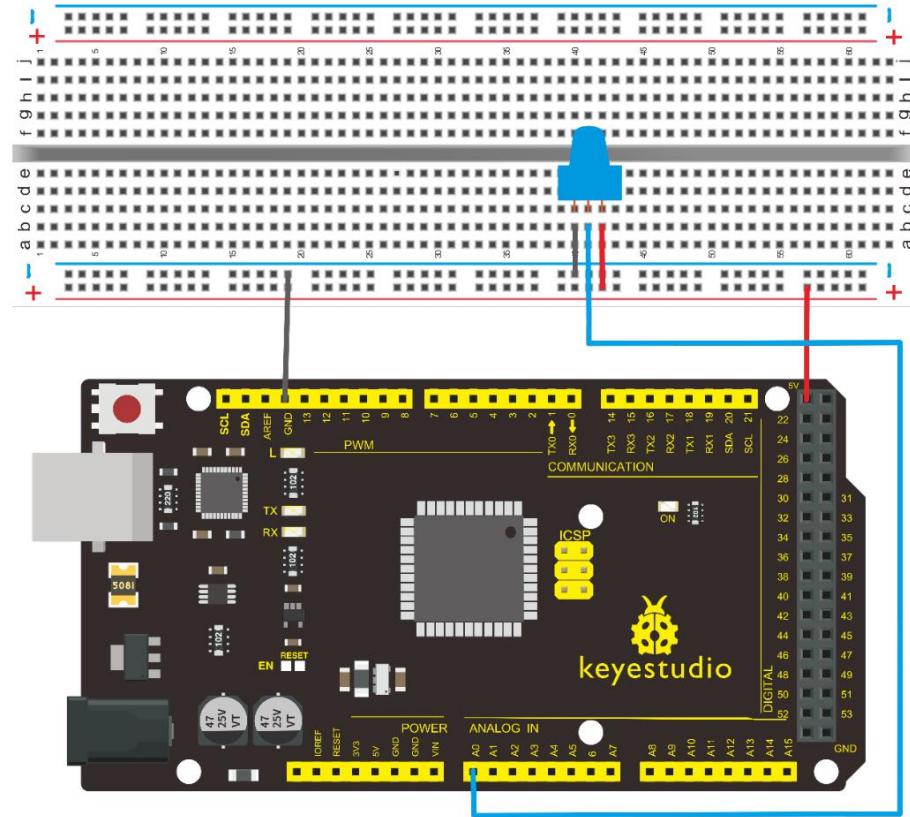


**Подключение для 2560**



## 5. образец кода

В        программа        составление        является просто. Ап        аналогЧитать

() Заявление может прочитать значение интерфейса. Сбор данных A / D Arduino 328 осуществляется в 10-битном формате, поэтому считываемое значение находится в диапазоне от 0 до 1023.

Одна из трудностей этого проекта - отобразить значение на экране, что на самом деле легко узнать.

Во-первых, вам нужно установить скорость передачи в **voidsetup ()**. Отображение значения - это связь между Arduino и ПК, поэтому скорость передачи Arduino должна совпадать со скоростью, установленной в программном обеспечении ПК. В противном случае на дисплее будут запутанные коды или вообще не будет дисплея.

В правом нижнем углу окна программного монитора Arduino есть кнопка для настройки скорости передачи данных. Настройка здесь должна совпадать с настройкой в программе. Заявление в программе **Serial.begin ()**; заключено значение скорости передачи данных, за которым следует инструкция для отображения. Вы можете использовать **Serial.print ()** или же **Serial.println ()** заявление.

/\*

супер обучающий комплект keyestudio

Project 15

## Потенциометр

<http://www.keyestudio.com>

\* /

```
int potpin = 0; // инициализируем аналоговый вывод 0 int ledpin = 13;
```

```
// инициализируем цифровой вывод 13
```

```
int val = 0; // определяем val, присваиваем начальное значение 0 void setup
```

```
()
```

```
{
```

```
  pinMode (ledpin, OUTPUT); // устанавливаем цифровой вывод как «выход» Serial.begin (9600);
```

```
  // устанавливаем скорость передачи 9600}
```

пустой цикл ()

```
{
```

```
  digitalWrite (ledpin, HIGH); // включаем светодиод на выводе 13 delay (50);
```

```
  // ждем 0,05 секунды
```

```
  digitalWrite (ledpin, LOW); // выключаем светодиод на выводе 13 delay (50);
```

```
  // ждем 0,05 секунды
```

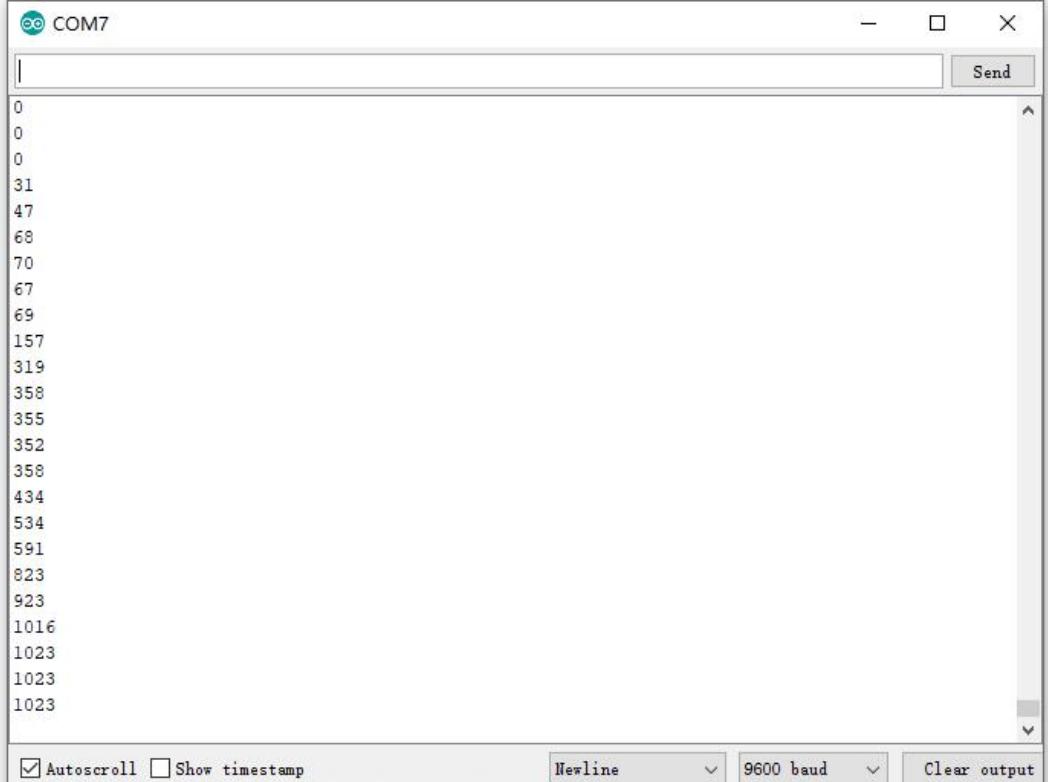
```
  val = analogRead (potpin); // считываем аналоговое значение аналогового вывода 0 и назначаем его val
```

```
  Serial.println (val); // выводим значение val}
```

```
|||||||||||||||||||||||||||||||||||||||||||
```

## 6. результат теста

В примере кода используется встроенный светодиод, подключенный к контакту 13. Каждый раз, когда устройство считывает значение, светодиод мигает. Когда вы вращаете ручку потенциометра, вы можете увидеть изменение отображаемого значения. Считывание аналогового значения - очень распространенная функция, поскольку большинство датчиков выдают аналоговое значение. После расчета вы можете получить нужное вам значение. На рисунке ниже показано считываемое аналоговое значение.

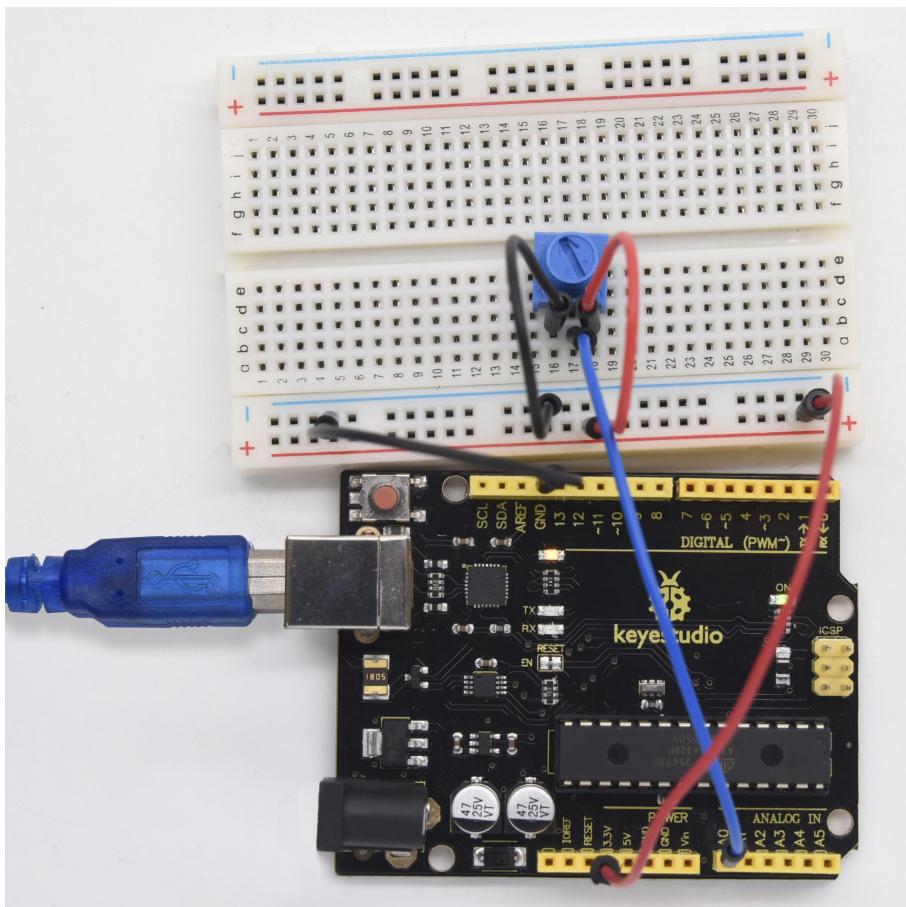


The screenshot shows a terminal window titled "COM7". The window displays a series of numerical values, likely representing analog readings from a potentiometer. The values are as follows:

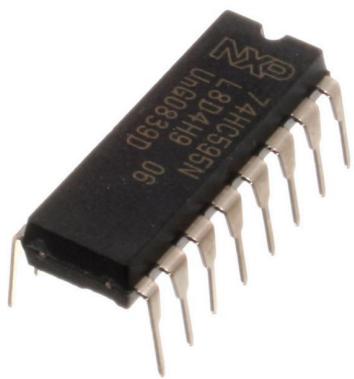
```
0
0
0
31
47
68
70
67
69
157
319
358
355
352
358
434
534
591
823
923
1016
1023
1023
1023
```

At the bottom of the window, there are several control buttons: "Autoscroll" (checked), "Show timestamp" (unchecked), "Newline" (dropdown menu), "9600 baud" (dropdown menu), and "Clear output".

На этом эксперимент завершен. Спасибо!



Проект 16: микросхема 74HC595



## 1. Введение

Микросхема 74HC595 представляет собой устройство последовательного и параллельного вывода. Чтобы

Проще говоря, микросхема 74HC595 представляет собой комбинацию из 8-значного регистра сдвига, запоминающего устройства и оснащена выходом с тремя состояниями.

Здесь мы используем его для управления 8 светодиодами. Вы можете задаться вопросом, зачем использовать 74HC595 для управления светодиодом? Ну, подумайте, сколько операций ввода-вывода требуется Arduino для управления 8 светодиодами? Да, 8.

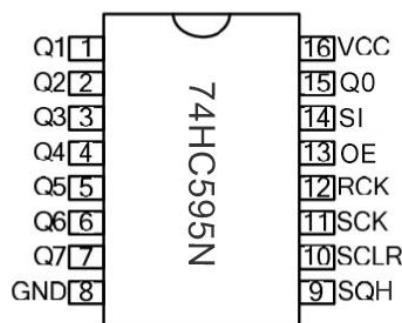
Для Arduino 328 у него всего 20 входов / выходов, включая аналоговые порты. Для экономии ресурсов порта использование 74HC595 позволяет нам использовать 3 цифровых порта ввода / вывода для управления 8 светодиодами!

## **2. Требуется оборудование**

- Плата V4.0 или плата MEGA 2650 \* 1 микросхема
- 74HC595 \* 1
- Красный светодиод M5 \* 4
- Зеленый светодиод M5 \* 4
- Резистор 220 Ом \* 8
- Макетная плата \* 1
- Провода перемычки макетной платы
- USB-кабель \* 1



3. Описание контактов :



Контакты Нет	имя	Функция
1-7, 15	Q0-Q7	Параллельный выход
8	GND	GND
9	Г-Н	Последовательный выход
10		Главный резерв, подключите выход тактовой
11	SH_CP	частоты регистра сдвига 5 В

12	ST_CP	Вход часов регистра памяти
13	OE	Разрешение выхода (активный НИЗКИЙ)
14	DS	Последовательный ввод данных
16	Vcc	Рабочее напряжение 5 В

- VCC и GND используются для питания микросхемы, рабочее напряжение 5

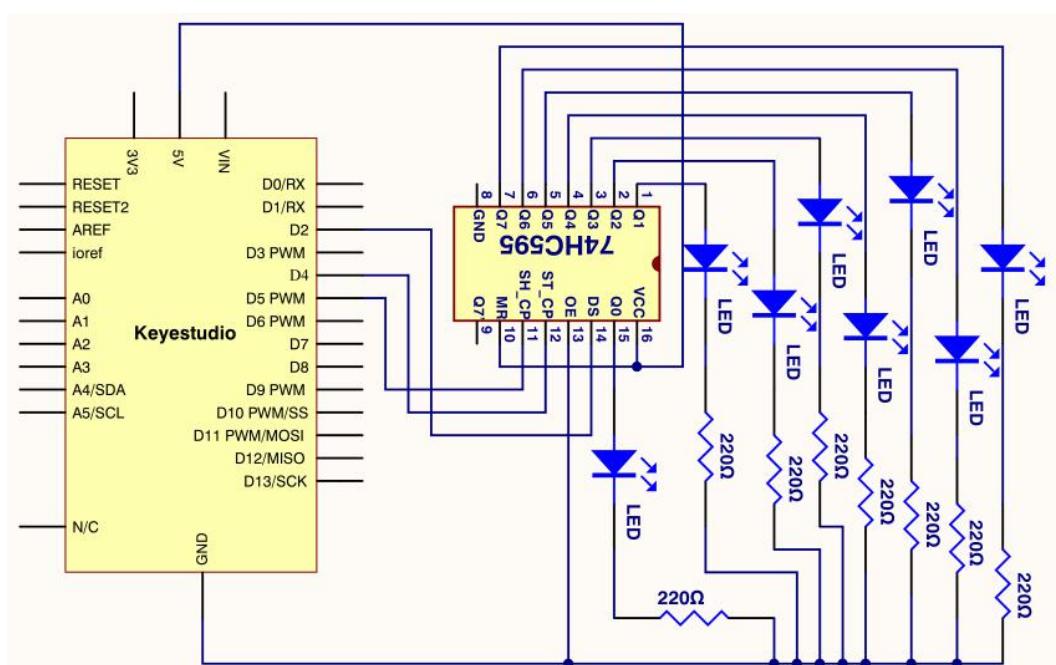
B.

- Q0 ~ Q7: Эти восемь контактов являются выходными контактами.
- Вывод DS - это вывод последовательного ввода, нам нужно записывать данные на этот вывод по битам.
- STCP - фиксатор. Данные могут быть скопированы в защелку и выведены параллельно после того, как будут переданы все 8-значные данные защелки.
- SHCP - это тактовый штифт. Данные могут быть записаны в регистр хранения.
- OE - это вывод разрешения выхода, который используется для проверки того, вводятся ли данные защелки на выводы Q0-Q7. При низком уровне высокий уровень не выводится. В этом эксперименте мы напрямую подключаемся к GND для сохранения выходных данных низкого уровня.
- MR - это вывод для инициализации вывода регистра памяти. Инициализируйте регистр внутренней памяти при низком уровне. В этом

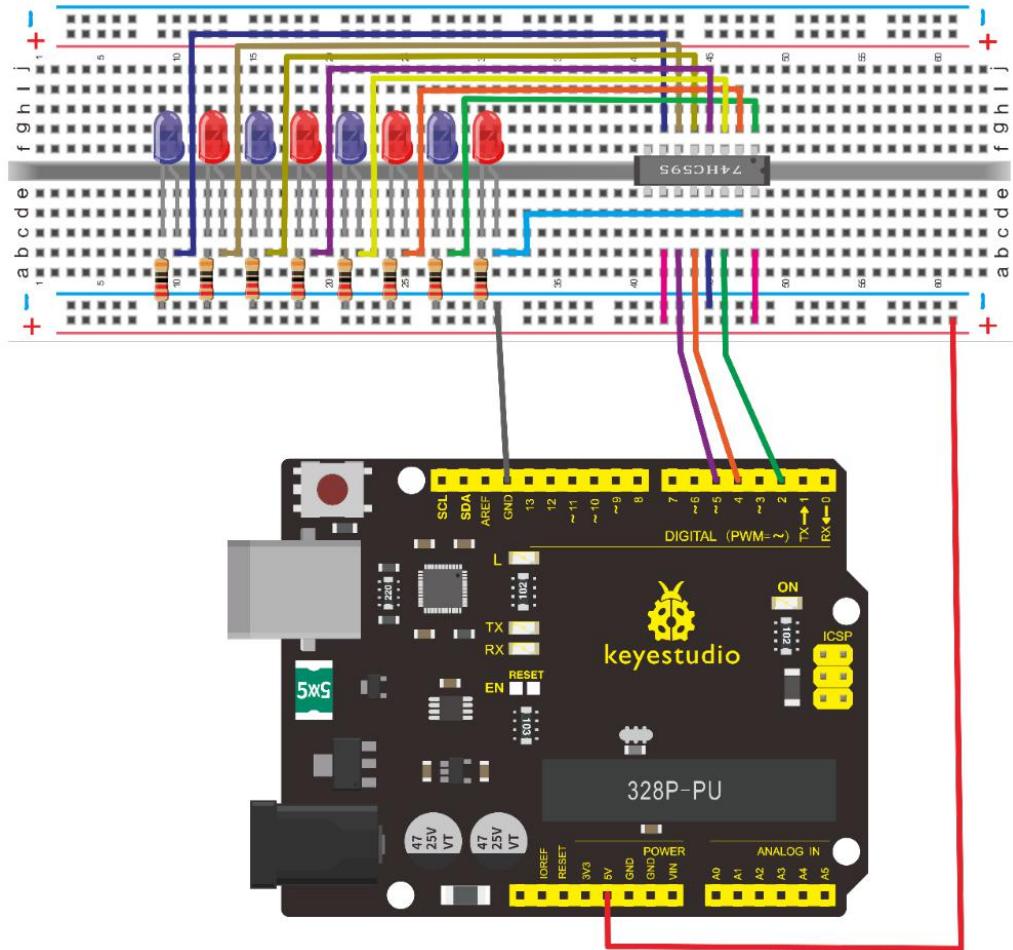
экспериментируем, подключаемся к VCC, чтобы поддерживать высокий уровень.

- Вывод Q7S - это вывод последовательного вывода, который специально используется для каскада микросхем.

#### 4. схема подключения



Подключение для V4.0



Подключение для 2560 R3

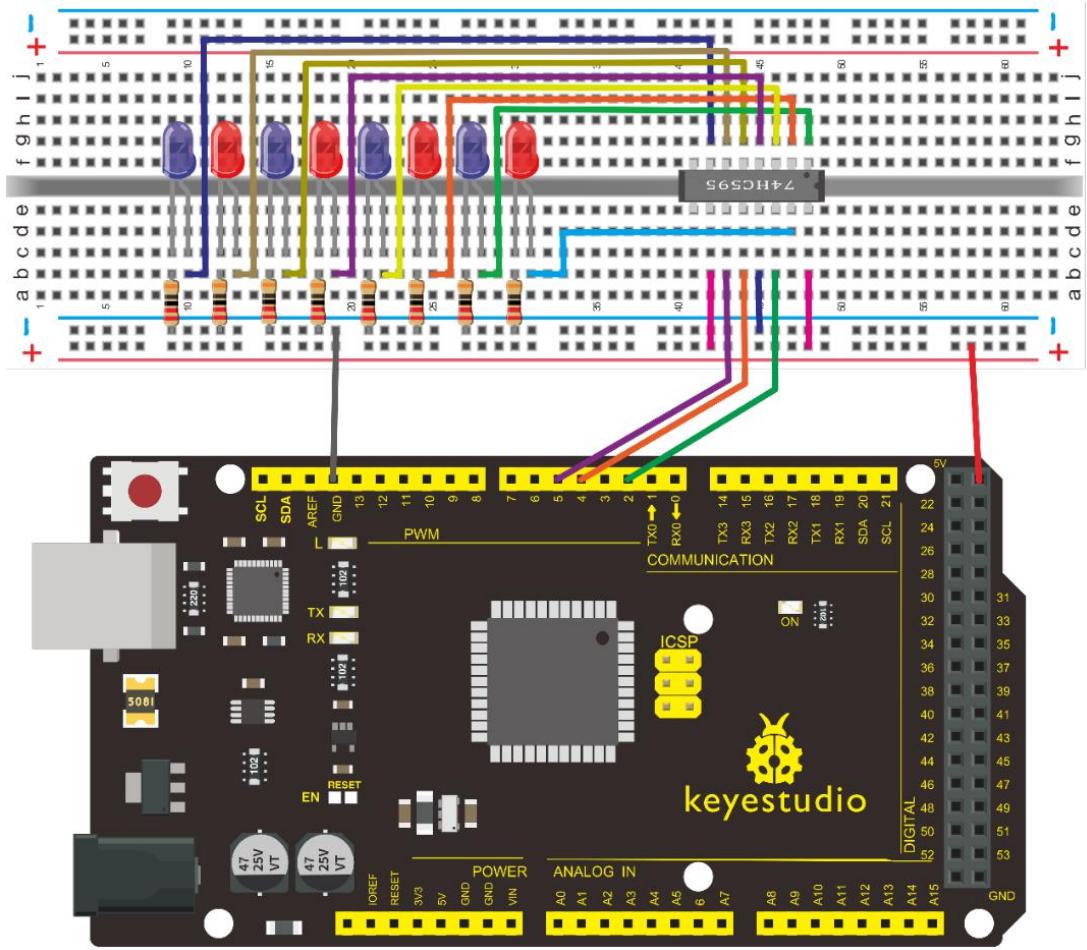


Схема может показаться сложной, но как только вы подключите ее по порядку, вам станет проще!

## 5. образец кода

```
/*
```

keyestudio супер обучающий комплект

Project 16

74hc595

<http://www.keyestudio.com>

```
* /  
  
int data = 2; // установить вывод 14 74HC595 как вывод данных ввода SI int clock = 5; //  
  
установить вывод 11 74hc595 как вывод синхронизации SCK  
  
int latch = 4; // устанавливаем вывод 12 74hc595 как выходную защелку RCK int ledState =  
0;  
  
const int ON = ВЫСОКИЙ; const  
  
int OFF = LOW; установка void  
  
()  
  
{  
  
pinMode (данные, ВЫХОД);  
  
pinMode (часы, ВЫХОД);  
  
pinMode (зашелка, ВЫХОД);  
  
}  
  
пустой цикл ()  
  
{  
  
for (int i = 0; i < 256; i++) {  
  
updateLEDs (i);  
  
задержка (500);  
  
}  
  
}  
  
void updateLEDs (значение int)
```

```
{
```

```
    digitalWrite (zaщелка, LOW); //
```

```
    shiftOut (data, clock, MSBFIRST, ~ value); // «вывод» последовательных данных, сначала высокий уровень
```

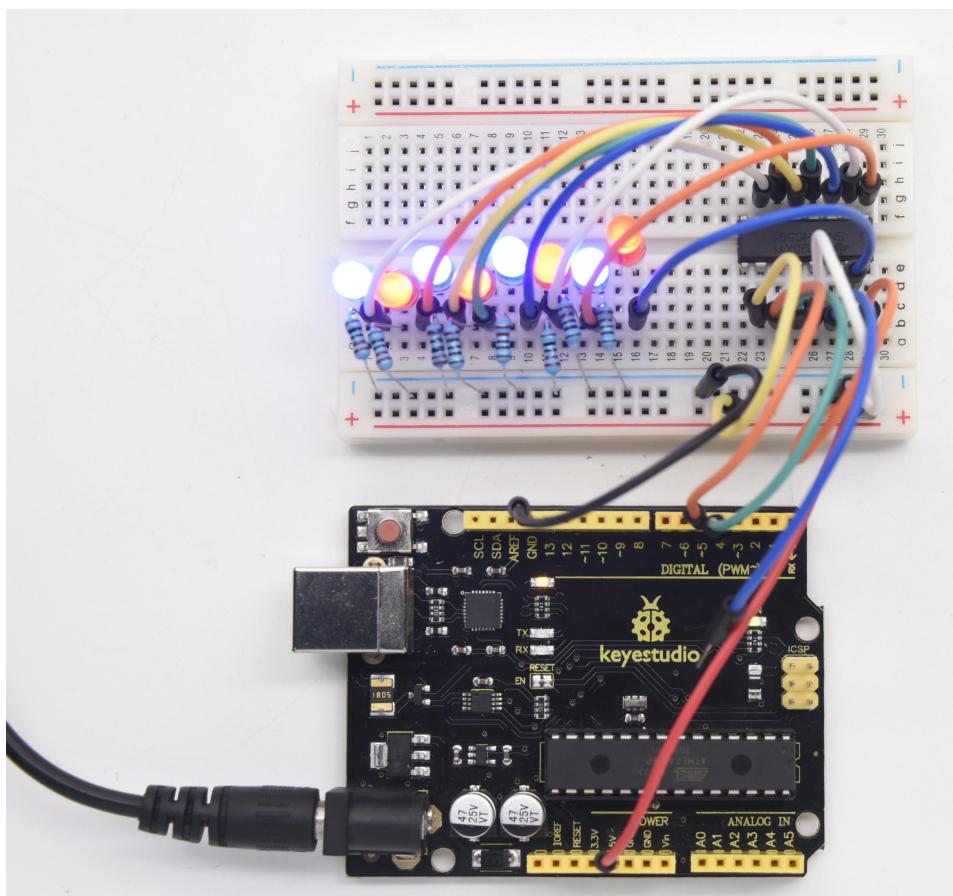
```
    digitalWrite (latch, HIGH); // защелка
```

```
}
```

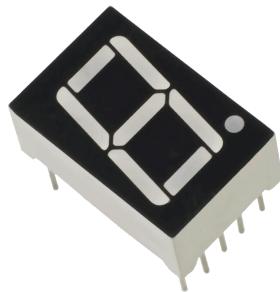
```
|||||||||||||||||||||||||||||||||||||||
```

## 6. результат теста

После загрузки программы вы увидите, что 8 светодиодов отображают 8-битное двоичное число.



## Проект 17: 1-значный сегментный светодиодный дисплей



### 1. Введение

Светодиодные сегментные дисплеи обычно используются для отображения числовой информации.

Это широко применяется на отображает  
электромагнитной печи, полностью автоматической стиральной машины,  
индикатора температуры воды, электронных часов и т. д. Нам необходимо  
изучить, как это работает.

### 2. Требуется оборудование

- Плата V4.0 или плата MEGA 2650 \* 1
- 1-значный сегментный светодиодный дисплей \* 1
- Резистор 220 Ом \* 8
- Макетная плата \* 1
- Провода перемычки макетной платы \* несколько
- USB-кабель \* 1

### **3. принцип отображения одноразрядного светодиодного сегментного дисплея**

Светодиодный сегментный дисплей представляет собой полупроводниковый светоизлучающий прибор.

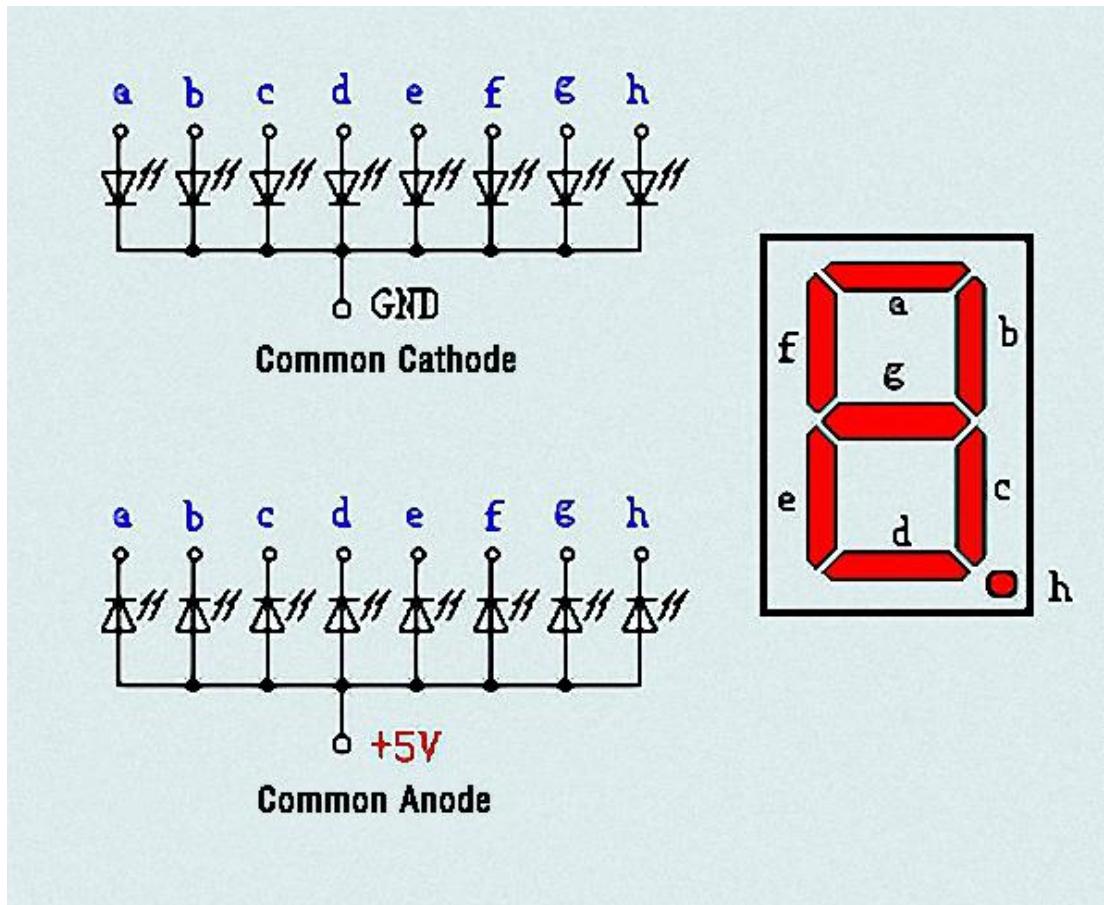
Его основной элемент - светодиод (LED).

Светодиодный сегментный дисплей можно разделить на 7-сегментный дисплей и 8-сегментный дисплей в зависимости от количества сегментов. 8-сегментный дисплей имеет на один светодиодный блок (для десятичной точки) больше, чем 7-сегментный.

Согласно способу подключения светодиодных блоков, сегментный светодиодный дисплей можно разделить на дисплей с общим анодом и дисплей с общим катодом. Дисплей с общим анодом - это дисплей, который объединяет все аноды светодиодных блоков в один общий анод (COM).

Для дисплея с общим анодом подключите общий анод (COM) к + 5В. Когда уровень катода определенного сегмента низкий, сегмент включен; когда уровень катода определенного сегмента высок, сегмент выключен.

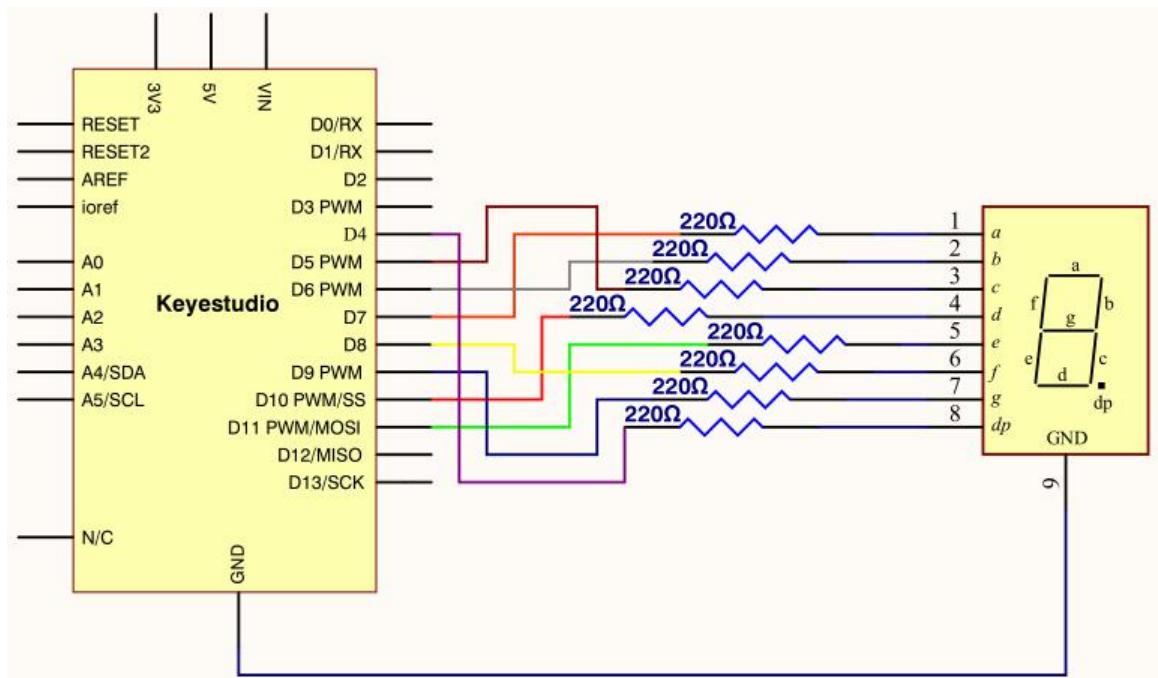
Для дисплея с общим катодом подключите общий катод (COM) к GND. Когда уровень анода определенного сегмента высок, сегмент включен; когда уровень анода определенного сегмента низкий, сегмент выключен.



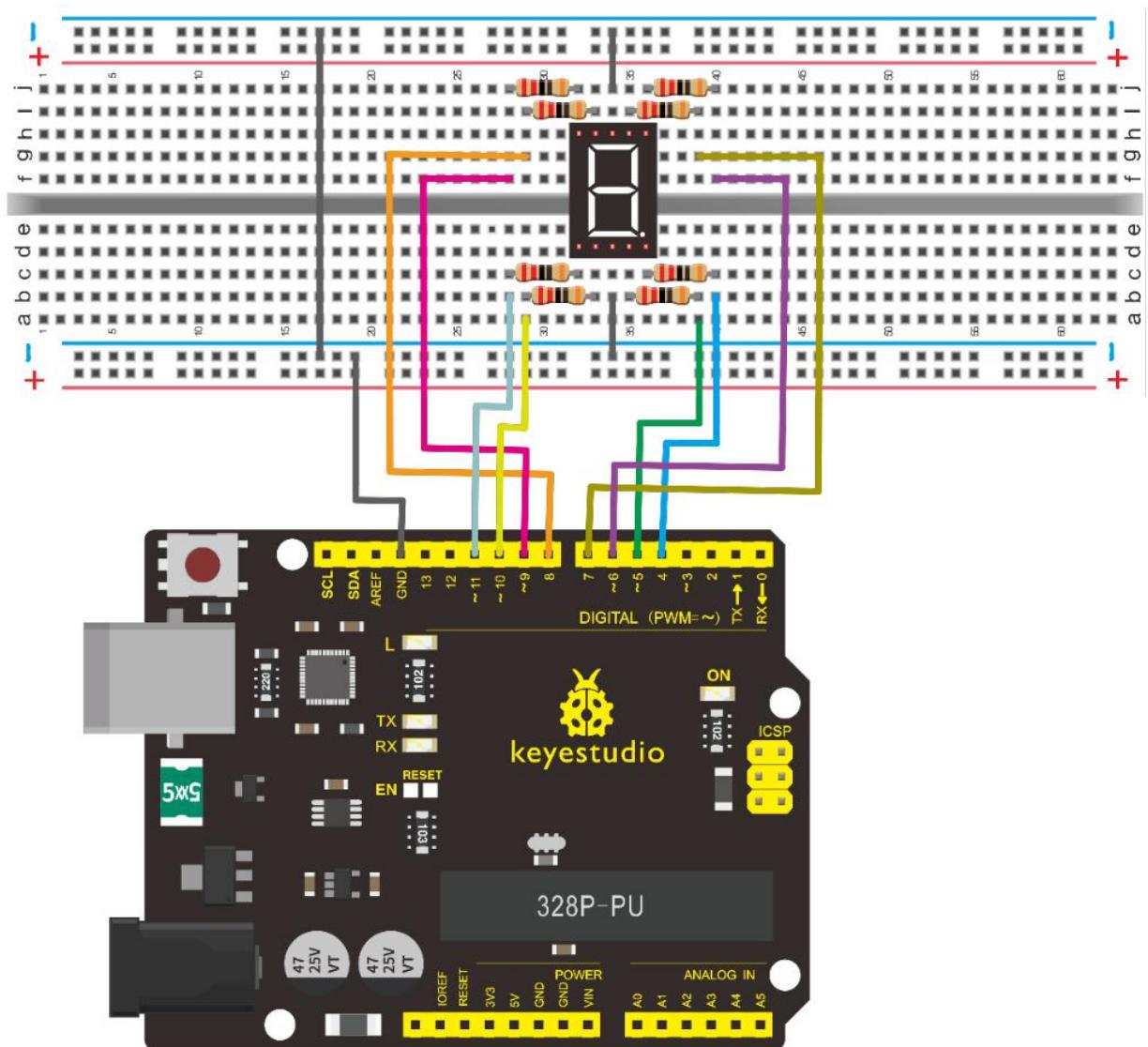
Каждый сегмент дисплея состоит из светодиода. Поэтому, когда вы его используете, вам также необходимо использовать резистор, ограничивающий ток. В противном случае светодиод перегорит.

В этом эксперименте мы используем обычный катодный дисплей. Как мы упоминали выше, для отображения общего катода подключите общий катод (COM) к GND. Когда уровень анода определенного сегмента высок, сегмент включен; когда уровень анода определенного сегмента низкий, сегмент выключен.

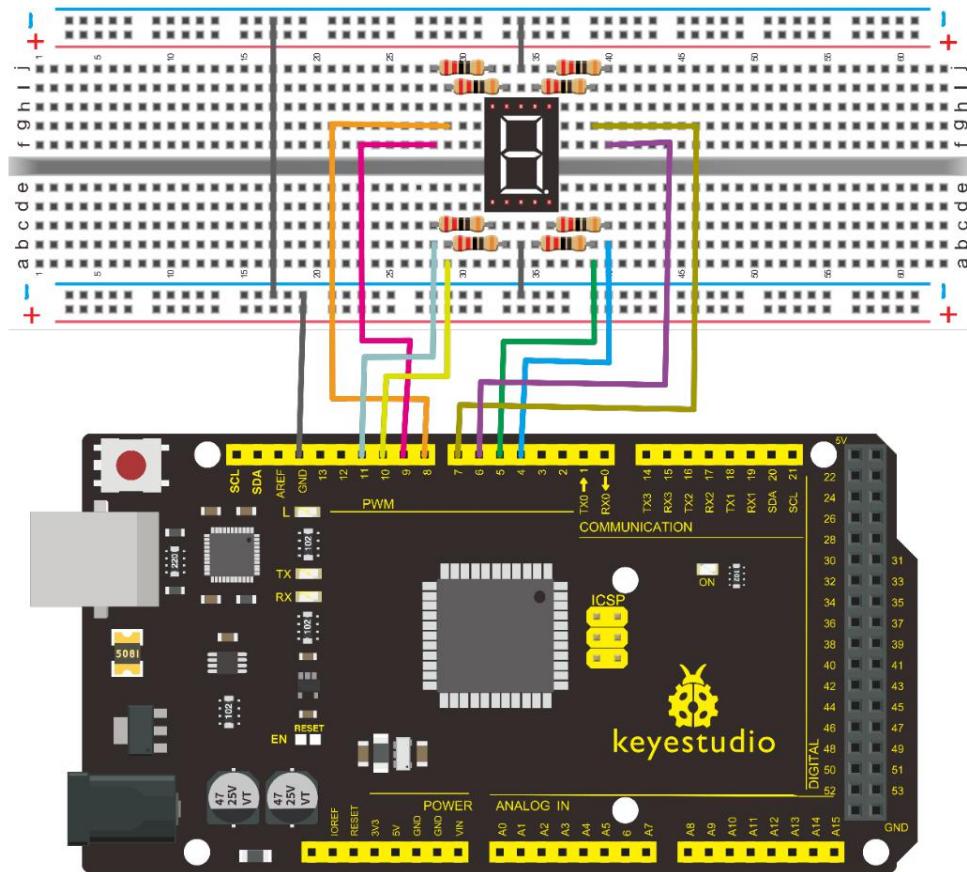
#### 4. схема подключения



Подключение для V4.0



Подключение для 2560



## 5. образец кода

Есть семь сегментов для числового отображения, один для отображения десятичной точки. Соответствующие сегменты будут включены при отображении определенных чисел.

Например, при отображении числа 1 будут включены сегменты b и c. Мы составляем подпрограмму для каждого числа и компилируем основную программу так, чтобы отображать одно число каждые 2 секунды, циклически меняя отображение числа 0 ~ 9.

Время отображения для каждого числа зависит от времени задержки, чем больше время задержки, тем дольше отображается

время.

/\*

супер обучающий комплект keyestudio

Project 17

1-значный сегментный светодиодный дисплей

<http://www.keyestudio.com>

\*/

// устанавливаем вывод IO для каждого сегмента

int a = 7; // установить цифровой контакт 7 для сегмента a int b = 6; //

установить цифровой контакт 6 для сегмента b int c = 5; // установить

цифровой контакт 5 для сегмента c

int d = 10; // установить цифровой контакт 10 для сегмента d int e = 11; //

установить цифровой контакт 11 для сегмента e int f = 8; // установить

цифровой контакт 8 для сегмента f int g = 9; // установить цифровой

вывод 9 для сегмента g int dp = 4; // установить цифровой вывод 4 для

сегмента dp void digital\_0 (void) // отобразить номер 5 {

беззнаковый символ j;

digitalWrite (a, ВЫСОКИЙ);

digitalWrite (b, HIGH);

```
digitalWrite (c, HIGH);

digitalWrite (d, HIGH);

digitalWrite (e, HIGH);

digitalWrite (f, ВЫСОКИЙ);

digitalWrite (g, LOW);

digitalWrite (dp, LOW);

}

void digital_1 (void) // отображение числа 1 {

    беззнаковый символ j;

    digitalWrite (c, HIGH); // устанавливаем «высокий» уровень для вывода 5, включаем сегмент c digitalWrite (b,
    HIGH); // включаем сегмент b

    for (j = 7; j <= 11; j++) // выключить другие сегменты digitalWrite
        (j, LOW);

    digitalWrite (dp, LOW); // выключить сегмент dp}

void digital_2 (void) // отображение числа 2 {

    беззнаковый символ j;

    digitalWrite (b, HIGH);

    digitalWrite (a, ВЫСОКИЙ);

    для (j = 9; j <= 11; j++)
```

```
digitalWrite (j, HIGH);

digitalWrite (dp, LOW);

digitalWrite (c, LOW);

digitalWrite (f, LOW);

}

void digital_3 (void) // отображение числа 3

{digitalWrite (g, HIGH);

digitalWrite (a, ВЫСОКИЙ);

digitalWrite (b, HIGH);

digitalWrite (c, HIGH);

digitalWrite (d, HIGH);

digitalWrite (dp, LOW);

digitalWrite (f, LOW);

digitalWrite (e, LOW);

}

void digital_4 (void) // отображение числа 4

{digitalWrite (c, HIGH);

digitalWrite (b, HIGH);

digitalWrite (f, ВЫСОКИЙ);

digitalWrite (g, HIGH);

digitalWrite (dp, LOW);

digitalWrite (a, LOW);
```

```
digitalWrite (e, LOW);

digitalWrite (d, LOW);

}
```

```
void digital_5 (void) // отображение числа 5 {
```

```
беззнаковый символ j;
```

```
digitalWrite (a, ВЫСОКИЙ);
```

```
digitalWrite (b, LOW);
```

```
digitalWrite (c, HIGH);
```

```
digitalWrite (d, HIGH);
```

```
digitalWrite (e, LOW);
```

```
digitalWrite (f, ВЫСОКИЙ);
```

```
digitalWrite (g, HIGH);
```

```
digitalWrite (dp, LOW);
```

```
}
```

```
void digital_6 (void) // отображение числа 6 {
```

```
беззнаковый символ j;
```

```
для (j = 7; j <= 11; j ++)
```

```
digitalWrite (j, HIGH);
```

```
digitalWrite (c, HIGH);
```

```
digitalWrite (dp, LOW);
```

```
digitalWrite (b, LOW);

}
```

```
void digital_7 (void) // отображение числа 7 {
```

```
беззнаковый символ j;
```

```
для (j = 5; j <= 7; j ++)
```

```
digitalWrite (j, HIGH);
```

```
digitalWrite (dp, LOW);
```

```
для (j = 8; j <= 11; j ++)
```

```
digitalWrite (j, LOW);
```

```
}
```

```
void digital_8 (void) // отображение числа 8 {
```

```
беззнаковый символ j;
```

```
для (j = 5; j <= 11; j ++)
```

```
digitalWrite (j, HIGH);
```

```
digitalWrite (dp, LOW);
```

```
}
```

```
void digital_9 (void) // отображение числа 5 {
```

```
беззнаковый символ j;
```

```
digitalWrite (a, ВЫСОКИЙ);
```

```
digitalWrite (b, HIGH);

digitalWrite (c, HIGH);

digitalWrite (d, HIGH);

digitalWrite (e, LOW);

digitalWrite (f, ВЫСОКИЙ);

digitalWrite (g, HIGH);

digitalWrite (dp, LOW);

}

установка void ()

{

int i; // устанавливаем переменную

для (i = 4; i <= 11; i ++)

pinMode (i, OUTPUT); // устанавливаем вывод 4-11 как «выход»}

пустой цикл ()

{

пока (1)

{

digital_0 (); // отображение числа 0 delay

(1000); // ожидание 1 с digital_1 (); //

отображение числа 1 delay (1000); //

ожидание 1 с
```

```
digital_2 (); // вывод числа 2 delay (1000);

// ждем 1 с digital_3 () // отображаем

цифру 3 delay (1000); // ждем 1 с digital_4

(); // выводим число 4 delay (1000); //

ждем 1 с digital_5 () // выводим цифру 5

delay (1000); // ждем 1 с digital_6 () //

выводим число 6 delay (1000); // ждем 1

с digital_7 () // вывод 7 delay (1000); //

ждем 1 с digital_8 () // вывод числа 8

delay (1000); // ждем 1 с digital_9 () //

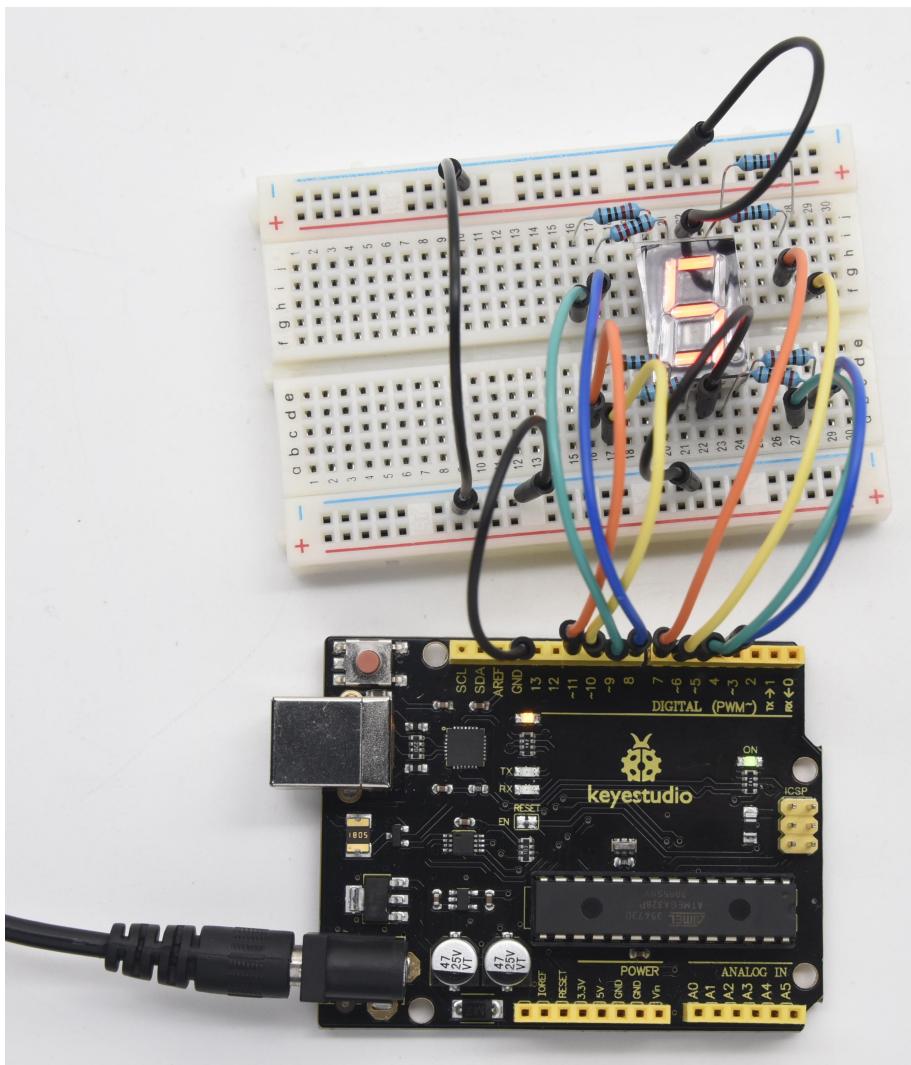
выводим число 9 delay (1000); // ждем 1

c} }
```

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

#### 6. результат теста

Светодиодный сегментный дисплей покажет число от 0 до 9.



Проект 18: 4-значный сегментный светодиодный дисплей



## 1. Введение

В этом эксперименте мы используем Arduino для управления общим катодным 4-значным 7-сегментным светодиодным дисплеем.

Для светодиодного дисплея незаменимы токоограничивающие резисторы. Существует два способа подключения токоограничивающего резистора. Один заключается в подключении одного резистора к каждому концу катода, всего 4 резистора для катода d1-d4.

Преимущество этого метода заключается в том, что для него требуется меньше резисторов, всего 4. Но он не может поддерживать постоянную яркость, 1, самую высокую; 8, наименее яркая. Другой метод - подключить по одному резистору к каждому выводу. Это гарантирует постоянную яркость, но требует больше резисторов.

В этом эксперименте мы используем 8 резисторов (220 Ом). Мы используем резисторы 220 Ом из-за отсутствия резистора 100 Ом. Если вы используете 100 Ом, изображение будет ярче.

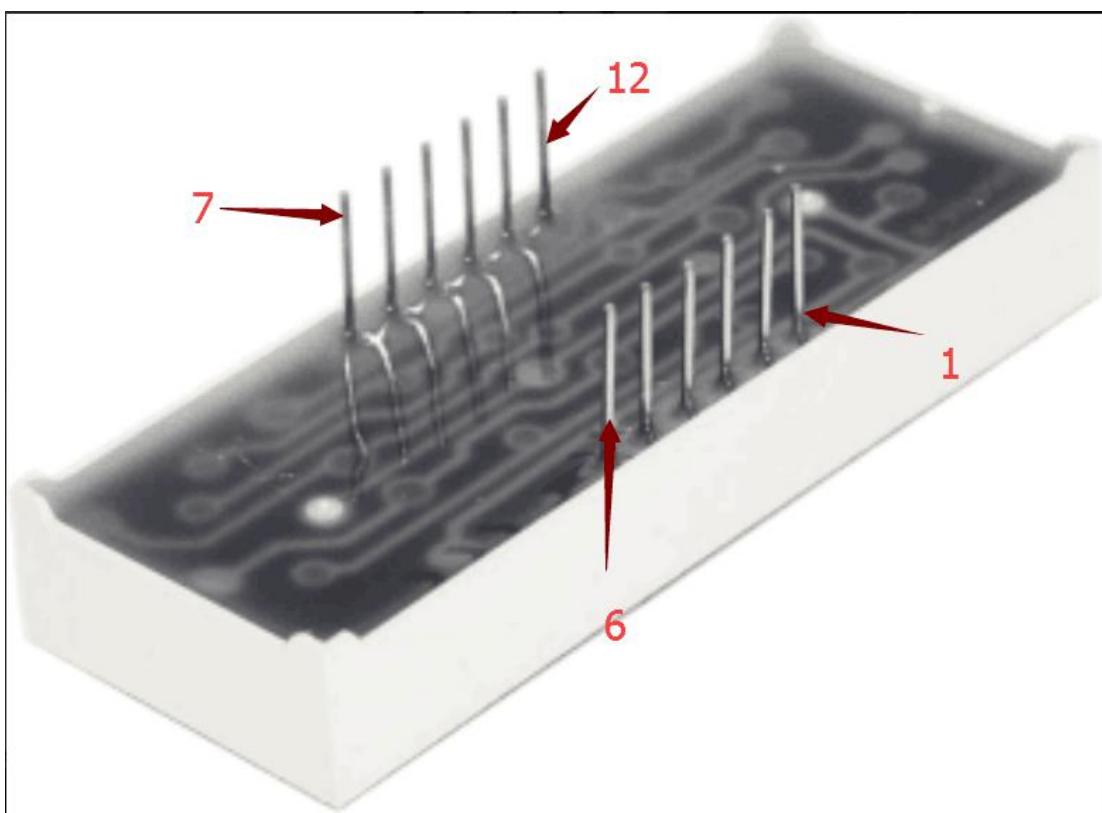
## **2. Требуется оборудование**

- Плата V4.0 или плата MEGA 2650 \* 1
- 4-значный сегментный светодиодный дисплей \* 1
- Резистор 220 Ом \* 8
- Макетная плата \* 1
- Провода перемычки макетной платы \* несколько

- USB-кабель \* 1

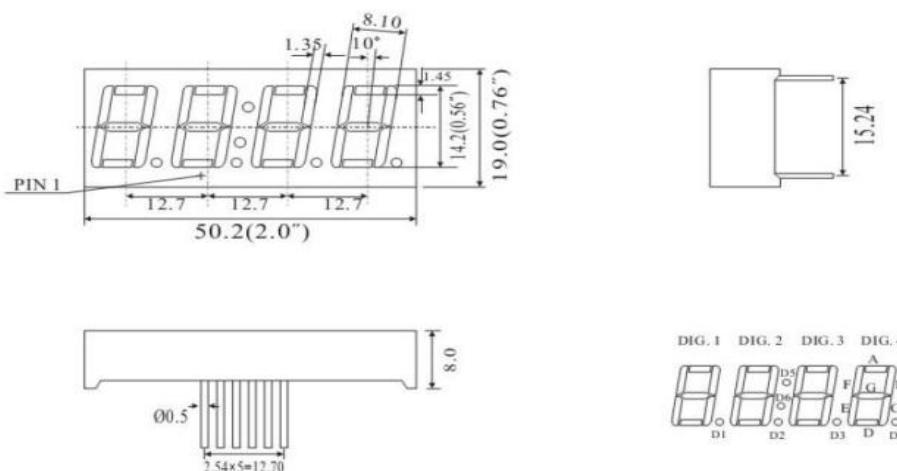
### 3. принцип отображения 4-значного дисплея

Для 4-значного дисплея всего 12 контактов. Когда вы помещаете десятичную точку вниз, штифт в левой нижней части обозначается как 1, а верхняя левая часть - 12. Показано ниже.



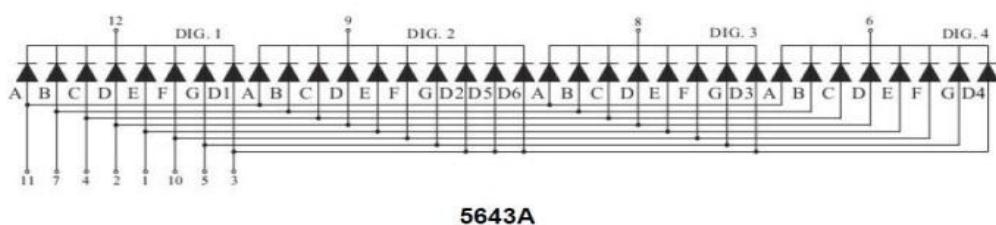
### Package Dimensions

**CPS05643AB**

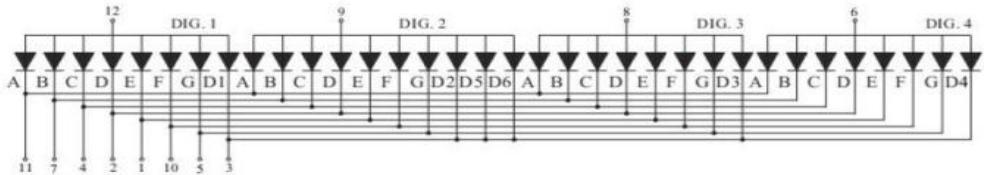


UNIT: MM(INCH) TOLERANCE:  $\pm 0.25(0.01")$

### Internal Circuit Diagram



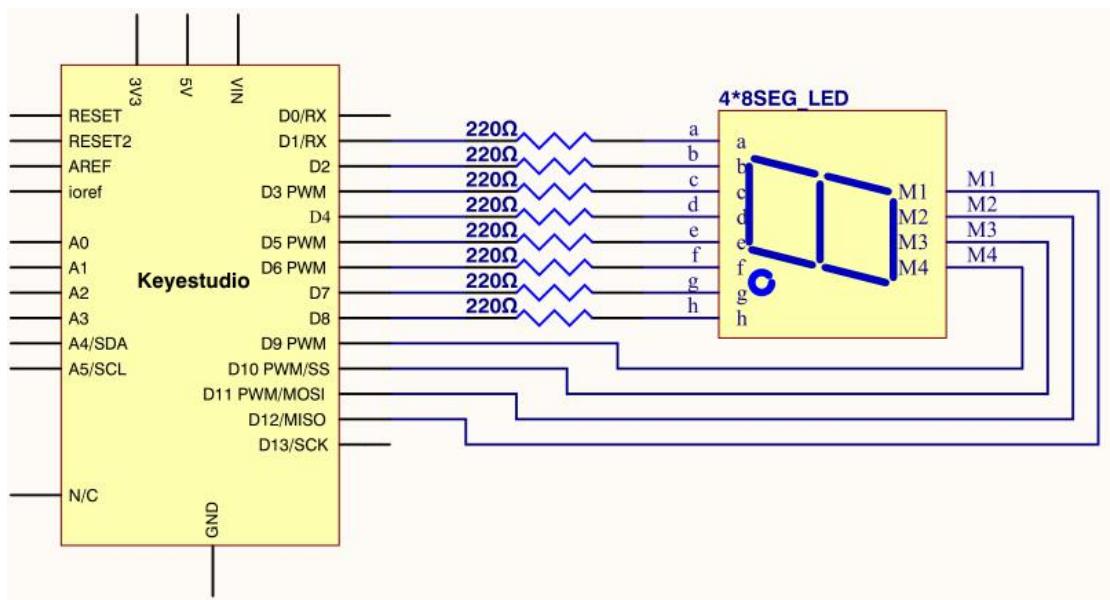
**5643A**



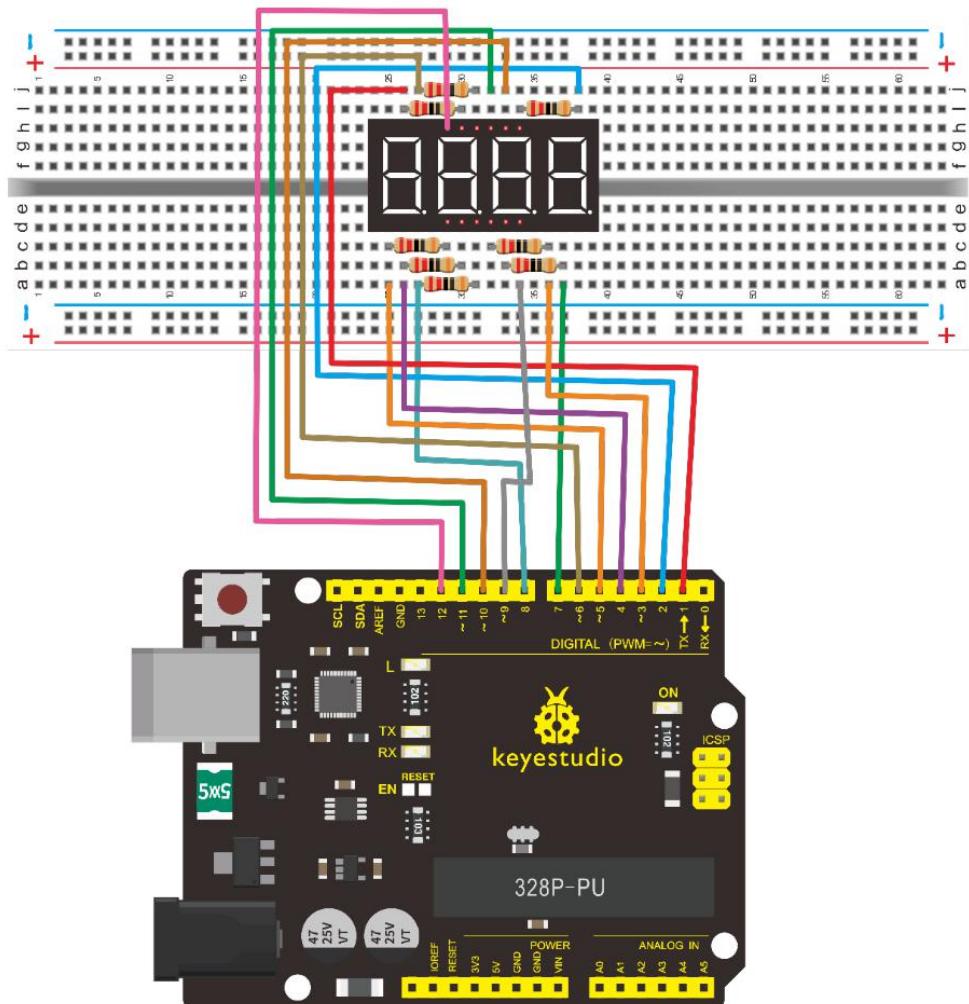
**5643B**

### Four Digits Displays Series

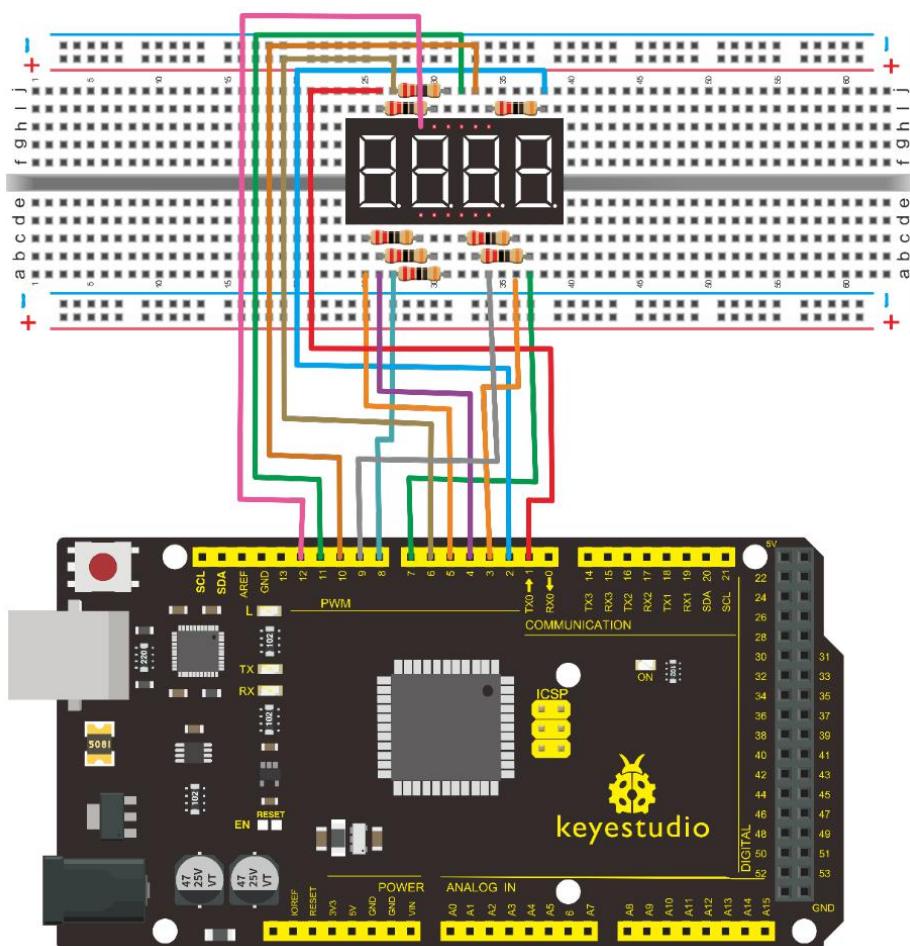
#### 4. схема подключения



### Подключение для V4.0



## Подключение для 2560 R3



## 5. образец кода

```
/*
```

```
супер обучающий комплект keyestudio
```

```
Project 18
```

```
4-значный сегментный светодиодный дисплей
```

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

```
* /  
  
// отображение 1234  
  
// выбираем вывод для катода int a  
  
= 1;  
  
int b = 2;  
  
int c = 3;  
  
int d = 4;  
  
int e = 5;  
  
int f = 6;  
  
int g = 7;  
  
int dp = 8;  
  
// выбираем вывод для анода int  
  
d4 = 9;  
  
int d3 = 10;  
  
int d2 = 11;  
  
int d1 = 12;  
  
// установить переменную  
  
длинные n = 1230;  
  
int x = 100;  
  
int del = 55; // точная настройка часов  
  
установка void ()
```

```
{  
  
pinMode (d1, ВЫХОД);  
  
pinMode (d2, ВЫХОД);  
  
pinMode (d3, ВЫХОД);  
  
pinMode (d4, ВЫХОД);  
  
pinMode (a, ВЫХОД);  
  
pinMode (b, ВЫХОД);  
  
pinMode (c, ВЫХОД);  
  
pinMode (d, ВЫХОД);  
  
pinMode (e, ВЫХОД);  
  
pinMode (f, ВЫХОД);  
  
pinMode (g, ВЫХОД);  
  
pinMode (дп, ВЫХОД);  
  
}  
  
/////////////////////////////// void loop ()  
  
{  
  
Дисплей (1, 1);  
  
Дисплей (2, 2);  
  
Дисплей (3, 3);  
  
Дисплей (4, 4);
```

```
}
```

```
|||||||||||||||||||||||||||||||||||| void WeiXuan (unsigned char n) //
```

```
{
```

```
    переключатель (n)
```

```
{
```

```
Случай 1:
```

```
    digitalWrite (d1, LOW);
```

```
    digitalWrite (d2, HIGH);
```

```
    digitalWrite (d3, HIGH);
```

```
    digitalWrite (d4, HIGH);
```

```
    сломать;
```

```
случай 2:
```

```
    digitalWrite (d1, HIGH);
```

```
    digitalWrite (d2, LOW);
```

```
    digitalWrite (d3, HIGH);
```

```
    digitalWrite (d4, HIGH);
```

```
    сломать;
```

```
случай 3:
```

```
    digitalWrite (d1, HIGH);
```

```
    digitalWrite (d2, HIGH);
```

```
    digitalWrite (d3, LOW);
```

```
digitalWrite (d4, HIGH);
```

```
сломать;
```

```
случай 4:
```

```
digitalWrite (d1, HIGH);
```

```
digitalWrite (d2, HIGH);
```

```
digitalWrite (d3, HIGH);
```

```
digitalWrite (d4, LOW);
```

```
сломать;
```

```
по умолчанию :
```

```
digitalWrite (d1, HIGH);
```

```
digitalWrite (d2, HIGH);
```

```
digitalWrite (d3, HIGH);
```

```
digitalWrite (d4, HIGH);
```

```
сломать;
```

```
}
```

```
}
```

```
void Num_0 ()
```

```
{
```

```
digitalWrite (a, ВЫСОКИЙ);
```

```
digitalWrite (b, HIGH);
```

```
digitalWrite (c, HIGH);
```

```
digitalWrite (d, HIGH);
```

```
digitalWrite (e, HIGH);

digitalWrite (f, ВЫСОКИЙ);

digitalWrite (g, LOW);

digitalWrite (dp, LOW);

}
```

```
void Num_1 ()
```

```
{

digitalWrite (a, LOW);

digitalWrite (b, HIGH);

digitalWrite (c, HIGH);

digitalWrite (d, LOW);

digitalWrite (e, LOW);

digitalWrite (f, LOW);

digitalWrite (g, LOW);

digitalWrite (dp, LOW);

}
```

```
void Num_2 ()
```

```
{

digitalWrite (a, ВЫСОКИЙ);

digitalWrite (b, HIGH);

digitalWrite (c, LOW);

digitalWrite (d, HIGH);
```

```
digitalWrite (e, HIGH);

digitalWrite (f, LOW);

digitalWrite (g, HIGH);

digitalWrite (dp, LOW);

}
```

```
void Num_3 ()
```

```
{

digitalWrite (a, ВЫСОКИЙ);

digitalWrite (b, HIGH);

digitalWrite (c, HIGH);

digitalWrite (d, HIGH);

digitalWrite (e, LOW);

digitalWrite (f, LOW);

digitalWrite (g, HIGH);

digitalWrite (dp, LOW);

}
```

```
void Num_4 ()
```

```
{

digitalWrite (a, LOW);

digitalWrite (b, HIGH);

digitalWrite (c, HIGH);

digitalWrite (d, LOW);
```

```
digitalWrite (e, LOW);

digitalWrite (f, ВЫСОКИЙ);

digitalWrite (g, HIGH);

digitalWrite (dp, LOW);

}
```

```
void Num_5 ()
```

```
{

digitalWrite (a, ВЫСОКИЙ);

digitalWrite (b, LOW);

digitalWrite (c, HIGH);

digitalWrite (d, HIGH);

digitalWrite (e, LOW);

digitalWrite (f, ВЫСОКИЙ);

digitalWrite (g, HIGH);
```

```
digitalWrite (dp, LOW);
```

```
}
```

```
void Num_6 ()
```

```
{

digitalWrite (a, ВЫСОКИЙ);

digitalWrite (b, LOW);

digitalWrite (c, HIGH);

digitalWrite (d, HIGH);
```

```
digitalWrite (e, HIGH);

digitalWrite (f, ВЫСОКИЙ);

digitalWrite (g, HIGH);

digitalWrite (dp, LOW);

}
```

```
void Num_7 ()
```

```
{

digitalWrite (a, ВЫСОКИЙ);

digitalWrite (b, HIGH);

digitalWrite (c, HIGH);

digitalWrite (d, LOW);

digitalWrite (e, LOW);

digitalWrite (f, LOW);

digitalWrite (g, LOW);

digitalWrite (dp, LOW);
```

```
}
```

```
void Num_8 ()
```

```
{

digitalWrite (a, ВЫСОКИЙ);

digitalWrite (b, HIGH);

digitalWrite (c, HIGH);

digitalWrite (d, HIGH);
```

```
digitalWrite (e, HIGH);

digitalWrite (f, ВЫСОКИЙ);

digitalWrite (g, HIGH);

digitalWrite (dp, LOW);

}
```

```
void Num_9 ()

{

digitalWrite (a, ВЫСОКИЙ);

digitalWrite (b, HIGH);

digitalWrite (c, HIGH);

digitalWrite (d, HIGH);

digitalWrite (e, LOW);

digitalWrite (f, ВЫСОКИЙ);

digitalWrite (g, HIGH);

digitalWrite (dp, LOW);

}
```

```
void Clear () // очистить экран {
```

```
digitalWrite (a, LOW);

digitalWrite (b, LOW);

digitalWrite (c, LOW);

digitalWrite (d, LOW);
```

```
digitalWrite (e, LOW);

digitalWrite (f, LOW);

digitalWrite (g, LOW);

digitalWrite (dp, LOW);

}

void pickNumber (unsigned char n) // выбрать число {

переключатель (п)

{

case 0: Num_0 ();

сломать;

случай 1: Num_1 ();

сломать;

случай 2: Num_2 ();

сломать;

случай 3: Num_3 ();

сломать;

случай 4: Num_4 ();

сломать;

case 5: Num_5 ();

сломать;

случай 6: Num_6 ();
```

```
    сломать;

    case 7: Num_7 ();

    сломать;

    case 8: Num_8 ();

    сломать;

    case 9: Num_9 ();

    сломать;

    по умолчанию: Clear ();

    сломать;

}

}

void Display (unsigned char x, unsigned char Number) // принять x как координату и
отобразить номер

{

    Вэйсюань (x);

    pickNumber (Число);

    задержка (1);

    Очистить() ; // очищаем экран

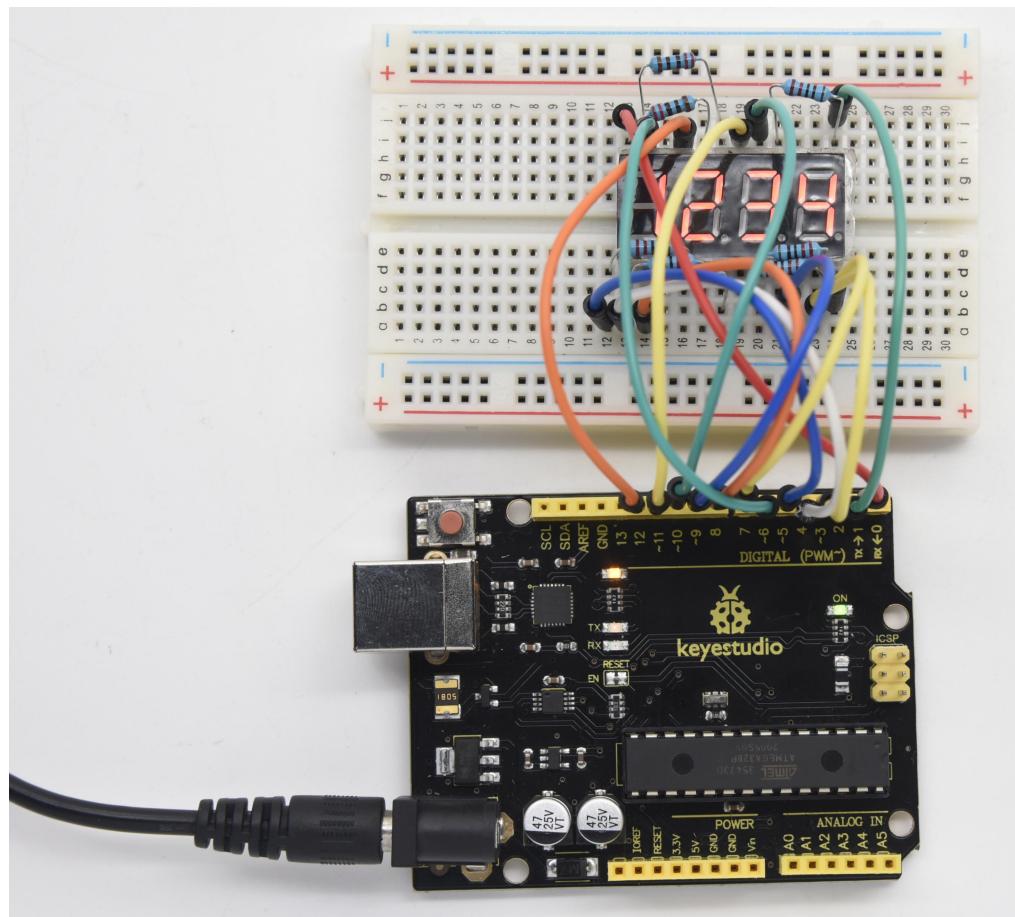
}

||||||||||||||||||||||||||||||||||||
```

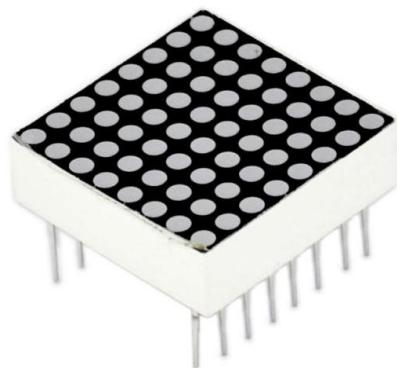
## 6. результат теста

Загрузите приведенный выше код на плату контроллера, вы увидите, что на светодиодном дисплее отображается число 1234.

**Запись** если он отображается неправильно, проверьте проводку.



#### Проект 19: светодиодная матрица 8 \* 8



## **1. Введение**

Светодиодный точечно-матричный дисплей может удовлетворить потребности различных приложений, поэтому имеет широкую перспективу развития. При низковольтном сканировании светодиодная матрица имеет некоторые преимущества, такие как энергосбережение, длительный срок службы, низкая стоимость, высокая яркость, широкий угол обзора, большой диапазон видимости, водонепроницаемость и многочисленные характеристики.

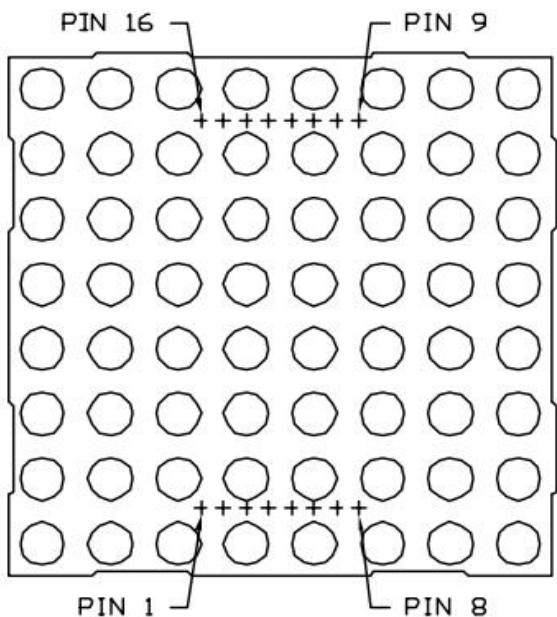
В этом проекте мы проведем эксперимент с точечной светодиодной матрицей, чтобы воочию убедиться в его очаровании.

## **2. Требуется оборудование**

- Плата V4.0 или плата MEGA 2650 \* 1
- 1 \* 8 \* 8 точечная матрица
- 8 \* резистор (220 Ом)
- 1 \* макет
- 1 \* USB-кабель
- Несколько \* перемычек

## **3. Принцип отображения точечной матрицы 8 \* 8.**

Внешний вид точечной матрицы показан следующим образом.

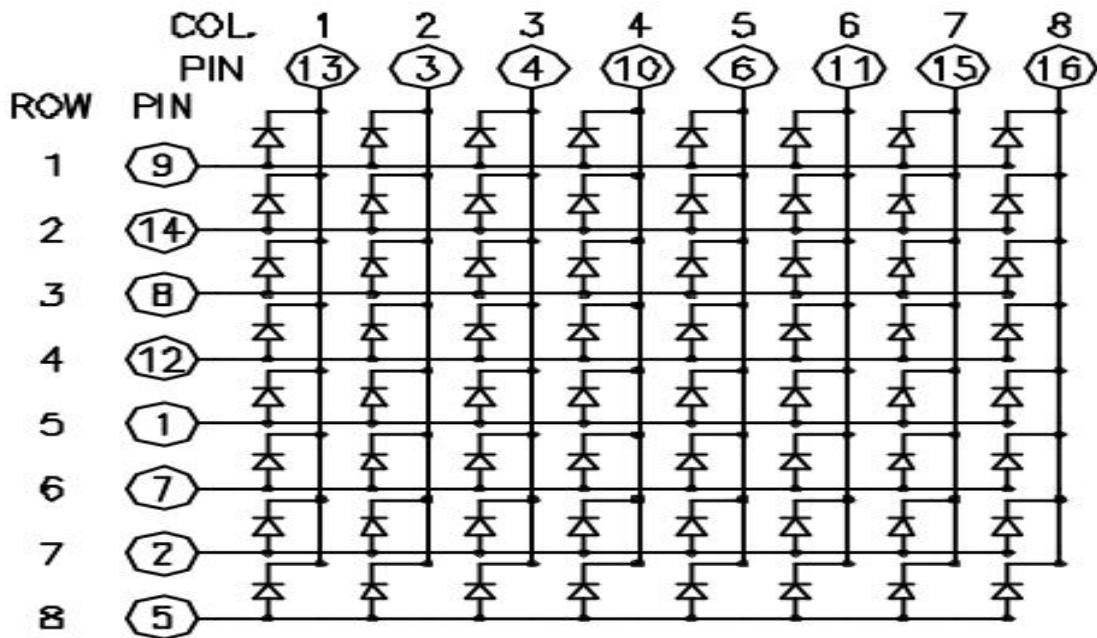


Точечная матрица 8 \* 8 состоит из шестидесяти четырех светодиодов, каждый из которых расположен в точке пересечения строки и столбца. Когда электрический уровень определенной строки равен 1, а электрический уровень определенного столбца равен 0, соответствующий светодиод загорается. Если вы хотите зажечь светодиод на первой точке, вы должны установить контакт 9 на высокий уровень, а контакт 13 на низкий уровень.

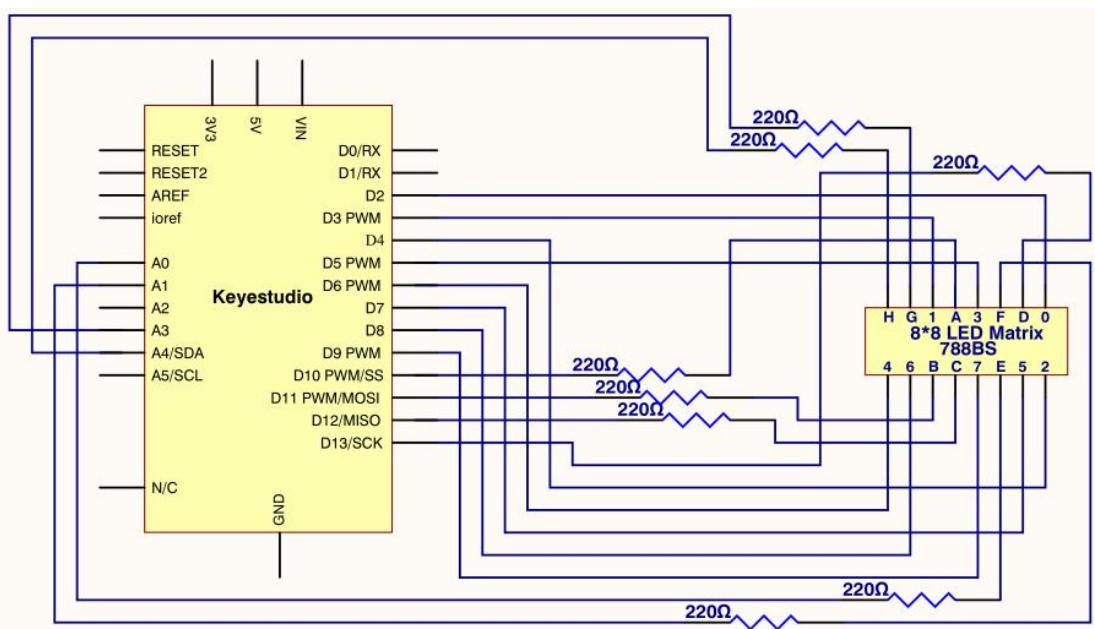
Если вы хотите зажечь светодиоды в первом ряду, вы должны установить контакт 9 на высокий уровень, а контакты 13, 3, 4, 10, 6, 11, 15 и 16 - на низкий уровень.

Если вы хотите зажечь светодиоды в первом столбце, установите вывод 13 на низкий уровень, а выводы 9, 14, 8, 12, 1, 7, 2 и 5 - на высокий уровень.

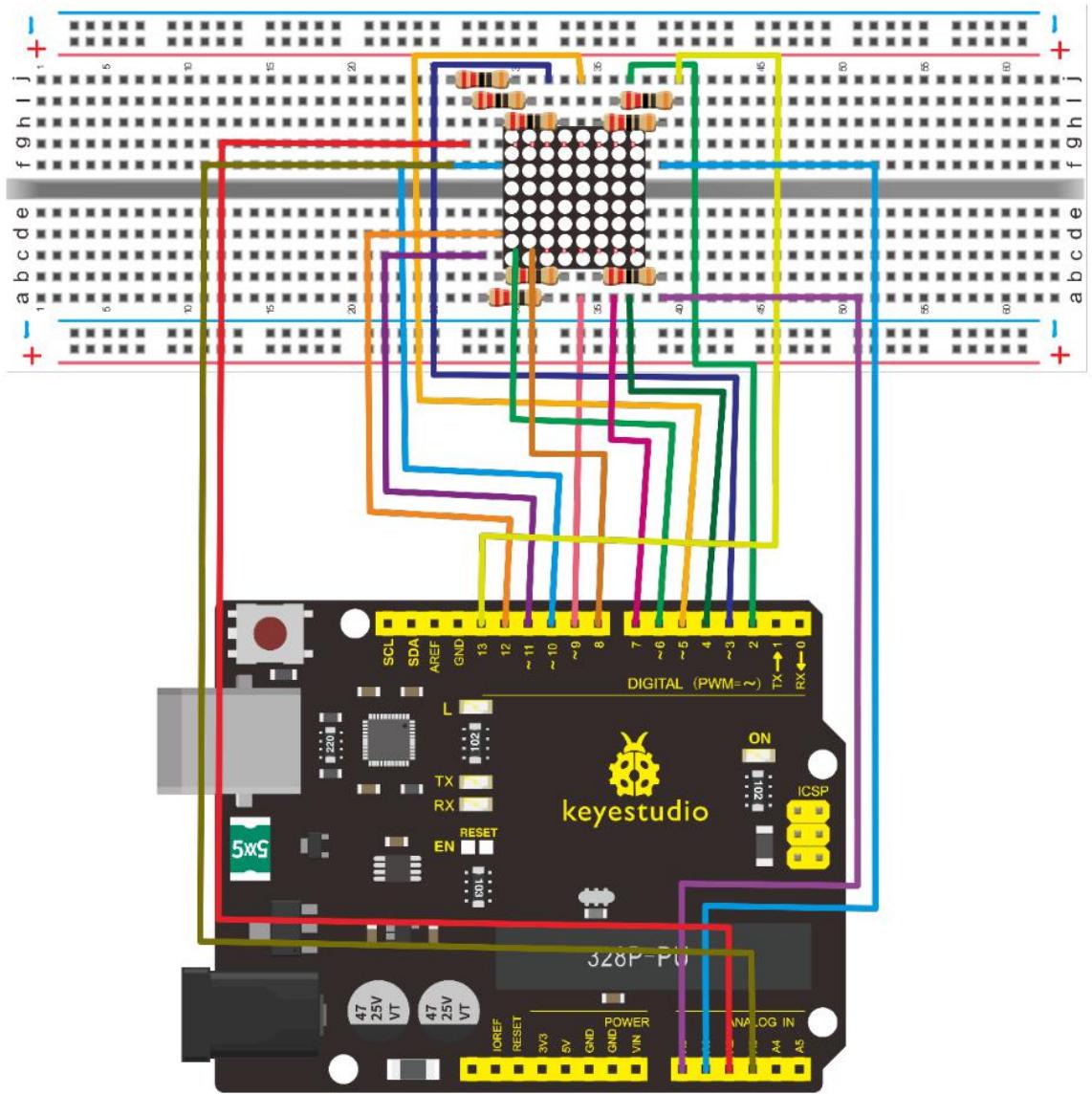
Внутренний вид точечной матрицы показан следующим образом.



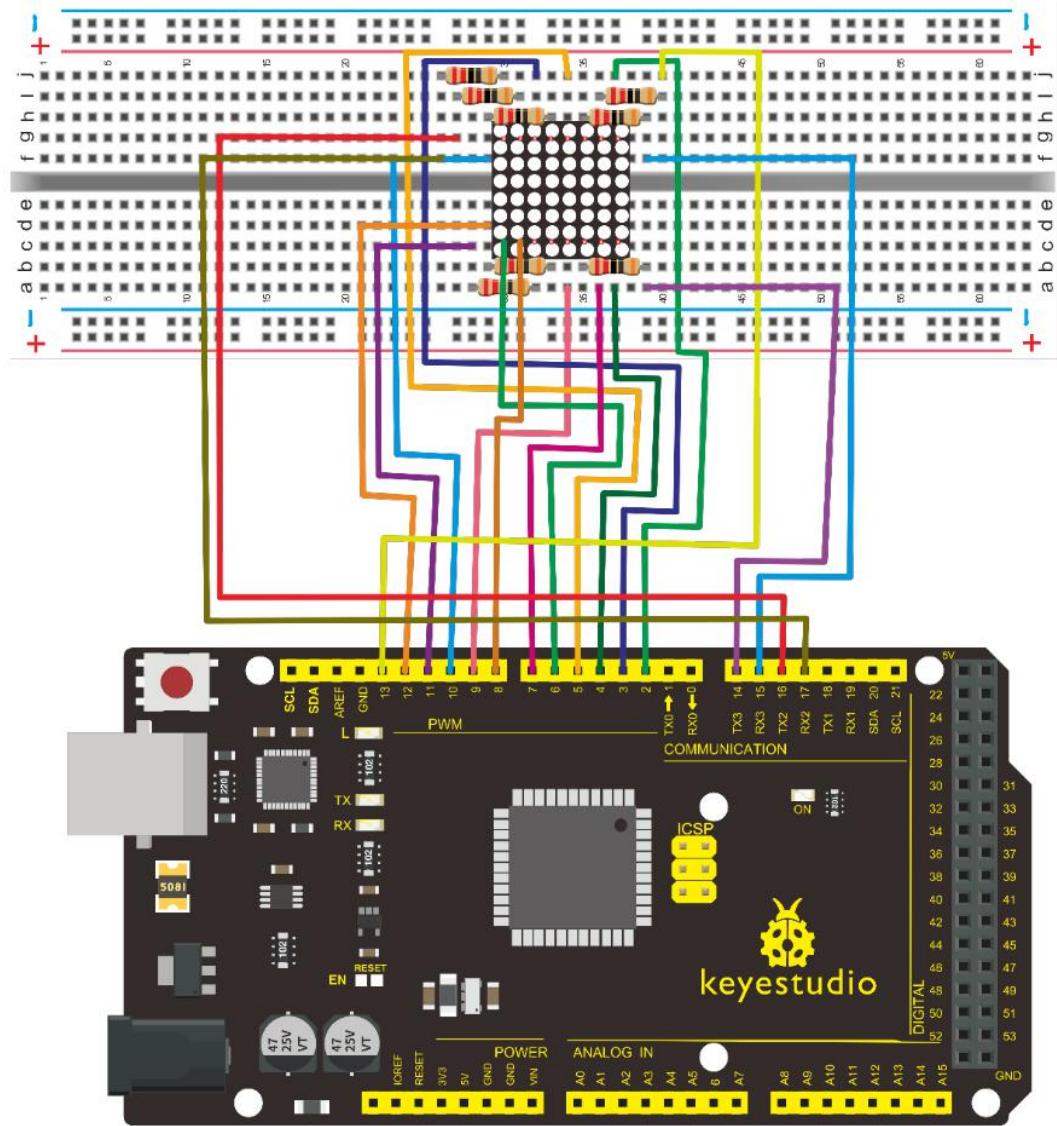
#### 4. схема подключения



Подключение для V4.0



Подключение для 2560 R3



## 5. Пример кода для отображения «0»

```
/*
```

супер обучающий комплект keyestudio

Project 19

Матрица 8 \* 8

<http://www.keyestudio.com>

```

* /



// устанавливаем массив для хранения символа «0» без знака

char



Текст [] = {0x00,0x1c, 0x22,0x22,0x22,0x22,0x22,0x1c};

void Draw_point (unsigned char x, unsigned char y) // функция рисования точки







{ Чисто_();

    digitalWrite (x + 2, ВЫСОКИЙ);

    digitalWrite (y + 10, LOW);

    задержка (1);

}

void show_num (void) // функция отображения, функция рисования точки вызова



{



    беззнаковый символ i, j, данные;

    для (я = 0; я <8; я ++)

    {

        data = Text [i];

        для (j = 0; j <8; j ++)

        {

            если (данные & 0x01) Draw_point (j, i);

            данные >> = 1;

```

```
    }

}

void setup () {
    int я = 0;

    для (я = 2; я <18; я++)
    {
        pinMode (я, ВЫХОД);

    }

    Чисто_();
}

пустой цикл ()

{show_num ();

}

void clear_ (void) // очистить экран {for (int
    i = 2; i <10; i ++)
    digitalWrite (я, LOW);

    для (int я = 0; я <8; я++)
        digitalWrite (я + 10, ВЫСОКИЙ);

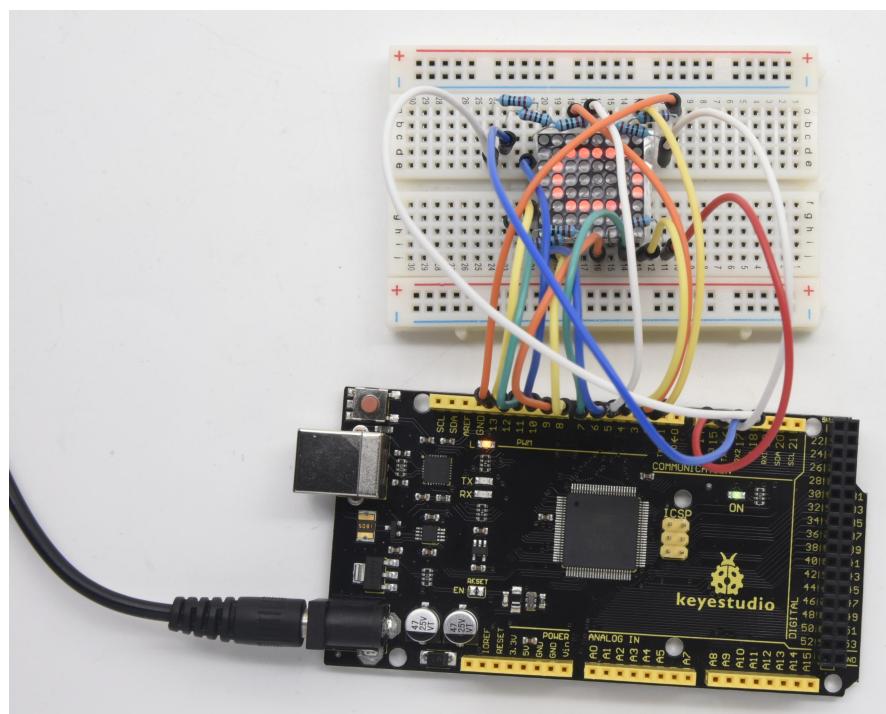
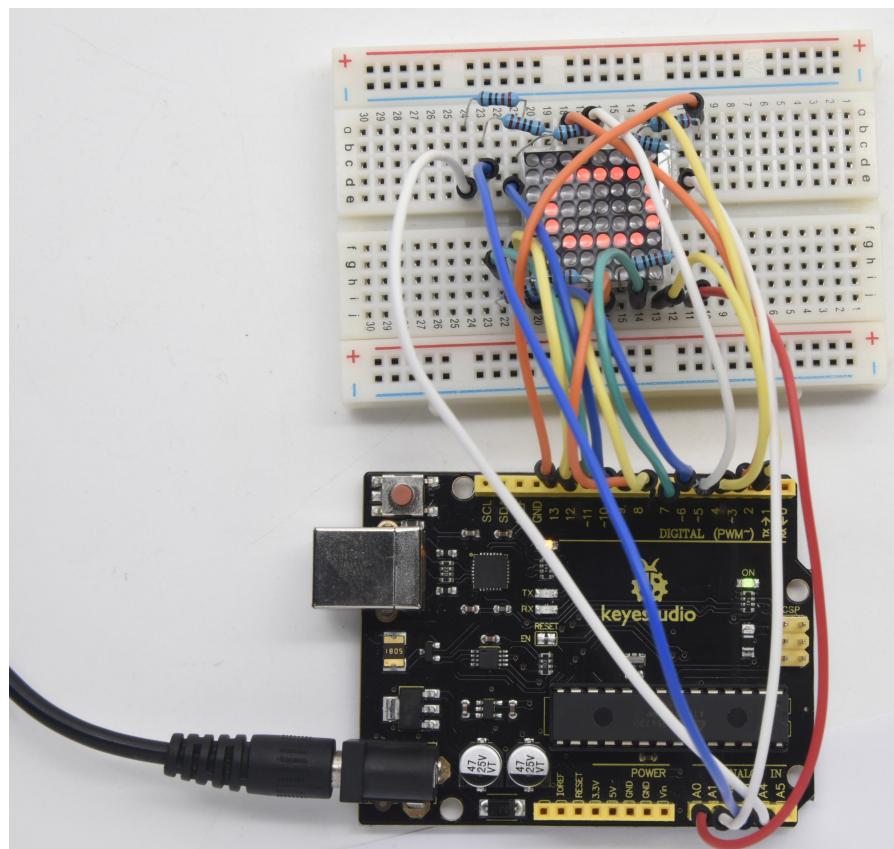
}

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

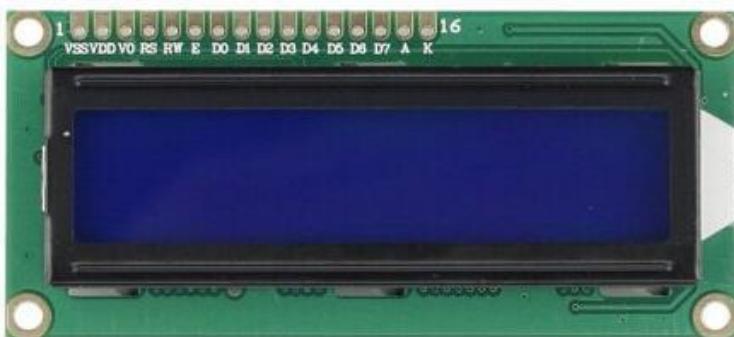
**6. результат теста**

Запишите программу на плату V4.0, точечная матрица отобразит

0.



## Проект 20: 1602 ЖК



### 1. Введение

В этом эксперименте мы используем Arduino для управления ЖК-дисплеем 1602. 1602 LCD имеет широкое применение. Вначале 1602 LCD использует контроллер HD44780.

Теперь почти все ЖК-модули 1602 используют совместимую ИС, поэтому их функции в основном одинаковы.

#### 2.1602LCD Основные параметры

- Плата V4.0 или плата MEGA 2650 \* 1
- Емкость дисплея 16 \* 2 символа
- Рабочее напряжение чипа 4,5 ~ 5,5 В
- Рабочий ток 2,0 мА (5,0 В)
- Оптимальное рабочее напряжение модуля 5,0 В.
- Размер символа 2,95 \* 4,35 (Ш \* В) мм

### 3. описание контактов 1602 LCD

Нет .	отметка	Описание пина	Нет .	отметка	Описание пина
1	VSS	Мощность GND	9	D2	Дата ввода / вывода
2	VDD	Положительная сила	10	D3	Дата ввода / вывода
3	VL	Смещение напряжения ЖК-дисплея сигнал	11	D4	Дата ввода / вывода
4	RS	Выбрать данные / команда (Об / л)	12	D5	Дата ввода / вывода
5	R / W	Выбрать чтение / запись (H / L)	13	D6	Дата ввода / вывода
6	E	Включить сигнал	14	D7	Дата ввода / вывода
7	D0	Дата ввода / вывода	15 БЛА		Мощность задней подсветки положительный
8	D1	Дата ввода / вывода	16 BLK		Мощность задней подсветки отрицательный

### 4. описание интерфейса

1. Два вывода питания, один для питания модуля, другой для подсветки, обычно используют 5 В. В этом проекте мы используем 3,3 В. для подсветки.

2. **VL** - это штифт для регулировки коэффициента контрастности. Обычно это последовательно подключает потенциометр (не более 5кОм) для его настройки.

В этом эксперименте мы используем резистор 1 кОм. Для его подключения есть два метода: высокий потенциал и низкий потенциал. Здесь мы используем метод низкого потенциала; подключите резистор, а затем GND.

3. **RS** - очень распространенный вывод на ЖК-дисплее. Это булавка для выбора команда / данные. Когда вывод находится на высоком уровне, он находится в режиме данных; когда он на низком уровне, он в командном режиме.

4. **RW** штырь является также очень общий в ЖК-дисплей. Это вывод выбора для чтения / записи. Когда вывод находится на высоком уровне, он находится в режиме чтения; если на низком уровне, это операция записи.

5. **E** Штырь также очень распространен в ЖК-дисплее. Обычно, когда сигнал в шине стабилизируется, он посылает положительный импульс, требующий операции чтения. Когда этот вывод находится на высоком уровне, шина не может иметь никаких изменений.

6. **D0-D7** это 8-битная двунаправленная параллельная шина, используемая для

передача команд и данных.

7. **BLA** анод для подсветки; BLK, катод для подсветки.

#### 5. Четыре основные операции для 1602LCD

Читать положение дел	inp ут	RS = L, R / W = H, E = H  D0-D7 = stat  нам слово	outp D0-D7 = stat  ут	
Написать командовать d	inp ут	RS = L, R / W = H,  D0-D7 = команда  код, E = высокий импульс	выход  ут	НИКТО
Читать данные	inp ут	RS = H, R / W = H, E = H	выход  ут	D0-D7 = данные
Написать данные	inp ут	RS = H, R / W = L,  D0-D7 = данные, E = высокий  пульс	выход  ут	НИКТО

#### 6. требуется аппаратное обеспечение

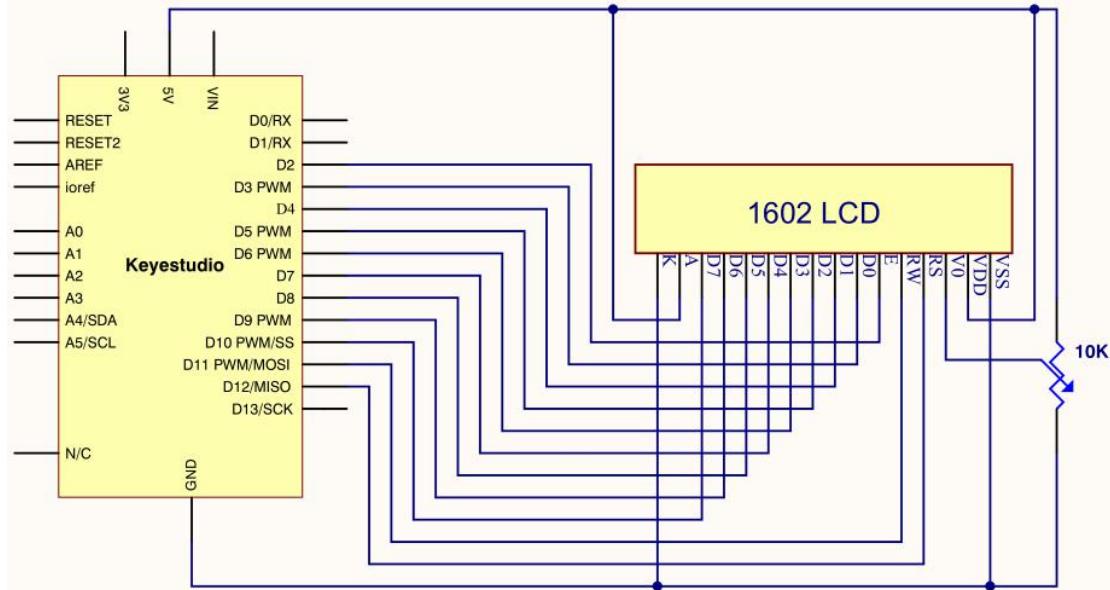
- Плата V4.0 или плата MEGA 2650 \* 1
- 1 \* 1602 ЖК-дисплей
- 1 \* потенциометр
- 1 \* макет

- 1 \* USB-кабель
- Несколько \* перемычек

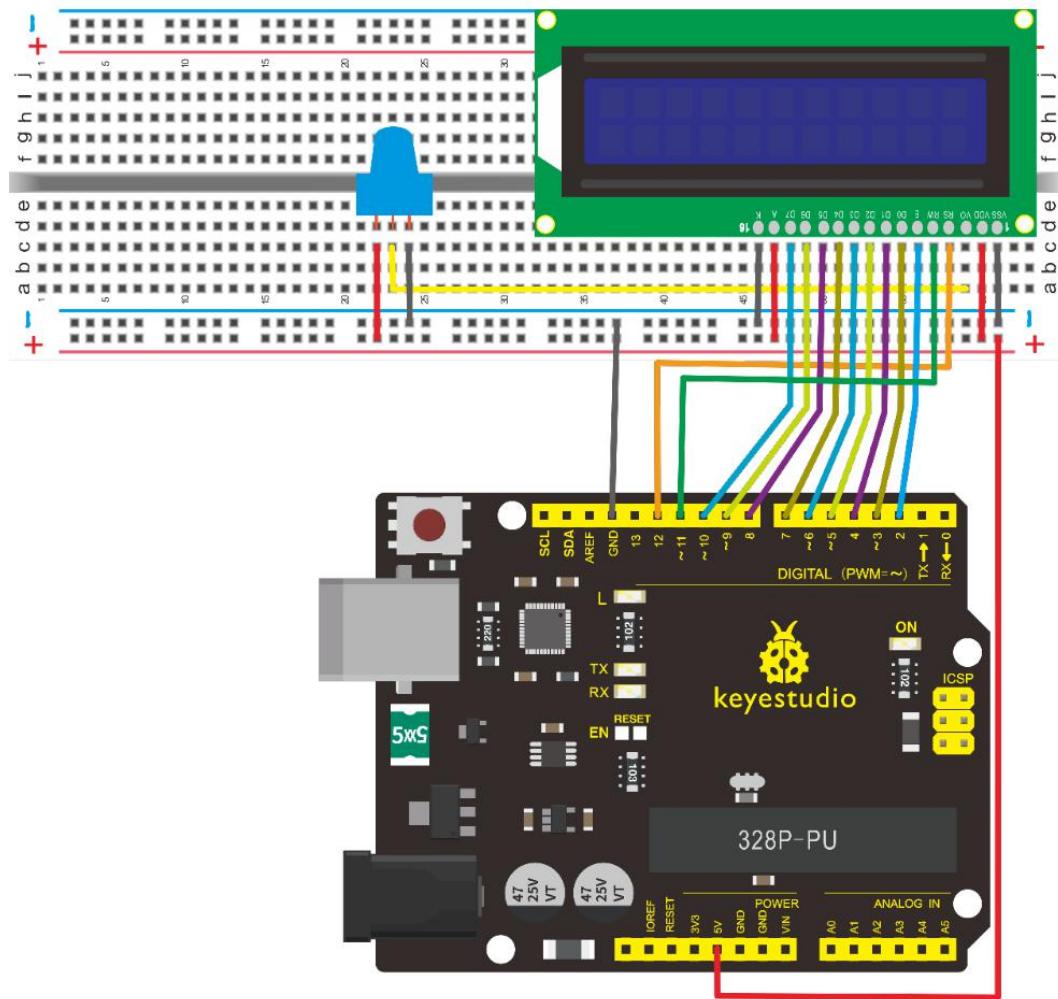
## 7. подключение

1602 может напрямую связываться с Arduino. Согласно руководству по продукту, у него есть два метода подключения, а именно 8-битное соединение и 4-битное соединение.

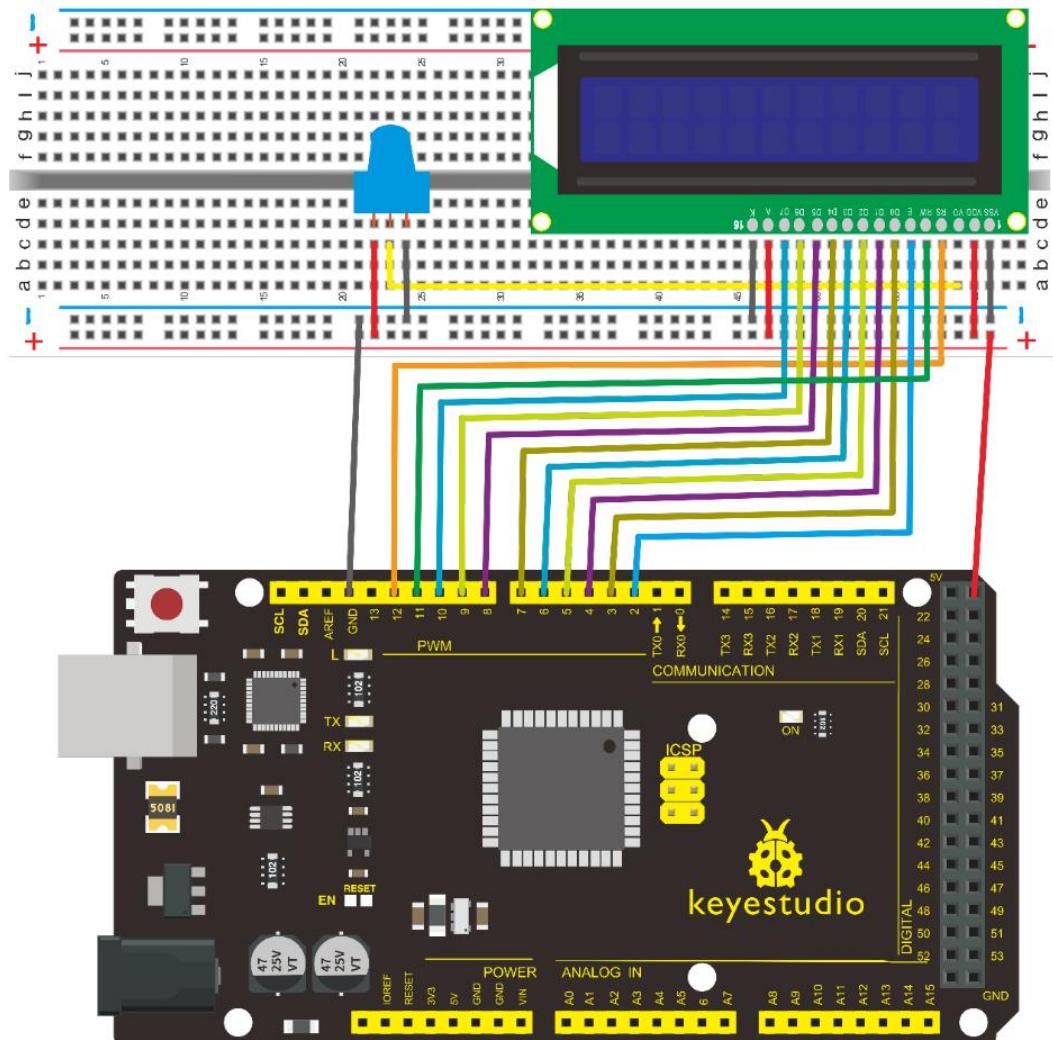
### 8-битный метод подключения



### Подключение для V4.0



Подключение для 2560 R3



## 8. образец кода А

/ \*

супер обучающий комплект keyestudio

## Project 20.1

## LCD1602

<http://www.keyestudio.com>

\* /

```
int DI = 12;  
  
int RW = 11;  
  
int DB [] = {3, 4, 5, 6, 7, 8, 9, 10}; // используем массив для выбора вывода для шины  
  
  
int Enable = 2;  
  
  
  
void LcdCommandWrite (int value) {// определяем все  
контакты  
  
int я = 0;  
  
for (i = DB [0]; i <= DI; i++) // присваиваем значение для шины {  
  
    digitalWrite (i, value & 01); // для ЖК-дисплея 1602 используется D7-D0 (не D0-D7)  
  
    для идентификации сигнала; здесь он используется для инверсии сигнала.  
  
    значение >> = 1;  
}  
  
digitalWrite (Включить, НИЗКИЙ);  
  
delayMicroseconds (1);  
  
digitalWrite (Включить, ВЫСОКИЙ);  
  
delayMicroseconds (1); // ждем 1 мс digitalWrite  
(Enable, LOW);  
  
delayMicroseconds (1); // ждем 1 мс
```

```
}
```

```
void LcdDataWrite (int value) { // инициализировать  
    все контакты  
  
    int я = 0;  
  
    digitalWrite (DI, HIGH);  
  
    digitalWrite (RW, LOW);  
  
    for (i = DB [0]; i <= DB [7]; i++) {  
  
        digitalWrite (я, значение & 01);  
  
        значение >> = 1;  
  
    }  
  
    digitalWrite (Включить, НИЗКИЙ);  
  
    delayMicroseconds (1);  
  
    digitalWrite (Включить, ВЫСОКИЙ);  
  
    delayMicroseconds (1);  
  
    digitalWrite (Включить, НИЗКИЙ);  
  
    delayMicroseconds (1); // ждем 1 мс}  
  
установка void (void) {int i  
= 0;  
for (i = Enable; i <= DI; i++) {
```

```
pinMode (я, ВЫХОД);

}

задержка (100);

// инициализируем ЖК-дисплей после короткой паузы // для

управления ЖК-дисплеем

LcdCommandWrite (0x38); // выбираем 8-битный интерфейс, 2-строчный дисплей, размер

символа 5x7

задержка (64);

LcdCommandWrite (0x38); // выбираем 8-битный интерфейс, 2-строчный дисплей, размер

символа 5x7

задержка (50);

LcdCommandWrite (0x38); // выбираем 8-битный интерфейс, 2-строчный дисплей, размер

символа 5x7

задержка (20);

LcdCommandWrite (0x06); // устанавливаем режим ввода

                    // автоинкремент, без отображения shiftt

НГ

задержка (20);

LcdCommandWrite (0x0E); // настройка отображения

                    // включаем монитор, курсор включен, нет

мерцание

задержка (20);
```

```
LcdCommandWrite (0x01); // очищаем осыпь, позиция курсора возвращается на 0

задержка (100);

LcdCommandWrite (0x80); // настройка отображения

// включаем монитор, курсор включен, нет

мерцание

задержка (20);

}

void loop (void) {

    LcdCommandWrite (0x01);           // очищаем осыпь, курсор

    позиция возвращается к 0

    задержке (10);

    LcdCommandWrite (0x80 + 3);

    задержка (10);

    // пишем приветственное сообщение

    LcdDataWrite ('W');

    LcdDataWrite ('e');

    LcdDataWrite ('л');

    LcdDataWrite ('c');

    LcdDataWrite ('o');

}
```

```
LcdDataWrite ('м');

LcdDataWrite ('е');

LcdDataWrite ("");

LcdDataWrite ('т');

LcdDataWrite ('о');

задержка (10);

LcdCommandWrite (0xc0 + 1);           // устанавливаем позицию курсора в

вторая строка, задержка второй позиции

(10);

LcdDataWrite ('к');

LcdDataWrite ('е');

LcdDataWrite ('ы');

LcdDataWrite ('е');

LcdDataWrite ('с');

LcdDataWrite ('т');

LcdDataWrite ('и');

LcdDataWrite ('д');

LcdDataWrite ('я');

LcdDataWrite ('о');

задержка (5000);

LcdCommandWrite (0x01);           // очищаем экран, курсор

возвращается к 0
```

```
задержка (10);

LcdDataWrite («Я»);

LcdDataWrite ("");

LcdDataWrite ('а');

LcdDataWrite ('м');

LcdDataWrite ("");

LcdDataWrite ('ч');

LcdDataWrite ('и');

LcdDataWrite ('н');

LcdDataWrite ('т');

LcdDataWrite ('е');

LcdDataWrite ('г');

задержка (3000);

LcdCommandWrite (0x02); // устанавливаем режим, когда новые персонажи воспроизводят старых, где нет новых, остаются прежними delay (10);
```

```
LcdCommandWrite (0x80 + 5); // установить позицию курсора в первой строке, шестой позиции

задержка (10);

LcdDataWrite ('т');

LcdDataWrite ('ч');

LcdDataWrite ('е');
```

```
LcdDataWrite ('"');

LcdDataWrite ('ш');

LcdDataWrite ('о');

LcdDataWrite ('г');

LcdDataWrite ('л');

LcdDataWrite ('д');

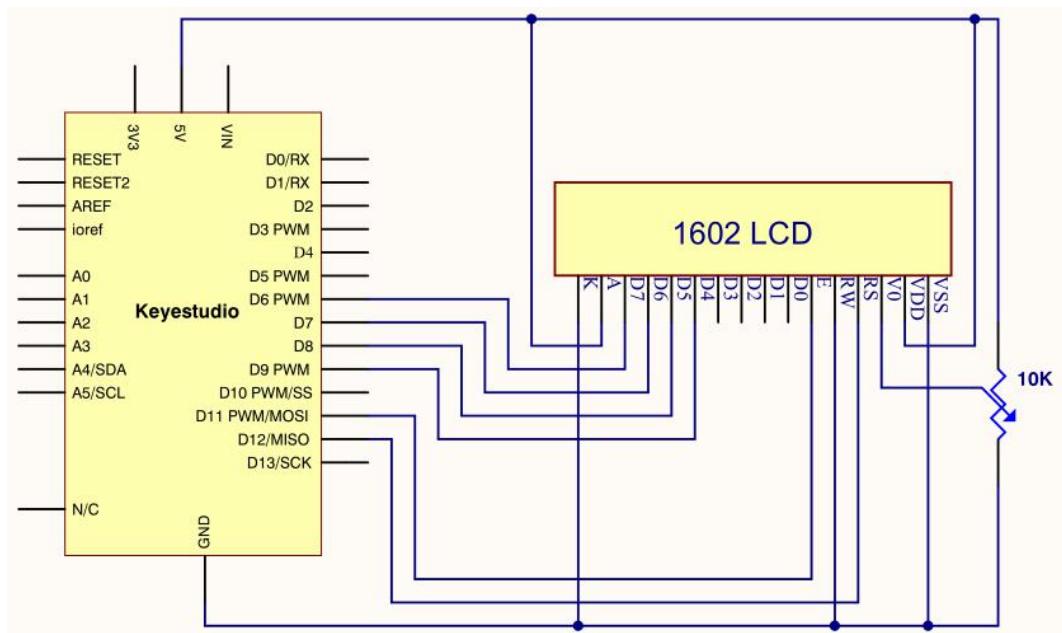
задержка (5000);

}

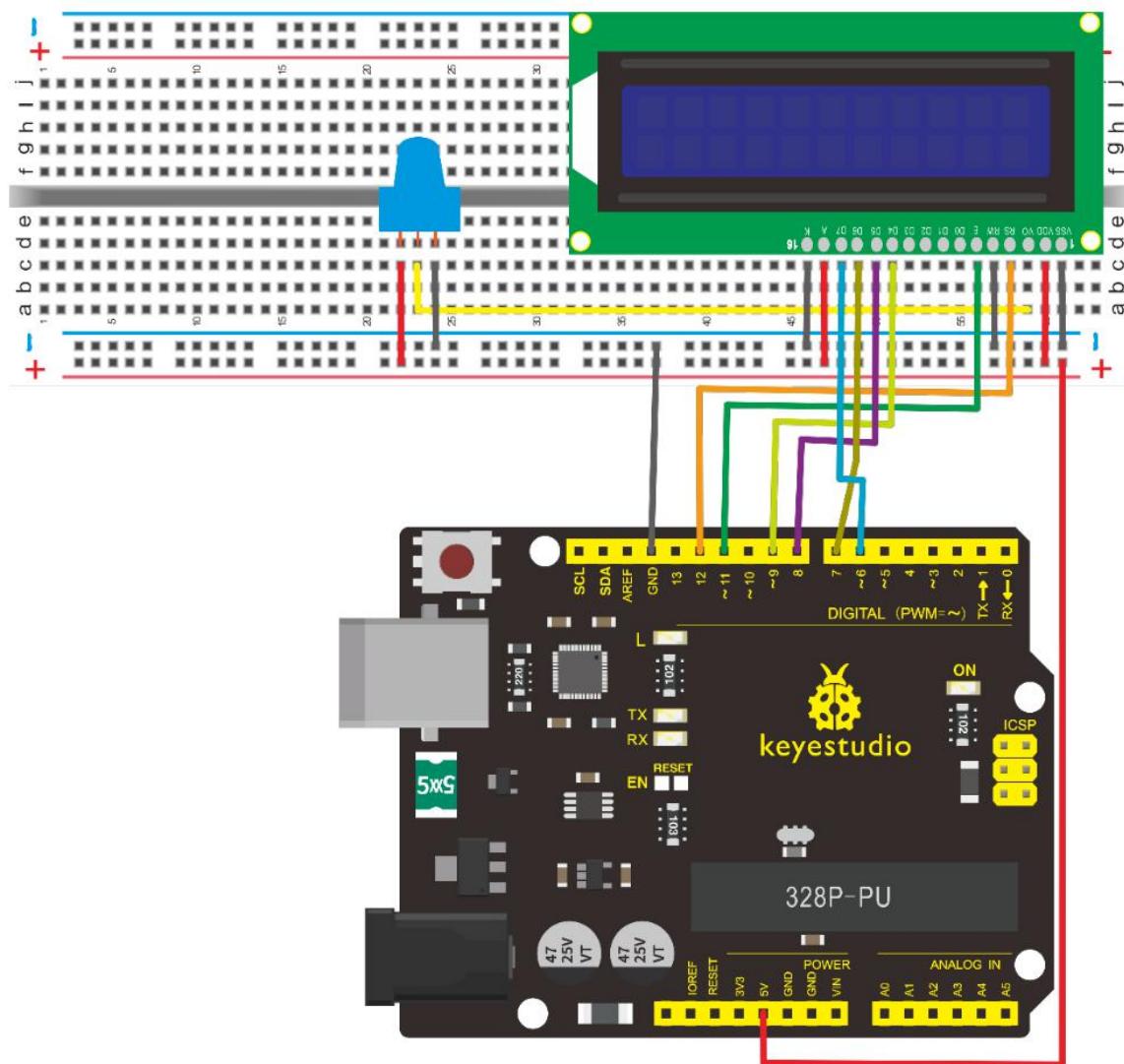
||||||||||||||||||||||||||||||||
```

#### **4- бит Метод подключения**

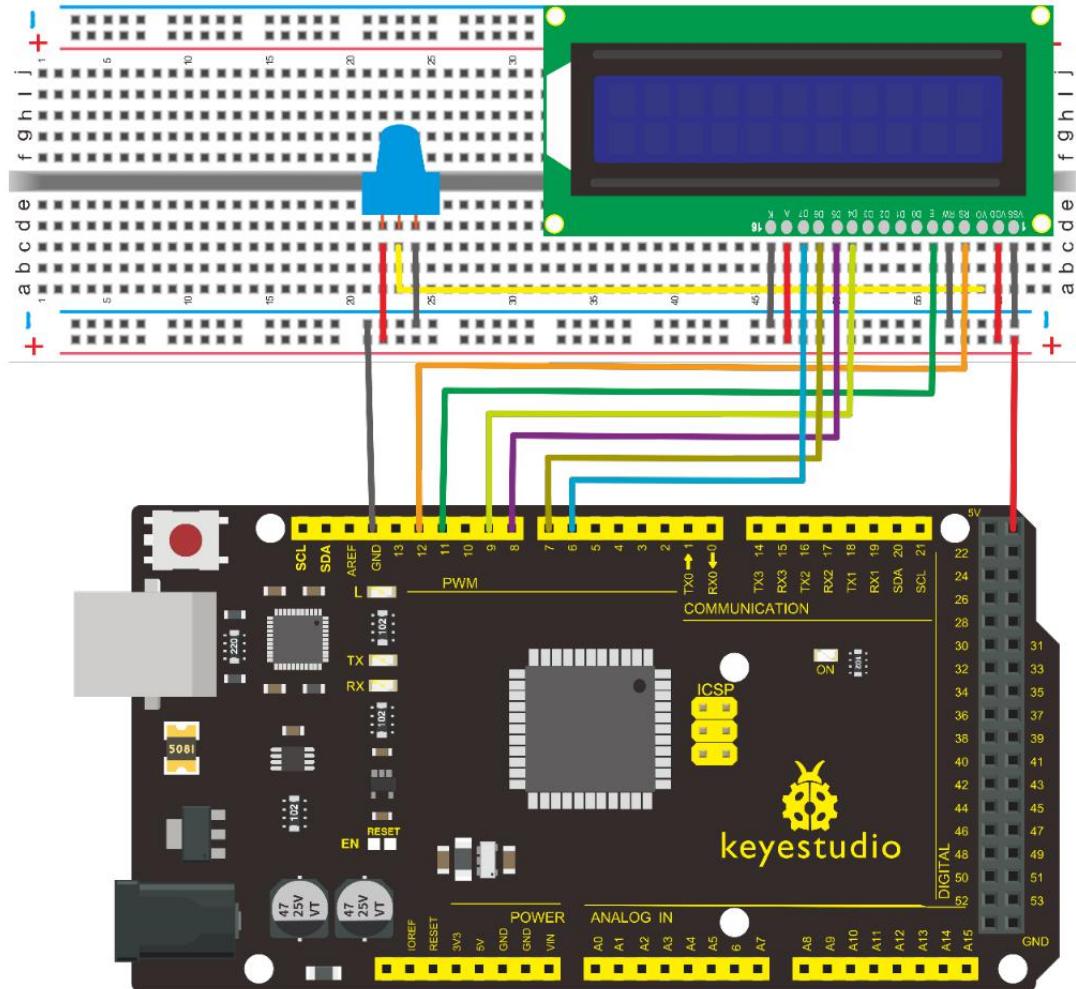
При использовании этого модуля 8-битное соединение использует все цифровые контакты Arduino, не оставляя контактов для датчиков. Что тогда? Вы можете использовать 4-битное соединение.



Подключение для V4.0



Подключение для 2560 R3



После подключения загрузите приведенный ниже код на плату контроллера и посмотрите, как

это работает.

## 9. образец кода В

```
/*
```

супер обучающий комплект keyestudio

[Project 20.2](#)

[LCD1602](#)

[http // www.keyestudio.com](http://www.keyestudio.com)

\* /

/\* Вывод RS LCD на цифровой вывод 12

\* ЖК-дисплей Включить вывод на цифровой вывод 11

\* Вывод LCD D4 на цифровой вывод 9

\* Вывод ЖК-дисплея D5 на цифровой вывод 8

\* Вывод LCD D6 на цифровой вывод 7

\* Вывод LCD D7 на цифровой вывод 6

\* LCD R / W контакт к земле

\* Вывод VSS ЖК-дисплея на землю

\* Вывод VCC ЖК-дисплея на 5 В

\* Резистор 10 кОм

\* заканчивается на +5 В и на землю

\* дворник к выводу LCD VO (вывод 3)

Этот пример кода находится в открытом доступе. [http //](http://www.arduino.cc/en/Учебник/LiquidCrystal)

[www.arduino.cc / en / Учебник / LiquidCrystal](http://www.arduino.cc/en/Учебник/LiquidCrystal)

\* /

// включаем код библиотеки

# include <LiquidCrystal.h>

// инициализируем библиотеку номерами контактов интерфейса LiquidCrystal

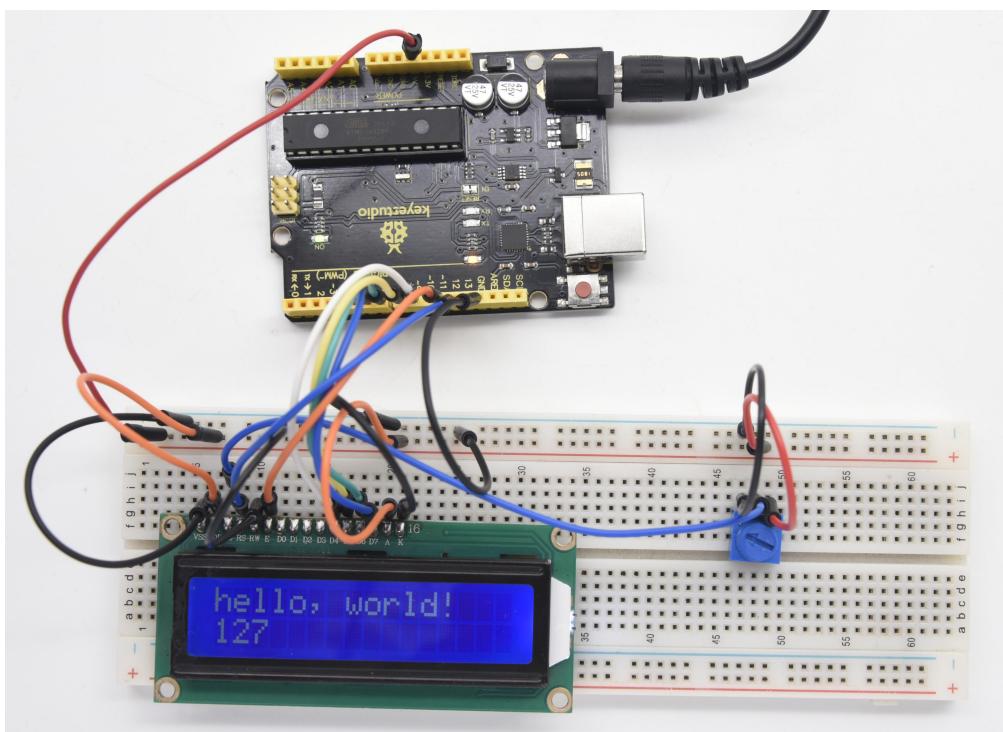
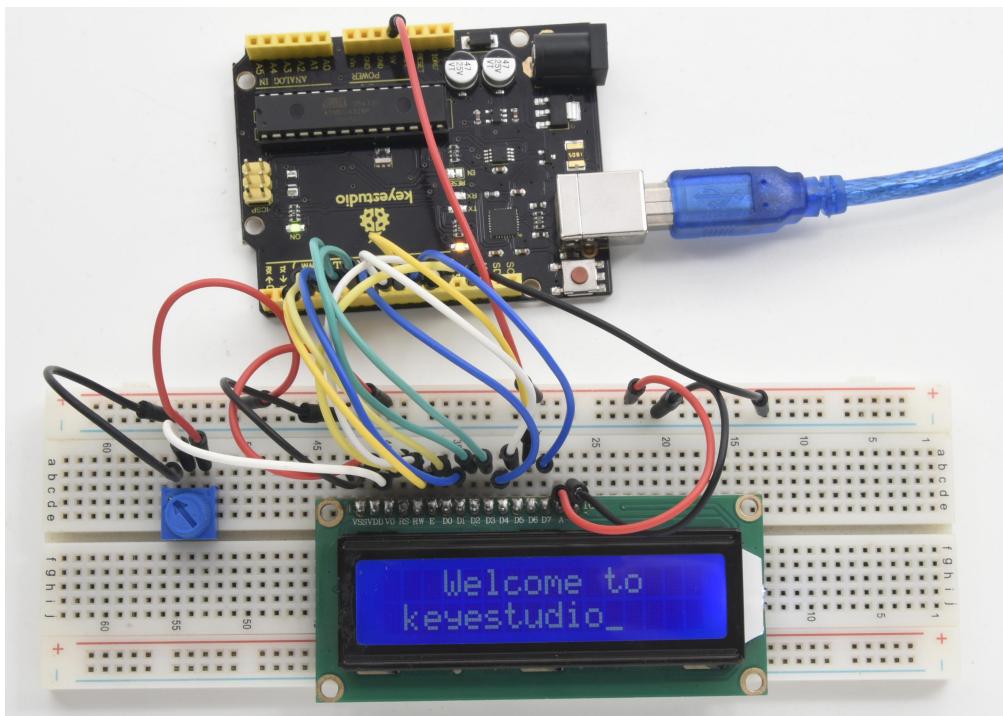
lcd (12, 11, 9, 8, 7, 6);

void setup () {

```
// устанавливаем количество столбцов и строк ЖК-дисплея lcd.begin  
(16, 2);  
  
// Выводим сообщение на ЖК-дисплей.  
  
lcd.print («привет, мир!»);  
  
}  
  
void loop () {  
  
    // установить курсор в столбец 0, строка 1  
  
    // ( обратите внимание, что строка 1 - это вторая строка, так как отсчет начинается с  
0)  
  
    lcd.setCursor (0, 1);  
  
    // выводим количество секунд с момента сброса lcd.print  
    (millis () / 1000);  
  
}
```

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

## **10. Результат испытаний**



## Проект 21: Сервоуправление



### 1. Введение

Серводвигатель - это поворотный привод с регулировкой положения. В основном он состоит из корпуса, печатной платы, двигателя без сердечника, редуктора и датчика положения. В этом уроке мы познакомим вас с сервоприводом.

### 2. принцип работы

MCU выдает сигнал на серводвигатель. Мотор имеет

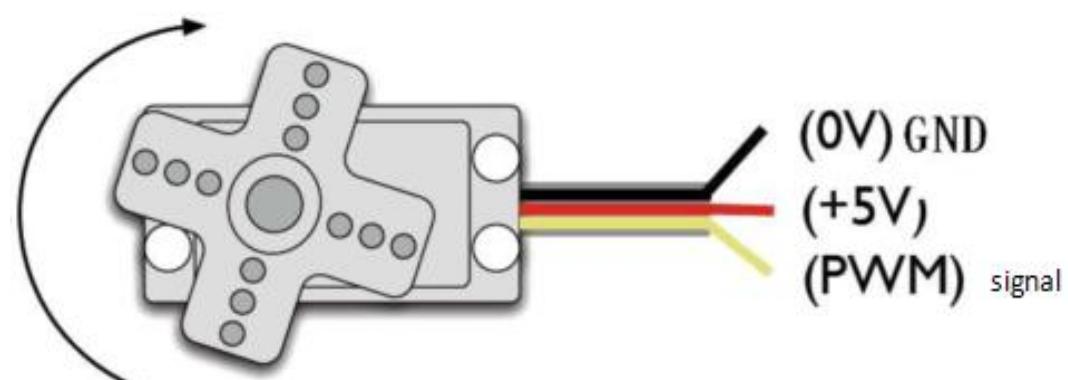
встроенный Справка цепь тот дает выходной опорный сигнал, цикл 20 мс и ширина 1,5 мс. Двигатель сравнивает полученное напряжение смещения постоянного тока с напряжением потенциометра и выводит разность напряжений. Микросхема на печатной плате соответственно определяет направление вращения и приводит в движение двигатель без сердечника. Затем шестерня передает усилие на вал. Датчик определит,

достиг заданного положения согласно сигналу обратной связи.

Серводвигатели используются в системах управления, которые требуют наличия и поддержания разных углов. Когда скорость двигателя определена, шестерня будет приводить во вращение потенциометр. Когда разница напряжений уменьшается до нуля, двигатель останавливается. Обычно диапазон угла поворота составляет от 0 до 180 градусов.

Серводвигатель имеет множество спецификаций. Но все они имеют три соединительных провода, которые различаются коричневым, красным, оранжевым (разные марки могут иметь разный цвет).

Коричневый - для GND, красный - для плюса питания, оранжевый - для сигнальной линии.

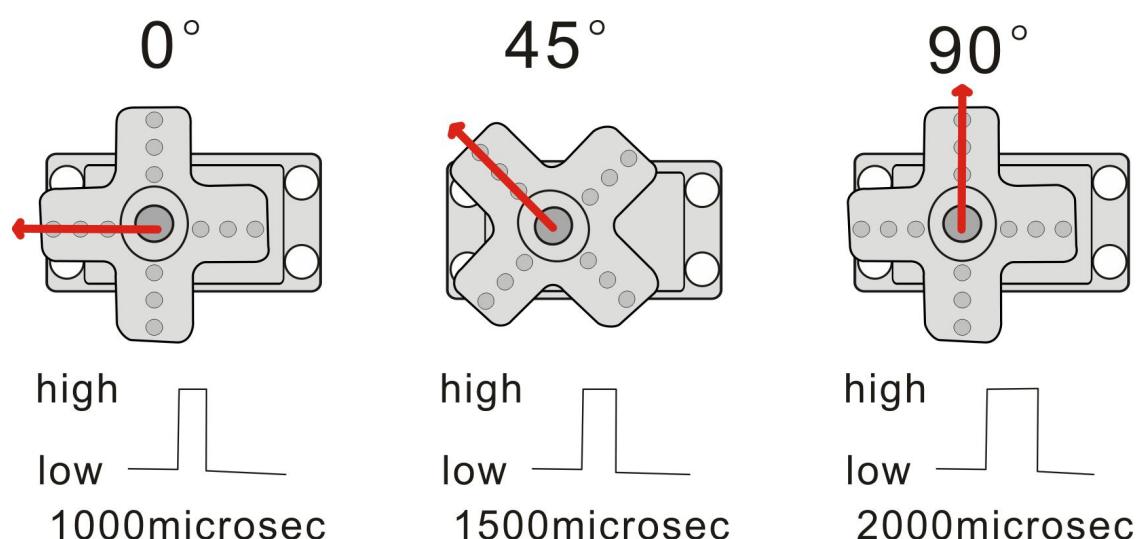


Угол поворота серводвигателя регулируется путем регулирования рабочего цикла ШИМ (широкото-импульсной модуляции).

сигнал.

Стандартный цикл сигнала ШИМ составляет 20 мс (50 Гц). Теоретически ширина распределяется от 1 до 2 мс. Ширина соответствует углу поворота от 0 ° до 90 °.

Но обратите внимание, что для двигателей разных производителей один и тот же сигнал может иметь разный угол поворота.



Обладая базовыми знаниями, давайте узнаем, как управлять серводвигателем. В этом эксперименте вам понадобится только серводвигатель и несколько перемычек.

### 3. требуется оборудование

- Плата V4.0 или плата MEGA 2650 \* 1
- Серводвигатель 9G \* 1
- Перемычка макетной платы \* 3

- USB-кабель \* 1

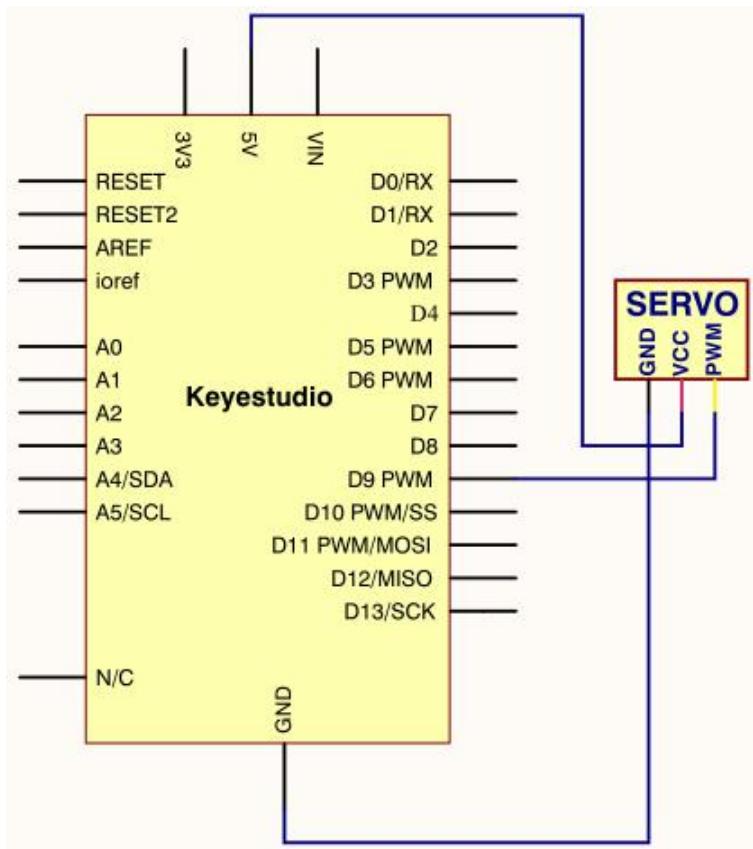
### **Подключение и пример кода**

Есть два способа управления серводвигателем с помощью Arduino. Один из них - использовать общий порт цифрового датчика Arduino для создания прямоугольной волны с разным рабочим циклом для имитации сигнала ШИМ и использования этого сигнала для управления позиционированием двигателя.

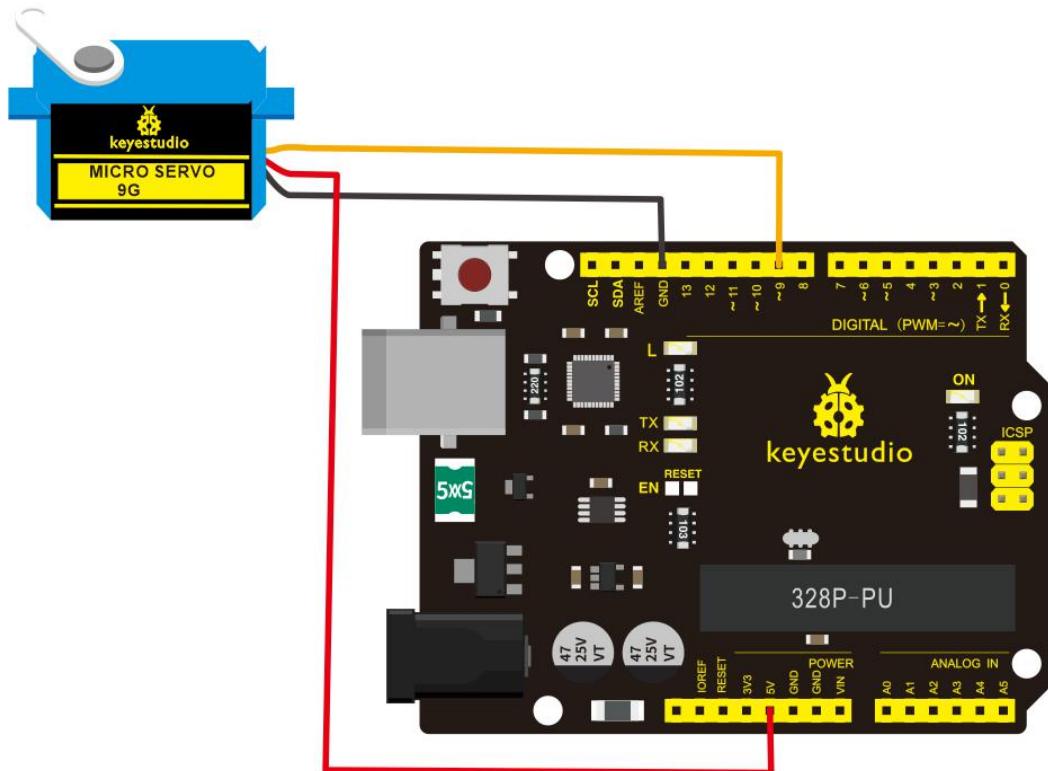
Другой способ - напрямую использовать серво-функцию Arduino для управления двигателем. Таким образом, программа будет более простой, но она может управлять только двухконтактным двигателем, поскольку из-за функции сервопривода можно использовать только цифровые выводы 9 и 10.

Емкость диска Arduino ограничена. Поэтому, если вам нужно управлять более чем одним двигателем, вам понадобится внешний источник питания.

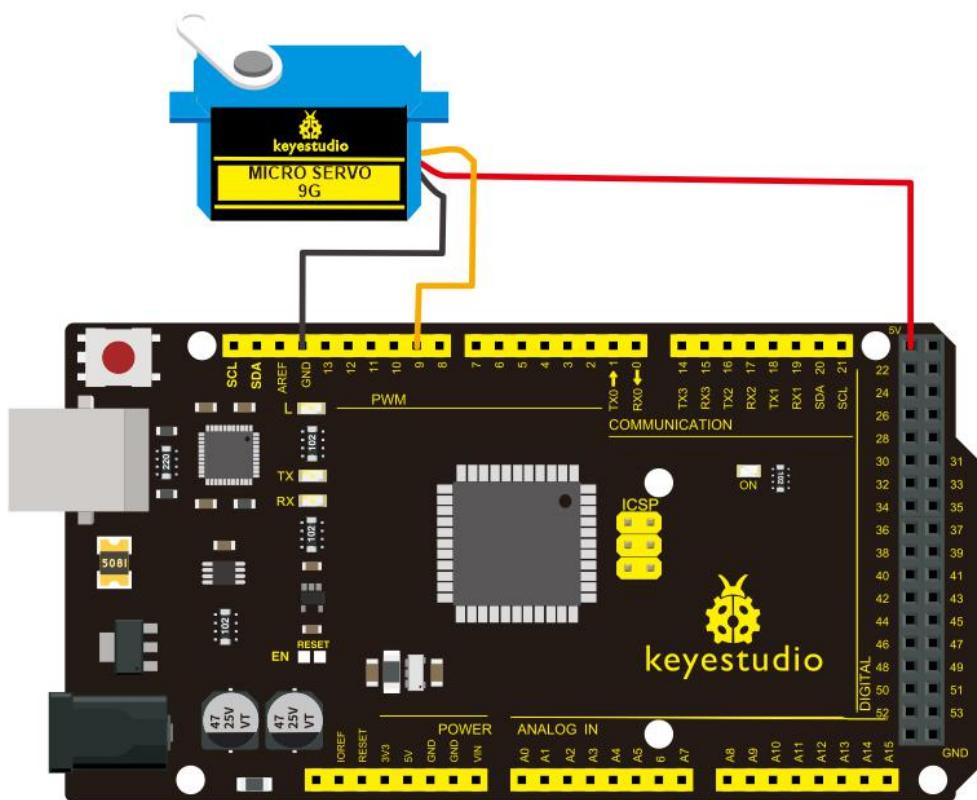
### **4. метод 1**



### Подключение для V4.0



## Подключение для 2560 R3



Подключите двигатель к цифровому выводу 9.

Составьте программу для управления двигателем, чтобы он вращался на заданный угол, и отобразите угол на экране.

## Пример кода А

```
/*  
супер обучающий комплект keyestudio
```

Project 21.1

сервопривод

[http // www.keyestudio.com](http://www.keyestudio.com)

\* /

int servopin = 9; // выбираем цифровой вывод 9 для сигнальной линии сервомотора int myangle; //

инициализируем угловую переменную

int pulsewidth; // инициализируем переменную ширины int val;

void servopulse (int servopin, int myangle) // определение функции сервоимпульса

{

    pulsewidth = (myangle \* 11) +500; // конвертируем угол в ширину импульса

    500-2480

    digitalWrite (servopin, HIGH); // устанавливаем уровень сервопривода как «высокий»

    delayMicroseconds (pulsewidth); // микросекундная задержка ширины импульса

    digitalWrite (servopin, LOW); // устанавливаем уровень серво штыря как «низкий»

    задержка (20 импульсов / 1000);

}

    установка void ()

{

        pinMode (servopin, OUTPUT); // устанавливаем вывод сервопривода как «выход»

```
Serial.begin (9600); // подключаемся к последовательному порту, устанавливаем скорость передачи  
«9600»  
  
Serial.println ("servo = o_serai_simple ready"); }  
  
  
  
void loop () // преобразовываем число от 0 до 9 в соответствующий угол 0–180 градусов,  
светодиод мигает соответствующее количество раз {  
  
  
  
val = Serial.read (); // считываем значение последовательного порта if  
(val> = '0' && val <= '9')  
  
{  
  
val = val-'0'; // преобразовываем характеристическое количество в числовую переменную  
  
  
  
val = val * (180/9); // преобразовываем число в угол Serial.print  
("перемещение сервопривода в");  
  
Serial.print (val, DEC);  
  
Serial.println ();  
  
for (int i = 0; i <= 50; i++) // даем сервоприводу время для поворота в заданное  
положение  
  
{  
  
servopulse (servopin, val); // использовать импульсную функцию}}}  
  
||||||||||||||||||||||||||||||||||||||||
```

## 5. способ 2

Давайте сначала взглянем на встроенную серво-функцию Arduino и некоторые общие утверждения.

1. **прикрепить (интерфейс)** —— выберите штифт для сервопривода, можно использовать только штырь 9 или 10.
2. **написать (угол)** —— используется для управления углом поворота сервопривода, можно установить угол от 0 до 180 градусов.
3. **читать()** —— используется для считывания угла сервопривода, считайте это функцией для чтения значения в функции write () .
4. **прилагается ( )** —— определить, передан ли параметр сервопривода на штифт сервопривода.
5. **отсоединить ( )** —— отсоедините серво и штырь, и штырь (цифровой штырь 9 или 10) можно использовать для порта ШИМ.

**Запись** письменная форма приведенных выше операторов - «имя сервопеременной.

конкретный оператор ()», например myservo. Присоедините (9).

Тем не менее, подключите сервопривод к выводу 9.

## Пример кода В

```
/*  
  супер обучающий комплект keyestudio
```

## Проект 21.2

сервопривод

<http://www.keyestudio.com>

\* /

# include <Servo.h>

Servo myservo; // определяем имя сервопеременной void setup

()

{

myservo.attach (9); // выбираем серво штифт (9 или 10)}

пустой цикл ()

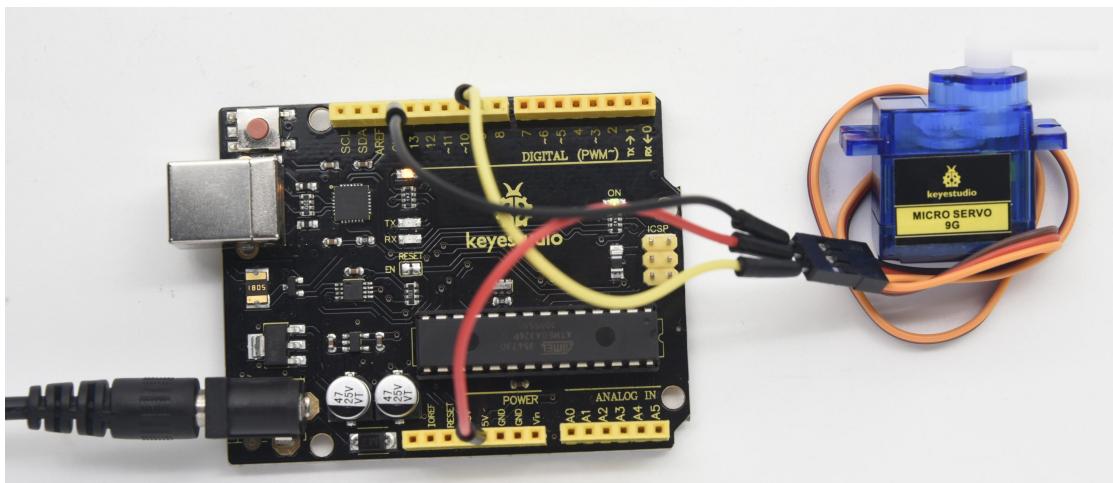
{

myservo.write (90); // устанавливаем угол поворота двигателя}

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

Выше приведены два метода управления сервоприводом. Вы можете выбрать один из них по своему вкусу или реальной потребности.

### 6. результат теста



## Проект 22: Шаговый двигатель



### 1. Введение

Шаговый двигатель - это электромеханическое устройство, которое может преобразовывать электрические импульсы в дискретные механические движения. Вал или шпиндель шагового двигателя вращается с дискретными шагами при подаче электрических командных импульсов.

к нему в правильной последовательности.

Вращение двигателей имеет несколько прямых отношений с этими приложенными входными импульсами. Последовательность подаваемых импульсов напрямую зависит от направления вращения валов двигателя. Скорость вращения валов двигателя напрямую связана с частотой входных импульсов, а длина вращения напрямую связана с количеством подаваемых входных импульсов.

Одним из наиболее значительных преимуществ шагового двигателя является его способность точно регулироваться в системе без обратной связи. Управление без обратной связи означает, что обратная связь о положении не требуется. Этот тип управления устраняет необходимость в дорогостоящих устройствах считывания и обратной связи, таких как оптические энкодеры. Ваша позиция известна, просто отслеживая входные пошаговые импульсы.

## **2. Требуется оборудование**

- Плата V4.0 или плата MEGA 2650 \* 1
- Шаговый двигатель \* 1
- Драйвер шагового двигателя \* 1
- Перемычка \* 1
- USB-кабель \* 1

### **3. особенности**

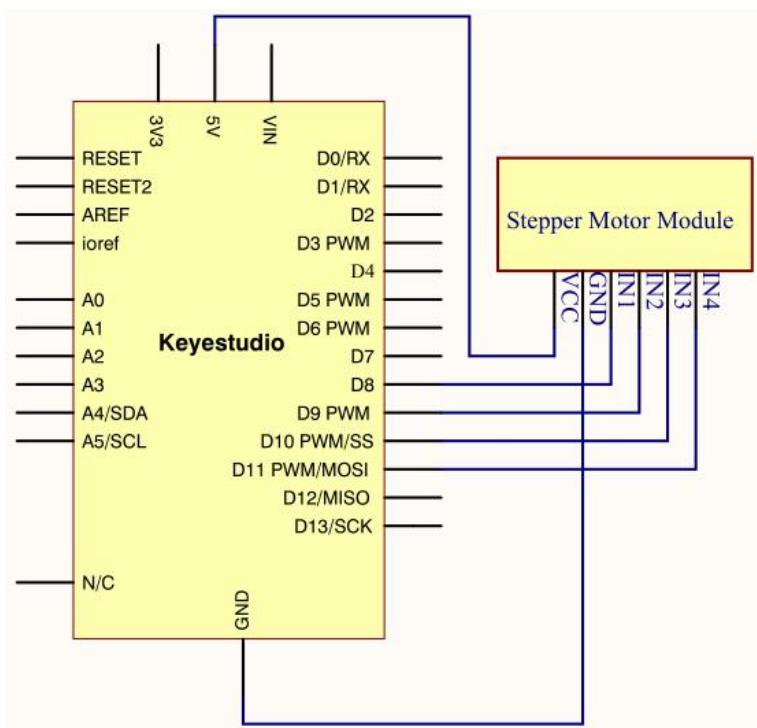
- Угол поворота двигателя пропорционален входному импульсу.
- Двигатель имеет полный крутящий момент в состоянии покоя (если обмотки находятся под напряжением)
- Точное позиционирование и повторяемость движения, так как хорошие шаговые двигатели имеют точность - 5% шага, и эта ошибка не накапливается от одного шага к другому. Отличная реакция на пуск / остановку / движение задним ходом.
- Очень надежен, так как в моторе нет контактных щеток. Следовательно, срок службы двигателя просто зависит от срока службы подшипника.
- Реакция двигателей на цифровые входные импульсы обеспечивает управление без обратной связи, что упрощает управление двигателем и снижает его стоимость.
- Можно добиться синхронного вращения на очень низкой скорости с нагрузкой, непосредственно связанной с валом.
- Может быть реализован широкий диапазон скоростей вращения, поскольку скорость пропорциональна частоте входных импульсов.

### **4. параметры шагового двигателя 28BYJ-48**

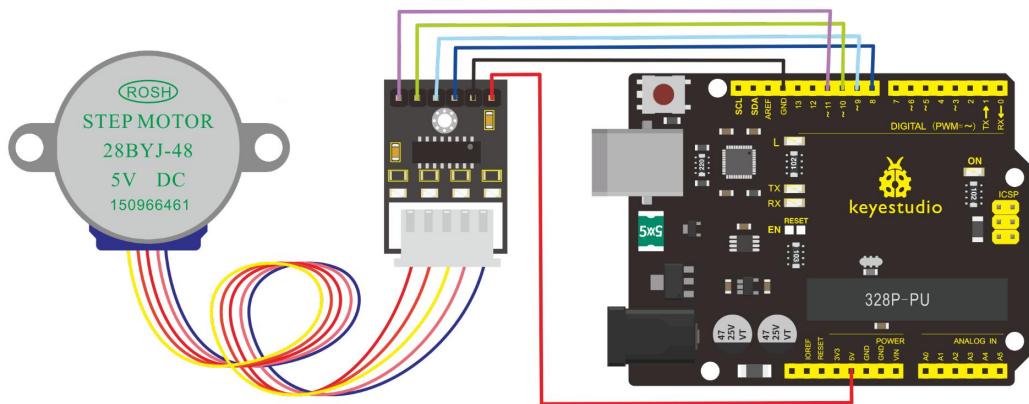
- Модель 28BYJ-48

- Номинальное напряжение 5 В постоянного тока
- Номер фазы 4
- Коэффициент изменения скорости
- 1/64 Угол шага 5,625 ° / 64 Частота
- 100 Гц
- Сопротивление постоянному току 50 Ом ± 7% (25 °C)
- Частота на холостом ходу > 600 Гц Частота на выходе на холостом ходу > 1000 Гц Крутящий момент на холостом ходу > 34,3 мН.м (120 Гц)
- Момент самопозиционирования > 34,3 мН.м
- Момент трения 600-1200 гс.см Момент затяжки 300 гс.см
- Изолированное сопротивление > 10 МОм (500 В)
- Изолированная электрическая мощность 600 В переменного тока / 1 мА / 1 с Класс изоляции А
- Повышение температуры <40 К (120 Гц) Шум <35 дБ (120 Гц, без нагрузки, 10 см)

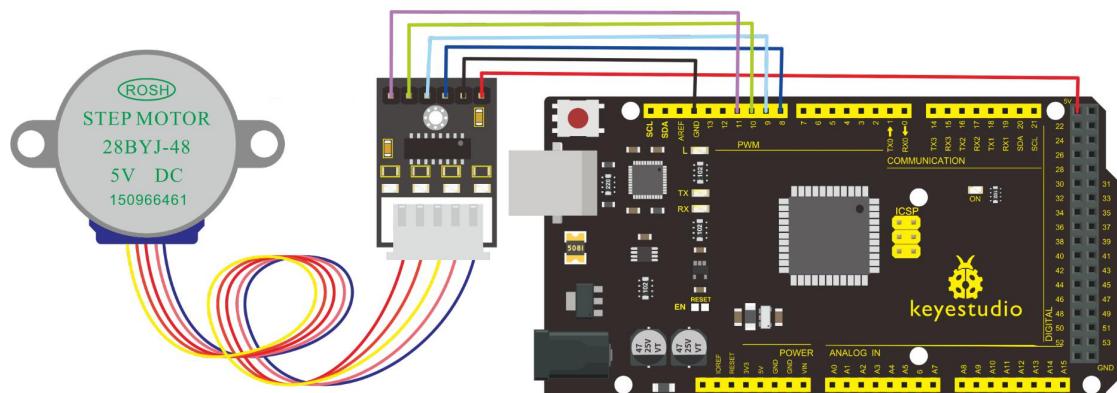
## **5. схема подключения**



Подключение для V4.0



Подключение для 2560 R3



## 6. образец кода

```
/*  
супер обучающий комплект keyestudio
```

Project 22

Шаговый двигатель

<http://www.keyestudio.com>

```
*/
```

```
# include <Stepper.h>
```

```
# определить ШАГИ 100
```

```
Шаговый шаговый (ШАГИ, 8, 9, 10, 11); int
```

```
предыдущий = 0;
```

```
установка void ()
```

```
{
```

```
stepper.setSpeed (90);
```

```
}
```

```
пустой цикл ()
```

```
{
```

```
int val = analogRead (0); stepper.step
```

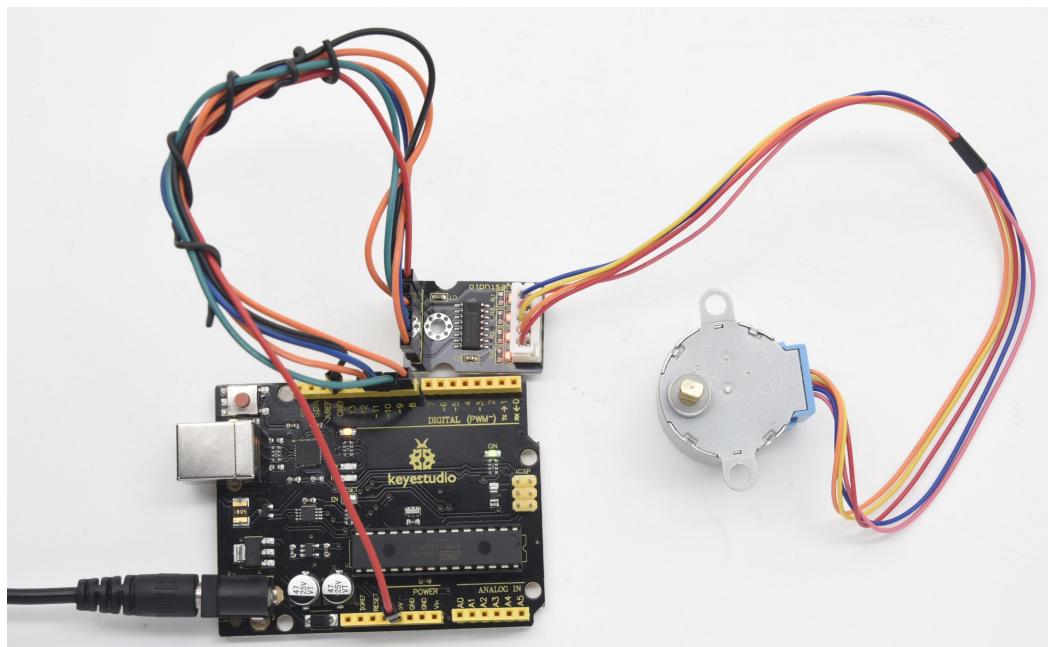
```
(val - предыдущий);
```

```
предыдущий = val;
```

```
}
```

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

## 7. результат теста



## Проект 23: Датчик движения PIR



### 1. Введение

Пироэлектрический инфракрасный датчик движения может обнаруживать инфракрасные сигналы

от движущегося человека или движущегося животного и выходных сигналов

переключения.

Его можно применять в различных случаях для обнаружения движения

человеческого тела.

Обычные пироэлектрические инфракрасные датчики требуют наличия пироэлектрического

инфракрасного детектора на теле, профессионального чипа, сложной периферийной схемы,

поэтому он больше по размеру со сложной схемой и с меньшей надежностью.

Теперь мы запускаем этот новый пироэлектрический инфракрасный датчик движения,

специально разработанный для Arduino.

Он использует встроенный цифровой пироэлектрический инфракрасный датчик тела и имеет

меньший размер, более высокую надежность, более низкое энергопотребление, а также более

простую периферийную схему.

## **2. Требуется оборудование**

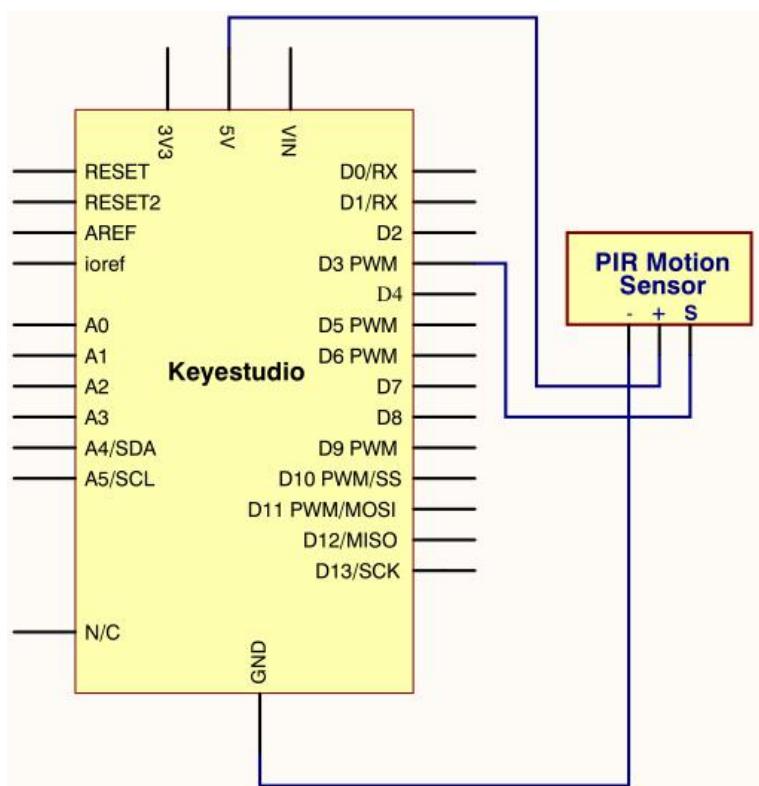
- Плата V4.0 или плата MEGA 2650 \* 1
- Датчик движения PIR \* 1
- Перемычка \* 1
- USB-кабель \* 1

## **3. Спецификация**

- Входное напряжение 3,3 ~ 5 В (максимум 6 В)
- Рабочий ток 15uA

- Рабочая температура -20 ~ 85 °C Выходное
- напряжение: высокое 3 В, низкое 0 В
- Время задержки вывода (высокий уровень) Около 2,3 - 3 секунды Угол
- обнаружения 100 °
- Расстояние обнаружения 7 метров
- Светодиодный индикатор выхода (когда выход ВЫСОКИЙ, он горит) Ограничение по
- контакту Ток 100 мА

#### 4. схема подключения



Подключение для V4.0