_ToCard (PCD_TRANSCEIVE, буфер, 9, буфер, & recvBits);

if ((status == MI_OK) && (recvBits == 0x18)) {

    размер = буфер [0];

}

```

```
еще

{

size = 0;

}

размер возврата;

}

uchar MFRC522_Auth (uchar authMode, uchar BlockAddr, uchar * Sectorkey, uchar
* serNum)

{

статус учар;

uint recvBits;

учар я;

учар бафф [12];

// Команды проверки + адрес блока + пароль сектора + порядковый номер карты

бафф [0] = authMode;

buff [1] = BlockAddr;

for (i = 0; i <6; i++) {
```

```
    buff [i + 2] = * (Sectorkey + i);

}

for (i = 0; i < 4; i++) {

    buff [i + 8] = * (serNum + i);

}

статус = MFRC522_ToCard (PCD_AUTHENT, buff, 12, buff, & recvBits);

if ((status != MI_OK) || (! (Read_MFRC522 (Status2Reg) & 0x08))) {

    статус = MI_ERR;

}

статус возврата;

}

uchar MFRC522_Read (uchar blockAddr, uchar * recvData) {

    статус учар;

    uint unLen;

    recvData [0] = PICC_READ;
```

```
recvData [1] = blockAddr;

CalulateCRC (recvData, 2, & recvData [2]);

статус = MFRC522_ToCard (PCD_TRANSCEIVE, recvData, 4, recvData, & unLen);

if ((status != MI_OK) || (unLen != 0x90)) {

    статус = MI_ERR;

}

статус возврата;

}

uchar MFRC522_Write (uchar blockAddr, uchar * writeData) {

    статус учар;

    uint recvBits;

    учар я;

    учар бафф [18];

    бафф [0] = PICC_WRITE;

    buff [1] = blockAddr;
```

```

CalulateCRC (buff, 2, & buff [2]);

статус = MFRC522_ToCard (PCD_TRANSCEIVE, бaфф, 4, бaфф, & recvBits);

if ((status! = MI_OK) || (recvBits! = 4) || ((buff [0] & 0x0F)! = 0x0A)) {

    статус = MI_ERR;

}

if (status == MI_OK) {

    for (i = 0; i <16; i++) {           // записываем 16-байтовые данные в FIFO

        бaфф [я] = * (writeData + я);

    }

    CalulateCRC (buff, 16, & buff [16]);

    статус = MFRC522_ToCard (PCD_TRANSCEIVE, бaфф, 18, бaфф, & recvBits);

}

if ((status! = MI_OK) || (recvBits! = 4) || ((buff [0] & 0x0F)! = 0x0A))

{

    статус = MI_ERR;

}

}

```

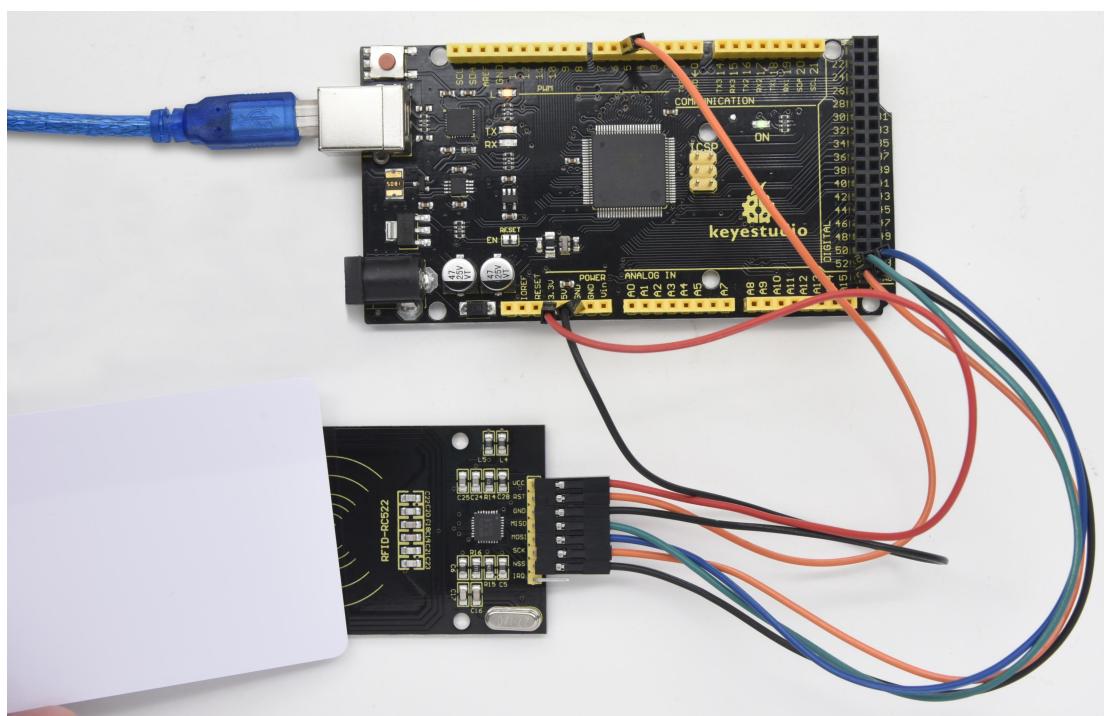
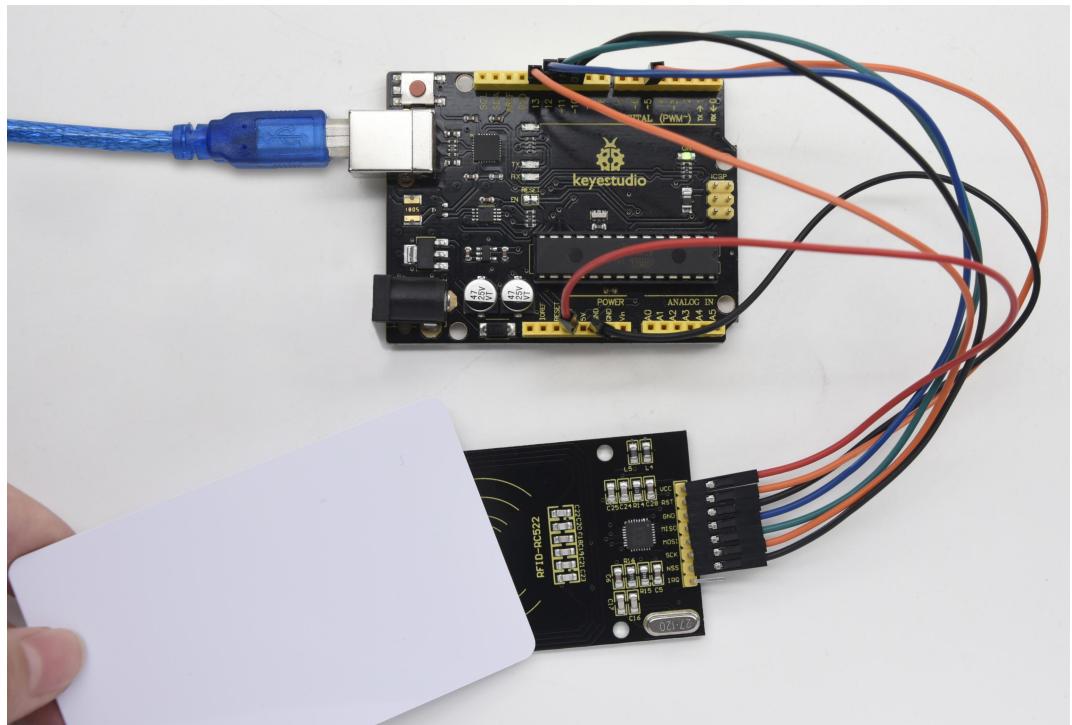
```
    статус возврата;  
}  
  
недействительным MFRC522_Halt (недействительным)  
  
{  
  
    статус учар;  
  
    uint unLen;  
  
    учар бафф [4];  
  
    бафф [0] = PICC_HALT;  
  
    бафф [1] = 0;  
  
    CalulateCRC (buff, 2, & buff [2]);  
  
    статус = MFRC522_ToCard (PCD_TRANSCEIVE, бафф, 4, бафф, & unLen);  
}
```

Обратите внимание: если вы хотите использовать MEGA 2560 R3, пожалуйста, измените код

```
const int chipSelectPin = 10; // если контроллер V4.0  
  
const int chipSelectPin = 53; // если контроллер MEGA2560
```

6. результат теста

В этом эксперименте, когда IC-карта приближается, RFID-модуль записывает данные на IC-карту и считывает данные с карты, вы можете видеть данные в окне монитора. Показано ниже.



The card's number is :
101100110000001010100010011000101001
set the new card password, and can modify the data of the Sector: 1OK!
Read from the card ,the data is :
841011101033266111000000000

Autoscroll Show timestamp Newline 9600 baud Clear output

6. Ресурсы

Скачать код, драйвер и библиотеку:

<https://fs.keyestudio.com/KS0077-78-79>