

Super Learning Kit for Arduino

Content

1. Introduction.....	13
2. Component List.....	14
3. Install Arduino IDE and Driver.....	21
(1) Installing Arduino IDE.....	21
(2) keyestudio V4.0 Development Board.....	23
(3) Keyestudio MEGA 2560 Board.....	26
(4) Installing V4.0 board Driver.....	29
(5) Arduino IDE Setting.....	35
(6) Start First Program.....	39
4. How to Add a Library?.....	43
5. Project Details.....	47

Project 1: Hello World.....	47
1.Introduction.....	47
2.Hardware Required.....	47
3.Sample Code.....	48
4.Test Result.....	49
Project 2: LED Blinking.....	50
1.Introduction.....	50
2. Hardware Required.....	50
3. Little Knowledge.....	51
4.Circuit Connection.....	54
5.Sample Code.....	57
6.Test Result.....	58
Project 3: PWM.....	59
1.Introduction.....	59
2.Working Principle.....	59
3.Hardware Required.....	61
4.Circuit Connection.....	61
5.Sample Code.....	63
6.Test Result.....	64
Project 4: Traffic Light.....	65
1.Introduction.....	65
2.Hardware Required.....	66

3.Circuit Connection.....	66
4..Sample Code.....	68
5.Test Result.....	70
Project 5: LED Chasing Effect.....	71
1.Introduction.....	71
2.Hardware Required.....	72
3.Circuit Connection.....	72
4.Sample Code.....	75
LED Chasing Effect.....	75
5.Test Result.....	77
Project 6: Button-Controlled LED.....	78
1.Introduction.....	78
2.Hardware Required.....	78
3. Little Knowledge.....	79
4. Circuit Connection.....	81
5.Sample Code.....	83
6.Test Result.....	84
Project 7: Active Buzzer.....	86
1.Introduction.....	86
2.Hardware Required.....	86
3. Little Knowledge.....	87
4.Circuit Connection.....	87

5.Sample Code.....	89
6.Test Result.....	90
Project 8: Passive Buzzer.....	91
1.Introduction.....	91
2.Hardware Required.....	92
3. Little knowledge.....	92
4. Circuit Connection.....	93
5.Sample Code.....	94
6.Test Result.....	96
Project 9: RGB LED.....	97
1.Introduction.....	97
2.Hardware Required.....	97
3. Little Knowledge.....	97
4.Circuit Connection.....	98
5.Sample Code.....	100
6.Test Result.....	102
Project 10: Photo Resistor.....	103
1.Introduction.....	103
2.Hardware Required.....	103
3. Little Knowledge.....	104
4.Circuit Connection.....	105
5.Sample Code.....	107

6.Test Result.....	108
Project 11: Flame Sensor.....	109
1.Introduction.....	109
2.Hardware Required.....	110
3. Little Knowledge.....	110
4.Experiment Principle.....	111
5.Circuit Connection.....	112
6.Sample Code.....	115
7.Test Result.....	116
Project 12: LM35 Temperature Sensor.....	117
1.Introduction.....	117
2. Working Principle.....	117
3.Hardware Required.....	118
4.Circuit Connection	119
5.Sample Code.....	121
6.Test Result.....	122
Project 13: Tilt Switch.....	124
1.Introduction.....	124
2.Working Principle.....	124
3.Hardware Required.....	125
4.Circuit Connection.....	125
5.Sample Code.....	127

6.Test Result.....	128
Project 14: IR Remote Control.....	129
1.Introduction.....	129
2.Working Principle.....	130
3.Pin and Wiring for Infrared Receiver.....	130
4.Hardware Required.....	131
5.Circuit Connection.....	132
6.Experimental Principle.....	135
7.Features.....	135
8.Protocol is as below.....	135
9.Sample Code.....	138
10.Test Result.....	144
Project 15: Analog Value Reading.....	145
1.Introduction.....	146
2.Hardware Required.....	146
3.Potentiometer Features.....	146
4.Circuit Connection.....	147
5.Sample Code.....	149
6.Test Result.....	151
Project 16: 74HC595 chip.....	152
1.Introduction.....	152
2.Hardware Required.....	153

3. Pins Description :	154
4.Circuit Connection.....	156
5.Sample Code.....	158
6.Test Result.....	160
Project 17: 1-digit LED Segment Display.....	161
1.Introduction.....	161
2.Hardware Required.....	161
3.The display principle of the 1-digit LED Segment Display.....	162
4.Circuit Connection.....	163
5.Sample Code.....	166
6.Test Result.....	173
Project 18: 4-digit LED Segment Display.....	174
1.Introduction.....	174
2.Hardware Required.....	175
3.The display principle of the 4-digit display.....	176
4.Circuit Connection.....	177
5.Sample Code.....	179
6.Test Result.....	190
Project 19: 8*8 LED Matrix.....	191
1.Introduction.....	192
2.Hardware Required.....	192
3. The display principle of the 8*8 dot-matrix.....	192

4.Circuit Connection.....	194
5.Sample Code for displaying “0”.....	196
6.Test Result.....	199
Project 20: 1602 LCD.....	200
1.Introduction.....	200
2.1602LCD Main Parameters.....	200
3.Pin description of 1602 LCD.....	201
4.Interface Description.....	201
5.Four Basic Operations for 1602LCD.....	203
6.Hardware Required.....	203
7.Connection.....	204
8.Sample Code A.....	206
9.Sample Code B.....	216
10. Test Result.....	218
Project 21: Servo Control.....	220
1.Introduction.....	220
2.Working Principle.....	220
3.Hardware Required.....	222
4.Method 1.....	223
5.Method 2.....	228
6.Test Result.....	229
Project 22: Stepper Motor.....	230

1.Introduction.....	230
2.Hardware Required.....	231
3.Features.....	232
4.Parameters of Stepper Motor 28BYJ-48.....	232
5.Circuit Connection.....	233
6.Sample Code.....	235
7.Test Result.....	236
Project 23: PIR Motion Sensor.....	236
1.Introduction.....	236
2.Hardware Required.....	237
3.Specification.....	237
4.Circuit Connection.....	238
5.Sample Code.....	239
6.Test Result.....	241
Project 24: Analog Gas Sensor.....	242
1.Introduction.....	242
2.Hardware Required.....	243
3.Specification.....	243
4.Circuit Connection.....	243
5.Sample Code.....	245
6.Test Result.....	246
Project 25: ADXL345 Three Axis Acceleration	246

1.Introduction.....	246
2.Hardware Required.....	247
3.Specification.....	247
4.Circuit Connection.....	248
5.Sample Code.....	249
6.Test Result.....	254
Project 26: HC-SR04 Ultrasonic Sensor.....	257
1.Introduction The HC-SR04 Ultrasonic Sensor is a very affordable proximity/distance sensor that is mainly used for object avoidance in various robotics projects.....	257
2.Hardware Required.....	257
3.Specification.....	258
4.Circuit Connection.....	258
5.Sample Code.....	260
6.Test Result.....	262
Project 27: Joystick Module.....	264
1.Introduction.....	264
2.Hardware Required.....	264
3.Specification.....	264
4.Circuit Connection.....	265
5.Sample Code.....	266
6.Test Result.....	267

Project 28: 5V Relay Module	269
1.Introduction.....	269
2.Hardware Required.....	269
3.Specification.....	270
4.Circuit Connection.....	270
5.Sample Code.....	271
6.Test Result.....	272
Project 29: DS3231 Clock Module.....	273
1.Introduction.....	273
2.Hardware Required.....	274
3.Specification.....	274
4.Circuit Connection.....	275
5.Sample Code.....	277
6.Test Result.....	279
Project 30: DHT11 Temperature and Humidity Sensor	281
1.Introduction.....	281
2.Hardware Required.....	282
3.Specification.....	283
4.Circuit Connection.....	283
5.Sample Code.....	284
6.Test Result.....	287
Project 31: Soil Humidity Sensor.....	288

1.Introduction.....	288
2.Hardware Required.....	289
3.Specification.....	290
4.Circuit Connection.....	290
5.Sample Code.....	292
6.Test Result.....	293
Project 32: RC522 RFID Module.....	293
1.Introduction.....	293
2.Hardware Required.....	294
3.Specification.....	294
4.Circuit Connection.....	295
5.Sample Code.....	296
6.Test Result.....	326
6. Resources.....	327



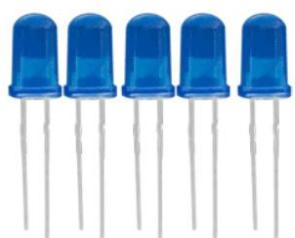
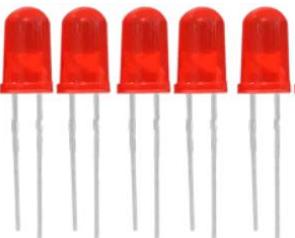
1. Introduction

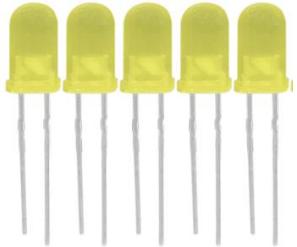
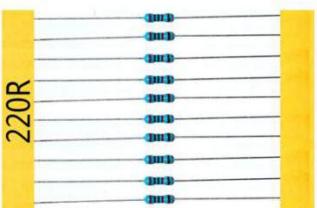
keyestudio super learning kit is suitable for Arduino

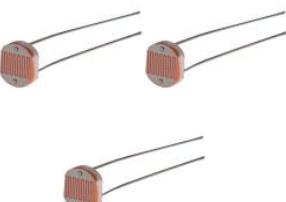
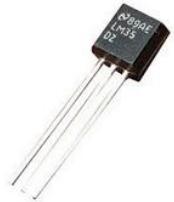
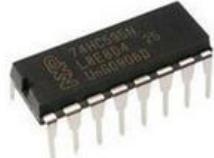
enthusiasts. This kit includes 32 projects with detailed tutorials, starting from the basics to more complex projects. Different from other kits, it adds some functional modules, such as RFID, temperature and humidity modules. There is connection diagram and code for each project, making it easy for you to learn.

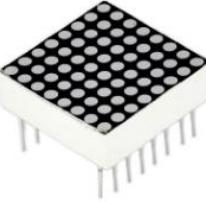
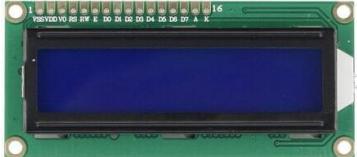
2.Component List

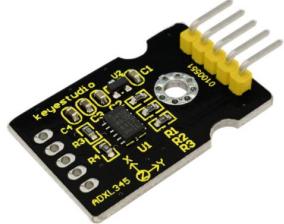
NOTE: KS0077 Kit doesn't Include Main board; KS0078 Kit Includes V4.0 board; KS0079 KIT Includes MEGA 2560 board.

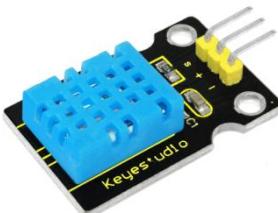
No.	Product Name	QT	Picture
1	LED - Blue	5	
2	LED - Red	5	

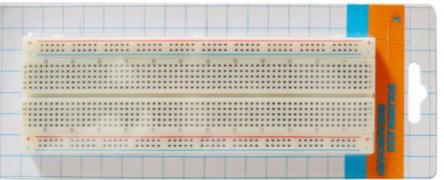
3	LED - Yellow	5	
4	LED - RGB	1	
5	220Ω Resistor	8	
6	10KΩ Resistor	5	
7	1KΩ resistor	5	
8	10KΩ Potentiometer	1	
9	Buzzer (Active)	1	
10	Buzzer (Passive)	1	

11	Large Button Switch	4	
12	Ball Tilt Sensor	2	
13	Photo Resistor	3	
14	Flame Sensor	1	
15	LM35 Temp Sensor	1	
16	IC 74HC595N 16-pin DIP	1	
17	7-seg LED Segment Display	1	
18	7-seg LED Segment Display	1	

19	8*8 LED Matrix	1	
20	2x16 LCD display	1	
21	IR Receiver	1	
22	IR Remote Control	1	
23	Servo Motor	1	
24	Stepper Driver	1	
25	Stepper Motor	1	

26	Joystick Module	1	
27	Relay Module	1	
28	PIR Motion Sensor	1	
29	Analog Gas Sensor	1	
30	ADXL345 Three Axis Acceleration Modul e	1	
31	HC-SR04 Ultrasonic Sensor	1	

32	DS3231 Clock Module	1	
33	DHT11 Temperature and Humidity Sensor	1	
34	Soil Humidity Sensor	1	
35	RC522 RFID Module	1	
36	RFID Card	1	
37	Access Key	1	
38	Pin Headers	40	

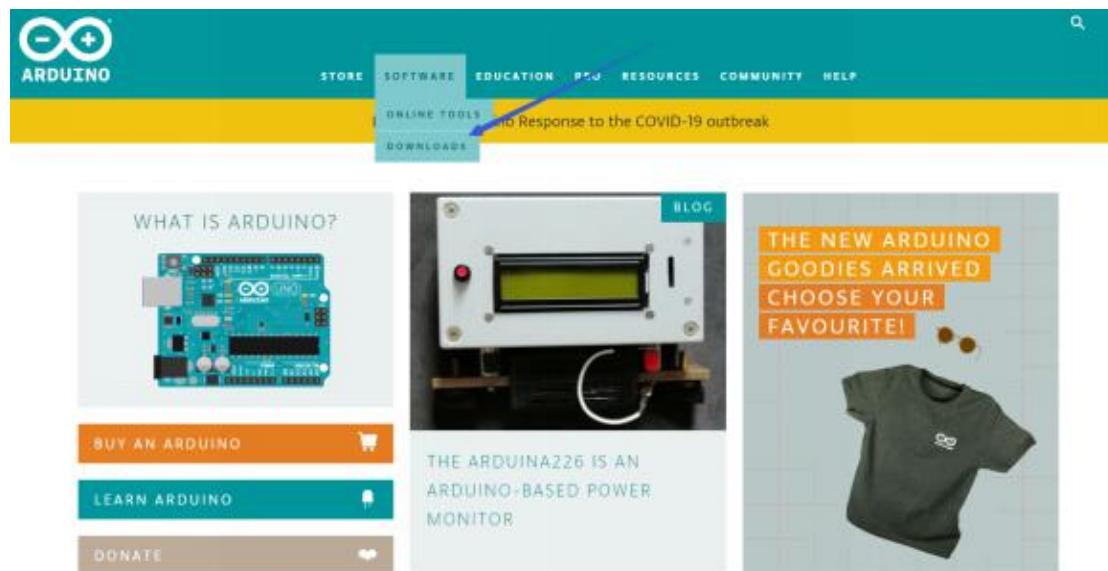
39	830-hole Breadboard	1	 A photograph of an 830-hole breadboard. It is a rectangular board with a grid of gold-colored pins. The top and bottom edges have rows of pins, while the sides have columns. A blue and orange plastic strip is attached to the right edge.
40	Dupont Wire	10	 A photograph of a bundle of Dupont wires. The wires are color-coded and stripped at both ends, with black plastic connectors attached to each end of the wires.
41	Jumper Wire	30	 A photograph of a coiled bundle of jumper wires. The wires are color-coded and stripped at both ends, with black plastic connectors attached to each end of the wires.
42	6-Slot AA Battery Holder	1	 A photograph of a black battery holder designed for six AA batteries. It has a central connection point and individual wires leading out for each battery slot.
43	USB Cable	1	 A photograph of a blue USB cable. It has a standard USB-A connector on one end and a micro-USB connector on the other.

3. Install Arduino IDE and Driver

(1) Installing Arduino IDE

When we get control board, we need to download Arduino IDE and driver firstly.

You could download Arduino IDE from the official website <https://www.arduino.cc/>, click the SOFTWARE on the browse bar, click “DOWNLOADS” to enter download page, as shown below



There are various versions Of IDE for Arduino, just download a version that compatible with your system, here we will show you how to download and install the windows version Arduino IDE.



There are two versions of IDE for WINDOWS system, you can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a portable installation.

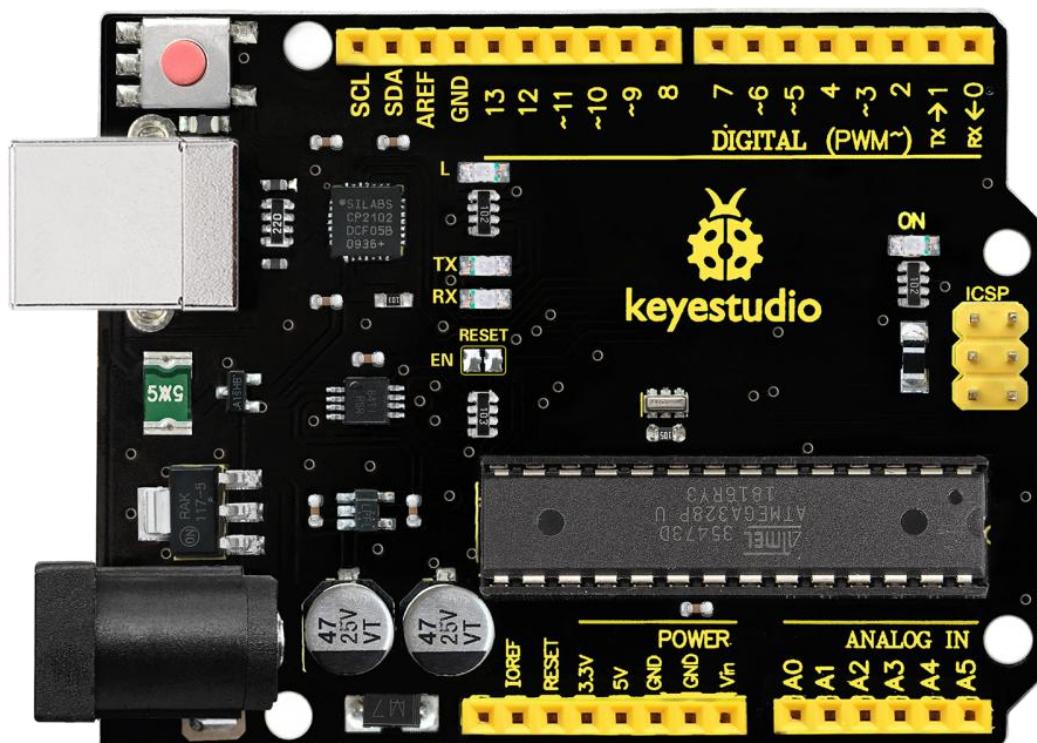
Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.



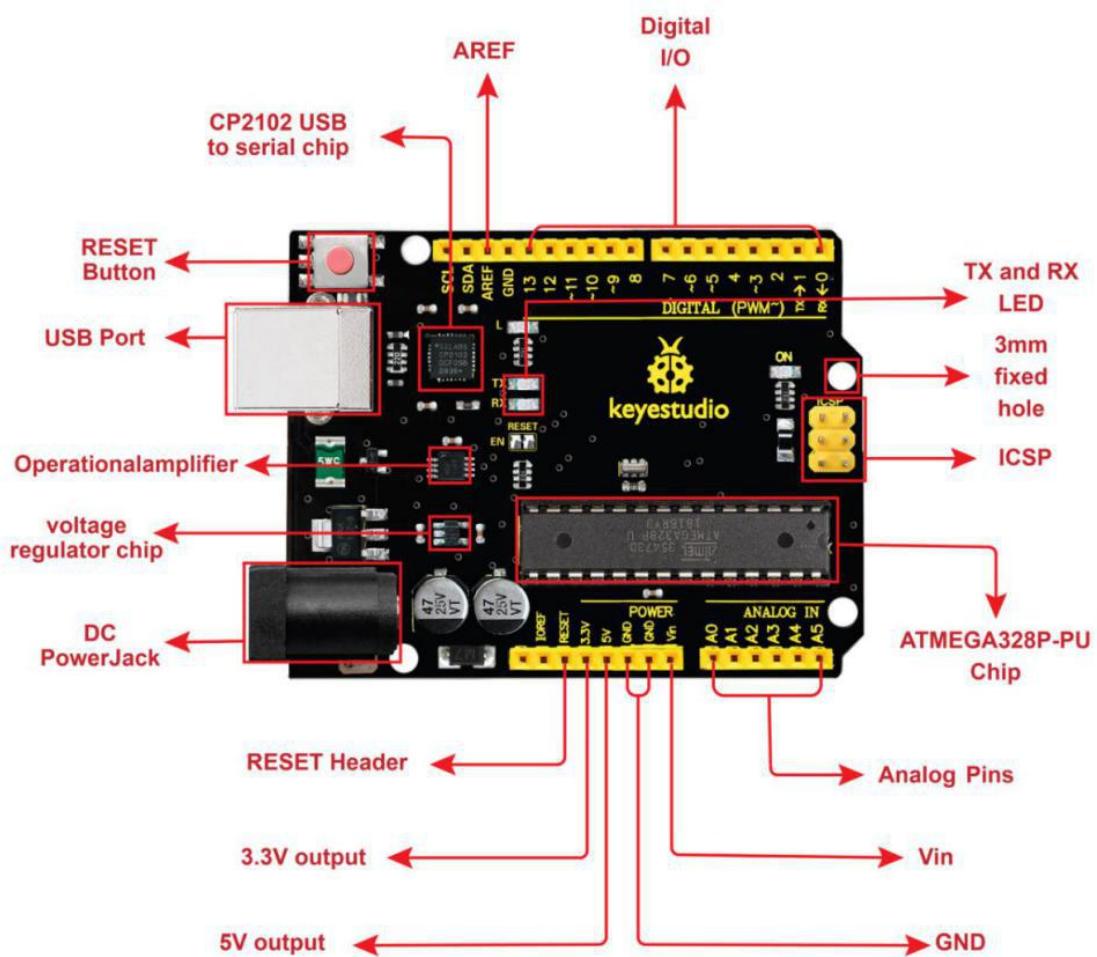
You just need to click JUST DOWNLOAD.

(2) keyestudio V4.0 Development Board

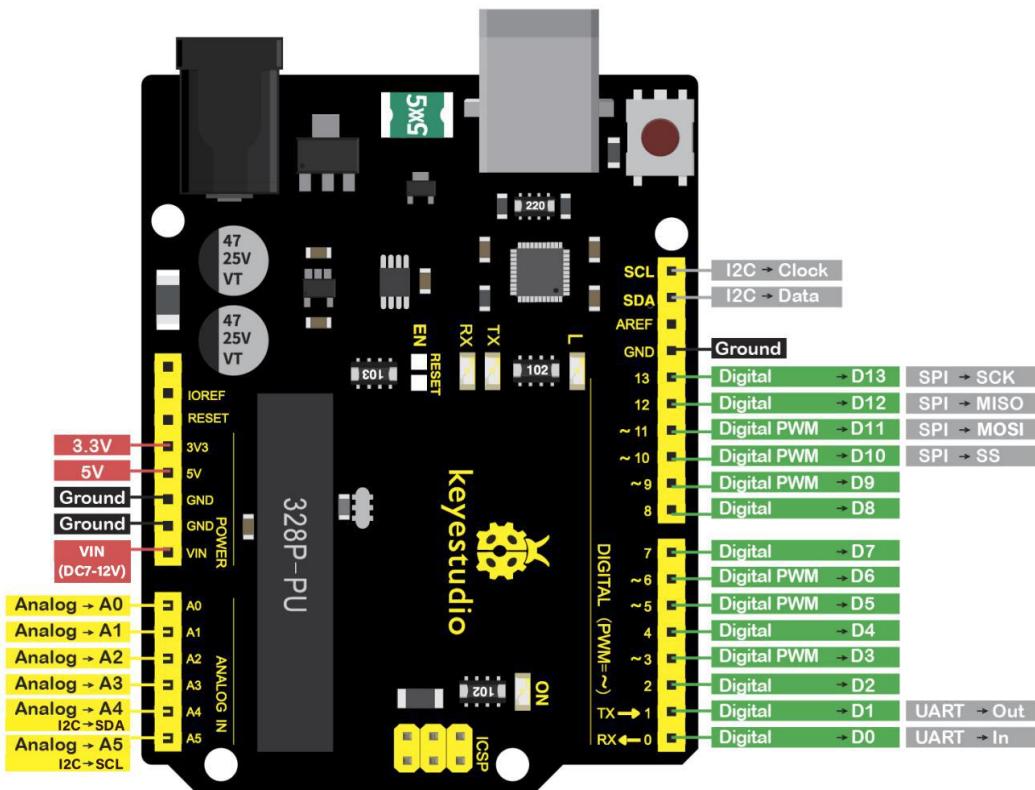
We need to know keyestudio V4.0 development board, as a core of this smart car.



keyestudio V4.0 development board is an Arduino board, which is based on ATmega328P MCU, and with a cp2102 Chip as a UART-to-USB converter.



It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, 2 ICSP headers and a reset button.

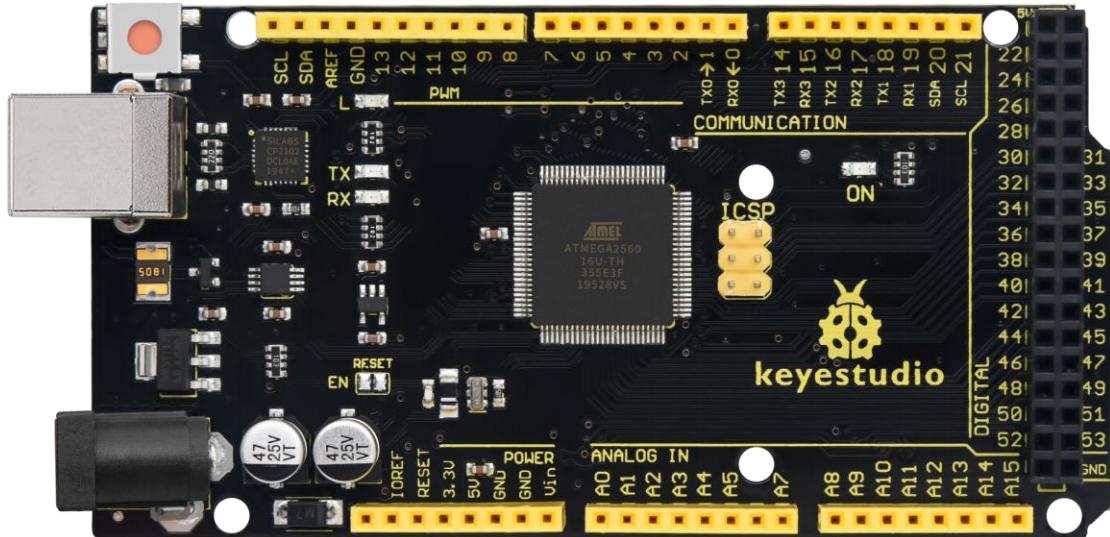


It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it via an external DC power jack (DC 7-12V) or via female headers Vin/ GND(DC 7-12V) to get started.

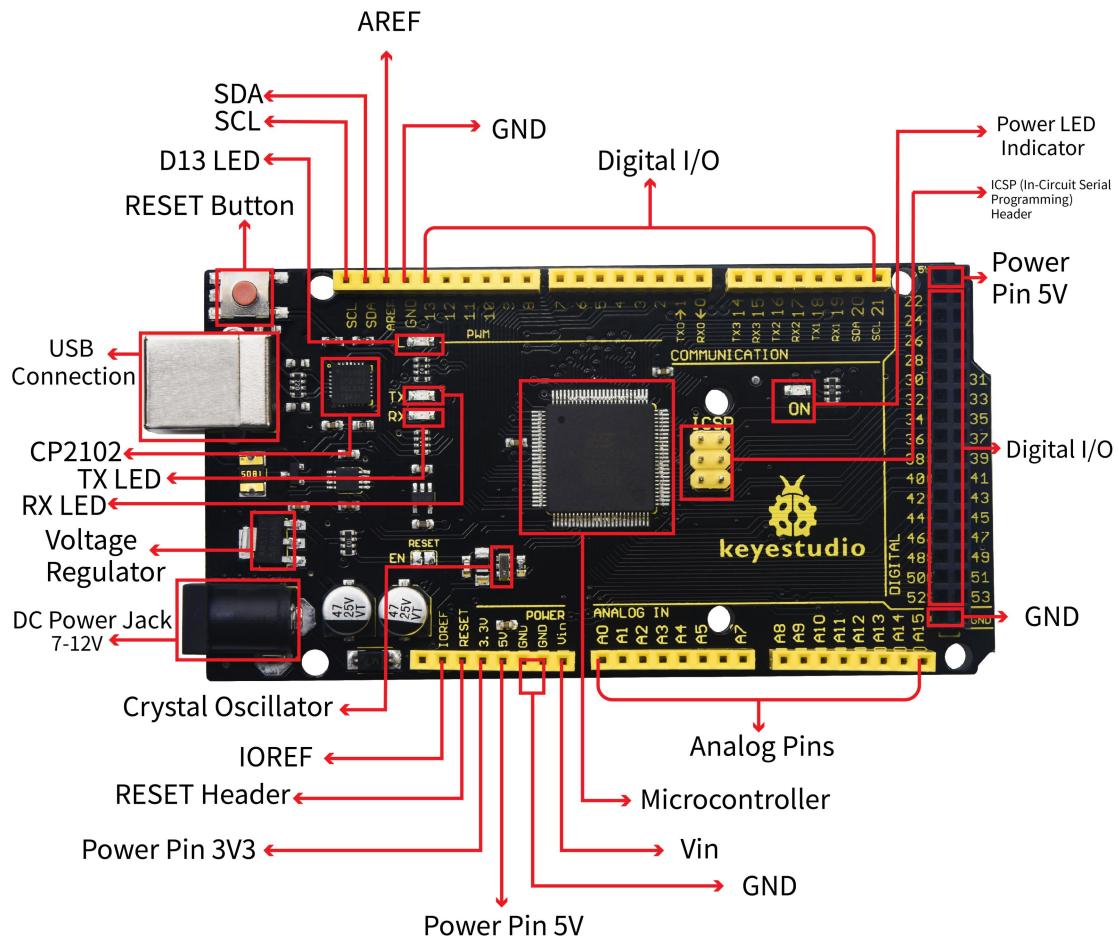
Microcontroller	ATmega328P-PU
Operating Voltage	5V
Input Voltage (recommended)	DC7-12V
Digital I/O Pins	14 (D0-D13) (of which 6 provide PWM output)
PWM Digital I/O Pins	6 (D3, D5, D6, D9, D10, D11)
Analog Input Pins	6 (A0-A5)
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P-PU) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P-PU)
EEPROM	1 KB (ATmega328P-PU)
Clock Speed	16 MHz
LED_BUILTIN	D13

(3) Keyestudio MEGA 2560 Board

Keyestudio Mega 2560 R3 is a microcontroller board based on the ATMEGA2560-16AU , fully compatible with ARDUINO MEGA 2560 R3.



It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, 1 ICSP header, and a reset button. The built-in ICSP port can burn the firmware for ATMEGA2560-16AU directly. This chip is burnt the firmware well before leaving the factory, therefore, we hardly use it. We can power on by USB wire, DC head and Vin GND pins. To facilitate wiring, a 0.5 m USB wire is provided for you.



Specialized Functions of Some Pins:

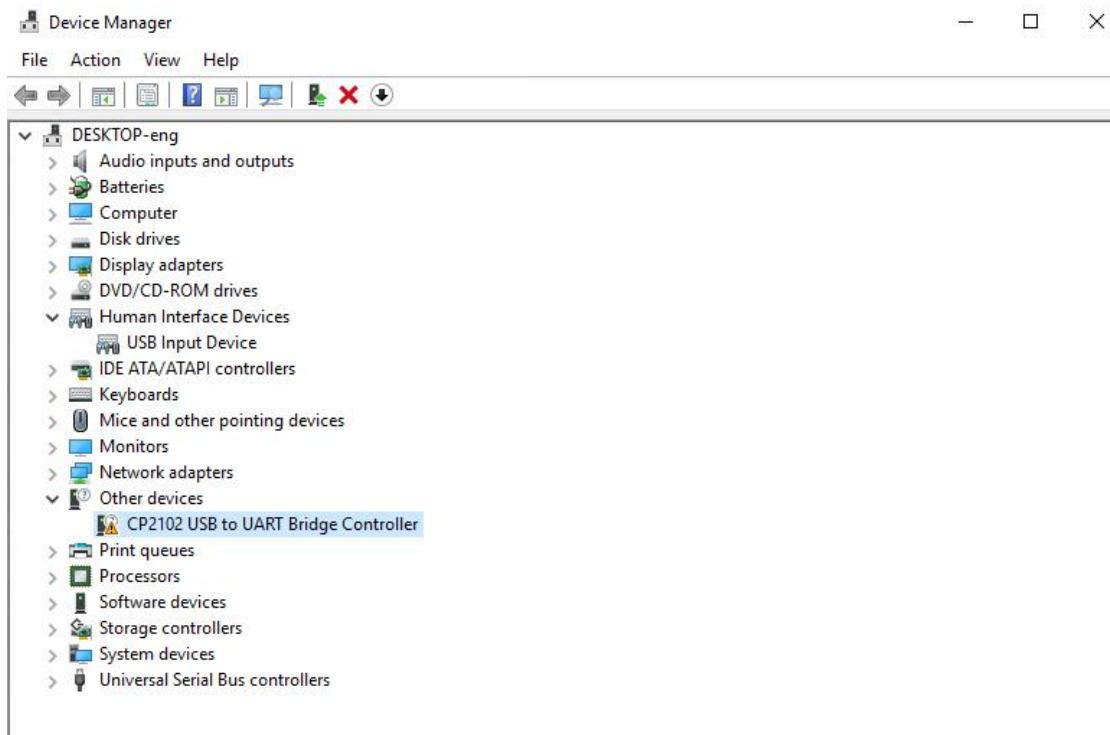
1. Serial Communication: D0 (RX0) and D1 (TX1); Serial 1: D19 (RX1) and D18 (TX1); Serial 2: D17 (RX2) and D16 (TX2); Serial 3: D15 (RX3) and D14 (TX3). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the CP2102 USB-to-TTL Serial chip.

2. PWM Pins (Pulse-Width Modulation): D2 to D13, and D44 to D46. Provide 8-bit PWM output with the `analogWrite()` function.
3. External Interrupts: D2 (interrupt 0), D3 (interrupt 1), D18 (interrupt 5), D19 (interrupt 4), D20 (interrupt 3), and D21 (interrupt 2). These pins can be configured to trigger an interrupt on a low level, a rising or falling edge, or a change in level. See the `attachInterrupt()` function for details.
4. SPI communication: D53 (SS), D52 (SCK), D51 (MOSI), D50 (MISO). These pins support SPI communication using the `SPI` library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Arduino Uno.
5. IIC communication: D20 (SDA); D21 (SCL). Support TWI communication using the `Wire` library.

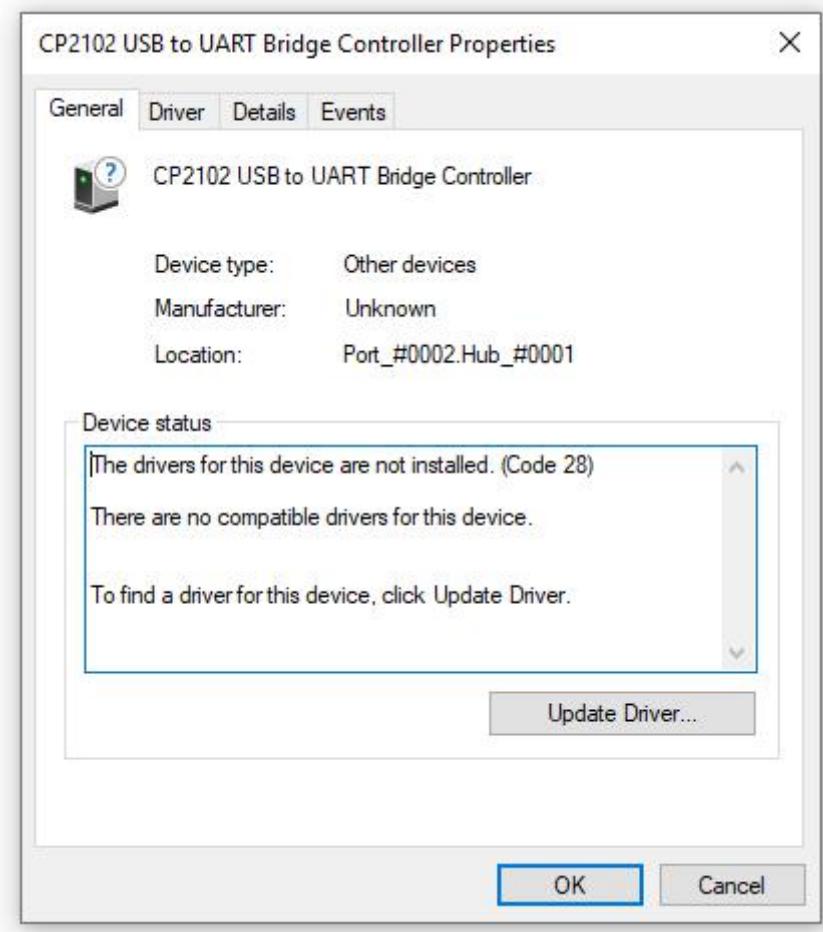
(4) Installing V4.0 board Driver

Let's install the driver of keyestudio V4.0 board. The USB-TTL chip on V4.0 board adopts CP2102 serial chip. The driver program of this chip is included in Arduino 1.8 version and above, which is convenient. Plug on USB port of board, the computer can recognize the hardware and automatically install the driver of CP2102.

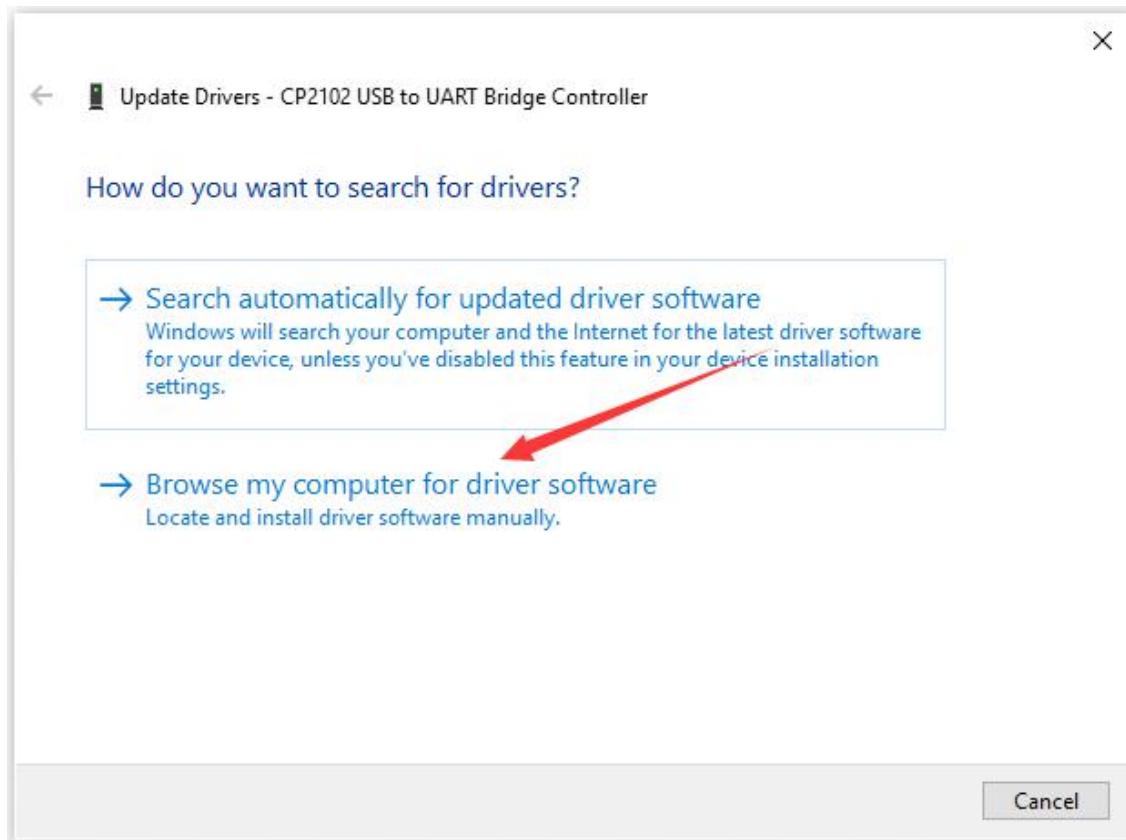
If install unsuccessfully, or you intend to install manually, open the device manager of computer. Right click Computer----- Properties----- Device Manager



There is a yellow exclamation mark on the page, which implies installing unsuccessfully. Then we double click the hardware and update the driver.

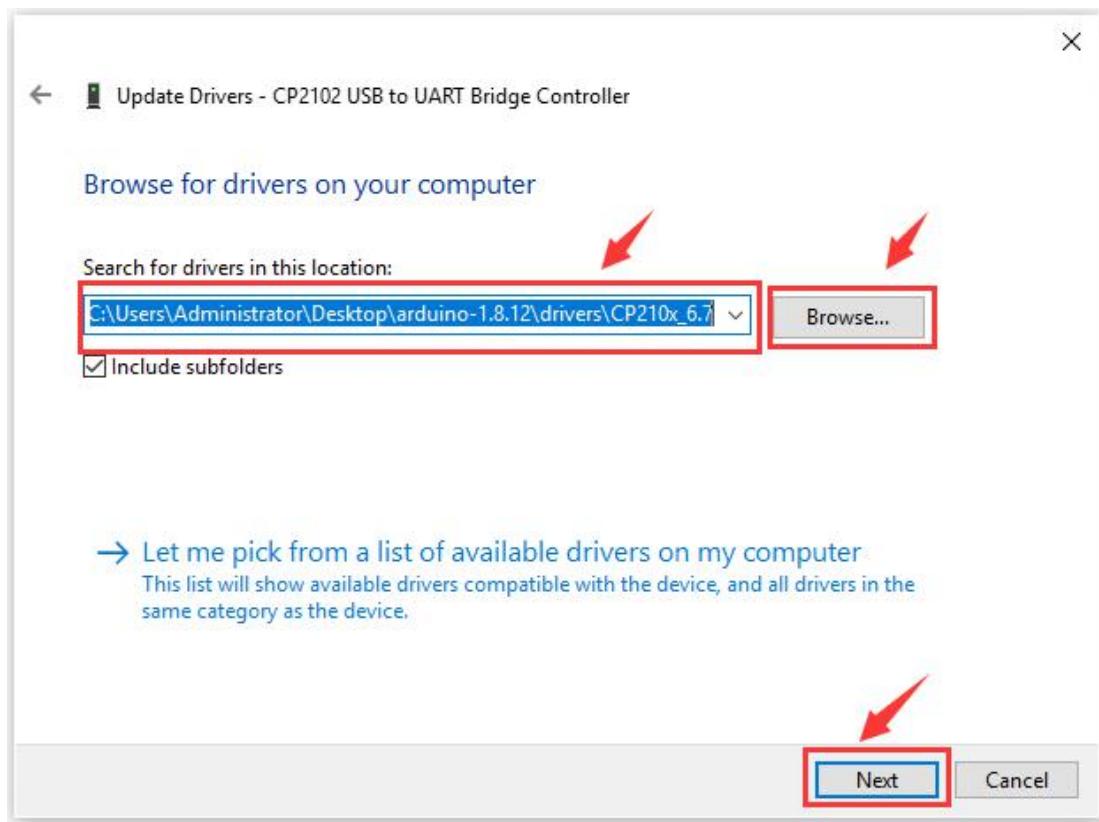


Click “OK” to enter the following page, click “browse my computer for updated driver software”, find out the installed or downloaded ARDUINO software. As shown below

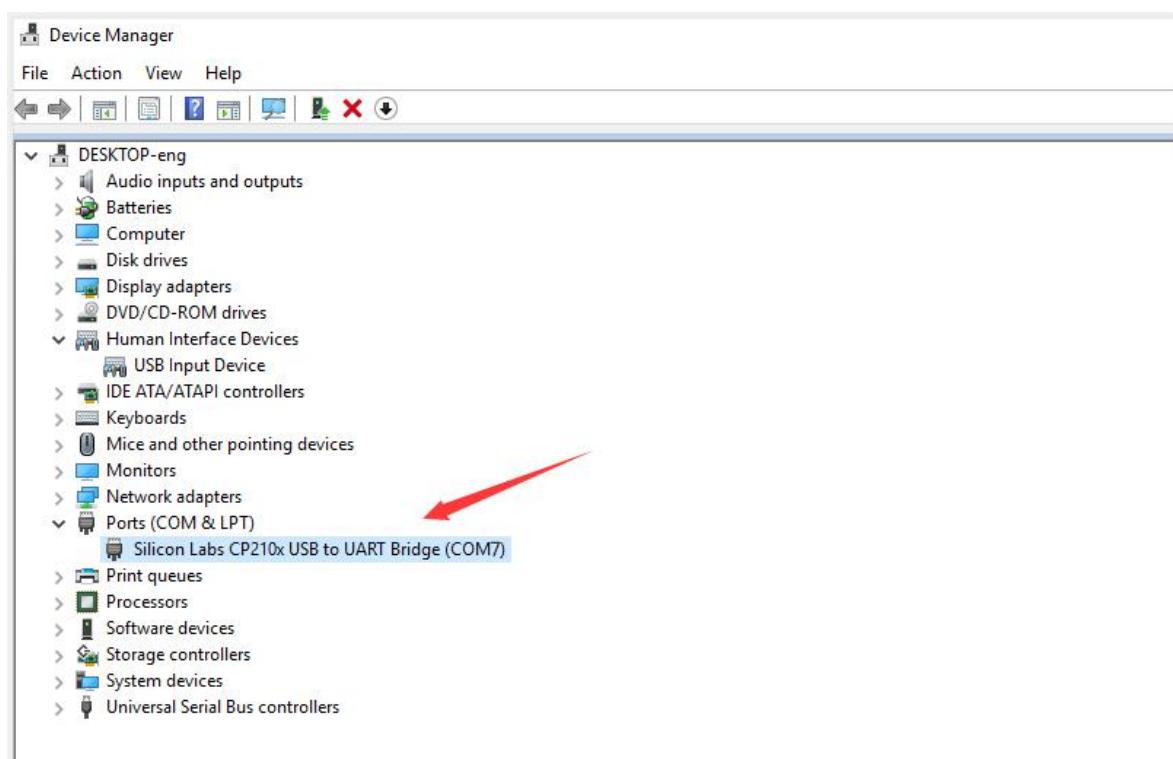
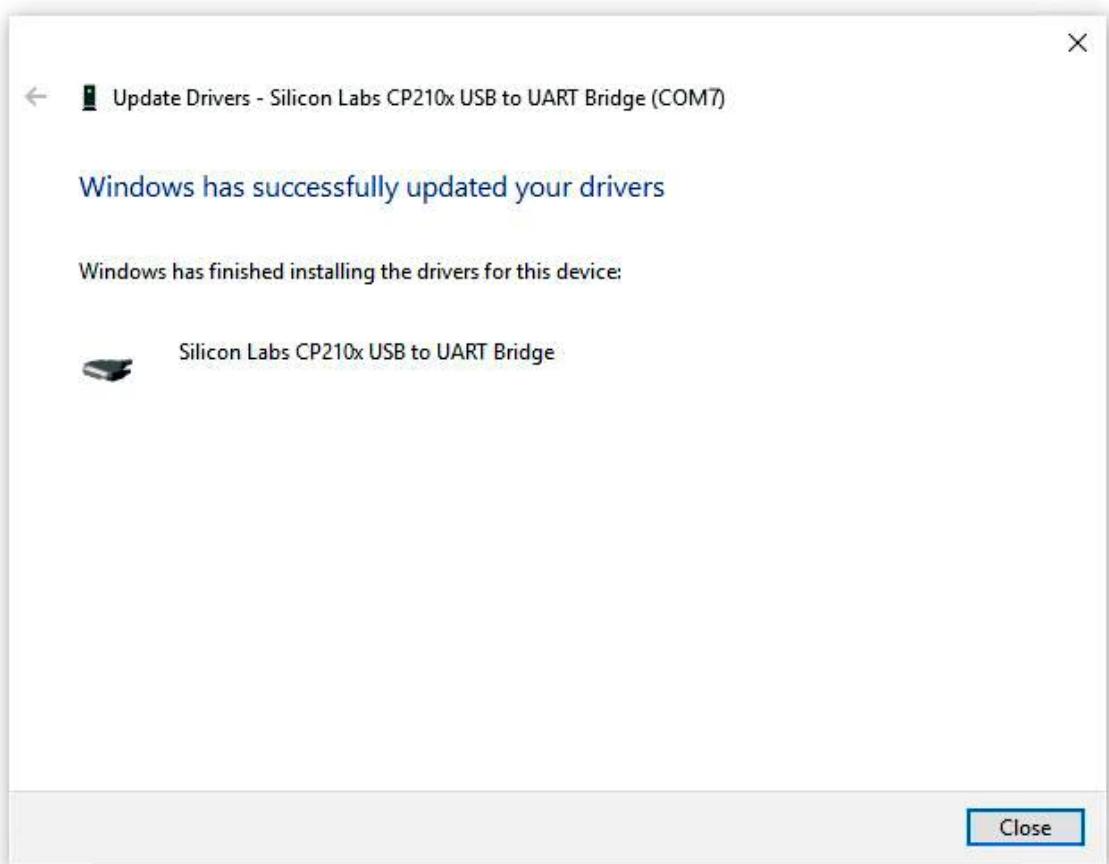


There is a **DRIVERS** folder in **Arduino software installed package** ( arduino-1.8.12), open driver folder and you can see the driver of CP210X series chips.

We click “Browse”, then find out the **driver** folder, or you could enter “driver” to search in rectangular box, then click “next”, the driver will be installed successfully. (I place Arduino software folder on the desktop, you could follow my way)



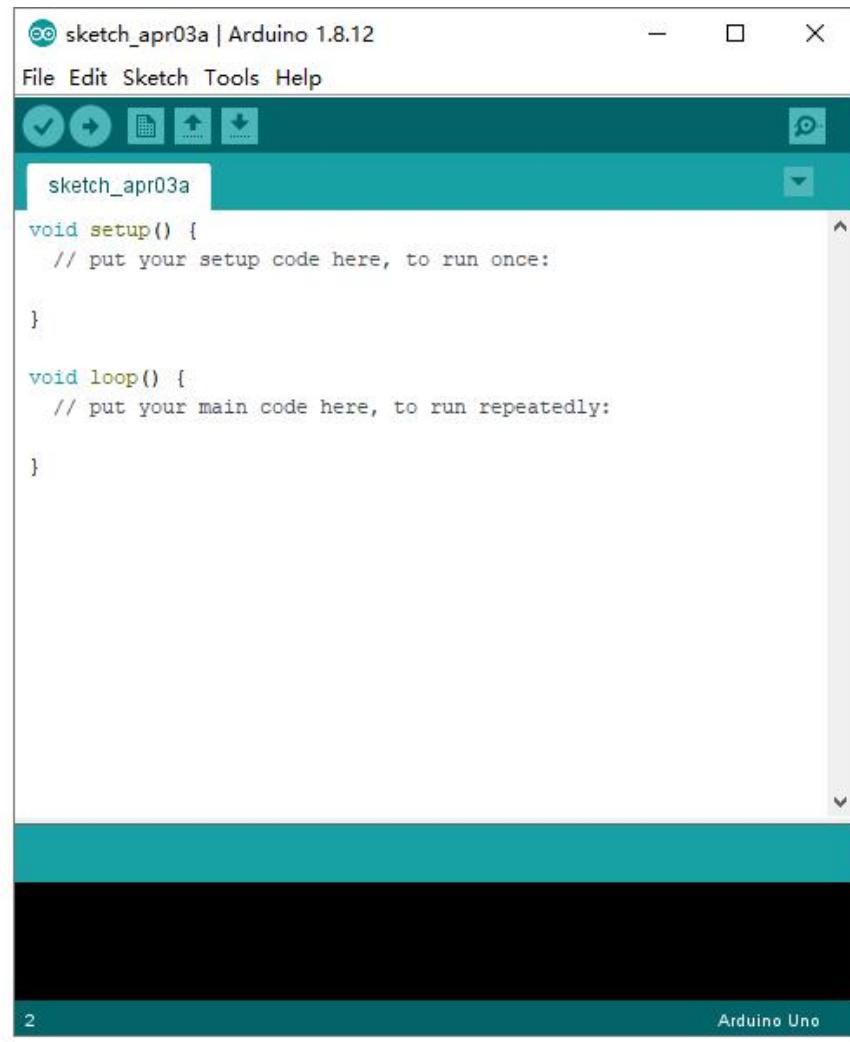
Open device manager, we will find the yellow exclamation mark disappear. The driver of CP2102 is installed successfully.



The installation method of keyestudio MEGA 2560 board and V4.0 board is same

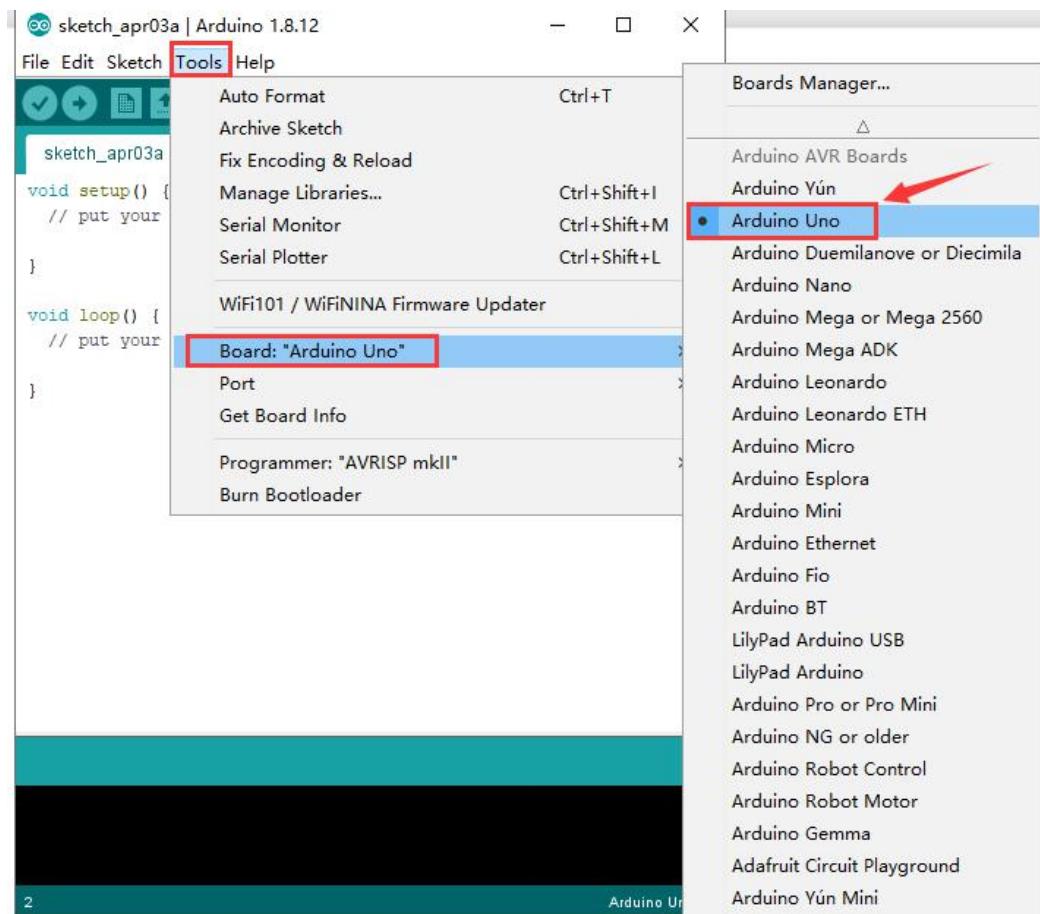
(5) Arduino IDE Setting

Click  icon, open Arduino IDE.

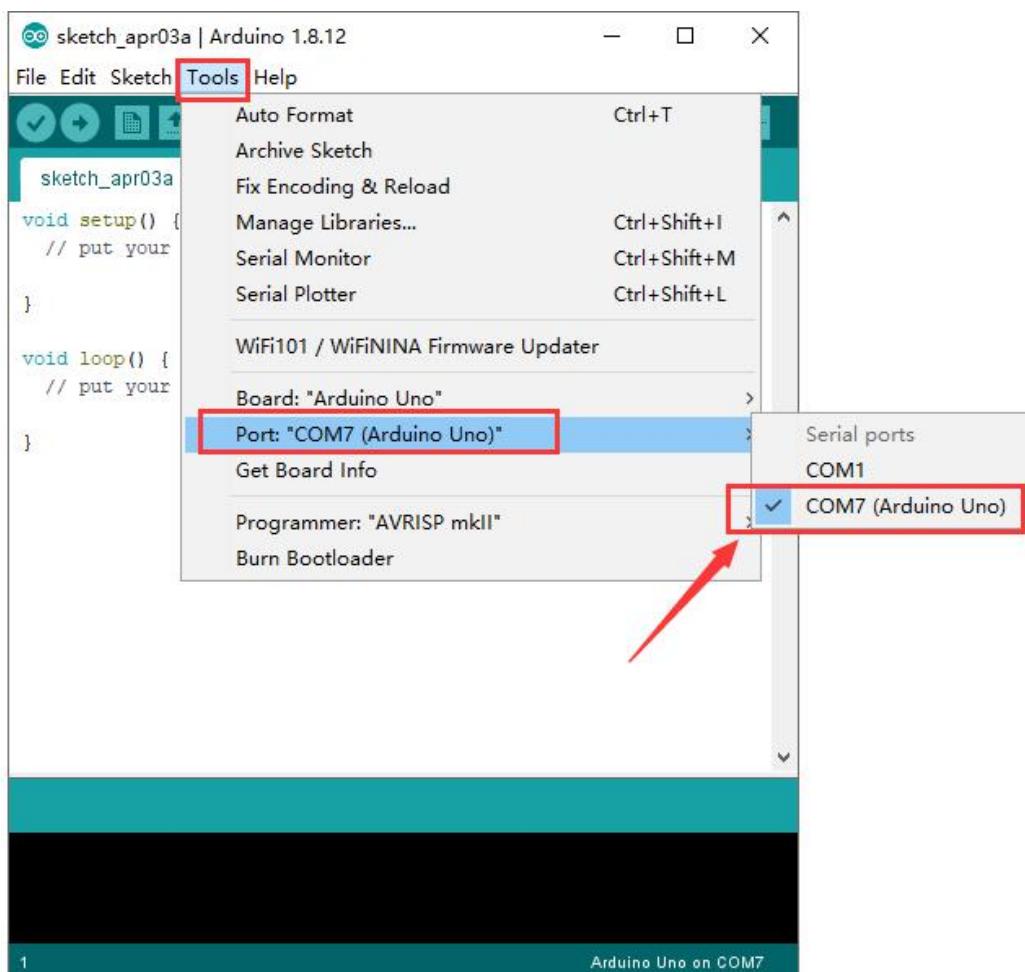


To avoid the errors when uploading the program to the board, you need to select the correct Arduino board that matches the board connected to your computer.

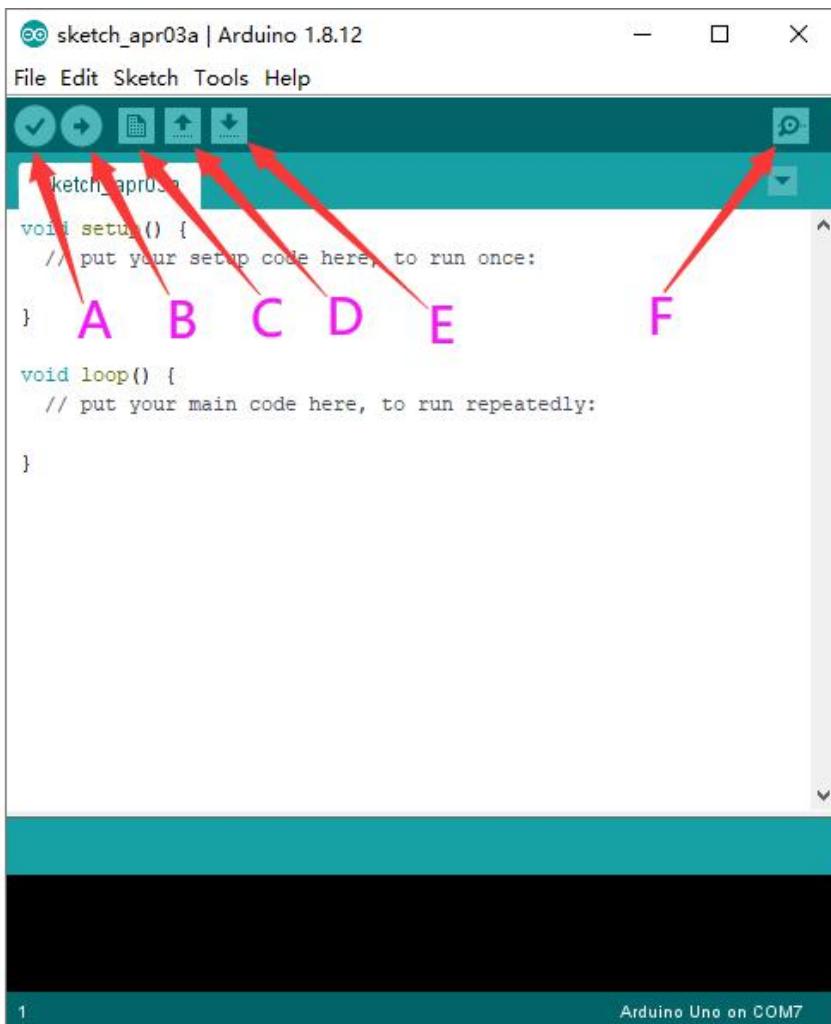
Then come back to the Arduino software, you should click Tools→Board, select the board. (as shown below)



Then select the correct COM port (you can see the corresponding COM port after the driver is successfully installed)



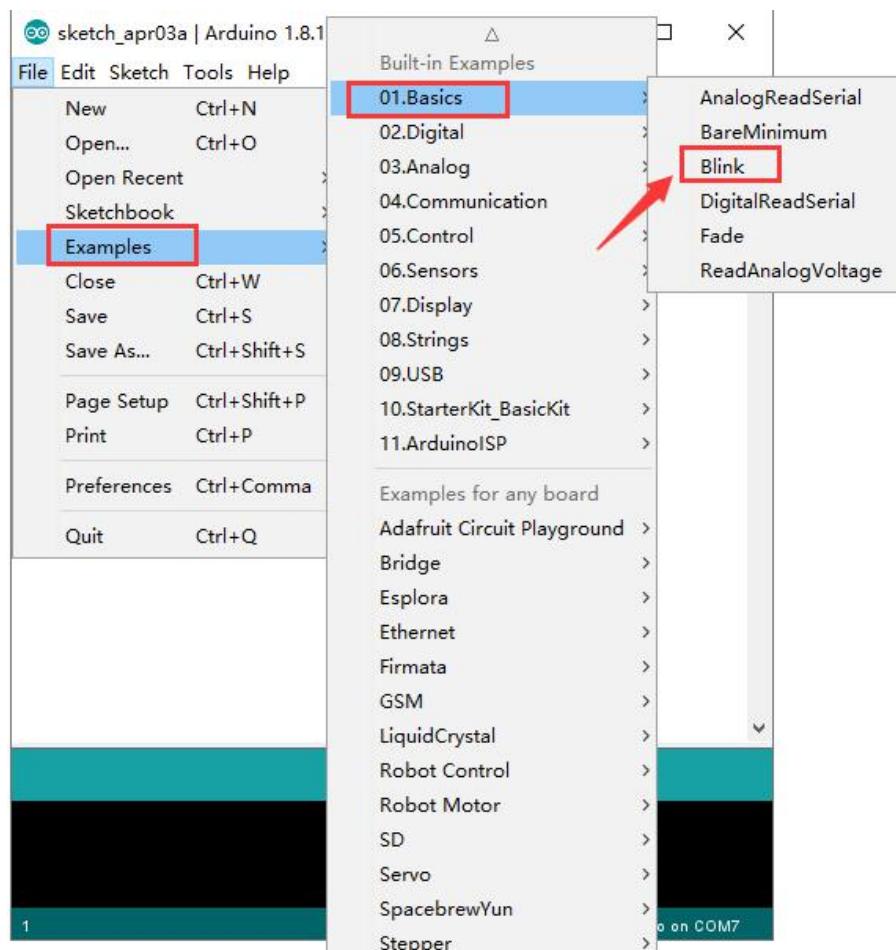
Before uploading the program to the board, let's demonstrate the function of each symbol in the Arduino IDE toolbar.



- A- Used to verify whether there is any compiling mistakes or not.
- B- Used to upload the sketch to your Arduino board.
- C- Used to create shortcut window of a new sketch.
- D- Used to directly open an example sketch.
- E- Used to save the sketch.
- F- Used to send the serial data received from board to the serial monitor.

(6) Start First Program

Open the file to select **Example**, choose **BLINK** from **BASIC**, as shown below

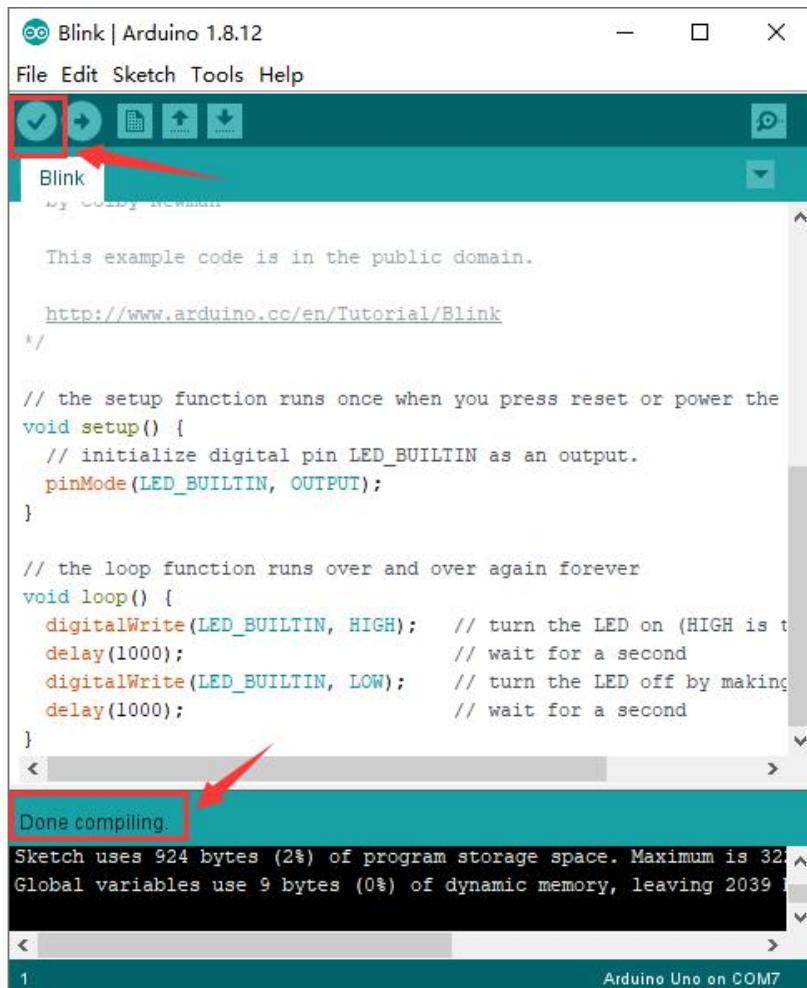


```
This example code is in the public domain.  
http://www.arduino.cc/en/Tutorial/Blink  
//  
  
// the setup function runs once when you press reset or power the  
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is t  
  delay(1000); // wait for a second  
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making  
  delay(1000); // wait for a second  
}
```

Set board and COM port, the corresponding board and COM port are shown on the lower right of IDE.

```
This example code is in the public domain.  
http://www.arduino.cc/en/Tutorial/Blink  
//  
  
// the setup function runs once when you press reset or power the  
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is t  
  delay(1000); // wait for a second  
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making  
  delay(1000); // wait for a second  
}
```

Click  to start compiling the program, check errors.



Blink | Arduino 1.8.12

File Edit Sketch Tools Help

Sketch: Blink

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/Blink>

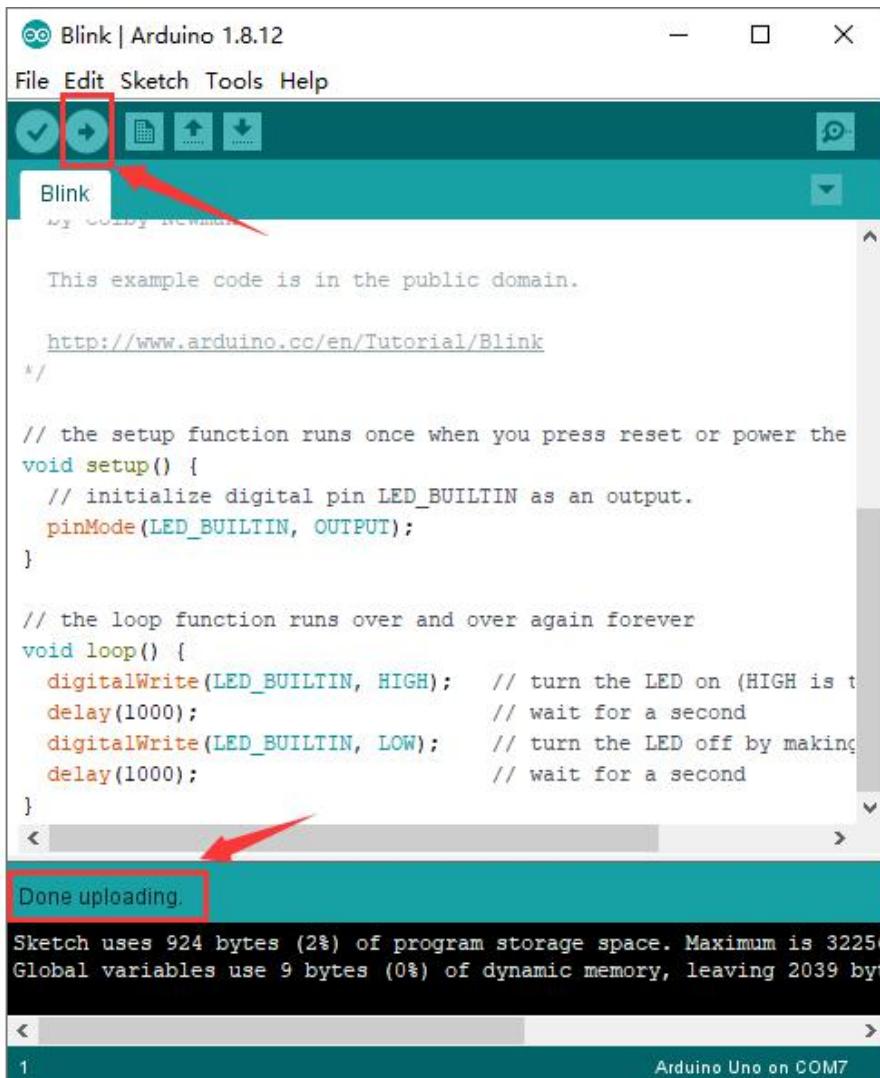
```
// the setup function runs once when you press reset or power the
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on (HIGH is t
    delay(1000);                      // wait for a second
    digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making
    delay(1000);                      // wait for a second
}
```

Done compiling.

Sketch uses 924 bytes (2%) of program storage space. Maximum is 32,128 bytes. Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes available for local variables. Arduino Uno on COM7

Click  to upload the program, upload successfully.



Upload the program successfully, the onboard LED lights on for 1s, lights off for 1s. Congratulation, you finish the first program.

If it is keyestudio MEGA 2560 board, please select **Arduino MEGA or MEGA 2560 board**

4. How to Add a Library?

What are Libraries ?

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc.

For example, the built-in LiquidCrystal library helps talk to LCD displays. There are hundreds of additional libraries available on the Internet for download.

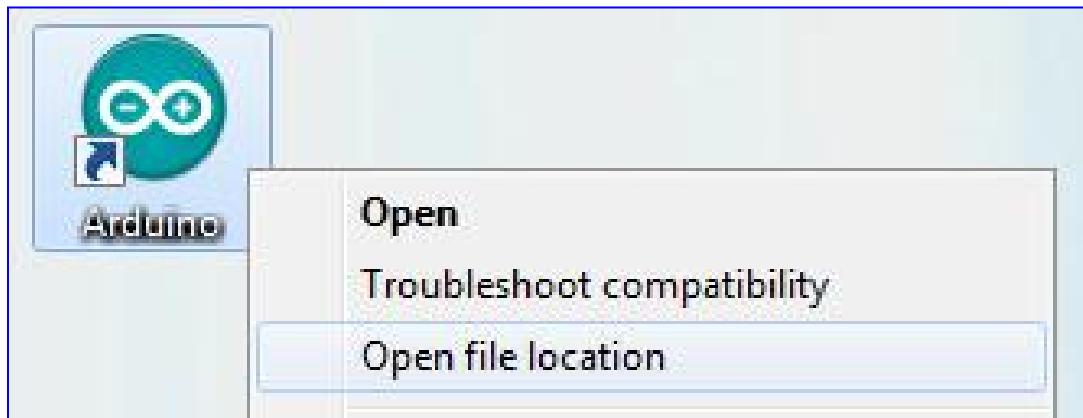
The built-in libraries and some of these additional libraries are listed in the reference.

How to Install a Library ?

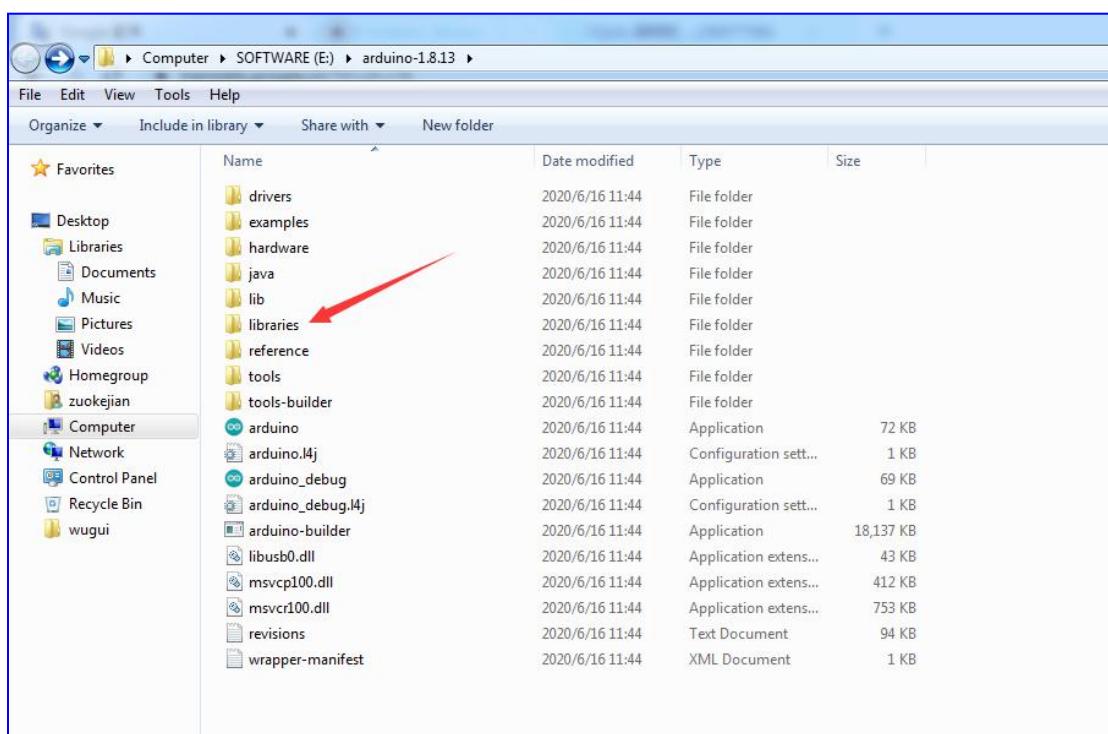
Here we will introduce the most simple way for you to add libraries .

Step 1 After downloading well the Arduino IDE, you can right-click the icon of Arduino IDE.

Find the option "Open file location" shown as below



Step 2 Enter it to find out libraries folder, this folder is the library file of Arduino.



Step 3 Next to find out the “libraries” folder of this kit(seen in the link <https://fs.keyestudio.com/KS0077-78-79>)

👤 > 1.Arduino > Starter kit >

KS0077 (78, 79) keyestudio Super Learning Kit for ARDU...

Click here to describe this folder and turn it into a Space [Show examples](#)

★ Pin or drag files and folders here for quick access

4 folders

Create ⌂ ⌂ ⌂ ⌂ ⌂

Name	Modified	Recent activity	Type
▶ 1. about keyestudio	5/21/20, 4:45 pm	--	File folder
▶ 2. Arduino software	5/21/20, 4:45 pm	--	File folder
▶ 3. Getting started with ard...	5/21/20, 4:45 pm	--	File folder
▶ 4. Tutorial	6/10/20, 2:13 pm	--	File folder

👤 > ... > Starter kit > KS0077 (78, 79) keyestudio Super Learning Kit for ARDUINO >

4. Tutorial

Click here to describe this folder and turn it into a Space [Show examples](#)

★ Pin or drag files and folders here for quick access

2 folders, 1 file

Create ⌂ ⌂ ⌂ ⌂ ⌂

Name	Modified	Recent activity	Type
▶ libraries	5/21/20, 4:45 pm	--	File folder
▶ project code	5/21/20, 4:45 pm	--	File folder
▶ KS0077 (78,79) Super Lear...	5/27/19, 3:40 pm	✖ You moved 7/22/20, 3:43 pm	WPS PDF 文档

libraries

Click here to describe this folder and turn it into a Space [Show examples](#)

* Pin or drag files and folders here for quick access

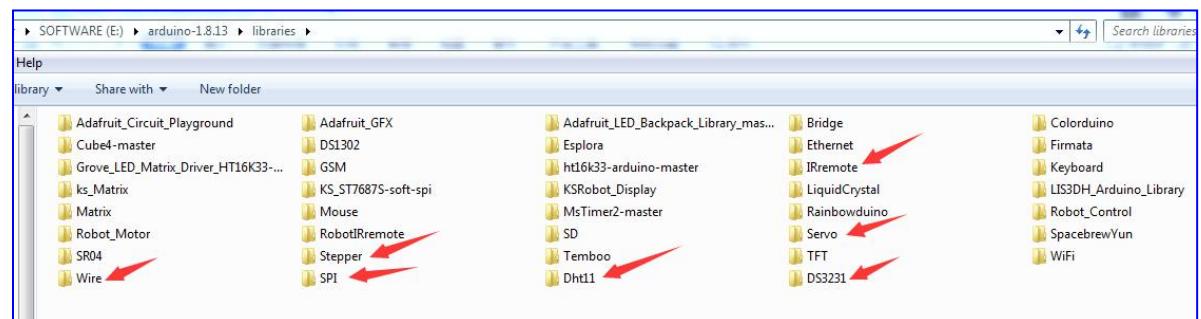
7 folders



Name	Modified	Recent activity	Type
Dht11	5/21/20, 4:45 pm	--	File folder
DS3231	5/21/20, 4:45 pm	--	File folder
IRremote	5/21/20, 4:45 pm	--	File folder
Servo	5/21/20, 4:45 pm	--	File folder
SPI	5/21/20, 4:45 pm	--	File folder
Stepper	5/21/20, 4:45 pm	--	File folder
Wire	5/21/20, 4:45 pm	--	File folder

You just need to replicate and paste above libraries into the **libraries** folder of Arduino IDE.

Then the libraries of this kit are installed successfully, as shown below



Note the Arduino software download and the driver installation of keyestudio Mega 2560 R3 board is similar to arduino V4.0 board.

5. Project Details

Project 1: Hello World



Hello world

1. Introduction

As for starters, we will begin with something simple. In this project, you only need an Arduino and a USB Cable to start the "Hello World!" experiment. It is not only a communication test of your Arduino and PC, but also a primer project for you to have your first try in the Arduino world!

2. Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- USB cable*1

3.Sample Code

After installing driver for Arduino, let's open Arduino software and compile code that enables Arduino to print "Hello World!" under your instruction. Of course, you can compile code for Arduino to continuously echo "Hello World!" without instruction.

A simple **If () statement** will do the instruction trick. when Arduino gets an instruction and then to print "Hello World!".

```
/*
keyestudio super learning kit
Project 1
Hello World
http://www.keyestudio.com
*/
int val;//define variable val
void setup()
{
Serial.begin(9600);// set the baud rate at 9600 .
}
void loop()
{
val=Serial.read();// read the instruction or character from PC to
```

Arduino, and assign them to Val.

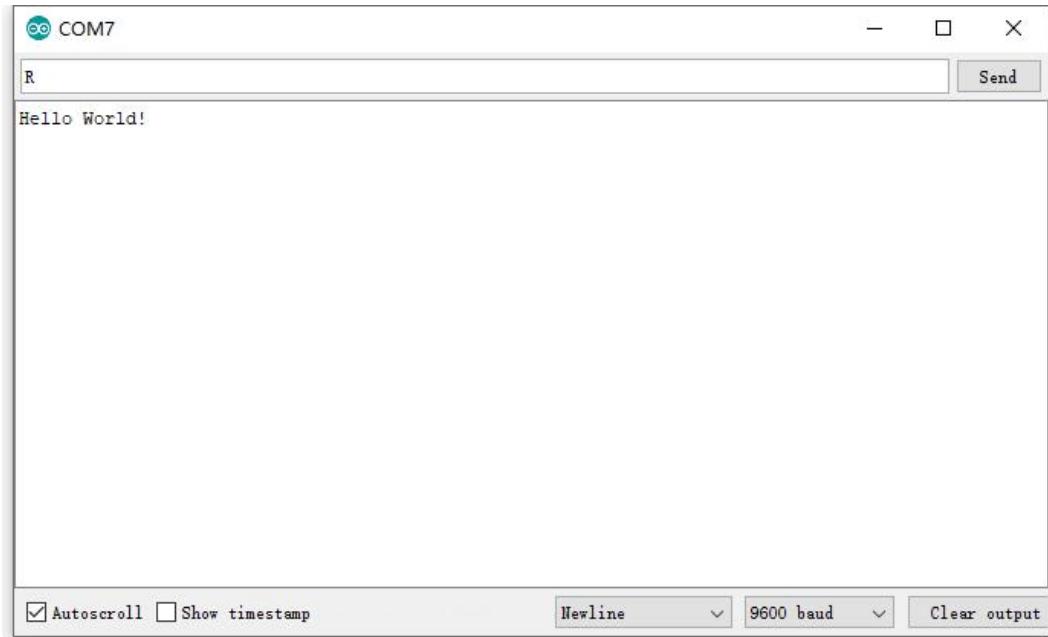
```
if(val=='R')// determine if the instruction or character received is  
"R".
```

```
{ // if it's "R",  
  
    Serial.println("Hello World!"); // display "Hello World!" string.  
  
} } //
```

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

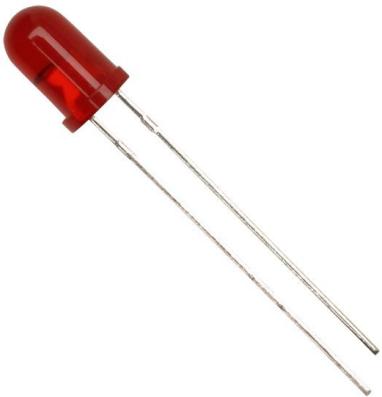
4. Test Result

Click to open the serial monitor, input an “R”, PC will receive the information from Arduino Hello World!



After choosing the proper port, the experiment is easy for you!

Project 2: LED Blinking



1. Introduction

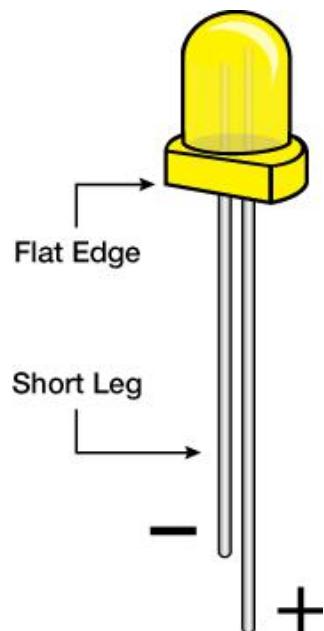
Blinking LED experiment is quite simple. In the "Hello World!" program, we have come across LED. This time, we are going to connect an LED to one of the digital pins rather than using LED13 soldered to the board. Apart from an Arduino and a USB cable, you will need extra parts as below

2. Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- Red M5 LED*1
- 220 Ω Resistor*1
- Breadboard*1
- Breadboard Jumper Wire*2
- USB cable*1

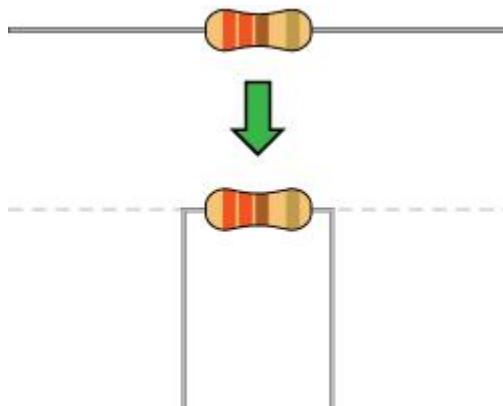
3. Little Knowledge

LED is a type of semiconductor called "Light Emitting Diode" which is an electronic device made of semiconductor materials (silicon, selenium, germanium, etc.). It is dubbed indicator, digital and word display in circuit and device. It has positive and negative poles. The short leg is negative pole, the long one is positive pole.

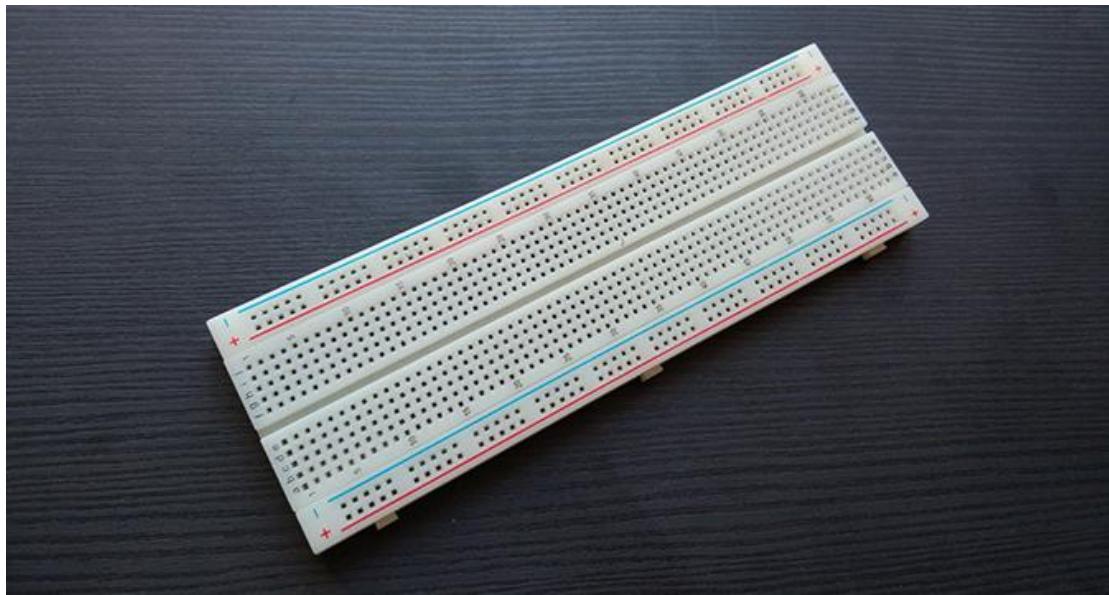


Resistor : Resistor is the electronic component in the circuit, which limits and regulates current flow. Its unit is (Ω).

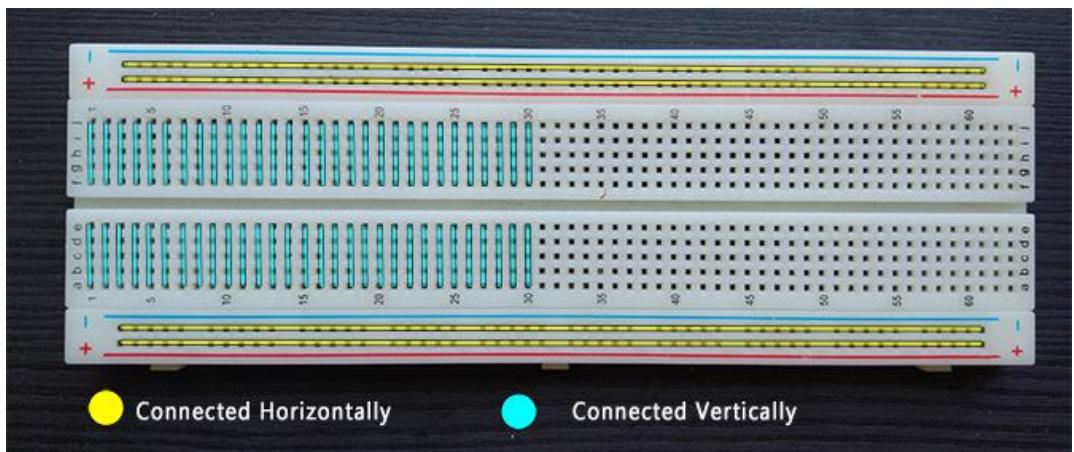
The units larger than ohms are kilohms ($K\Omega$) and megaohms ($M\Omega$). When in use, in addition to the size of the resistance, you must also pay attention to its power. In the project, the leads at both ends of the resistor should be bent at a 90° angle to fit the breadboard properly. If the lead is too long, it can be cut to an appropriate length.



A breadboard is used to build and test circuits quickly before finalizing any circuit design. The breadboard has many holes into which circuit components like ICs and resistors can be inserted. A typical breadboard is shown below:



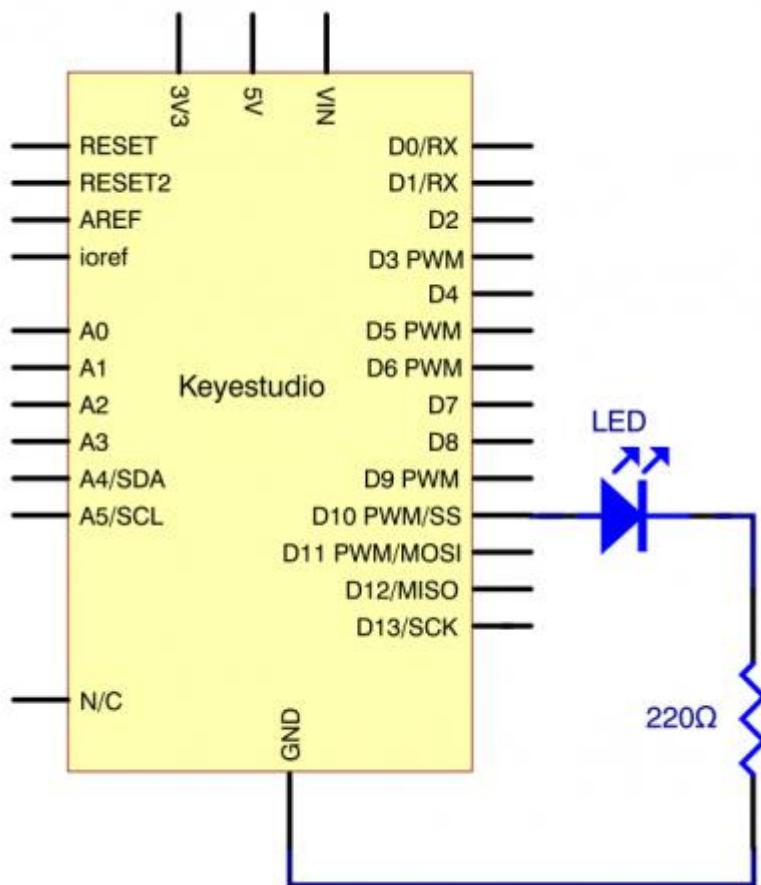
The bread board has strips of metal which run underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally while the remaining holes are connected vertically.



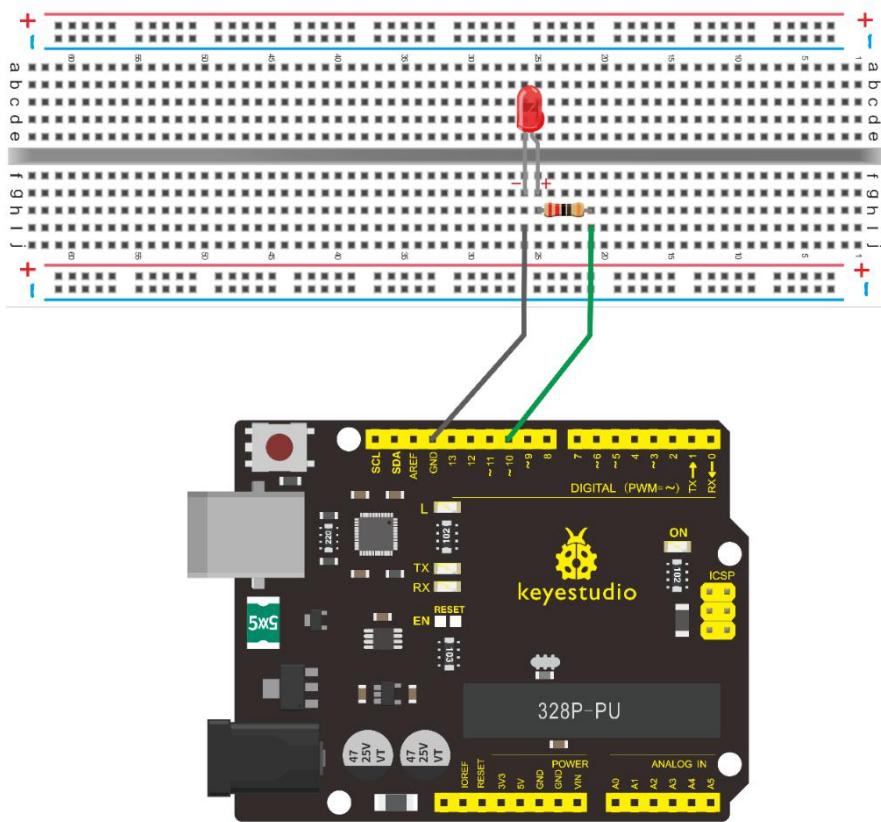
To use the bread board, the legs of components are placed in the holes. Each set of holes connected by a metal a strip underneath forms anode

4.Circuit Connection

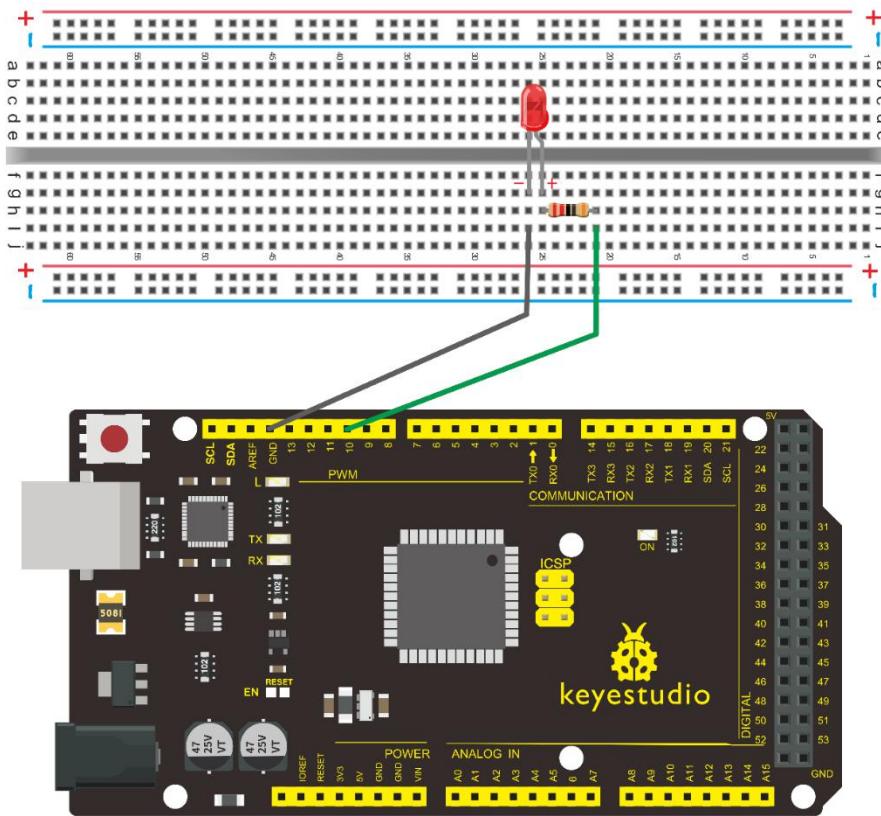
We follow below diagram from the experimental schematic link. Here we use digital pin 10. We connect LED to a 220 ohm resistor to avoid high current damaging the LED.



Connection for V4.0



Connection for 2560



5.Sample Code

```
/*
keyestudio super learning kit
Project 2
Blink
http://www.keyestudio.com
*/
int ledPin = 10; // define digital pin 10.

void setup()
{
pinMode(ledPin, OUTPUT);// define pin with LED connected as output.

}

void loop()
{
digitalWrite(ledPin, HIGH); // set the LED on.

delay(1000); // wait for a second.

digitalWrite(ledPin, LOW); // set the LED off.

delay(1000); // wait for a second
}

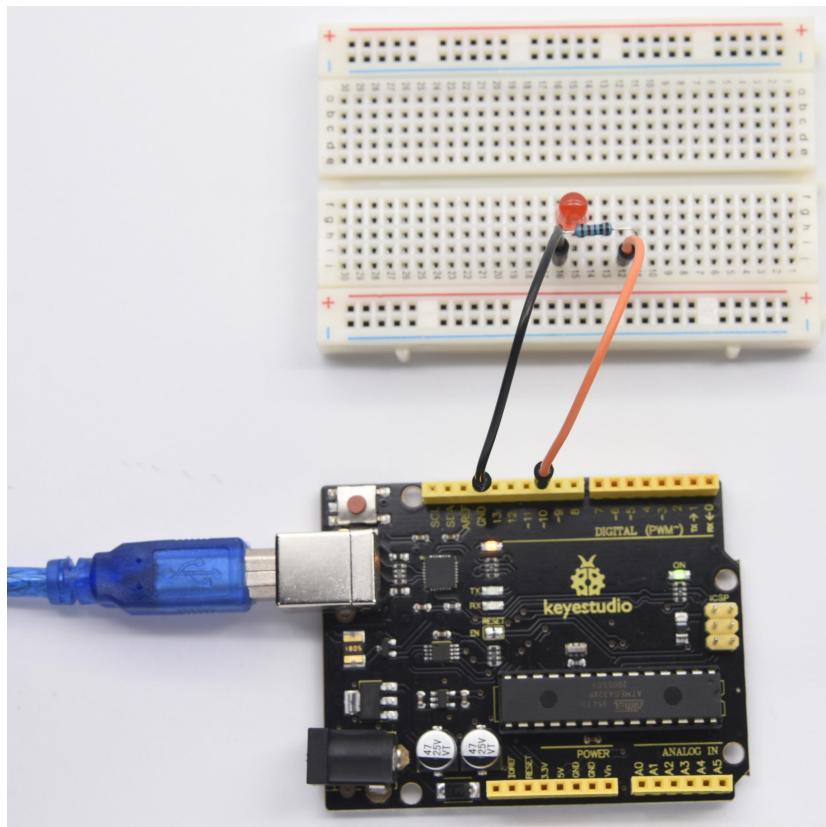
///////////////////////////////
```

6. Test Result

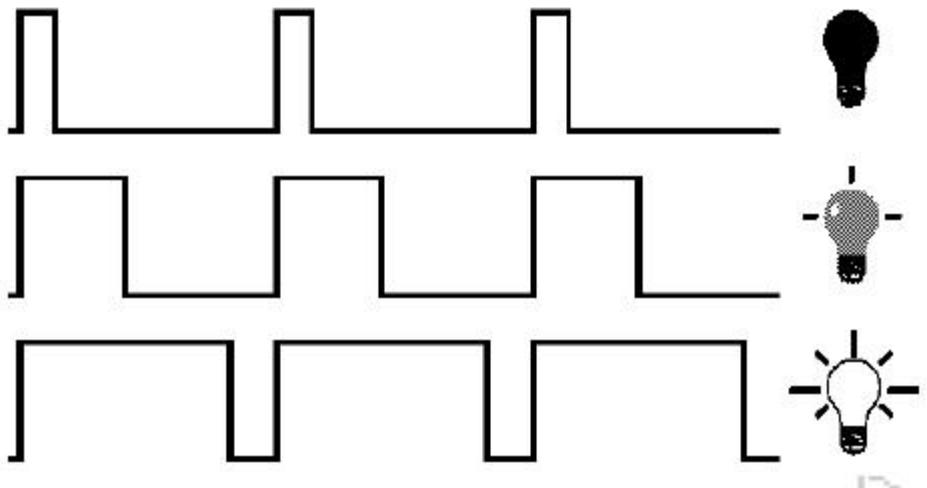
After uploading this program, in the experiment, you will see the LED connected to pin 10 turning on and off, with an interval of approximate one second.

In this way, blinking LED experiment is now completed.

Thank you!



Project 3: PWM



1. Introduction

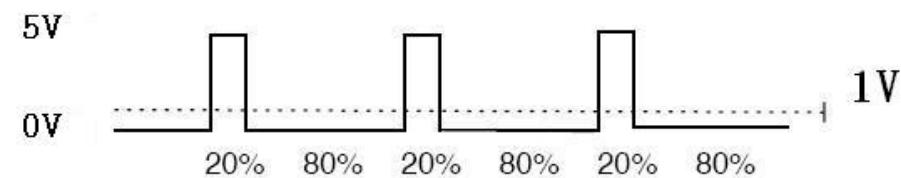
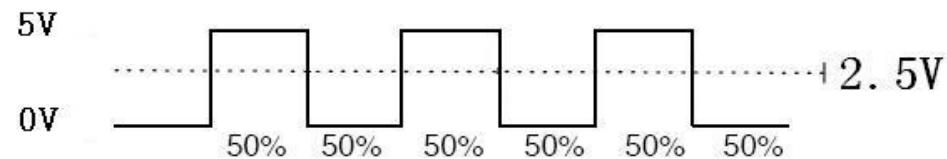
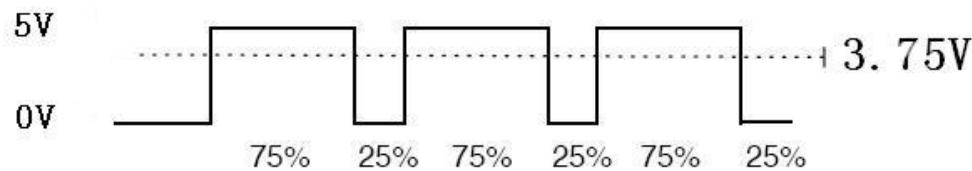
PWM, short for Pulse Width Modulation, is a technique used to encode analog signal level into digital ones. A computer cannot output analog voltage but only digital voltage values such as 0V or 5V. So we use a high resolution counter to encode a specific analog signal level by modulating the duty cycle of PMW.

2. Working Principle

The PWM signal is also digitalized because in any given moment, fully on DC power supply is either 5V (ON), or 0V (OFF). The voltage or current is fed to the analog load (the device that uses the power) by repeated pulse sequence being ON or OFF. Being on, the current is fed to the

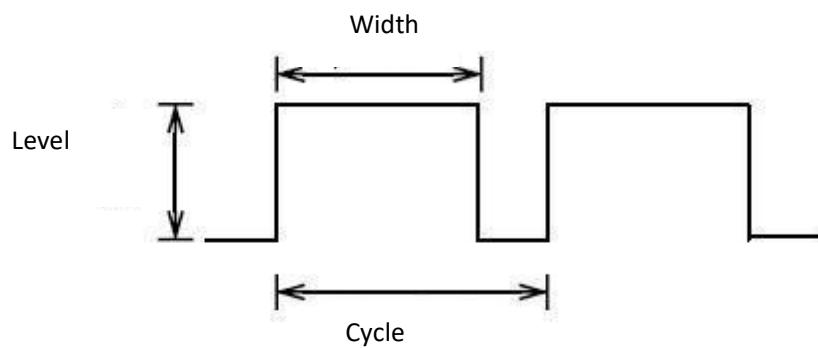
load; being off, it's not. With adequate bandwidth, any analog value can be encoded using PWM. The output voltage value is calculated via the on and off time.

Output voltage = (turn on time/pulse time) * maximum voltage value



PWM has many applications lamp brightness regulating, motor speed regulating, sound making, etc.

The following are the three basic parameters of PWM



1. The amplitude of pulse width (minimum / maximum)
2. The pulse period (The reciprocal of pulse frequency in one second)
3. The voltage level (such as 0V-5V)

There are 6 PMW interfaces on Arduino, namely digital pin 3, 5, 6, 9, 10, and 11.

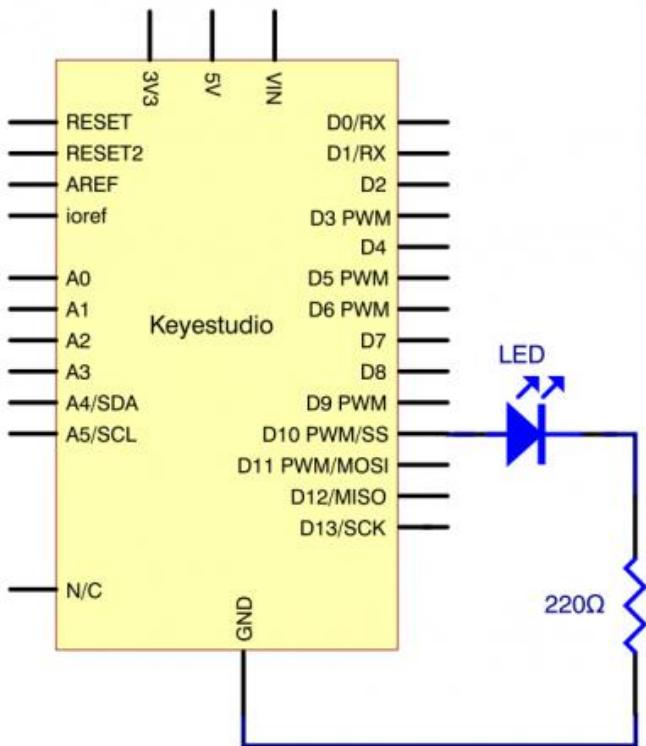
In previous experiments, we have done "button-controlled LED", using digital signal to control digital pin, also one about potentiometer.

This time, we will use a potentiometer to control the brightness of the LED.

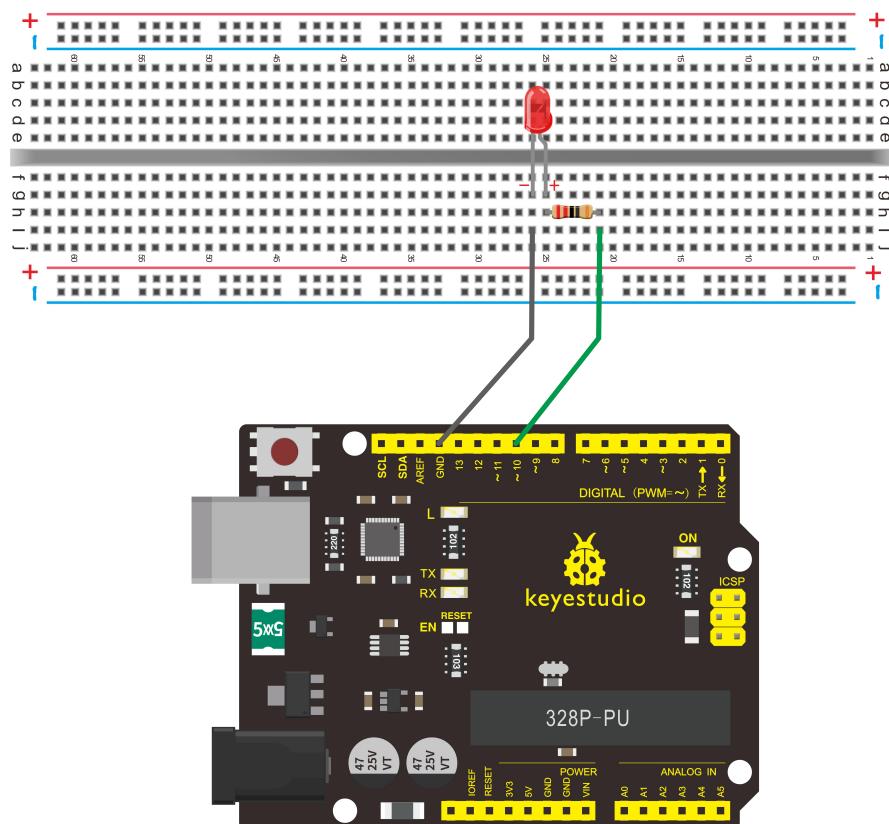
3.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- Red M5 LED*1
- 220 Ω Resistor
- Breadboard*1
- Breadboard Jumper Wire*6
- USB cable*1

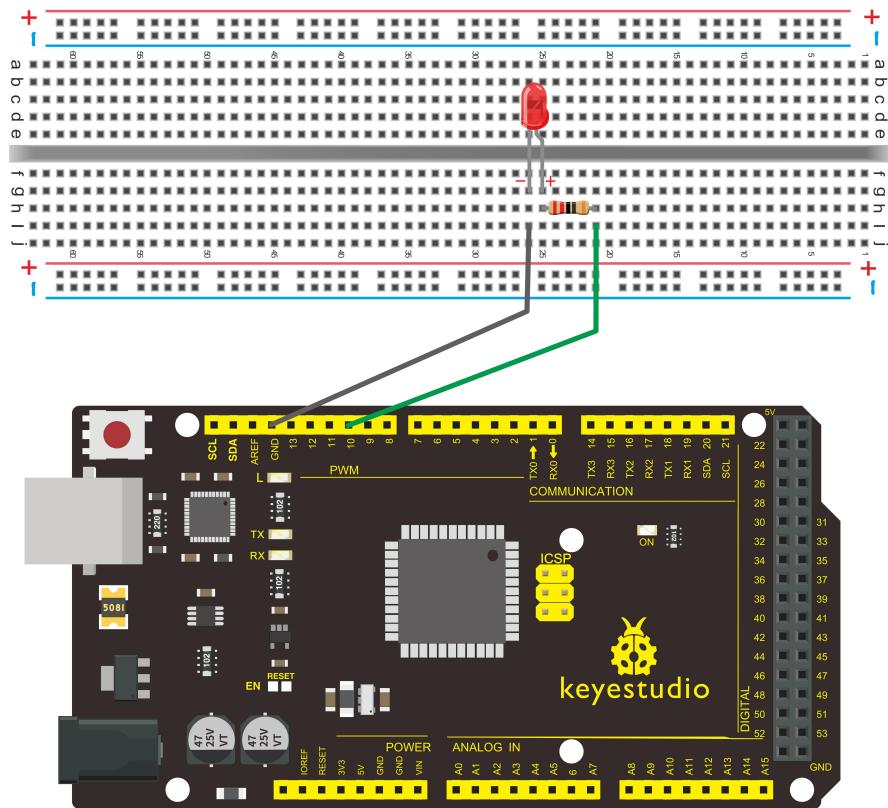
4.Circuit Connection



Connection for V4.0



Connection for 2560 R3



5. Sample Code

```

/*
keyestudio super learning kit
Project 3
pwm
http://www.keyestudio.com
*/
int ledPin = 10;

void setup() {
    pinMode(ledPin,OUTPUT);

```

```
}

void loop(){

    for (int value = 0 ; value < 255; value=value+1){

        analogWrite(ledPin, value);

        delay(5);

    }

    for (int value = 255; value >0; value=value-1){

        analogWrite(ledPin, value);

        delay(5);

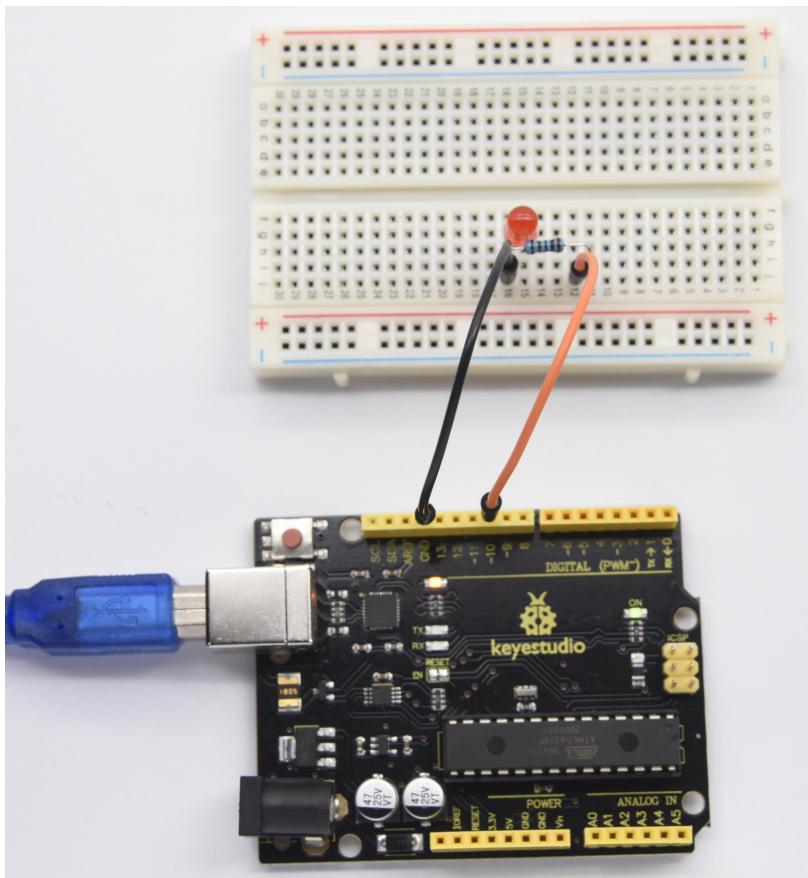
    }

}

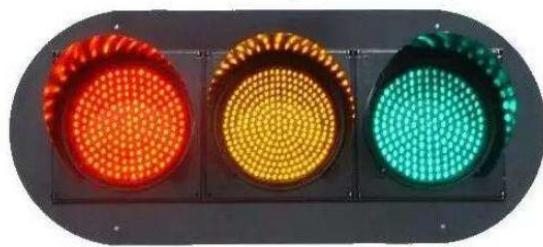
//////////
```

6.Test Result

Hook up via connection diagram, upload the code and plug in power. The external LED lights up then gets dark gradually, which looks like human breath.



Project 4: Traffic Light



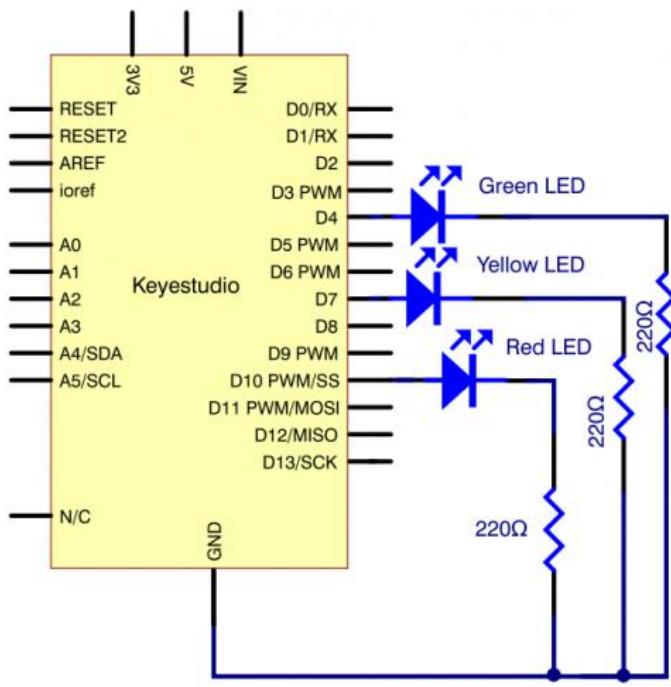
1. Introduction

In the previous program, we have done the LED blinking experiment with one LED. Now, it's time to up the stakes to do a bit more complicated experiment-traffic light. Actually, these two experiments are similar. While in this traffic light experiment, we use three LEDs with different colors rather than an LED.

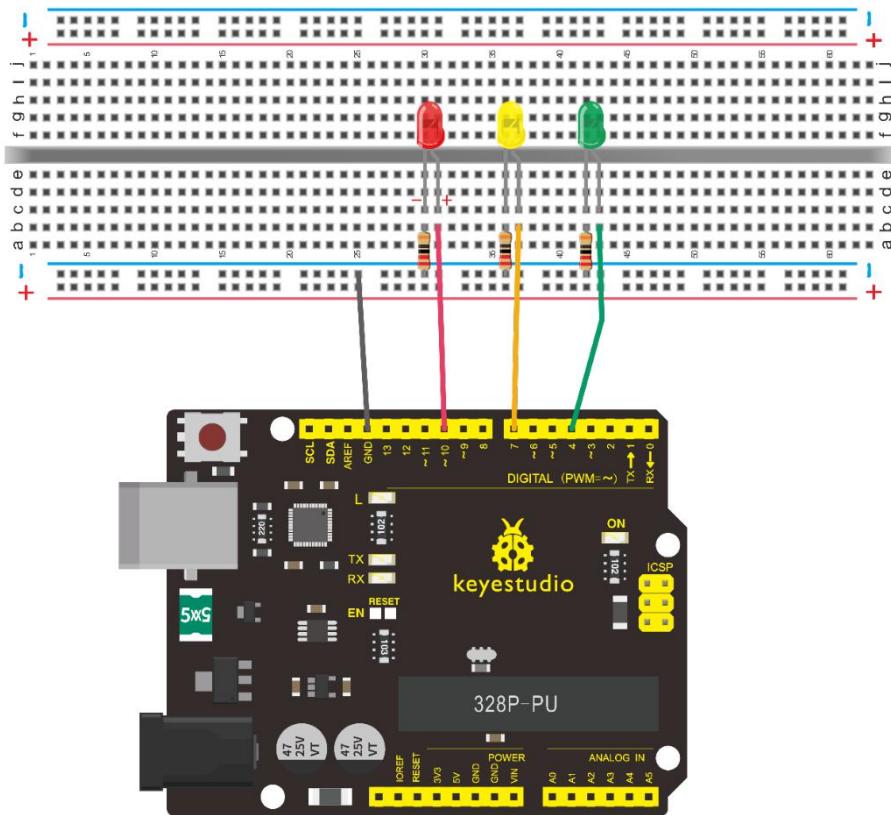
2.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- USB Cable *1
- Red M5 LED*1
- Yellow M5 LED*1
- Blue M5 LED*1
- 220 Ω Resistor *3
- Breadboard*1
- Breadboard Jumper Wire *4

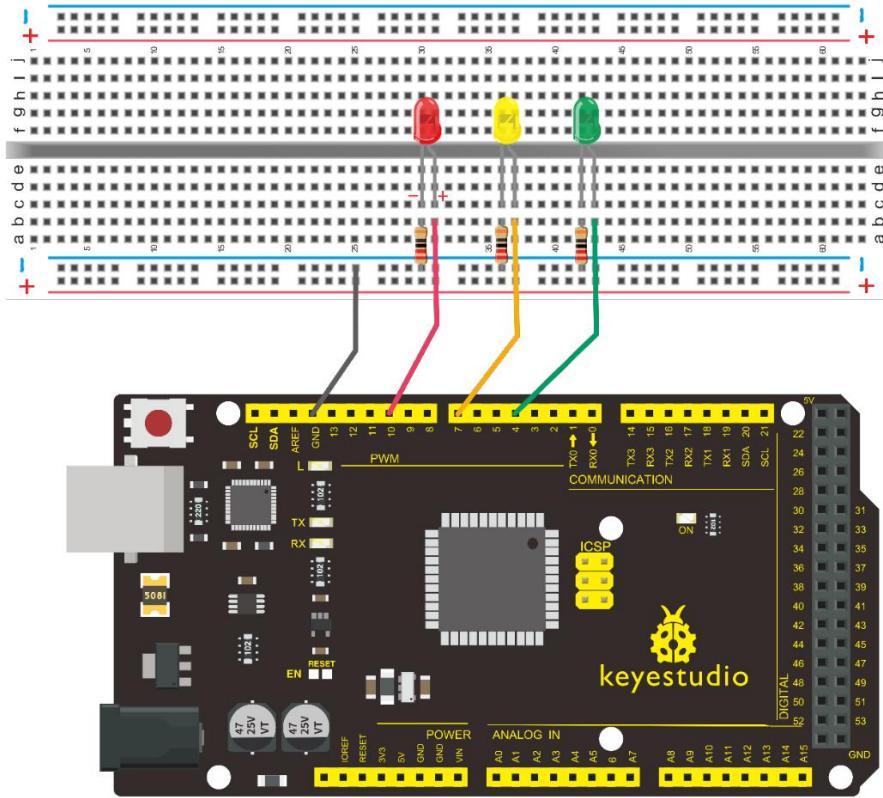
3.Circuit Connection



Connection for V4.0



Connection for 2560



4..Sample Code

Since it is a simulation of traffic lights, the blinking time of each LED should be the same with those in traffic lights system.

In this program, we use Arduino **delay ()** function to control delay time, which is much simpler than C language.

```
/*
keyestudio super learning kit
Project 4
traffic light
```

<http://www.keyestudio.com>

```
*/  
  
int redled =10; // initialize digital pin 10.  
  
int yellowled =7; // initialize digital pin 7.  
  
int blueled =4; // initialize digital pin 4.  
  
void setup()  
{  
  
pinMode(redled, OUTPUT);// set the pin with red LED as "output"  
  
pinMode(yellowled, OUTPUT); // set the pin with yellow LED as  
"output"  
  
pinMode(blueled, OUTPUT); // set the pin with blue LED as "output"  
}  
  
void loop()  
{  
  
digitalWrite(blueled, HIGH);//// turn on blue LED  
  
delay(5000);// wait 5 seconds  
  
digitalWrite(blueled, LOW); // turn off blue LED  
  
for(int i=0;i<3;i++)// blinks for 3 times  
{  
  
delay(500);// wait 0.5 second  
  
digitalWrite(yellowled, HIGH);// turn on yellow LED  
  
delay(500);// wait 0.5 second
```

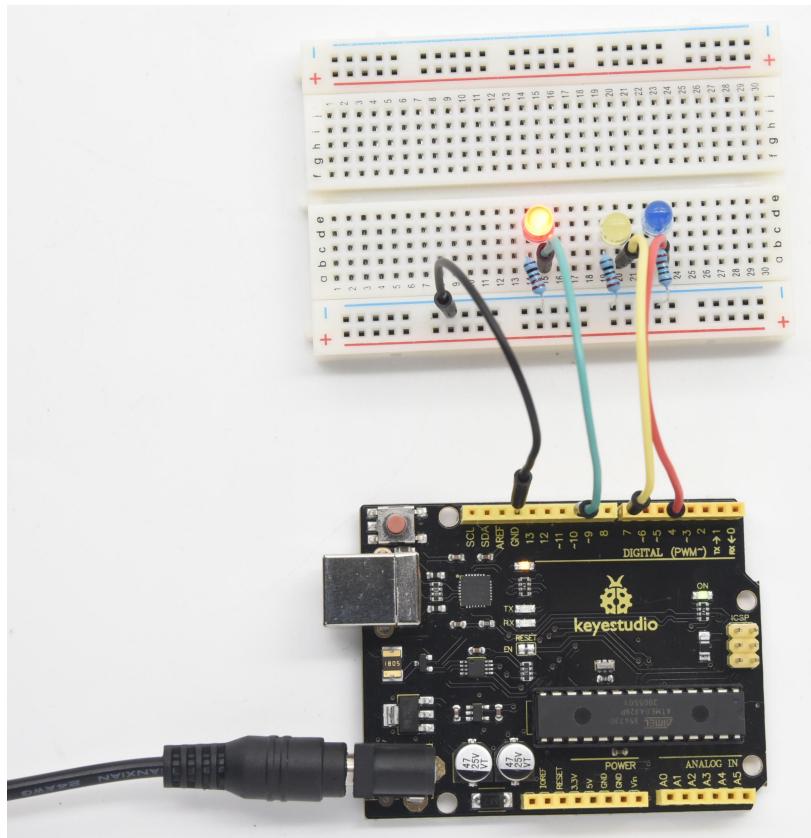
```
digitalWrite(yellowled, LOW);// turn off yellow LED  
}  
  
delay(500);// wait 0.5 second  
  
digitalWrite(redled, HIGH);// turn on red LED  
  
delay(5000);// wait 5 second  
  
digitalWrite(redled, LOW);// turn off red LED  
}  
||||||||||||||||||||||||||||||||||||||||||||||||
```

5. Test Result

When the uploading process is completed, you can see traffic lights of your own design. Note this circuit design is very similar with the one in LED chasing effect.

The blue light will be on for 5 seconds, and then off, followed by the yellow light blinking for 3 times, and then the red light is on for 5 seconds, repeatedly forming a cycle.

Experiment is now completed, thank you!



Project 5: LED Chasing Effect



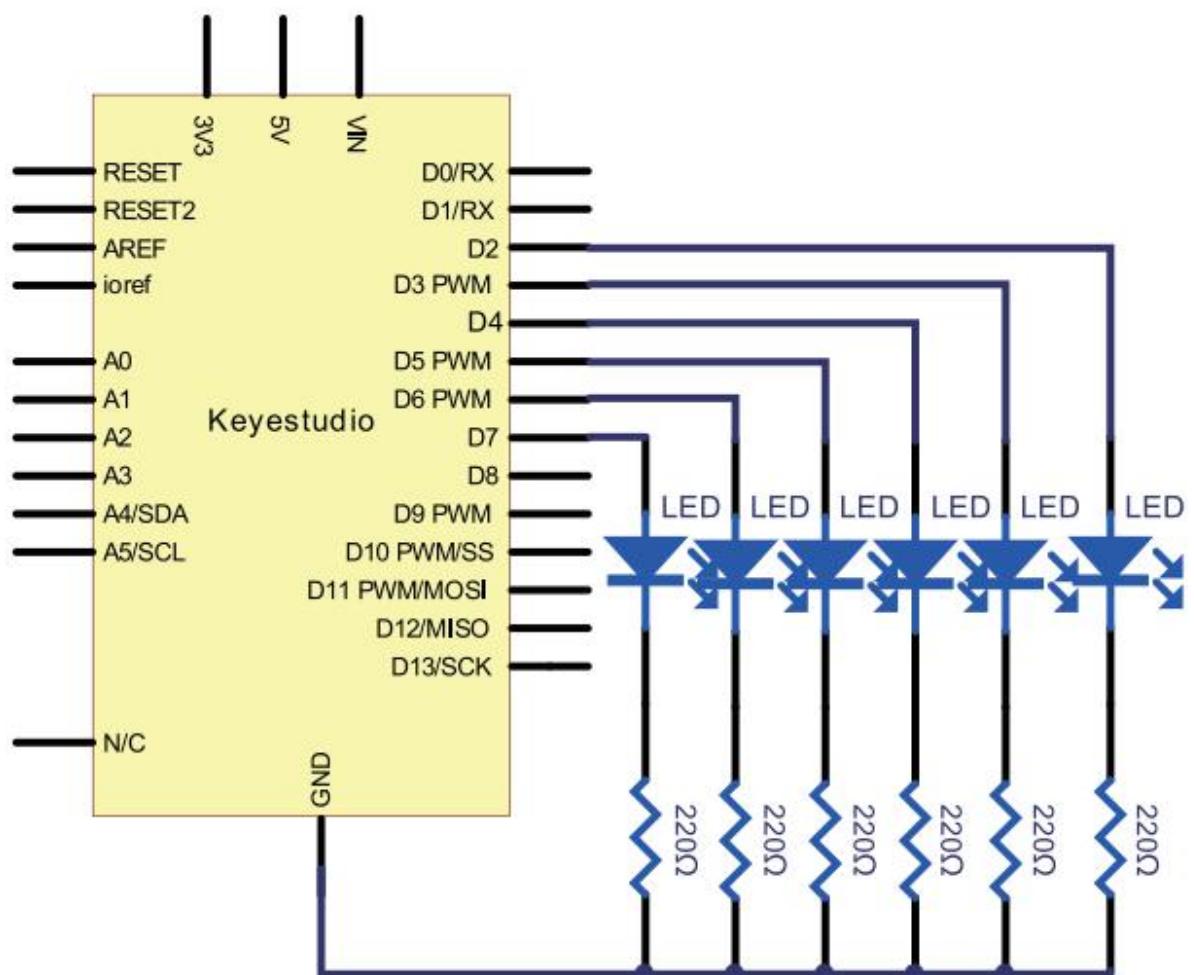
1. Introduction

We can see many billboards composed of colorful LEDs. They are constantly changing to form various effects. In this experiment, we compile a program to simulate chase effect.

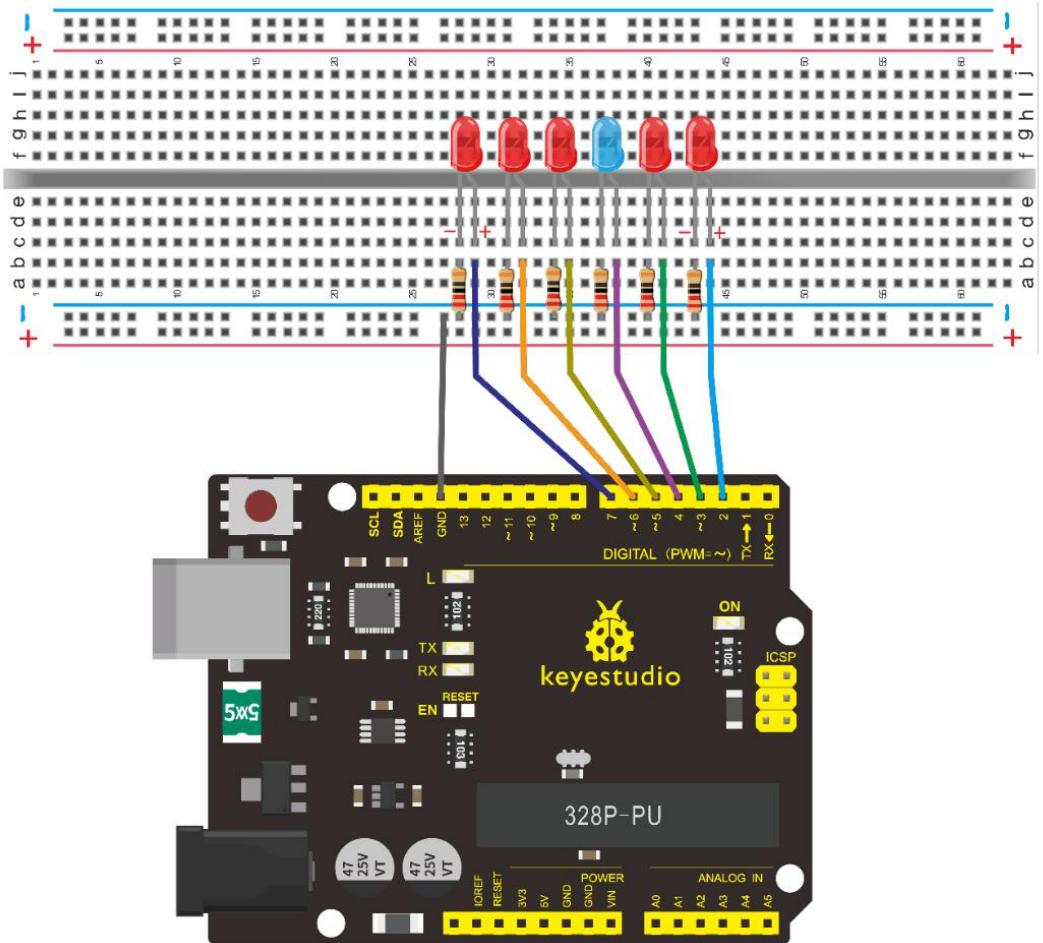
2.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- Red LED*6
- 220 Ω Resistor *6
- Breadboard Jumper Wire*12
- USB cable*1

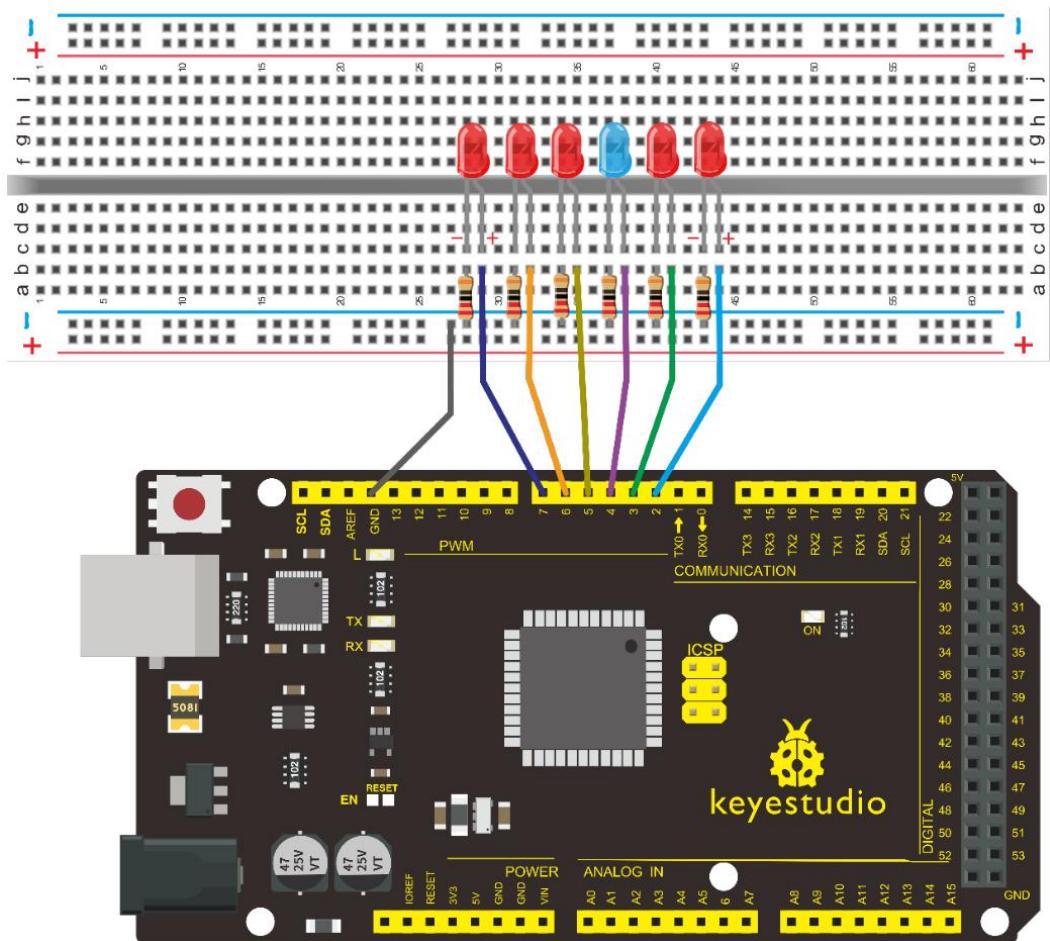
3.Circuit Connection



Connection for V4.0



Connection for 2560



4. Sample Code

```
/*
keyestudio super learning kit
```

Project 5

LED Chasing Effect

<http://www.keyestudio.com>

```
*/
```

```
int BASE = 2 ; // the I/O pin for the first LED
```

```
int NUM = 6; // number of LEDs
```

```

void setup()
{
    for (int i = BASE; i < BASE + NUM; i++)
    {
        pinMode(i, OUTPUT); // set I/O pins as output
    }
}

void loop()
{
    for (int i = BASE; i < BASE + NUM; i++)
    {
        digitalWrite(i, LOW); // set I/O pins as "low", turn off LEDs one
        by one.

        delay(200); // delay
    }

    for (int i = BASE; i < BASE + NUM; i++)
    {
        digitalWrite(i, HIGH); // set I/O pins as "high", turn on LEDs
        one by one

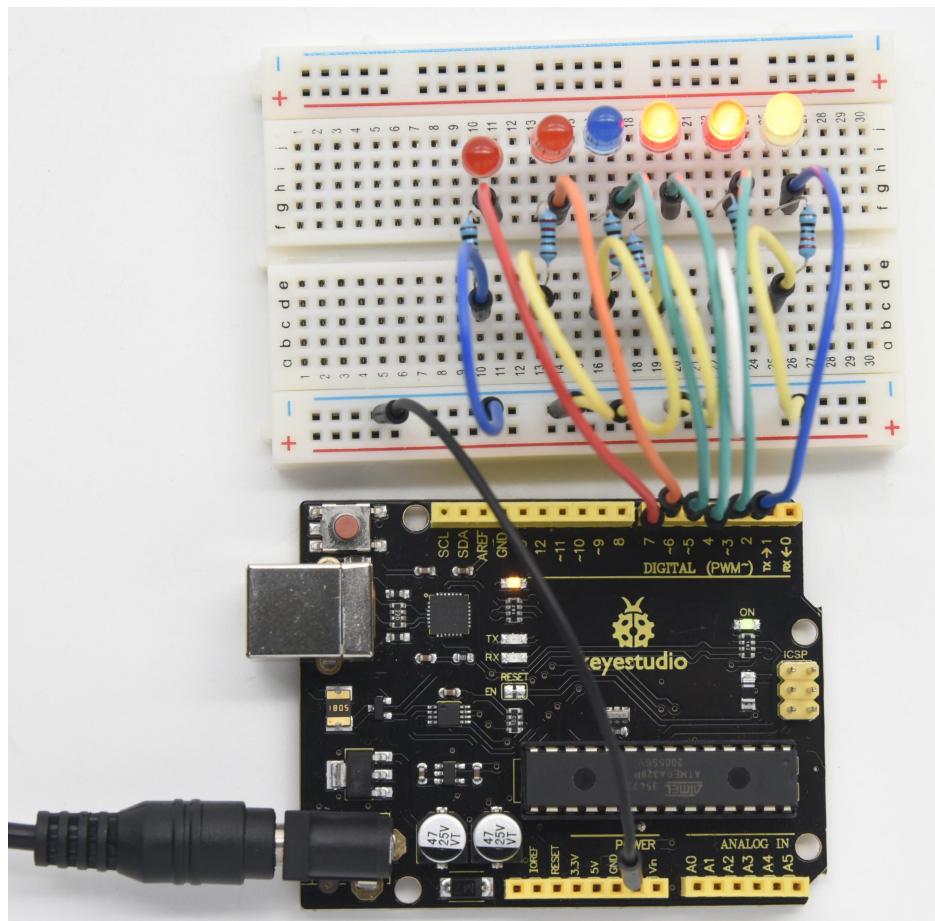
        delay(200); // delay
    }
}

```



5. Test Result

You can see the LEDs blink by sequence.



Project 6: Button-Controlled LED



1. Introduction

I/O port means interface for INPUT and OUTPUT. Up to now, we have only used the OUTPUT function.

In this experiment, we will try to use the INPUT function, which is to read the output value of device connecting to it.

We use 1 button and 1 LED using both input and output to give you a better understanding of the I/O function.

Button switch, familiar to most of us, is a switch value (digital value) component. When it's pressed, the circuit is in closed (conducting) state.

2. Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- Button switch*1
- Red M5 LED*1
- 220Ω Resistor*1
- $10K\Omega$ Resistor*1
- Breadboard*1
- Breadboard Jumper Wire*6
- USB cable*1

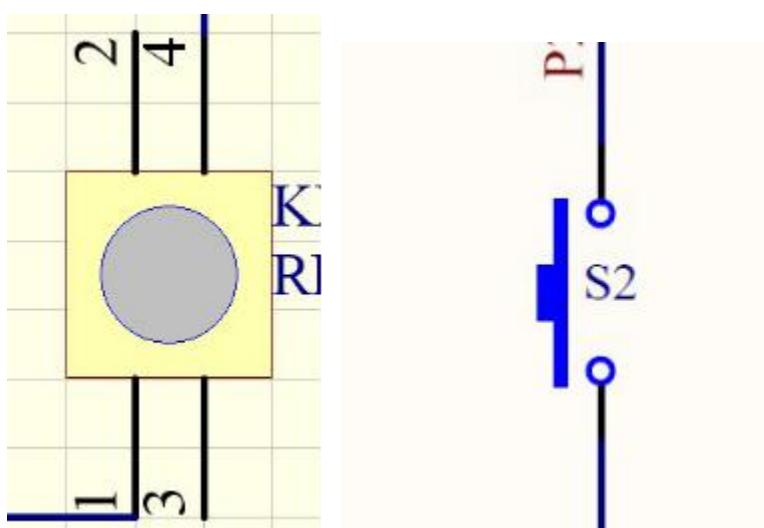
3. Little Knowledge

I believe that button switch is common and popular for people. It belongs to switch quantity(digital quantity)component. Composed of normally open contact and normally closed contact, its working principle is similar with ordinary switch.

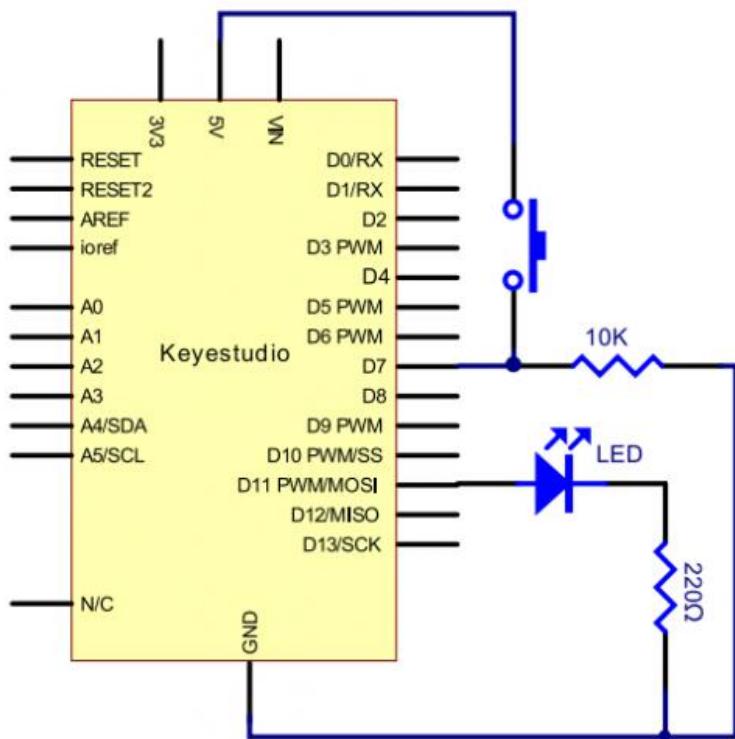
When the normally open contact bears pressure, the circuit is on state ; however, when this pressure disappears, the normally open contact goes back to initial state, that is, off state. The pressure is the act we switch the button.



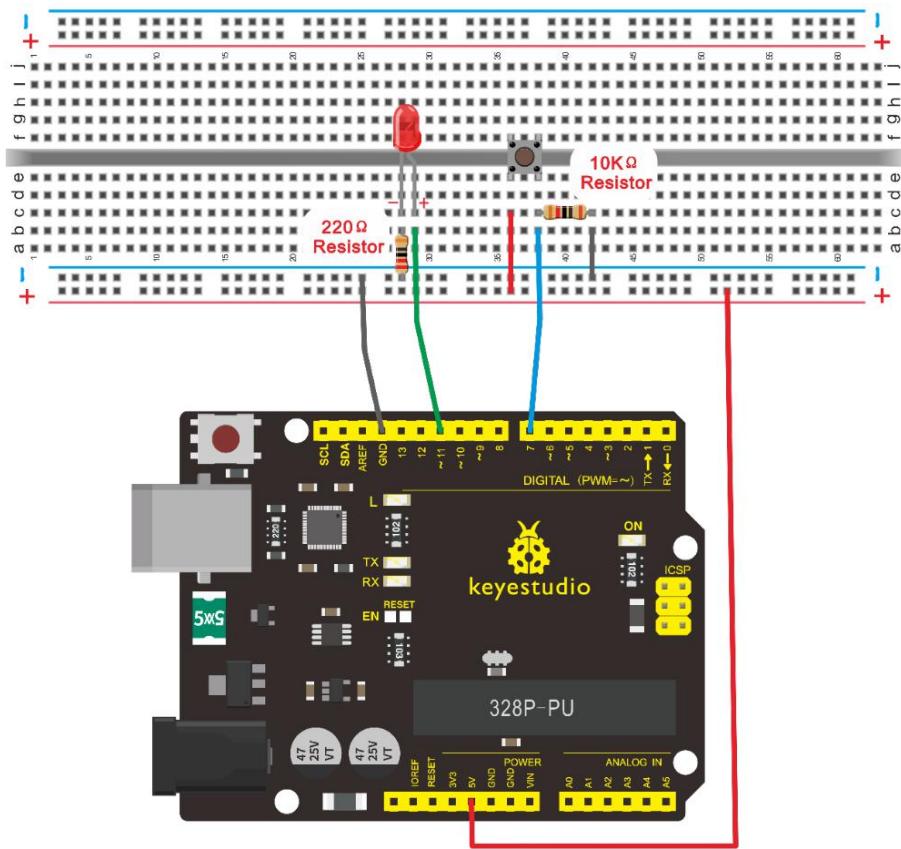
Schematic Diagrams:



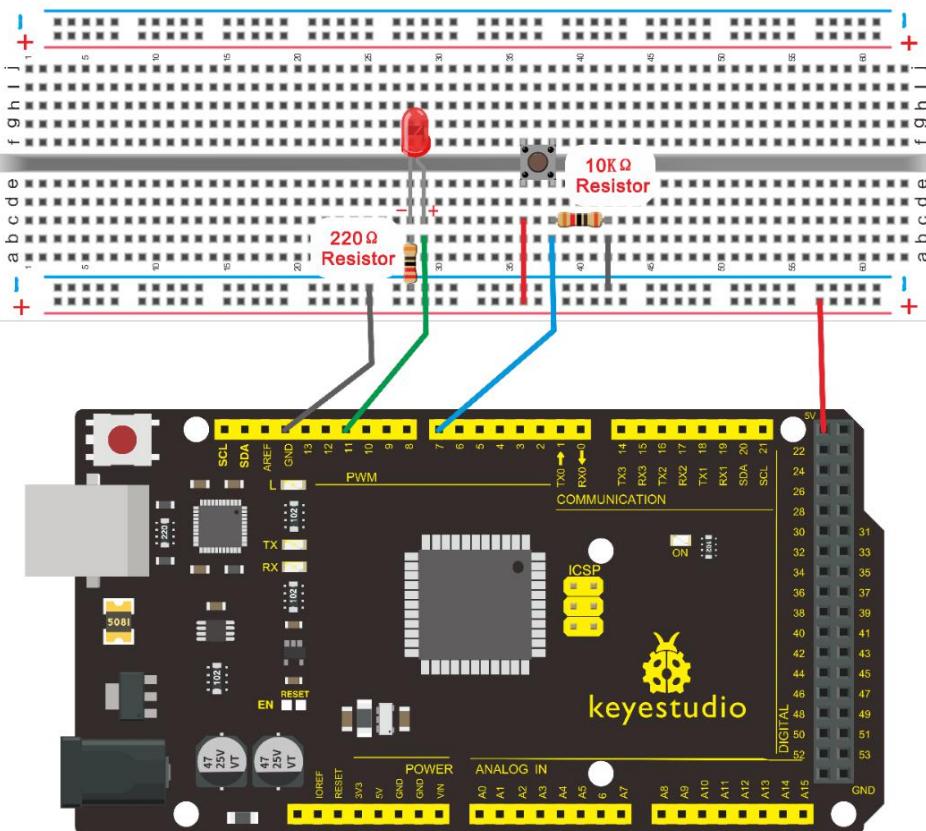
4. Circuit Connection



Connection for V4.0



Connection for 2560



5.Sample Code

Now, let's begin the compiling. When the button is pressed, the LED will be on. Based on the previous study, the coding should be easy for you.

In this program, we add a statement of judgment. Here, we use an **if ()** statement.

Arduino IDE is based on C language, so statements of C language such as while, switch etc. can certainly be used for Arduino program.

When we press the button, pin 7 will output high level. We can program pin 11 to output high level and turn on the LED.

When pin 7 outputs low level, pin 11 also outputs low level and the LED remains off.

```
/*
keyestudio super learning kit
Project 6
Button
http://www.keyestudio.com
*/
int ledpin=11;// initialize pin 11
int inpin=7;// initialize pin 7
```

```
int val;// define val

void setup()

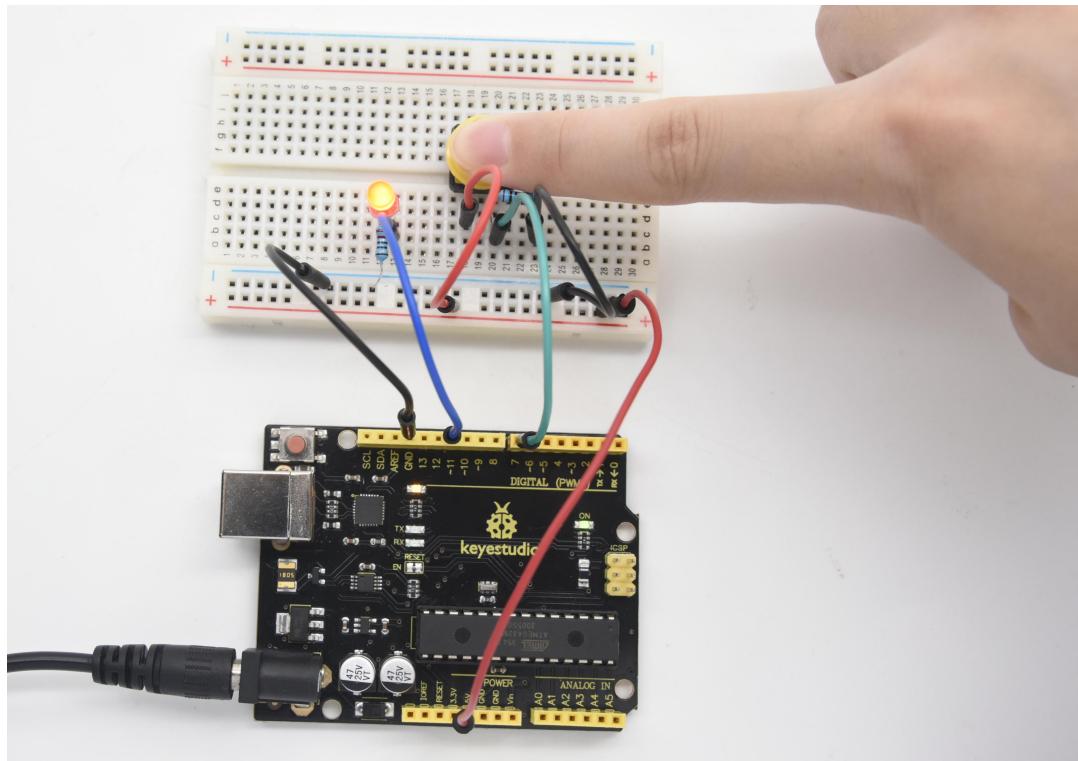
{
    pinMode(ledpin,OUTPUT);// set LED pin as "output"
    pinMode(inpin,INPUT);// set button pin as "input"
}

void loop()

{
    val=digitalRead(inpin);// read the level value of pin 7 and assign it to val
    if(val==LOW)// check if the button is pressed, if yes, turn on the LED
    {
        digitalWrite(ledpin,LOW);
    }
    else
    {
        digitalWrite(ledpin,HIGH);
    }
}
//////////
```

6. Test Result

When the button is pressed, LED is on, otherwise, LED remains off. In this way, the button controlled LED experiment is completed.



The simple principle of this experiment is widely used in a variety of circuit and electric appliances. You can easily come across it in your daily life. One typical example is when you press a certain key on your phone, the backlight will be on.

Project 7: Active Buzzer



1. Introduction

Active buzzer is widely used as a sound making element on computer, printer, alarm, electronic toy, telephone, timer and more. It has an inner vibration source. Simply connect it with 5V power supply, it can buzz continuously.

2. Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- Buzzer*1
- Breadboard*1
- Breadboard Jumper Wire*2
- USB cable*1

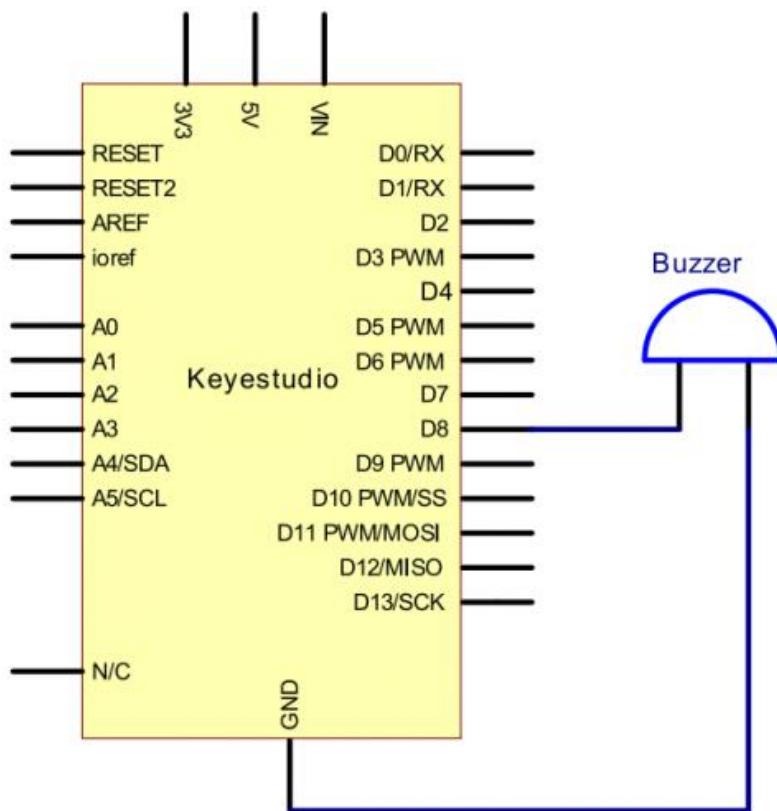
3. Little Knowledge

There are two kinds of buzzer, active buzzer and passive buzzer.

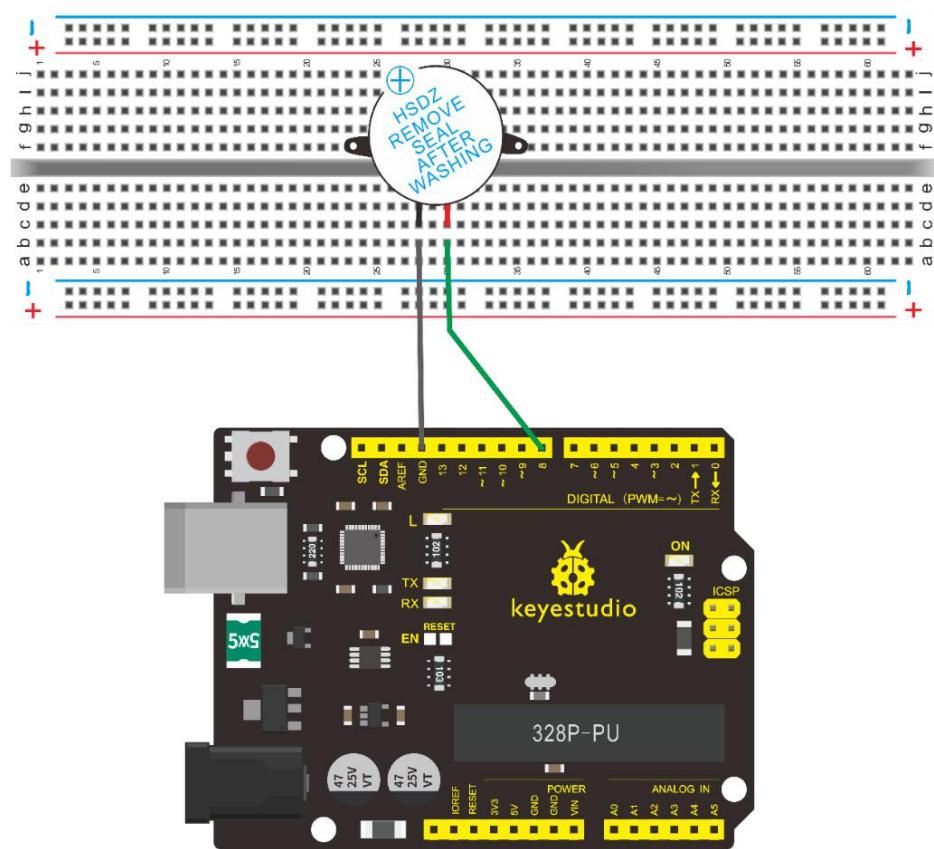
In this lesson, we will use Micro:bit to drive an active buzzer.

The active buzzer inside has a simple oscillator circuit which can convert constant direct current into a certain frequency pulse signal. Once active buzzer receives a high level, it will produce an audible beep.

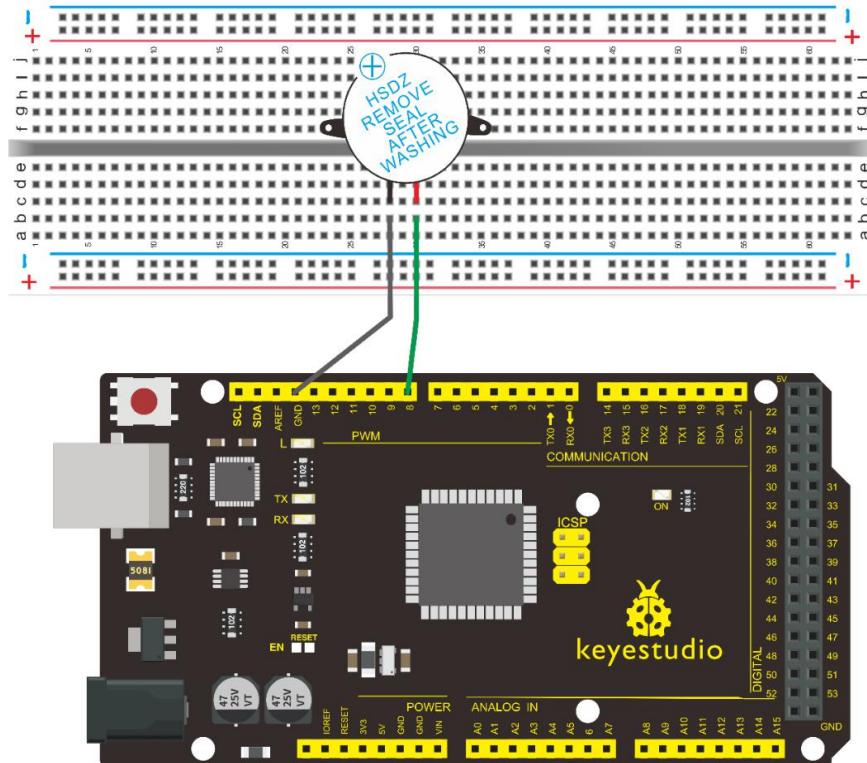
4. Circuit Connection



Connection for V4.0



Connection for 2560



When connecting the circuit, pay attention to the positive and negative poles of the buzzer. In the photo, you can see there are red and black lines. When the circuit is finished, you can begin the programming.

5. Sample Code

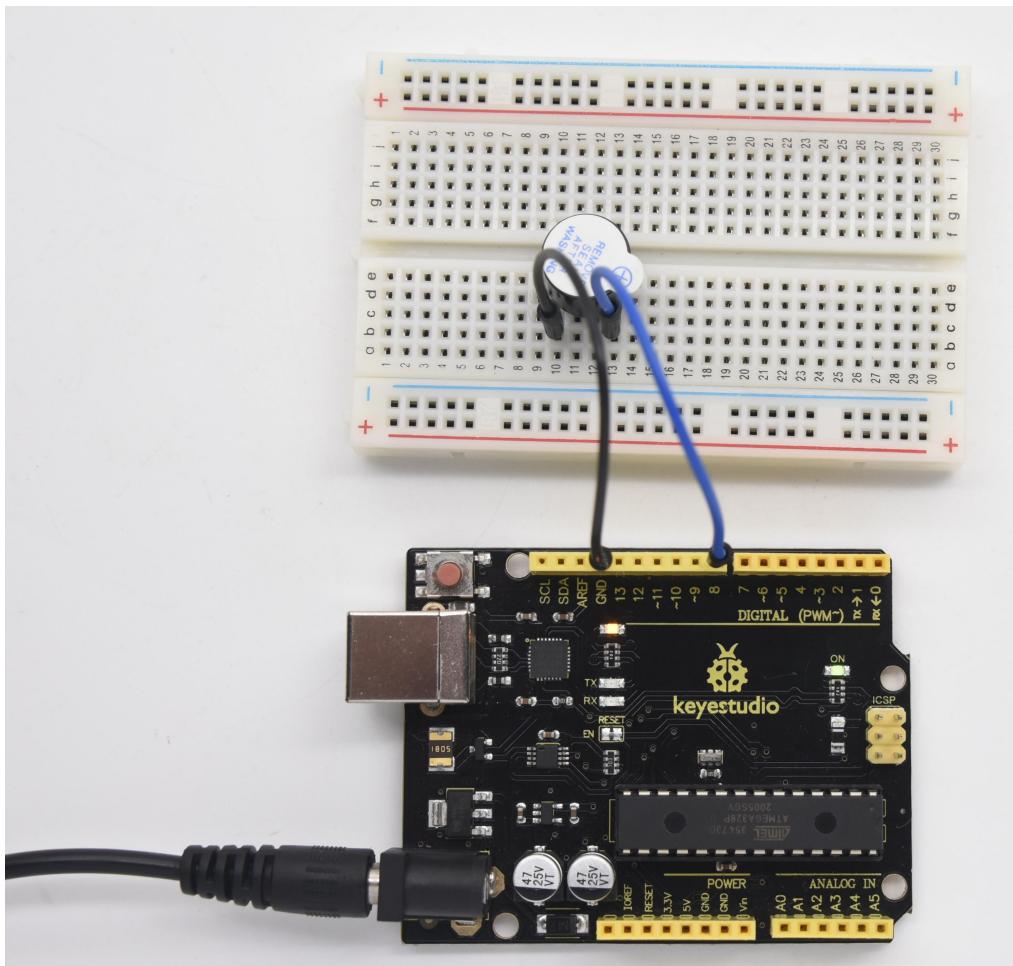
Program is simple. You control the buzzer by outputting high/low level.

```
/*
keyestudio super learning kit
Project 7
Active Buzzer
http://www.keyestudio.com
```

```
*/  
int buzzer=8;// initialize digital IO pin that controls the buzzer  
void setup()  
{  
    pinMode(buzzer,OUTPUT);// set pin mode as "output"  
}  
void loop()  
{  
    digitalWrite(buzzer, HIGH); // produce sound  
}  
//////////
```

6. Test Result

After uploading the program, the buzzer experiment is completed. You can see the buzzer is ringing.



Project 8: Passive Buzzer



1. Introduction

We can use Arduino to make many interactive works. The most commonly used one is acoustic-optic display. All the

previous experiment has something to do with LED. However, the circuit in this experiment can produce sound. Normally, the experiment is done with a buzzer but not a speaker while buzzer is more simpler and easier to use.

The buzzer we introduced here is a passive buzzer. It cannot be actuated by itself, but by external pulse frequencies. Different frequency produces different sound. We can use Arduino to code the melody of a song, which is quite fun and simple.

2. Hardware Required

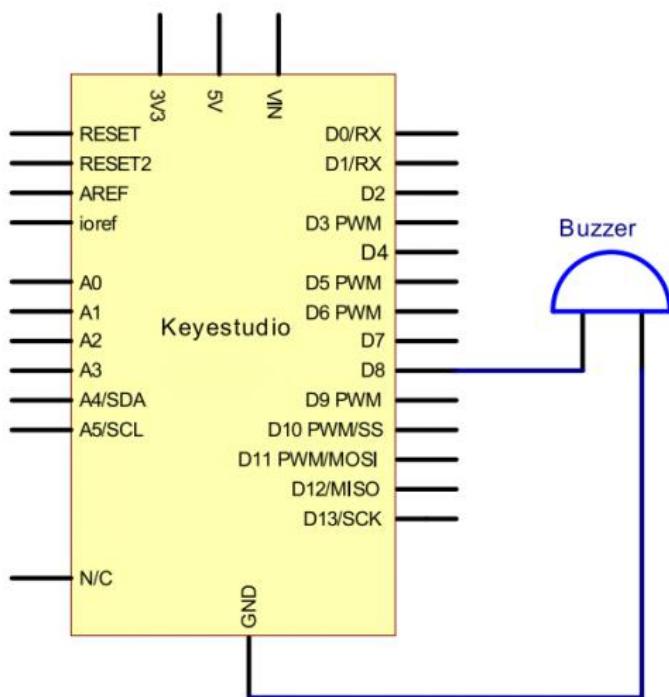
- V4.0 Board or MEGA 2650 Board*1
- Passive Buzzer*1
- Breadboard*1
- Breadboard Jumper Wire*2
- USB cable*1

3. Little knowledge

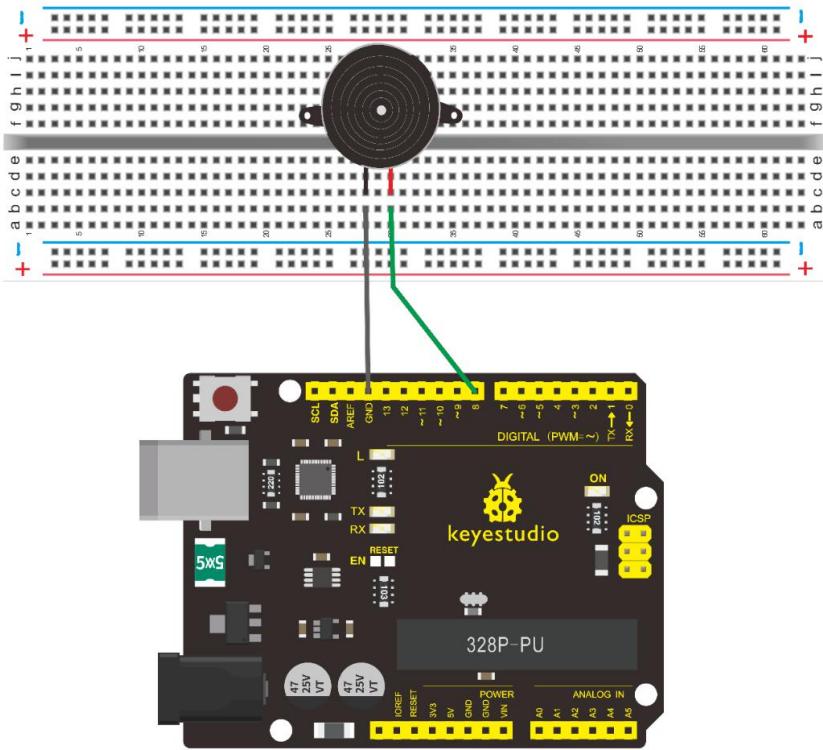
Passive buzzer is an integrated electronic buzzer without vibration source inside. It must be driven by 2K-5K square wave instead of direct current signals. There is little difference between the two buzzers, but when the pins of the two buzzers are placed up, the passive buzzer comes with green

circuit board, and the one sealed with vinyl is an active buzzer.

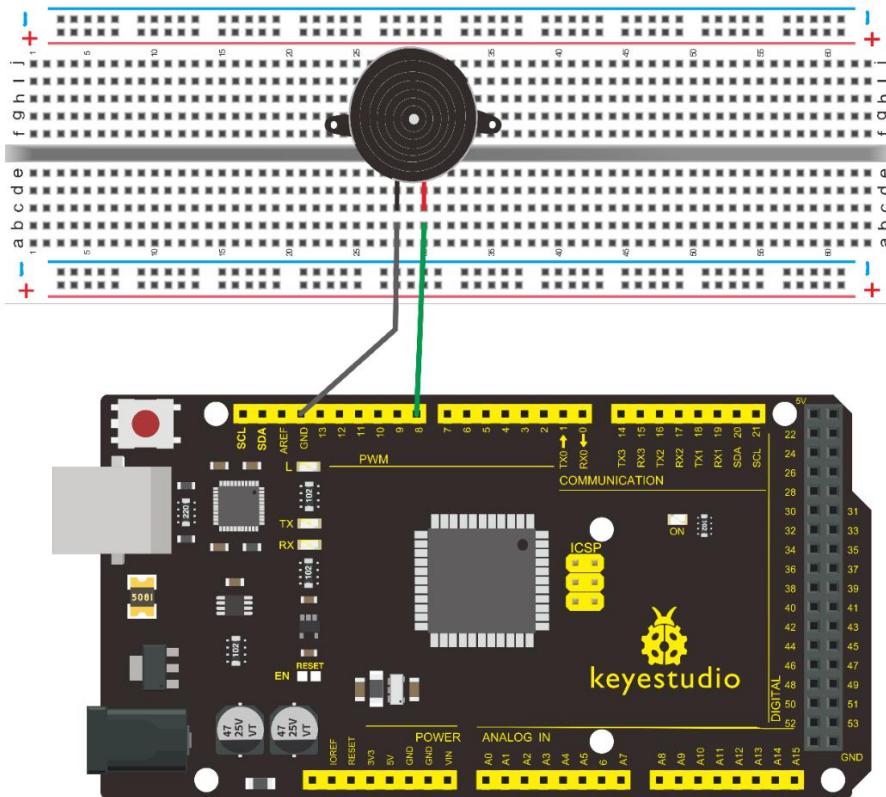
4. Circuit Connection



Connection for V4.0



Connection for 2560



5. Sample Code

```
/*
keyestudio super learning kit
Project 8
Passive Buzzer
http://www.keyestudio.com
*/
int buzzer=8;// select digital IO pin for the buzzer
void setup()
{
pinMode(buzzer,OUTPUT);// set digital IO pin pattern, OUTPUT to be
output
}
void loop()
{
unsigned char i,j;//define variable
while(1)
{
for(i=0;i<80;i++)// output a frequency sound
{
digitalWrite(buzzer,HIGH);// sound
delay(1);//delay1ms
digitalWrite(buzzer,LOW);//not sound
delay(1);//ms delay
}
for(i=0;i<100;i++)// output a frequency sound
```

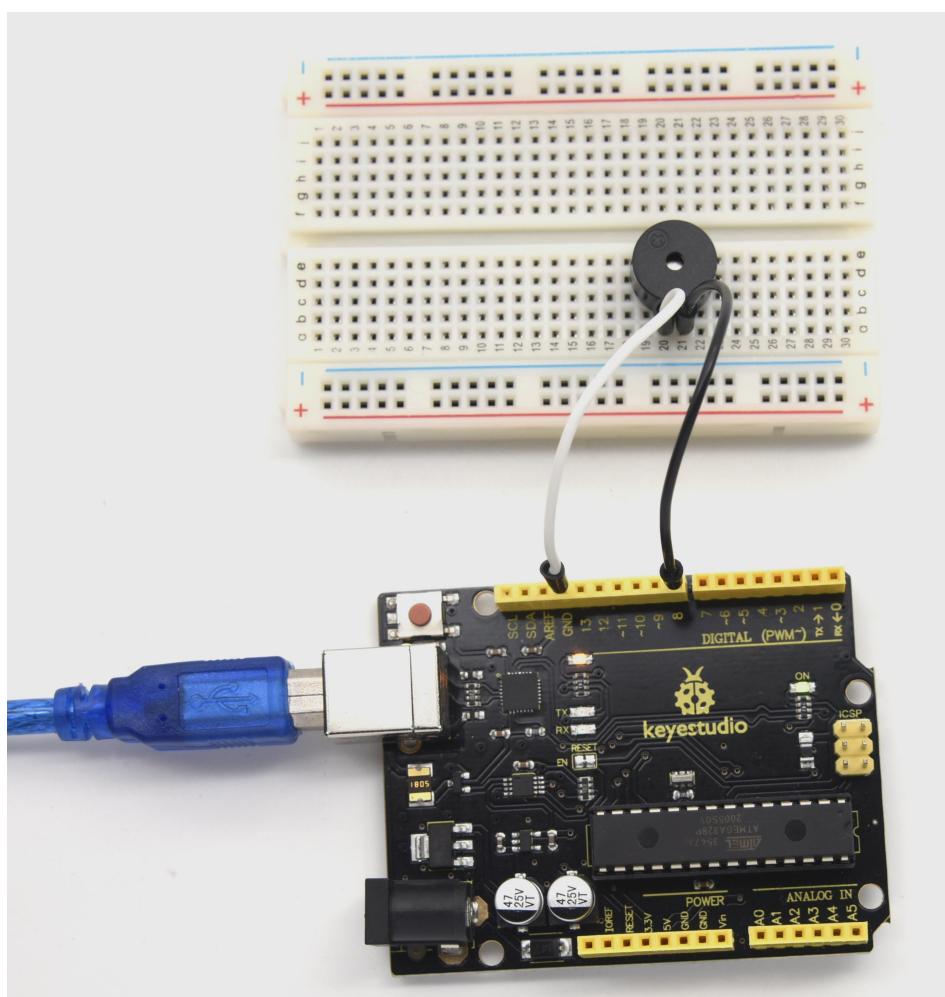
```

{ digitalWrite(buzzer,HIGH);// sound
delay(2);//2ms delay
digitalWrite(buzzer,LOW);//not sound
delay(2);//2ms delay
}}}

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

6. Test Result

After uploading the program, buzzer experiment is finished, you can hear the buzzer sound.



Project 9: RGB LED



1. Introduction

The RGB color mode is a color standard in the industry. It obtains various colors by changing the three color channels of red (R), green (G), and blue (B) and integrating them. RGB denotes the three colors of red, green and blue.

In this project, we use Arduino to mix these three colors in equal amounts to produce white light.

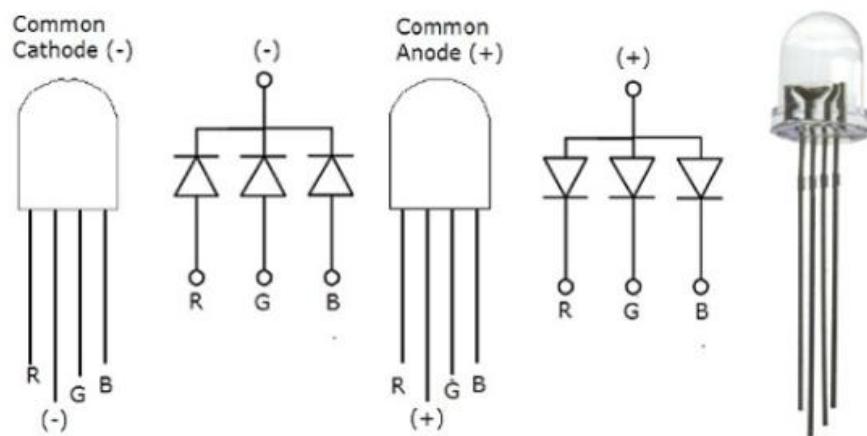
2. Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- USB Cable * 1
- RGB LED * 1
- Resistor *3
- Breadboard jumper wire*5

3. Little Knowledge

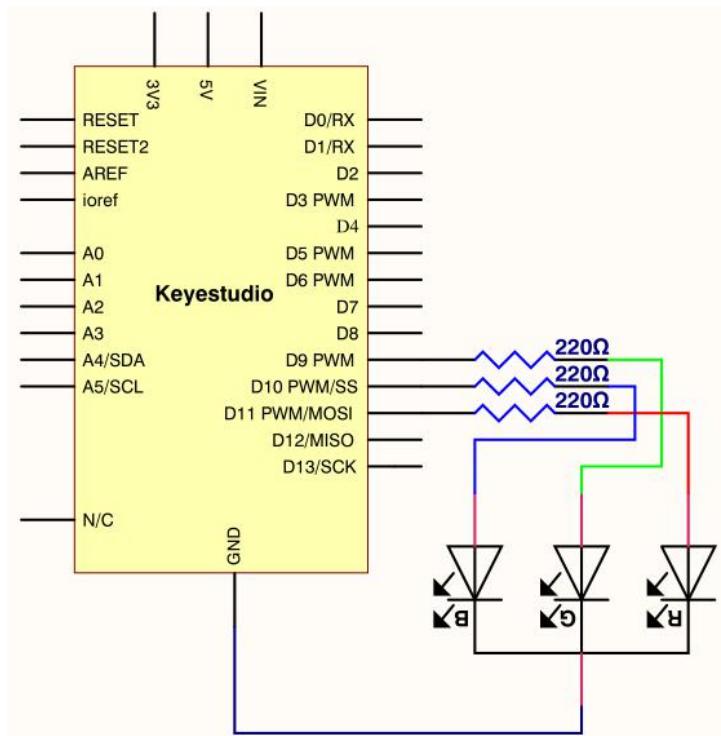
The monitors mostly adopt the RGB color standard, and all the colors on the computer screen are composed of the three colors of red, green and blue mixed in different proportions.

RGB is inclusive of common cathode RGB and common anode RGB.

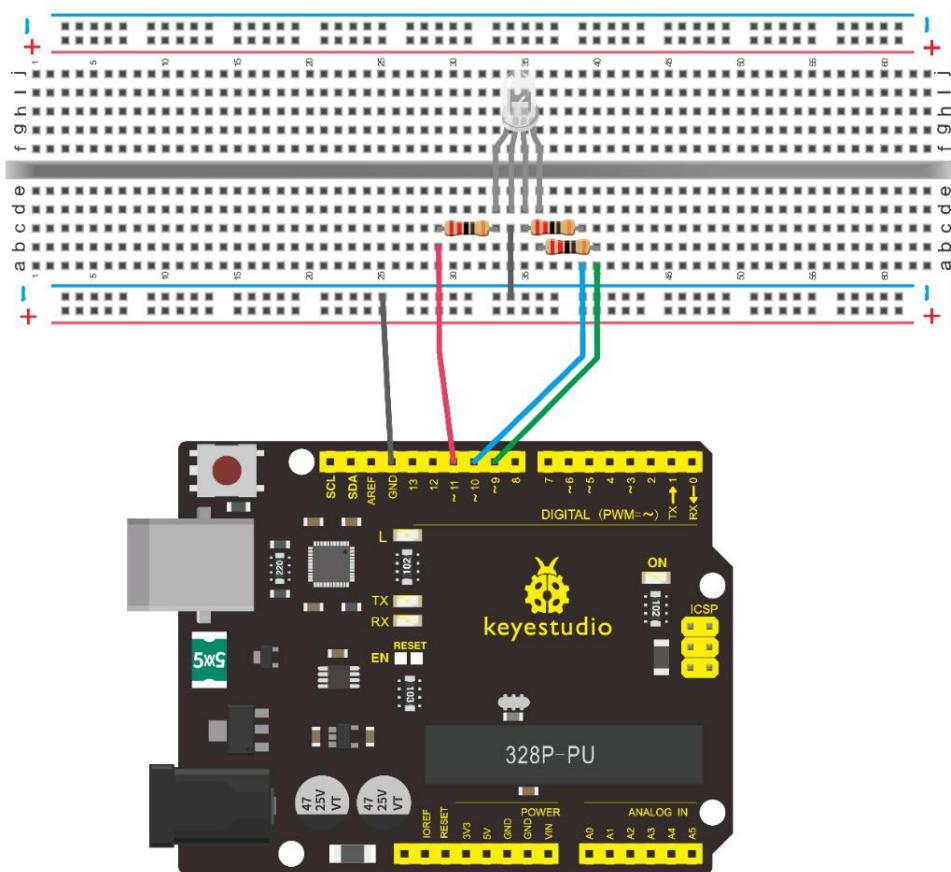


And we could adjust the LED brightness by PWM

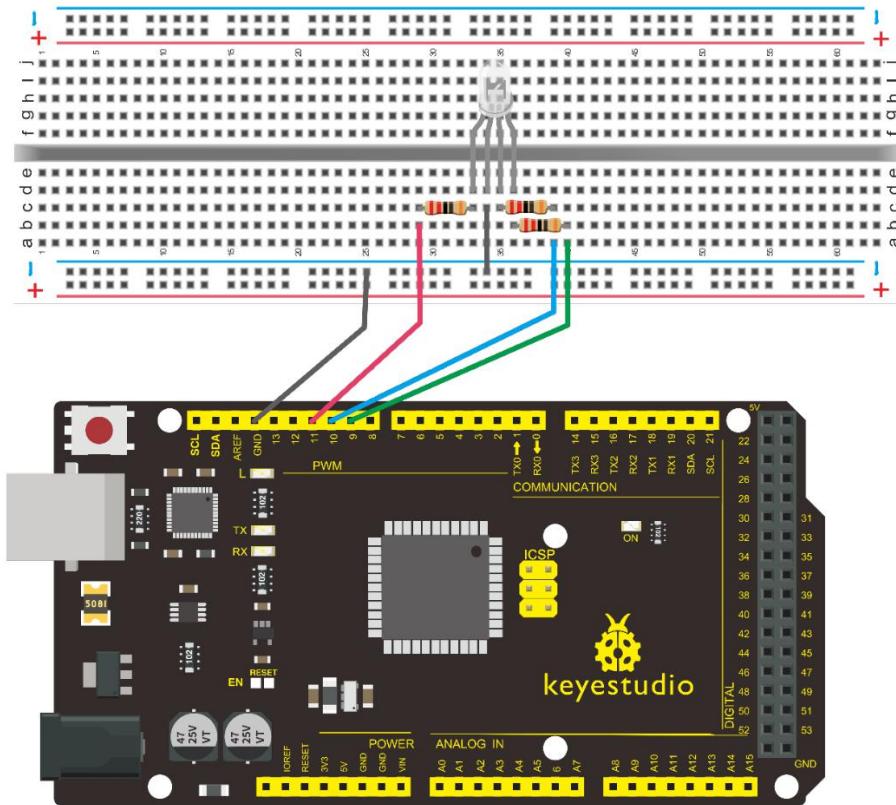
4.Circuit Connection



Connection for V4.0



Connection for 2560



5. Sample Code

```
/*
keyestudio super learning kit
Project 9
RGB
http://www.keyestudio.com
*/
int redpin = 11; //select the pin for the red LED
int bluepin =10; // select the pin for the blue LED
```

```
int greenpin =9;// select the pin for the green LED  
  
int val;  
  
void setup() {  
    pinMode(redpin, OUTPUT);  
    pinMode(bluepin, OUTPUT);  
    pinMode(greenpin, OUTPUT);  
    Serial.begin(9600);  
}  
  
void loop()  
{  
    for(val=255; val>0; val--)  
    {  
        analogWrite(11, val);  
        analogWrite(10, 255-val);  
        analogWrite(9, 128-val);  
        delay(1);  
    }  
    for(val=0; val<255; val++)  
    {  
        analogWrite(11, val);  
        analogWrite(10, 255-val);  
        analogWrite(9, 128-val);  
    }  
}
```

```

delay(1);

}

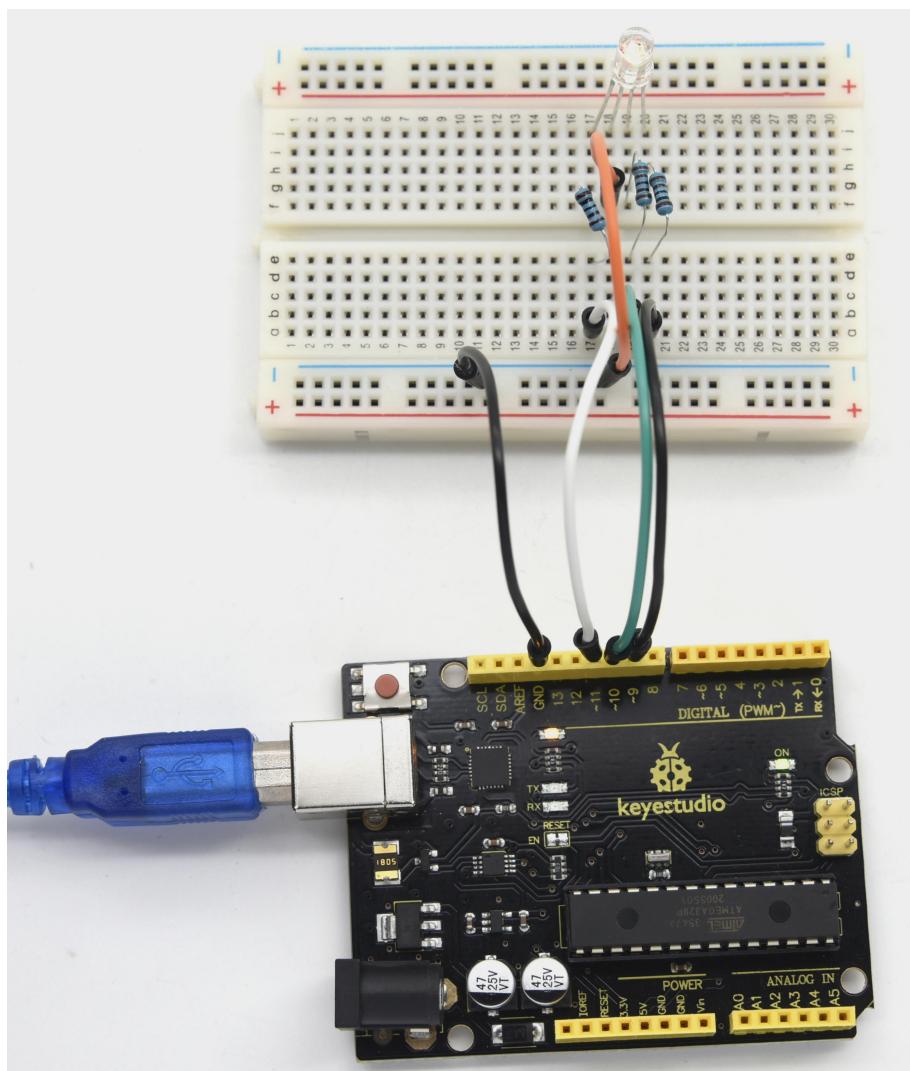
Serial.println(val, DEC);

}

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

6. Test Result

Directly copy the above code into arduino IDE, and click upload, wait for a few seconds, you can see a full-color LED.



Project 10: Photo Resistor



1. Introduction

After completing all the previous experiments, you may acquire some basic understanding and knowledge about Arduino application. We have introduced digital input and output, analog input and PWM.

Now, let's begin the learning of sensor applications.

Photo Resistor (Photovaristor) is a resistor whose resistance varies from different incident light strength. It's based on the photoelectric effect of semiconductor. If the incident light is intense, its resistance reduces; if the incident light is weak, the resistance increases.

2. Hardware Required

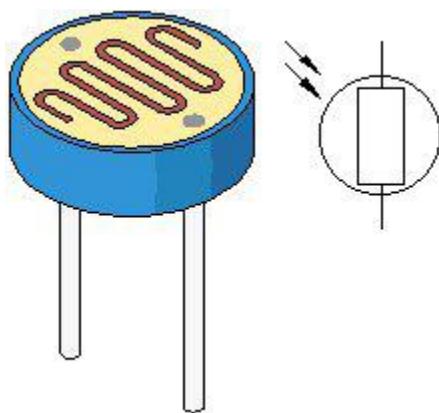
- V4.0 Board or MEGA 2650 Board*1
- Photo Resistor*1
- Red M5 LED*1

- 10K Ω Resistor*1
- 220 Ω Resistor*1
- Breadboard*1
- Breadboard Jumper Wire*5
- USB cable*1

3. Little Knowledge

Photovaristor is commonly applied in the measurement of light, light control and photovoltaic conversion (convert the change of light into the change of electricity).

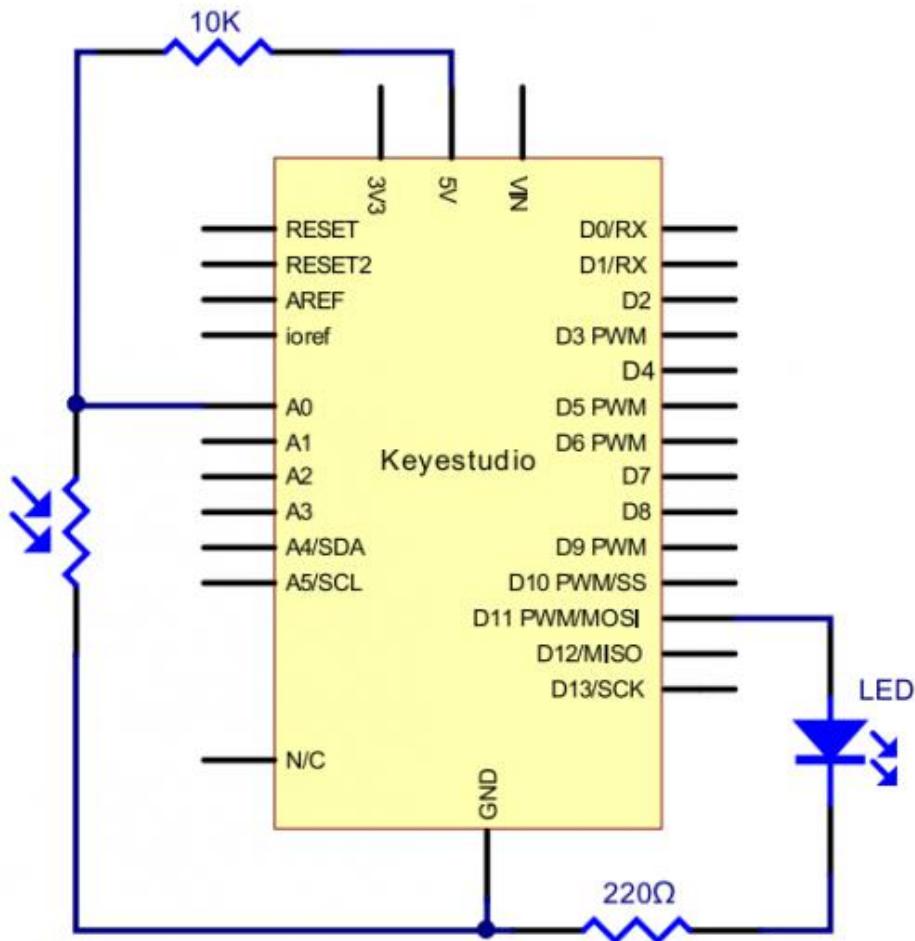
Photo resistor is also being widely applied to various light control circuit, such as light control and adjustment, optical switches, etc.



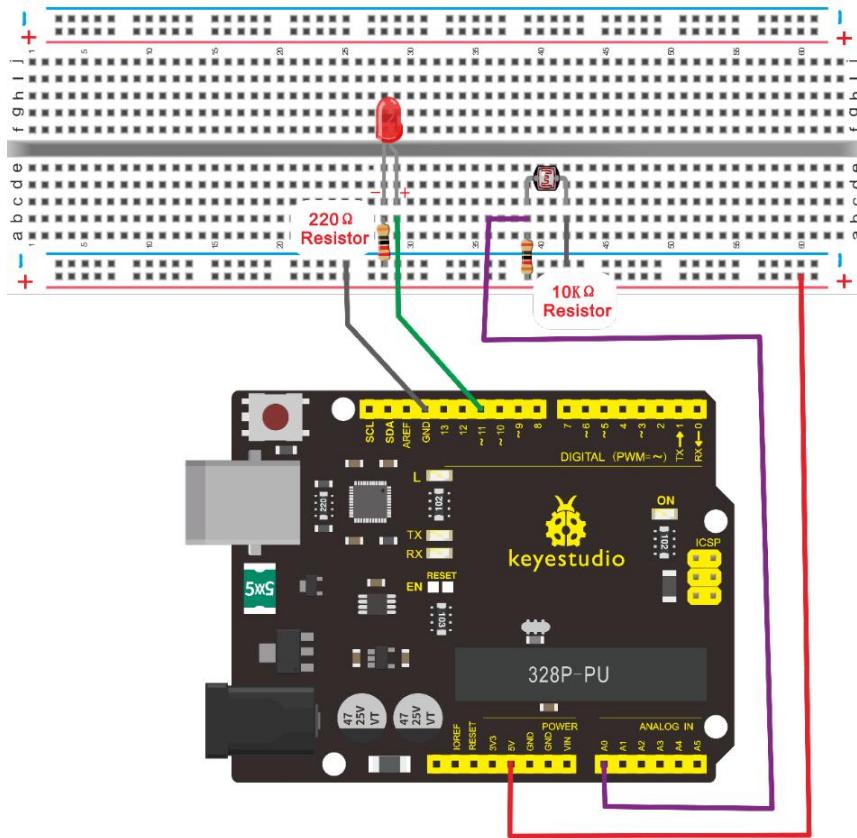
We will start with a relatively simple experiment regarding to photovaristor application.

Photovaristor is an element that can change its resistance as light strength changes. So need to read the analog value. You can refer to the PWM experiment, replacing the potentiometer with photovaristor. When there is change in light strength, it will make corresponding change on the LED.

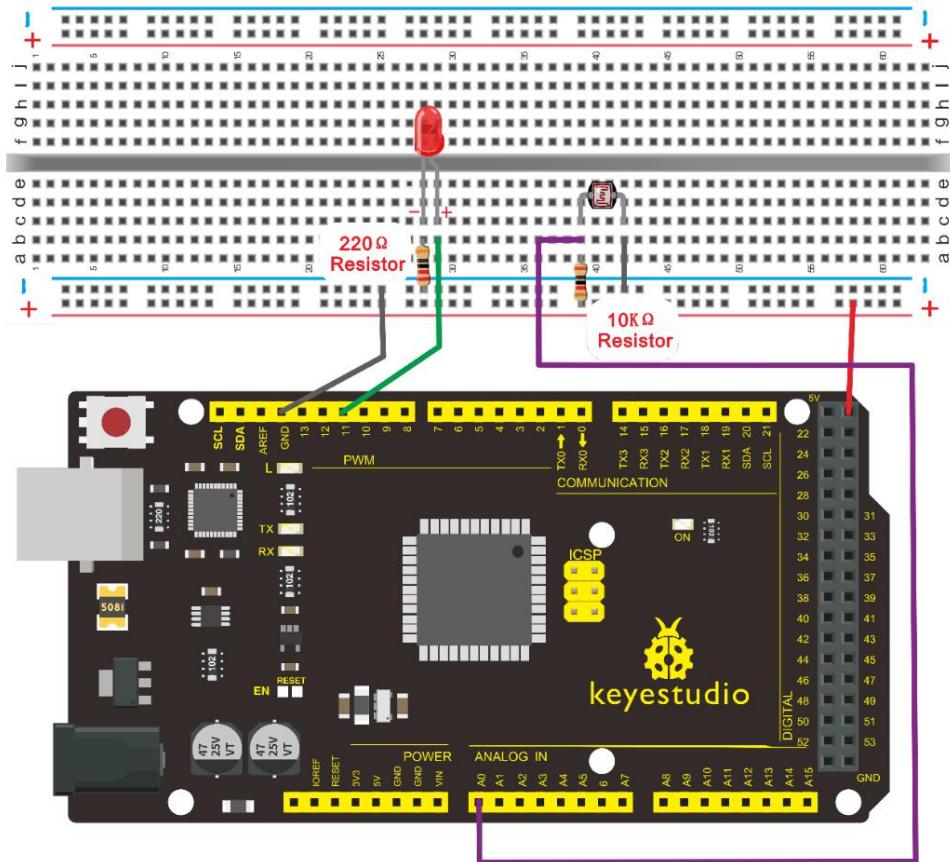
4.Circuit Connection



Connection for V4.0



Connection for 2560



5.Sample Code

After wiring, let's begin the program compiling. The program is similar to the PWM.

For change detail, please refer to the Sample Code below.

```
/*
keyestudio super learning kit
Project 10
Photo Resistor
http://www.keyestudio.com
*/
int potpin=0;// initialize analog pin 0, connected with
photovaristor
int ledpin=11;// initialize digital pin 11,
int val=0;// initialize variable va
void setup()
{
pinMode(ledpin,OUTPUT);// set digital pin 11 as "output"
Serial.begin(9600);// set baud rate at "9600"
}
void loop()
{
val=analogRead(potpin);// read the value of the sensor and
```

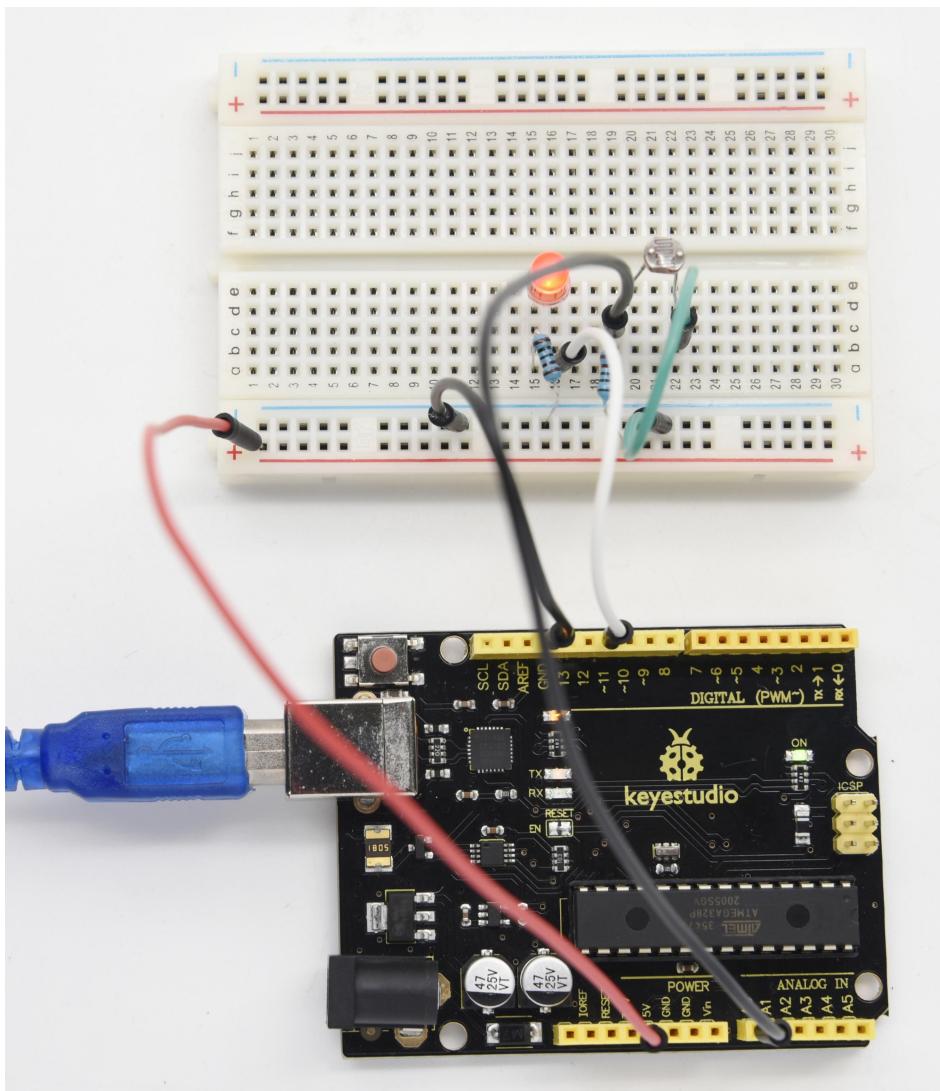
assign it to val

```
Serial.println(val);// display the value of val  
analogWrite(ledpin,val/4);// set up brightness (maximum  
value 255)  
delay(10);// wait for 0.01  
}  
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

6. Test Result

After downloading the program, you can change the light strength around the photovaristor, and see the corresponding brightness change of the LED.

Photovaristors has various applications in our everyday. You can make other interesting interactive projects based on this one.



Project 11: Flame Sensor



1. Introduction

Flame sensor (infrared receiving triode) is specially used for robots to find the fire source. This sensor is of high sensitivity to flame.

2. Hardware Required

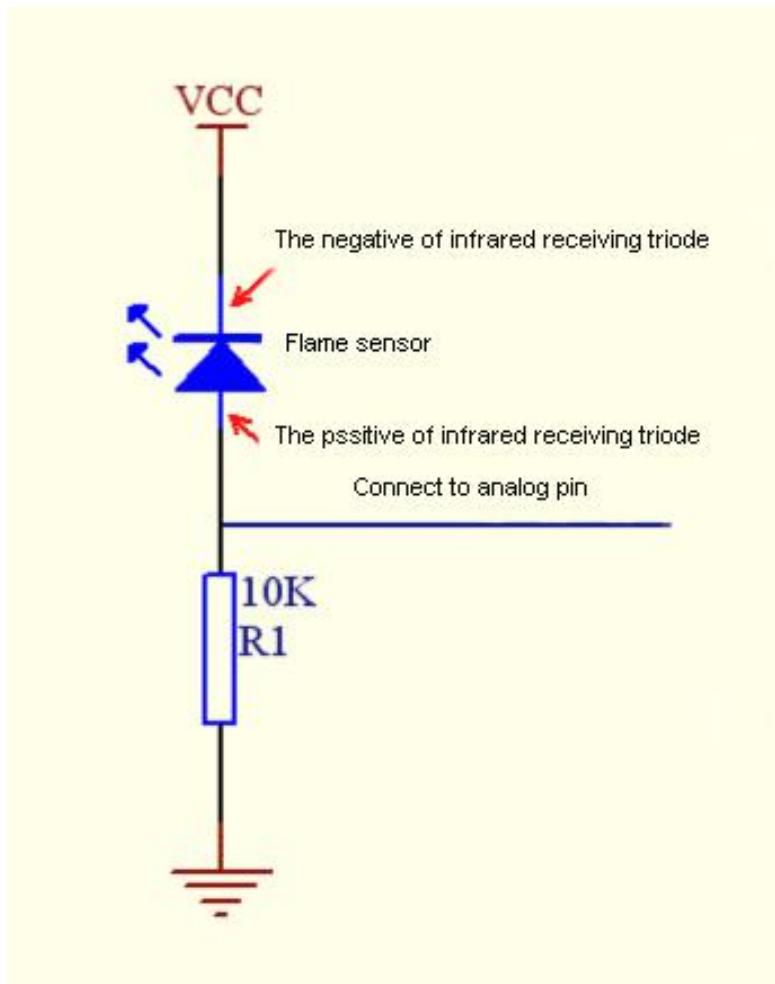
- V4.0 Board or MEGA 2650 Board*1
- Flame Sensor *1
- Buzzer *1
- 10K Resistor *1
- Breadboard Jumper Wire*6
- USB cable*1

3. Little Knowledge

Flame sensor is based on the principle that infrared ray is highly sensitive to flame. It has an infrared receiving tube specially designed to detect fire, and then to convert the flame brightness into fluctuating level signal. The signals are then input into the central processor and be dealt with accordingly.

The shorter lead of the receiving triode is for negative, the other one for positive. Connect negative to 5V pin, positive to resistor; connect the other end of the resistor to GND,

connect one end of a jumper wire to a clip which is electrically connected to sensor positive, the other end to analog pin. As shown below



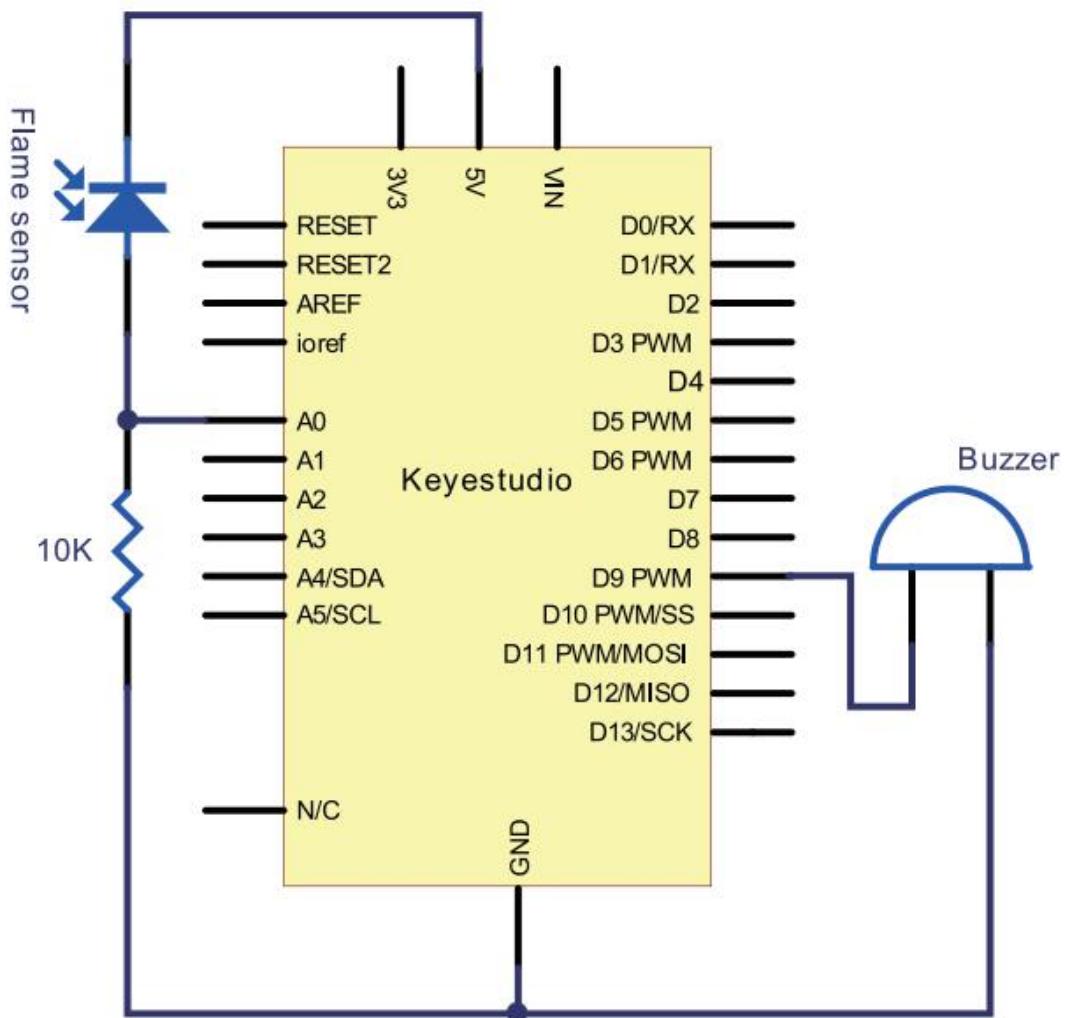
4.Experiment Principle

When it's approaching a fire, the voltage value read from the analog port will differ. If you use a multimeter, you can see that when there is no fire approaching, the voltage it reads is

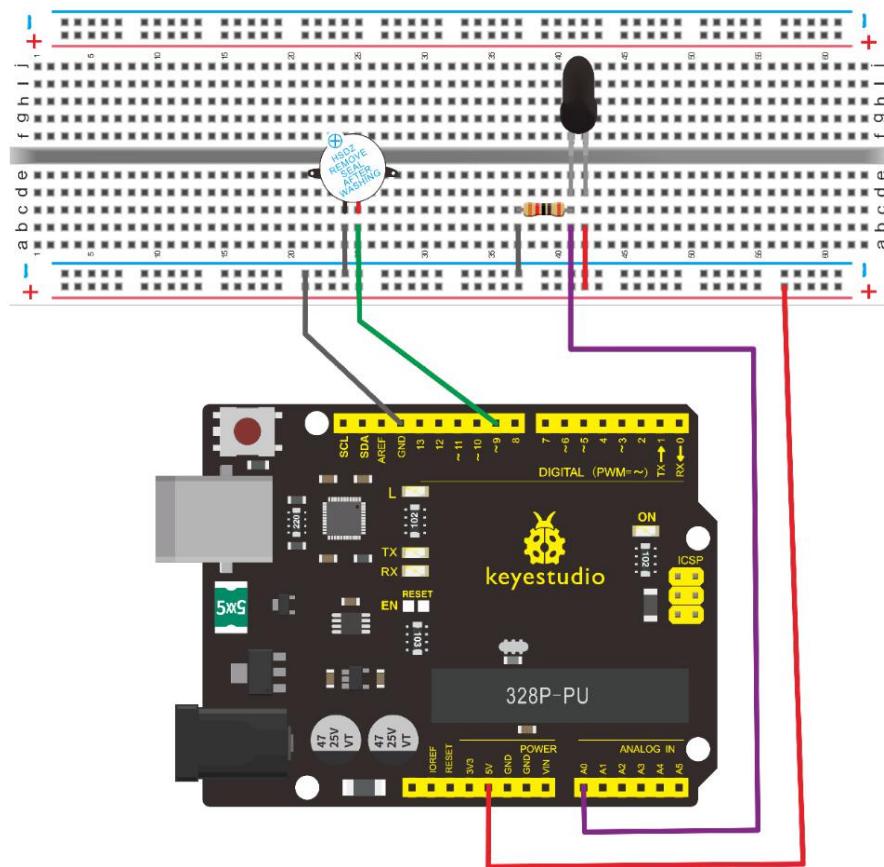
around 0.3V; when there is fire approaching, the voltage it reads is around 1.0V. The nearer the fire is, the higher the voltage is.

So in the beginning of the program, you can initialize voltage value i (no fire value); Then, continuously read the analog voltage value j and obtain difference value $k=j-i$; compare k with 0.6V (123 in binary) to determine whether there is a fire approaching or not; if yes, the buzzer will buzz.

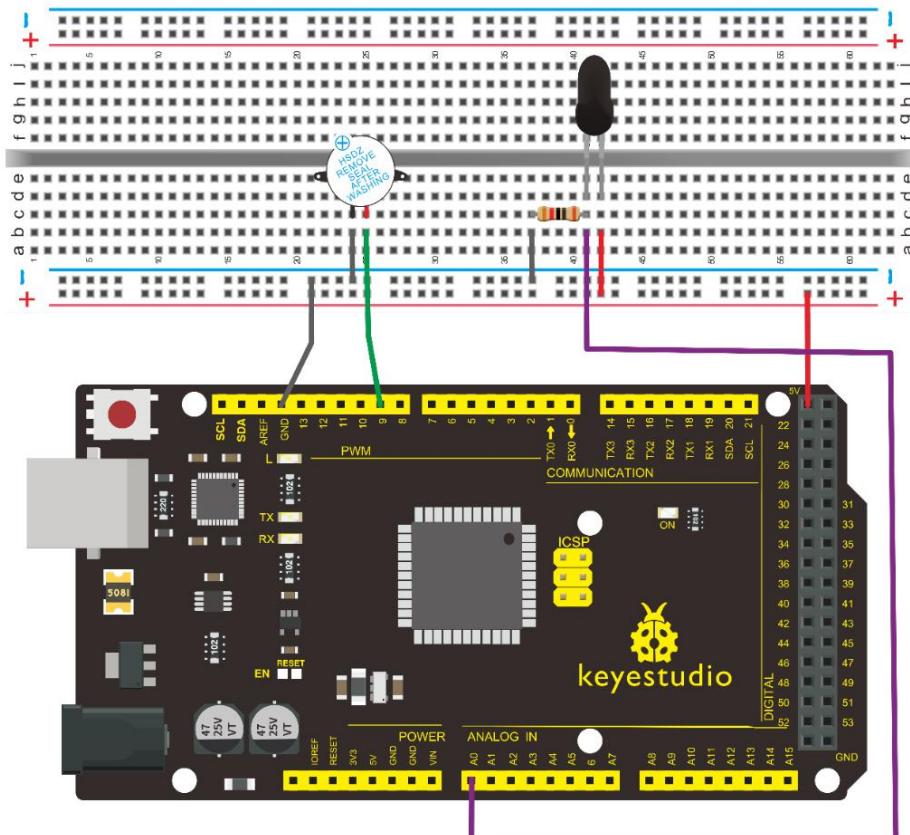
5.Circuit Connection



Connection for V4.0



Connection for MEGA 2560



6.Sample Code

```
/*
keyestudio super learning kit
Project 11
Flame
http://www.keyestudio.com
*/
int flame=0;// select analog pin 0 for the sensor
int Beep=9;// select digital pin 9 for the buzzer
int val=0;// initialize variable
void setup()
{
    pinMode(Beep,OUTPUT);// set LED pin as "output"
    pinMode(flame,INPUT);// set buzzer pin as "input"
    Serial.begin(9600);// set baud rate at "9600"
}
void loop()
{
    val=analogRead(flame);// read the analog value of the sensor
    Serial.println(val);// output and display the analog value
    if(val>=600)// when the analog value is larger than 600, the buzzer
        will buzz
}
```

```

{

digitalWrite(Beep,HIGH);

}else

{

digitalWrite(Beep,LOW);

}

delay(500);

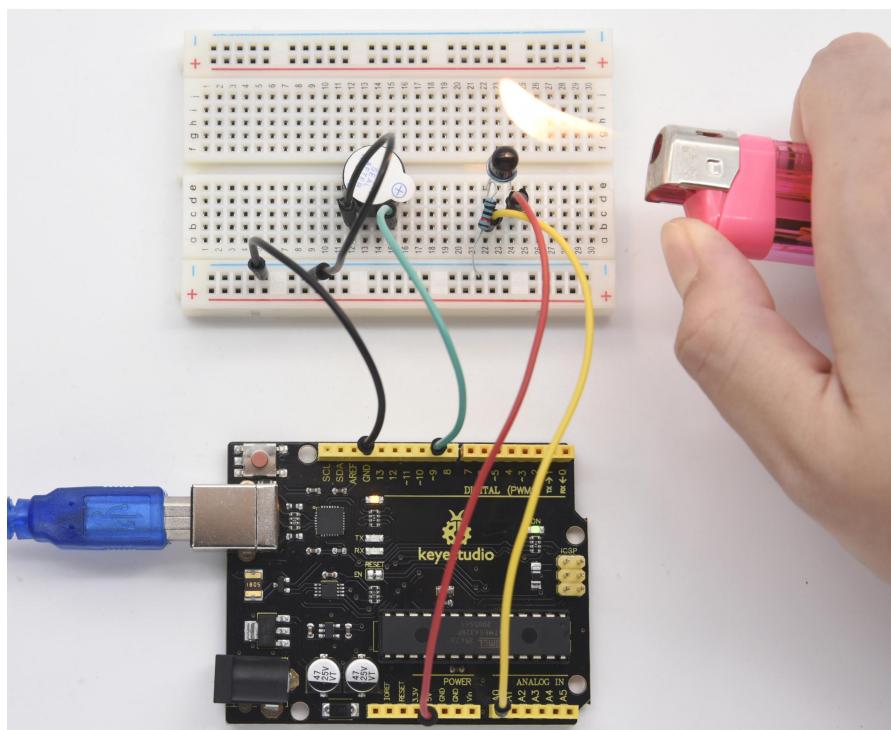
}

///////////////////////////////

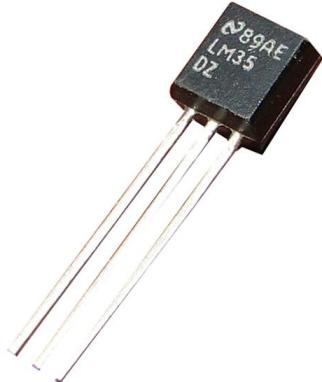
```

7. Test Result

This program can simulate an alarm when there is a fire. Everything is normal when there is no fire; when there is fire, the alarm will be set off immediately.



Project 12: LM35 Temperature Sensor

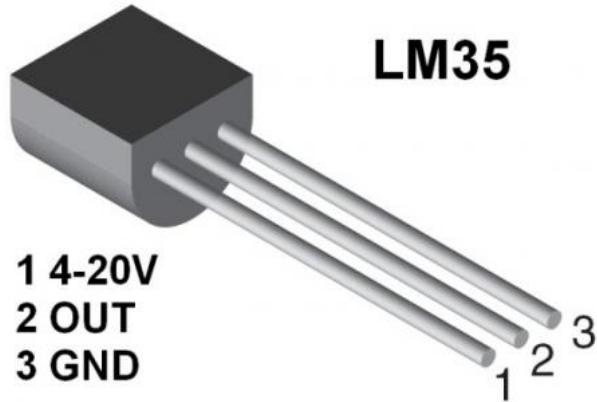


1. Introduction

LM35 is a common and easy-to-use temperature sensor. It does not require other hardware. You just need an analog port to make it work. The difficulty lies in compiling the code to convert the analog value it reads into Celsius temperature. In this project, we will guide you how to use LM35 temperature sensor.

2. Working Principle

LM35 is a widely used temperature sensor with many different package types. At room temperature, it can achieve the accuracy of $\pm 1/4^{\circ}\text{C}$ without additional calibration processing.



LM35 temperature sensor can produce different voltage by different temperature

When temperature is 0 °C, it outputs 0V; if increasing 1 °C, the output voltage will increase 10 mv.

The output temperature is 0 °C ~ 100 °C , the conversion formula is as follows:

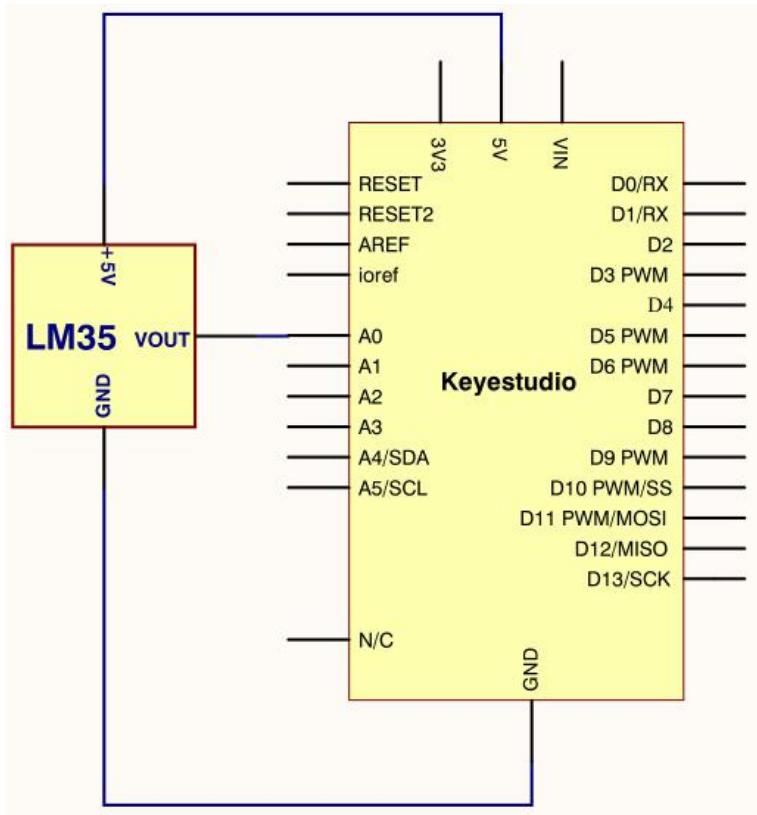
$$V_{\text{out_LM35}}(T) = 10 \text{ mV}/\text{°C} \times T^{\circ}\text{C}$$

3.Hardware Required

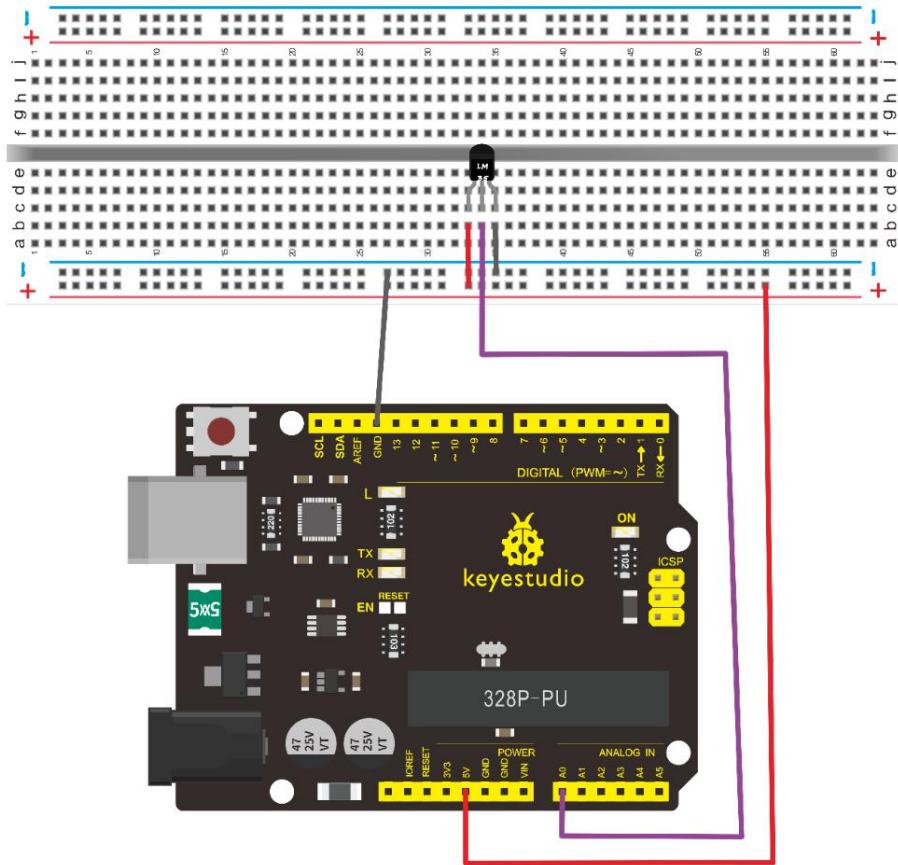
- V4.0 Board or MEGA 2650 Board*1
- LM35*1
- Breadboard*1
- Breadboard Jumper Wire*5

- USB cable*

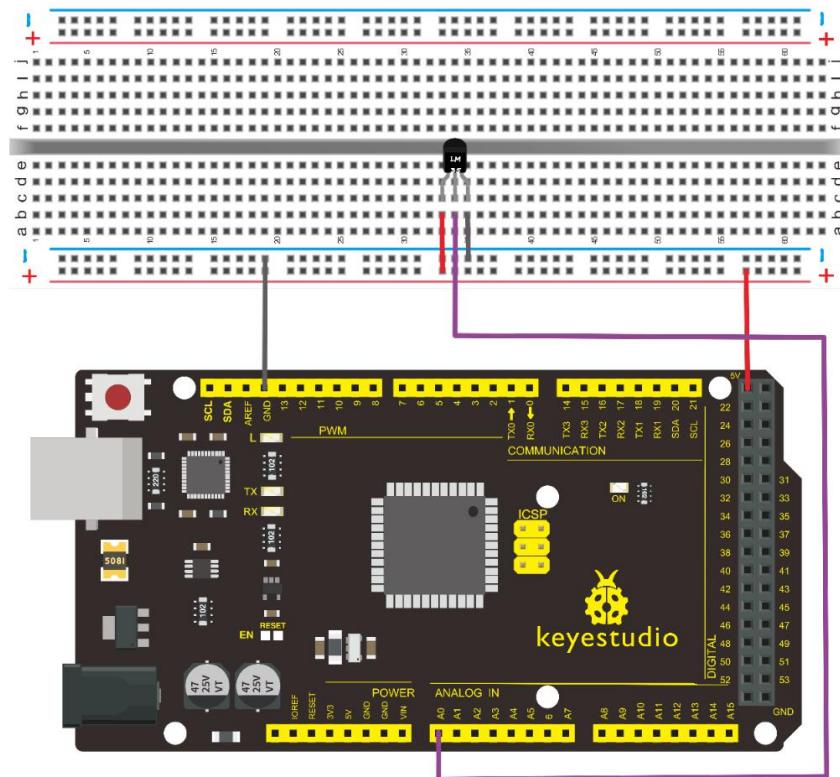
4.Circuit Connection



Connection for V4.0



Connection for 2560



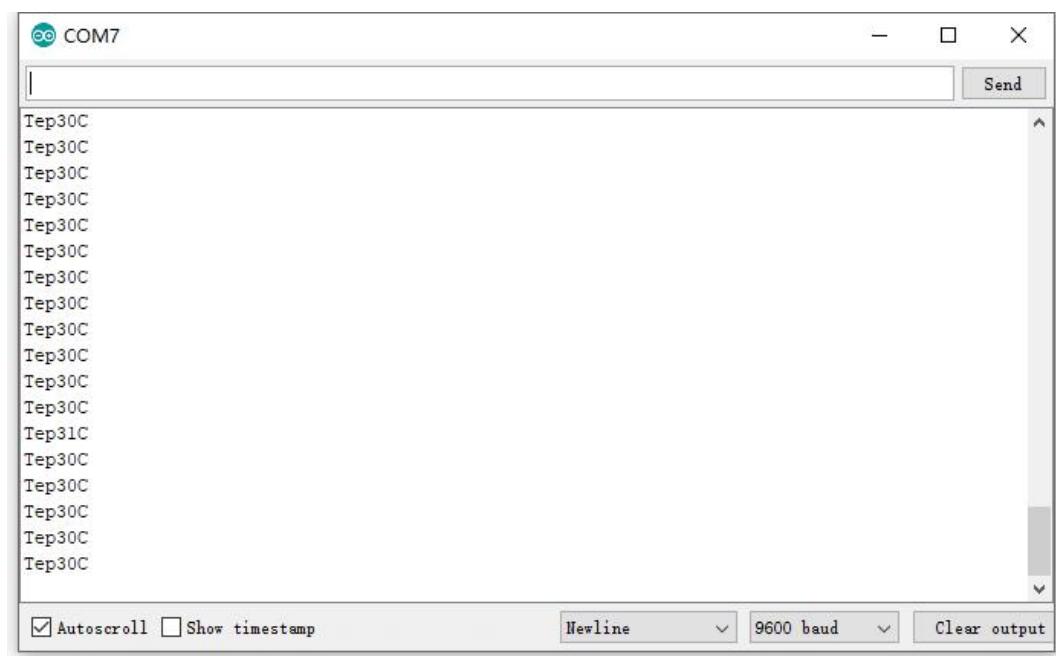
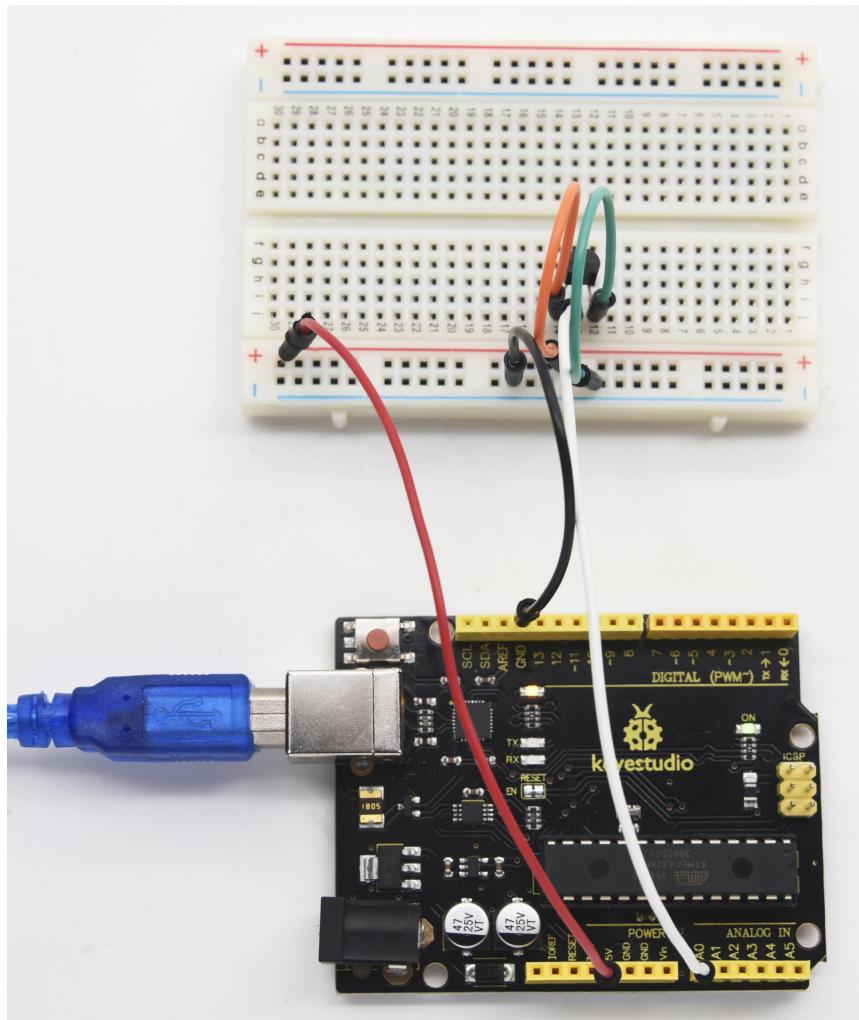
5.Sample Code

```
/*
keyestudio super learning kit
Project 12
LM35
http://www.keyestudio.com
*/
int potPin = 0; // initialize analog pin 0 for LM35 temperature
sensor
void setup()
{
Serial.begin(9600);// set baud rate at"9600"
}
void loop()
{
int val;// define variable
int dat;// define variable
val=analogRead(0);// read the analog value of the sensor
and assign it to val
dat=(125*val)>>8;// temperature calculation formula
```

```
Serial.print("Tep");// output and display characters  
beginning with Tep  
Serial.print(dat);// output and display value of dat  
Serial.println("C");// display "C" characters  
delay(500);// wait for 0.5 second  
}  
//////////
```

6. Test Result

After uploading the program, you can open the monitoring window to see the current temperature.



Project 13: Tilt Switch



1. Introduction

This is a ball switch experiment . The ball switch is also dubbed a steel ball switch. It controls the circuit by connecting guide pin with rolling ball. In this project, we control the LED light by reading the state of the ball switch.

2. Working Principle

When one end of the switch is below horizontal position, the switch is on. The voltage of the analog port is about 5V (1023 in binary). The LED will be on.

When the other end of the switch is below horizontal position, the switch is off. The voltage of the analog port is about 0V (0 in binary). The LED will be off.

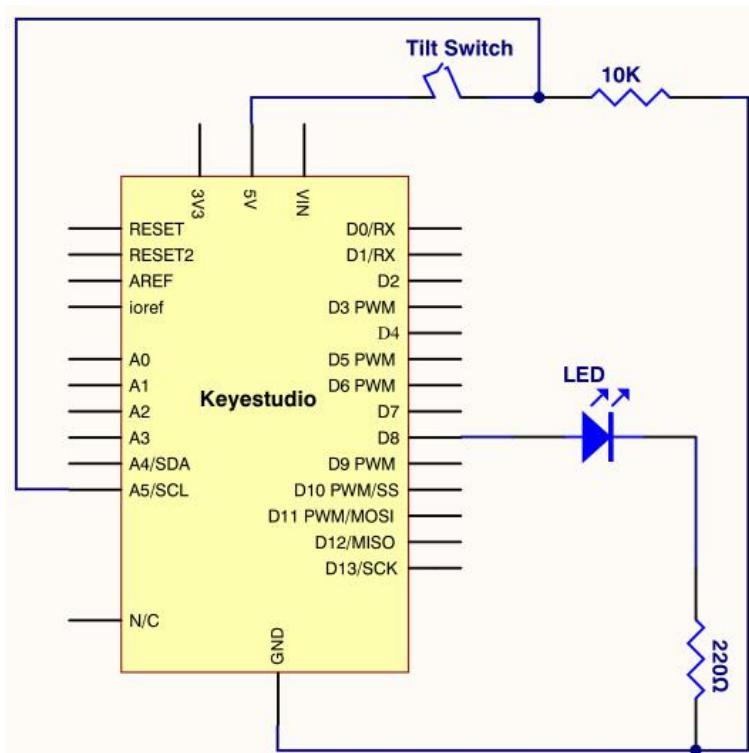
In the program, we determine whether the switch is on or off

according to the voltage value of the analog port, whether it's above 2.5V (512 in binary) or not.

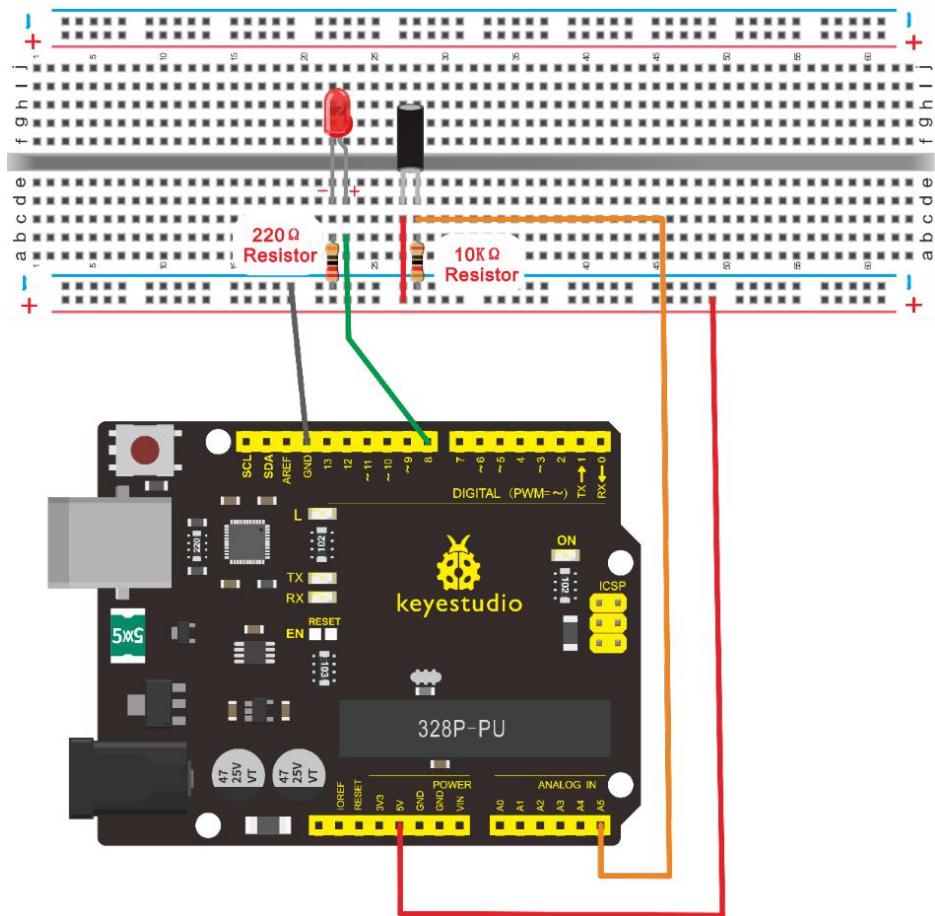
3.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- Ball switch*1
- Led *1
- 220Ω Resistor*1
- $10K\Omega$ resistor*1
- Breadboard Jumper Wire*5
- USB cable*1

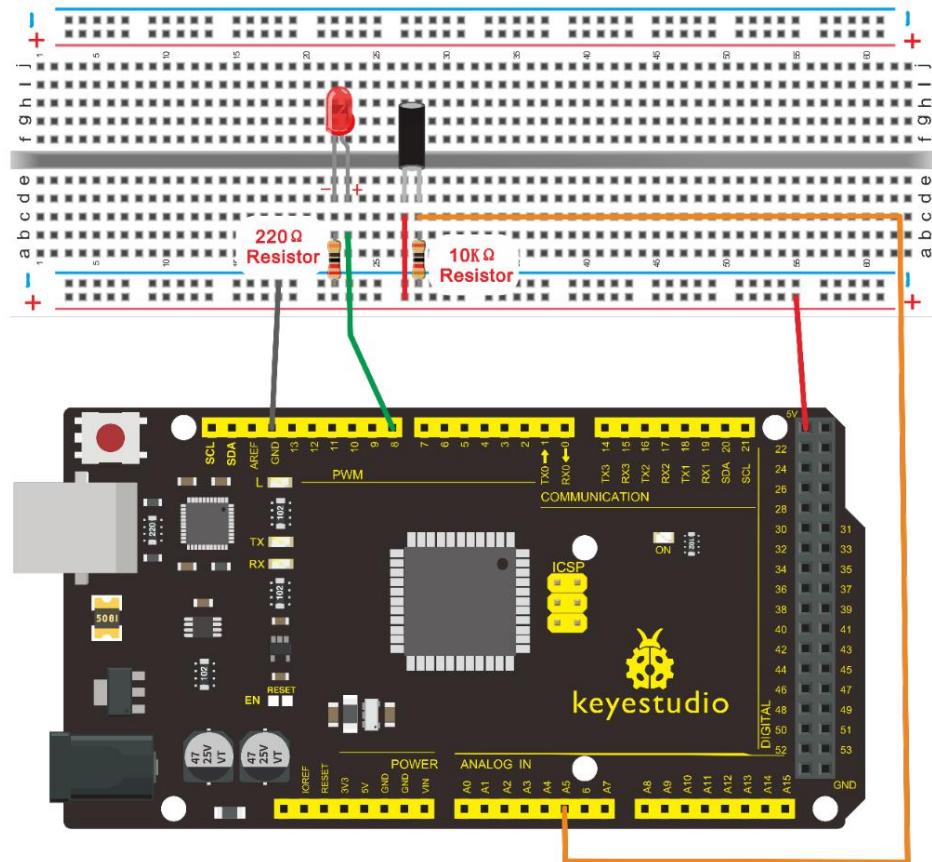
4.Circuit Connection



Connection for V4.0



Connection for 2560



5. Sample Code

```
/*
keyestudio super learning kit
Project 13
Tilt Switch
http://www.keyestudio.com
*/
void setup()
{
    pinMode(8,OUTPUT);// set digital pin 8 as "output"
}
```

```
}

void loop()

{
int i;// define variable i

while(1)

{
    i=analogRead(5);// read the voltage value of analog pin 5

    if(i>512)// if larger than 512 (2.5V)

    {
        digitalWrite(8,LOW);// turn on LED

    }

    else// otherwise

    {
        digitalWrite(8,HIGH);// turn off LED

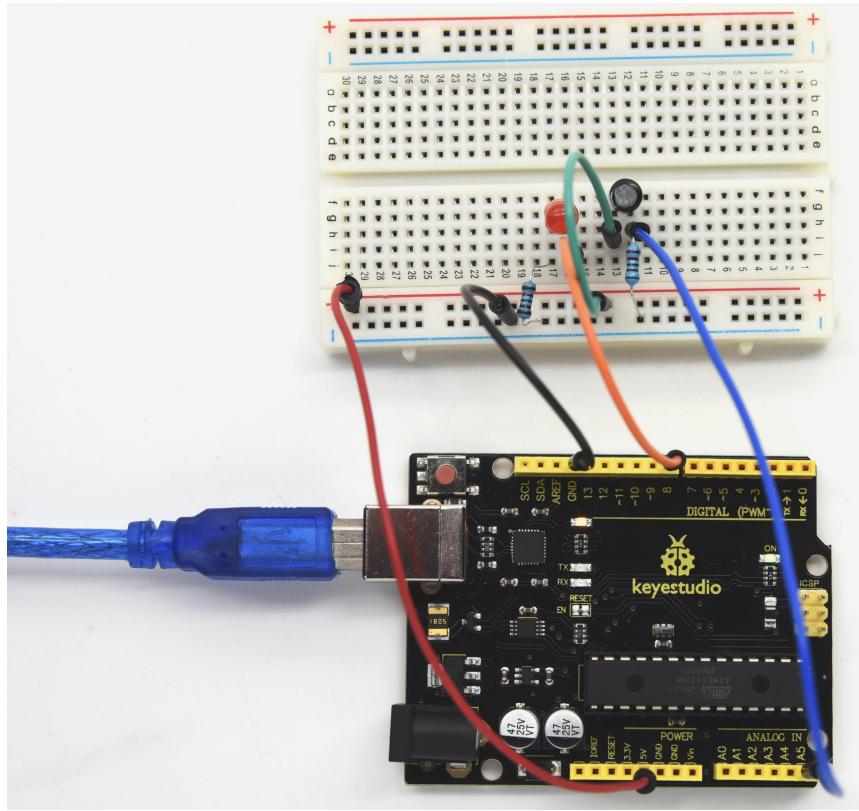
    } } }

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

6. Test Result

Hold the breadboard with your hand. Tilt it to a certain extent, the LED will be on. If there is no tilt, the LED will be off. The principle of this experiment can be also applied to relay control.

Experiment now is completed. Thank you!



Project 14: IR Remote Control



1. Introduction

What is an infrared receiver?

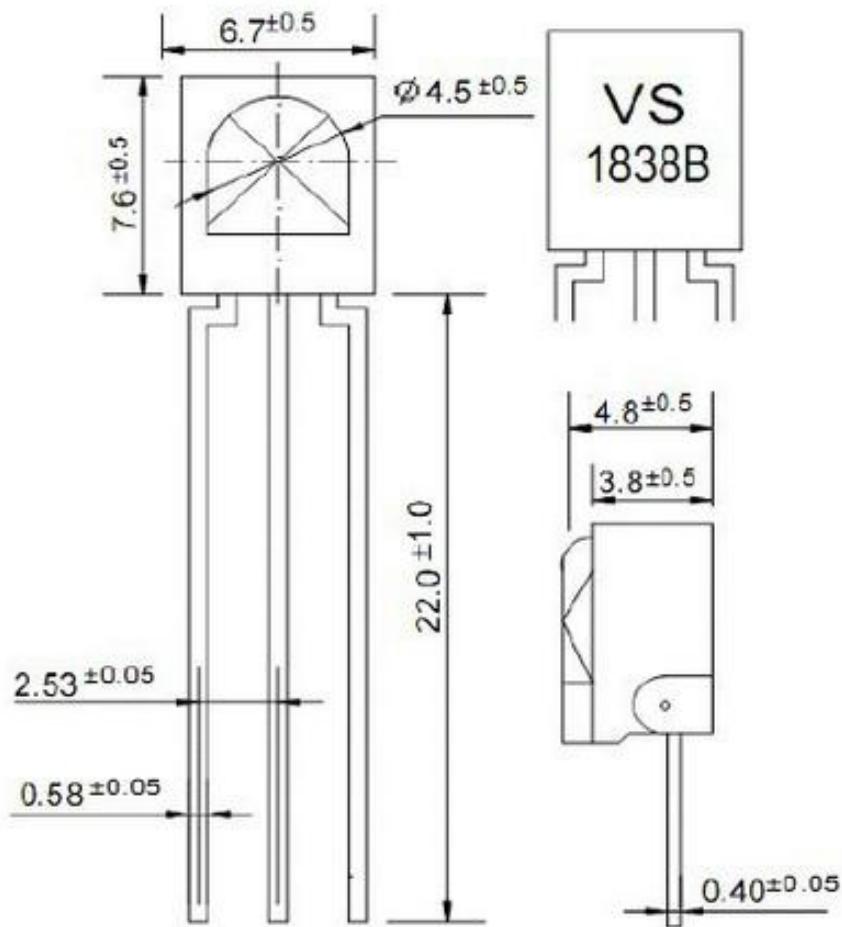
The signal from the infrared remote controller is a series of binary pulse code. To avoid the other infrared signal interference during the wireless transmission, the signal is pre-modulated at a specific carrier frequency and then send out by an infrared emission diode.

The infrared receiving device needs to filter out other waves and receive signals at that specific frequency and to modulate it back to binary pulse code, known as demodulation.

2.Working Principle

The built-in receiver converts the light signal it received from the sender into feeble electrical signal. The signal will be amplified by the IC amplifier. After automatic gain control, band-pass filtering, demodulation, wave shaping, it returns to the original code. The code is then input to the code identification circuit by the receiver's signal output pin.

3.Pin and Wiring for Infrared Receiver



Infrared receiver has 3 pins. When you use it, connect VOUT to analog pin, GND to GND, VCC to +5V.



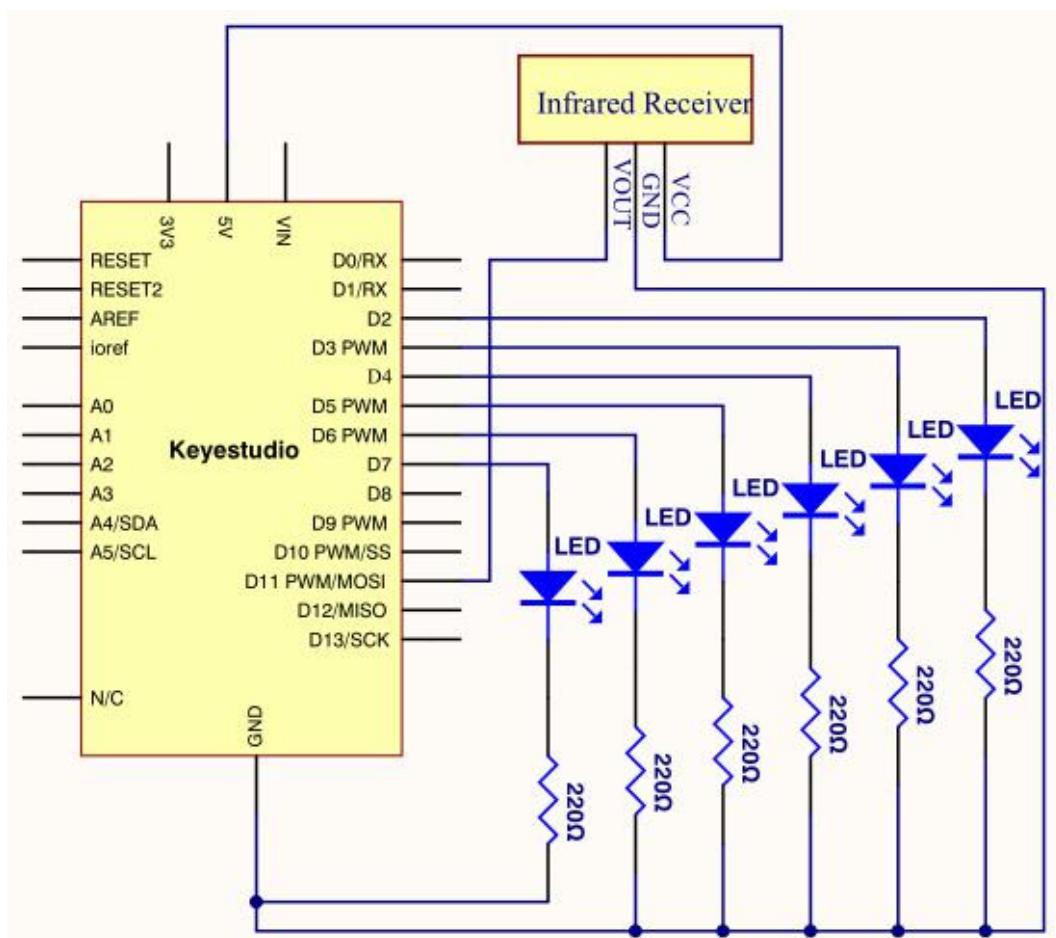
4. Hardware Required

- V4.0 Board or MEGA 2650 Board*1

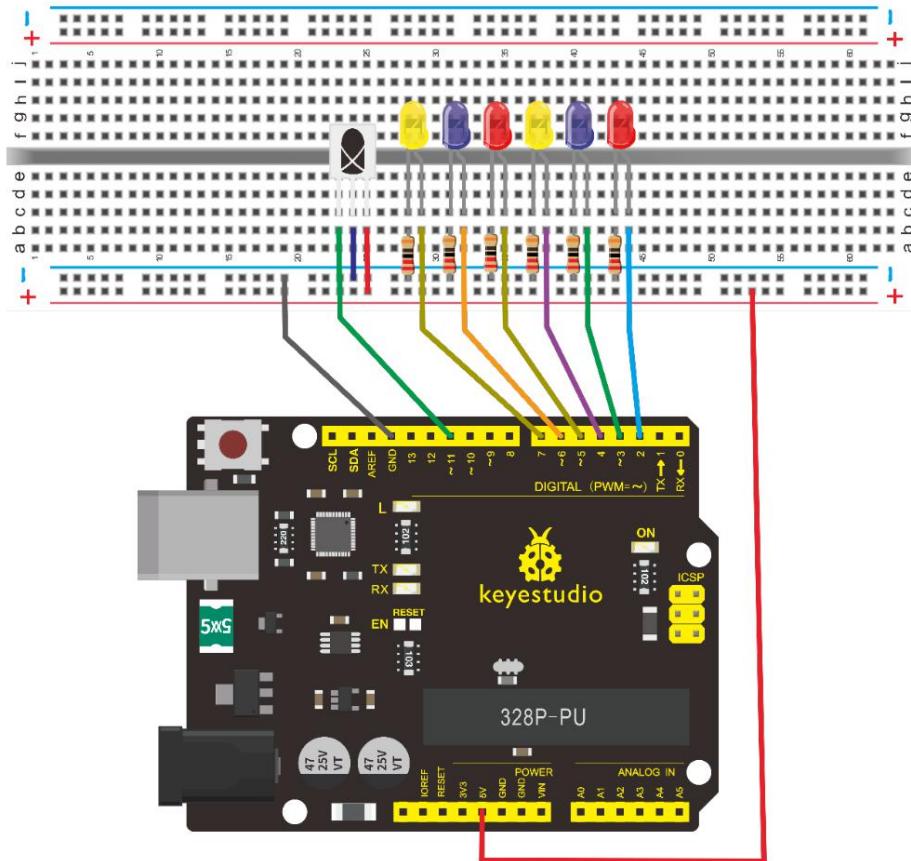
- Infrared Remote Controller *1
- Infrared Receiver *1
- LED *6
- 220ΩResistor *6
- Breadboard Wire *11
- USB cable*1

5.Circuit Connection

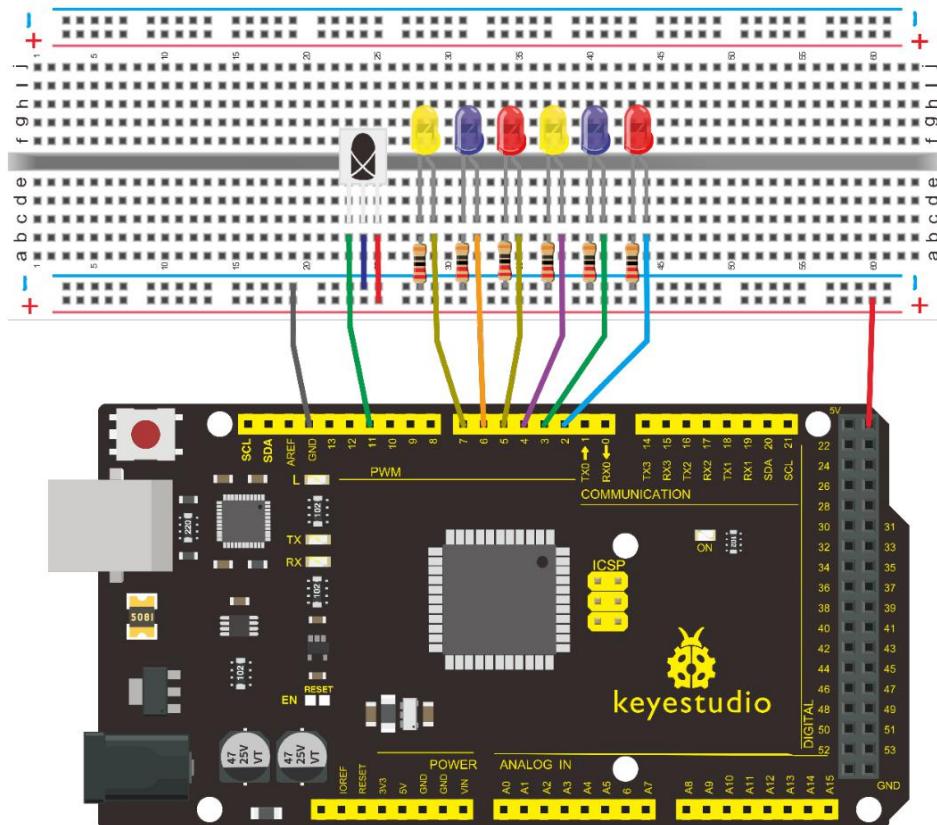
First, connect the controller board; then connect the infrared receiver as the above mentioned, connect VOUT to digital pin 11, connect the LEDs with resistors and connect the resistors to pin 2,3,4,5,6,7.



Connection for V4.0



Connection for 2560



6.Experimental Principle

If you want to decode the code from the remote controller, you must first know how it's coded. The coding method we use here is NEC protocol. Below is a brief introduction.

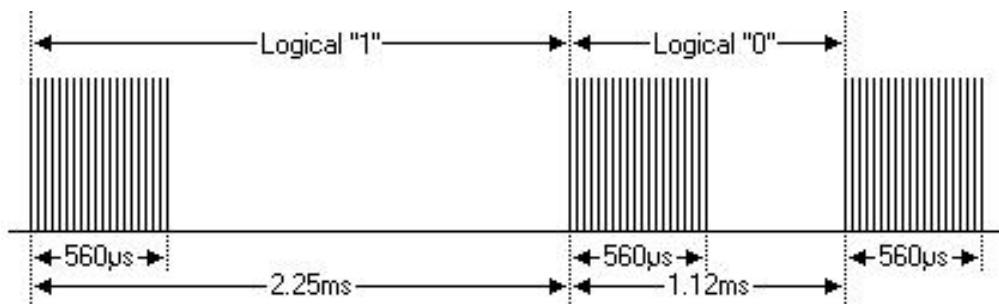
- [NEC protocol](#)

7.Features

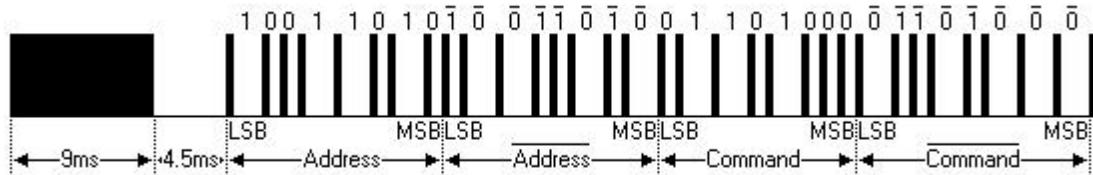
- (1) 8 bit address and 8 bit command length
- (2) address and command are transmitted twice for reliability
- (3) pulse distance modulation
- (4) carrier frequency of 38 KHZ
- (5) bit time of 1.125ms or 2.25ms

8.Protocol is as below

- Definition of logical 0 and 1 is as below

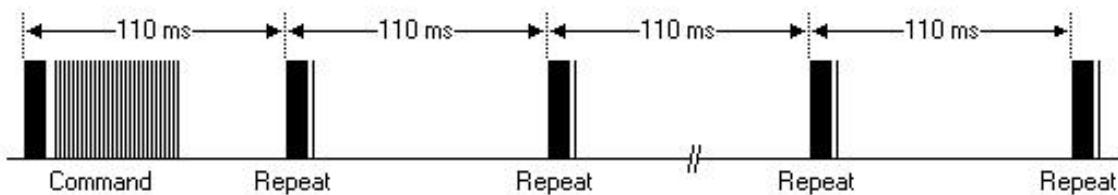


- Pulse transmitted when button is pressed and immediately released



The picture above shows a typical pulse train of the NEC protocol. With this protocol the LSB is transmitted first. In this case Address \$59 and Command \$16 is transmitted. A message is started by a 9ms AGC burst, which was used to set the gain of the earlier IR receivers. This AGC burst is then followed by a 4.5ms space, which is then followed by the address and command. Address and Command are transmitted twice. The second time all bits are inverted and can be used for verification of the received message. The total transmission time is constant because every bit is repeated with its inverted length. If you are not interested in this reliability, you can ignore the inverted values, or you can expend the Address and Command to 16 bits each!

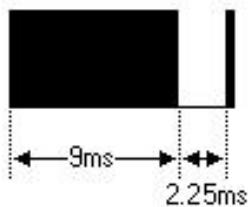
- Pulse transmitted when button is pressed and released after a period of time



A command is transmitted only once, even when the key on the remote

control remains pressed. Every 110ms a repeat code is transmitted for as long as the key remains down. This repeat code is simply a 9ms AGC pulse followed by a 2.25ms space and a $560\mu\text{s}$ burst.

- Repeat pulse



Note when the pulse enters the integrated receiver, there will be decoding, signal amplifying and wave shaping process. So you need to make sure the level of the output is just the opposite from that of the signal sending end.

That is when there is no infrared signal, the output end is in high level; when there is infrared signal, the output end is in low level. You can see the pulse of the receiving end in the oscilloscope. Try to better understand the program based on what you see.

9.Sample Code

```
/*
keyestudio super learning kit
Project 14
Remote
http://www.keyestudio.com
*/
#include <IRremote.h>

int RECV_PIN = 11;

int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
int LED5 = 6;
int LED6 = 7;

long on1 = 0x00FF6897;
long off1 = 0x00FF9867;
long on2 = 0x00FFB04F;
long off2 = 0x00FF30CF;
long on3 = 0x00FF18E7;
long off3 = 0x00FF7A85;
long on4 = 0x00FF10EF;
```

```
long off4 = 0x00FF38C7;

long on5 = 0x00FF5AA5;

long off5 = 0x00FF42BD;

long on6 = 0x00FF4AB5;

long off6 = 0x00FF52AD;

IRrecv irrecv(RECV_PIN);

decode_results results;

// Dumps out the decode_results structure.

// Call this after IRrecv::decode()

// void * to work around compiler issue

//void dump(void *v) {

// decode_results *results = (decode_results *)v

void dump(decode_results *results) {

    int count = results->rawlen;

    if (results->decode_type == UNKNOWN)

    {

        Serial.println("Could not decode message");

    }

    else

    {

        if (results->decode_type == NEC)
```

```
    Serial.print("Decoded NEC: ");

}

else if (results->decode_type == SONY)

{

    Serial.print("Decoded SONY: ");

}

else if (results->decode_type == RC5)

{

    Serial.print("Decoded RC5: ");

}

else if (results->decode_type == RC6)

{

    Serial.print("Decoded RC6: ");

}

Serial.print(results->value, HEX);

Serial.print(" (");

Serial.print(results->bits, DEC);

Serial.println(" bits)");

}

Serial.print("Raw (");

Serial.print(count, DEC);

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

```
for (int i = 0; i < count; i++)  
{  
    if ((i % 2) == 1) {  
        Serial.print(results->rawbuf[i]*USECPERTICK, DEC);  
    }  
    else  
{  
        Serial.print(-(int)results->rawbuf[i]*USECPERTICK, DEC);  
    }  
    Serial.print(" ");  
}  
Serial.println("");  
}  
  
void setup()  
{  
    pinMode(RECV_PIN, INPUT);  
    pinMode(LED1, OUTPUT);  
    pinMode(LED2, OUTPUT);  
    pinMode(LED3, OUTPUT);  
    pinMode(LED4, OUTPUT);  
    pinMode(LED5, OUTPUT);  
    pinMode(LED6, OUTPUT);
```

```
pinMode(13, OUTPUT);

Serial.begin(9600);

irrecv.enableIRIn(); // Start the receiver

}

int on = 0;

unsigned long last = millis();

void loop()

{

if (irrecv.decode(&results))

{

// If it's been at least 1/4 second since the last

// IR received, toggle the relay

if (millis() - last > 250)

{

on = !on;

// digitalWrite(8, on ? HIGH : LOW);

digitalWrite(13, on ? HIGH : LOW);

dump(&results);

}

if (results.value == on1 )

digitalWrite(LED1, HIGH);

if (results.value == off1 )


```

```
    digitalWrite(LED1, LOW);

    if (results.value == on2 )

        digitalWrite(LED2, HIGH);

    if (results.value == off2 )

        digitalWrite(LED2, LOW);

    if (results.value == on3 )

        digitalWrite(LED3, HIGH);

    if (results.value == off3 )

        digitalWrite(LED3, LOW);

    if (results.value == on4 )

        digitalWrite(LED4, HIGH);

    if (results.value == off4 )

        digitalWrite(LED4, LOW);

    if (results.value == on5 )

        digitalWrite(LED5, HIGH);

    if (results.value == off5 )

        digitalWrite(LED5, LOW);

    if (results.value == on6 )

        digitalWrite(LED6, HIGH);

    if (results.value == off6 )

        digitalWrite(LED6, LOW);

last = millis();
```

```
irrecv.resume(); // Receive the next value  
}  
}  
//////////
```

Note add IRremote folder into installation directory

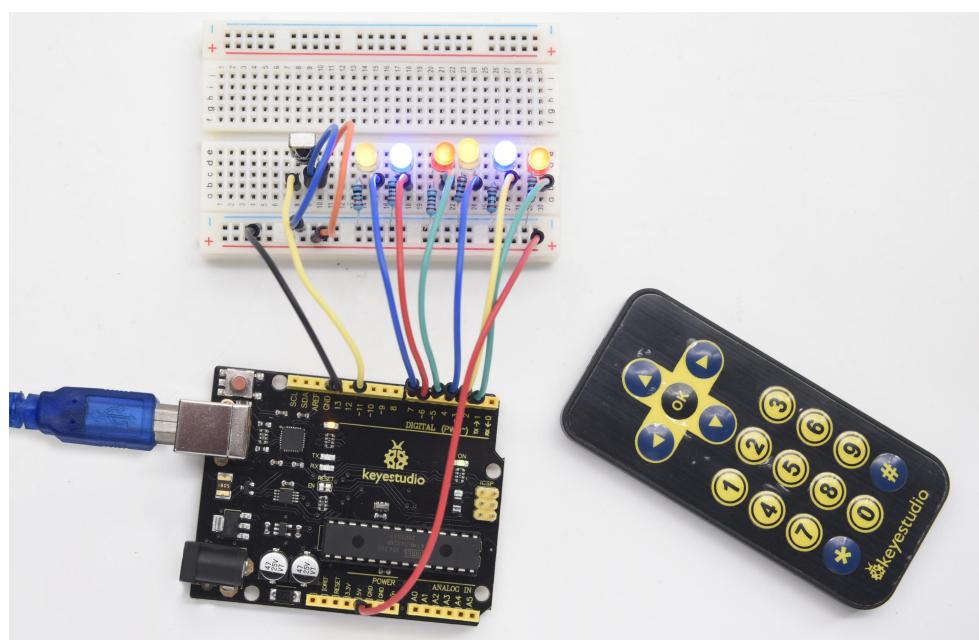
¥Arduino¥compiler libraries, or you will fail to compile it.

Infrared remote library

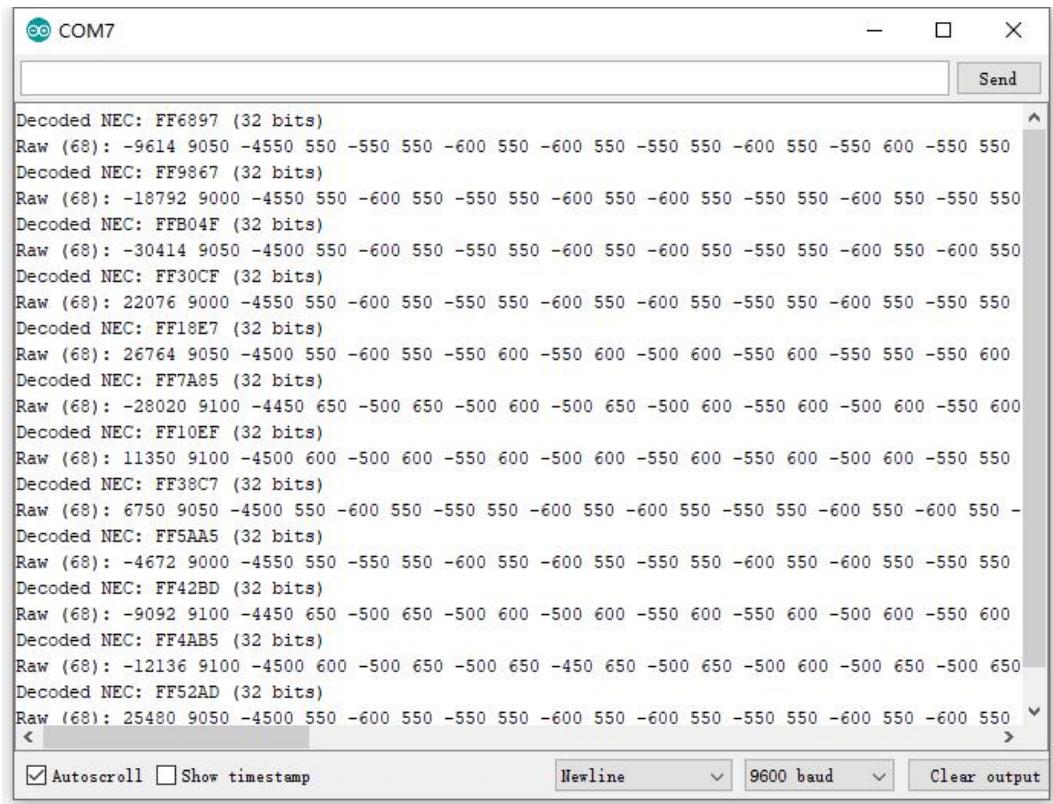
<https://github.com/shirriff/Arduino-IRremote>

10. Test Result

Decode the coded pulse signal emitted by the remote controller, then execute corresponding action according to the results of the decoding. In this way, you will be able to control your device with remote controller.



Done uploading, open the serial monitor, you can see the result as below.



The screenshot shows a Windows-style serial monitor window titled "COM7". The window displays a list of decoded NEC remote control messages. Each message consists of a "Decoded NEC" header followed by a "Raw (68)" value. The raw values are long strings of alternating 550 and 600 bits. The decoded messages include various codes such as FF6897, FF9867, FFB04F, FF30CF, FF18E7, FF7A85, FF10EF, FF38C7, FF5AA5, FF42BD, FF4AB5, and FF52AD. At the bottom of the window, there are checkboxes for "Autoscroll" and "Show timestamp", and dropdown menus for "Newline", "9600 baud", and "Clear output".

```
Decoded NEC: FF6897 (32 bits)
Raw (68): -9614 9050 -4550 550 -550 550 -600 550 -600 550 -550 550 -600 550 -550 600 -550 550
Decoded NEC: FF9867 (32 bits)
Raw (68): -18792 9000 -4550 550 -600 550 -550 550 -600 550 -550 550 -600 550 -550 550 -600 550
Decoded NEC: FFB04F (32 bits)
Raw (68): -30414 9050 -4500 550 -600 550 -550 550 -600 550 -600 550 -550 550 -600 550 -600 550
Decoded NEC: FF30CF (32 bits)
Raw (68): 22076 9000 -4550 550 -600 550 -550 550 -600 550 -600 550 -550 550 -600 550 -550 550
Decoded NEC: FF18E7 (32 bits)
Raw (68): 26764 9050 -4500 550 -600 550 -550 600 -550 600 -500 600 -550 600 -550 550 -550 600
Decoded NEC: FF7A85 (32 bits)
Raw (68): -28020 9100 -4450 650 -500 650 -500 600 -500 650 -500 600 -550 600 -500 600 -550 600
Decoded NEC: FF10EF (32 bits)
Raw (68): 11350 9100 -4500 600 -500 600 -550 600 -500 600 -550 600 -550 600 -500 600 -550 550
Decoded NEC: FF38C7 (32 bits)
Raw (68): 6750 9050 -4500 550 -600 550 -550 550 -600 550 -600 550 -550 550 -600 550 -600 550 -
Decoded NEC: FF5AA5 (32 bits)
Raw (68): -4672 9000 -4550 550 -550 550 -600 550 -600 550 -550 550 -600 550 -600 550 -550 550
Decoded NEC: FF42BD (32 bits)
Raw (68): -9092 9100 -4450 650 -500 650 -500 600 -500 600 -550 600 -550 600 -500 600 -550 600
Decoded NEC: FF4AB5 (32 bits)
Raw (68): -12136 9100 -4500 600 -500 650 -500 650 -450 650 -500 650 -500 600 -500 650 -500 650
Decoded NEC: FF52AD (32 bits)
Raw (68): 25480 9050 -4500 550 -600 550 -550 550 -600 550 -600 550 -550 550 -600 550 -600 550
< >
 Autoscroll  Show timestamp  Newline  9600 baud  Clear output
```

Project 15: Analog Value Reading



1.Introduction

In this experiment, we will begin the study of analog I/O interfaces. On an Arduino, there are 6 analog interfaces numbered from A0 to A5. Next, let's begin our project. Potentiometer used here is a typical output component of analog value that is familiar to us.

2.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- Potentiometer *1
- Breadboard*1
- Breadboard Jumper Wire*3
- USB cable*1

3.Potentiometer Features

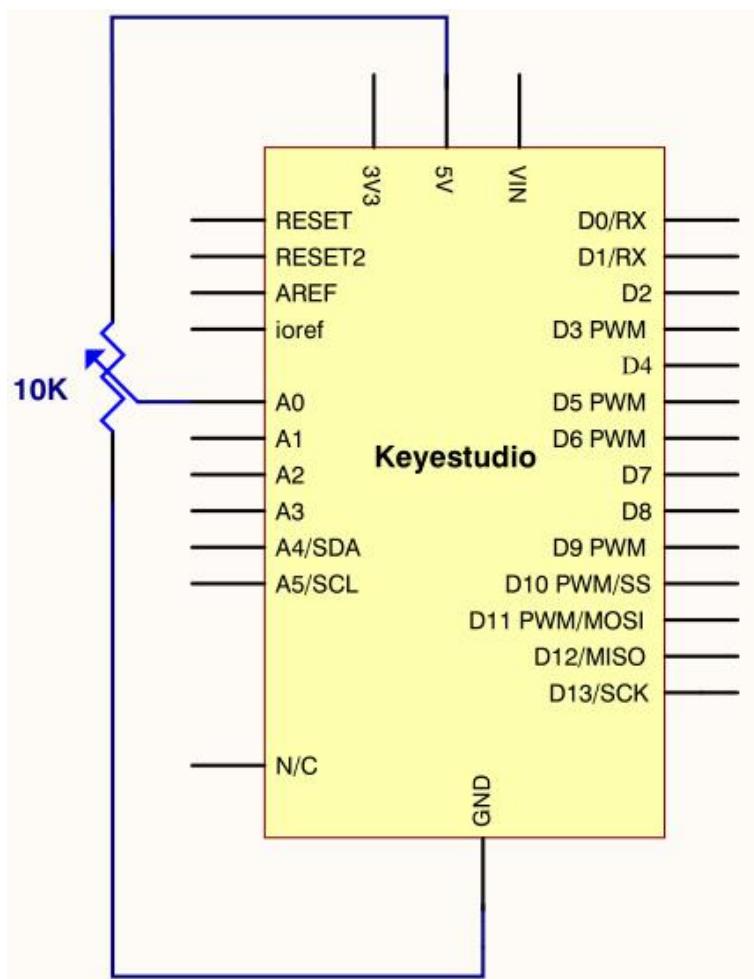
Adjustable potentiometer is just a kind of resistor. The resistance is changed by rotating the potentiometer, so is the voltage, speed, brightness and temperature. It is an analog electronic component, which has two states of 0 and 1(high level and low level). The analog quantity is different. Its data

state presents a linear state such as 1 to 1000.

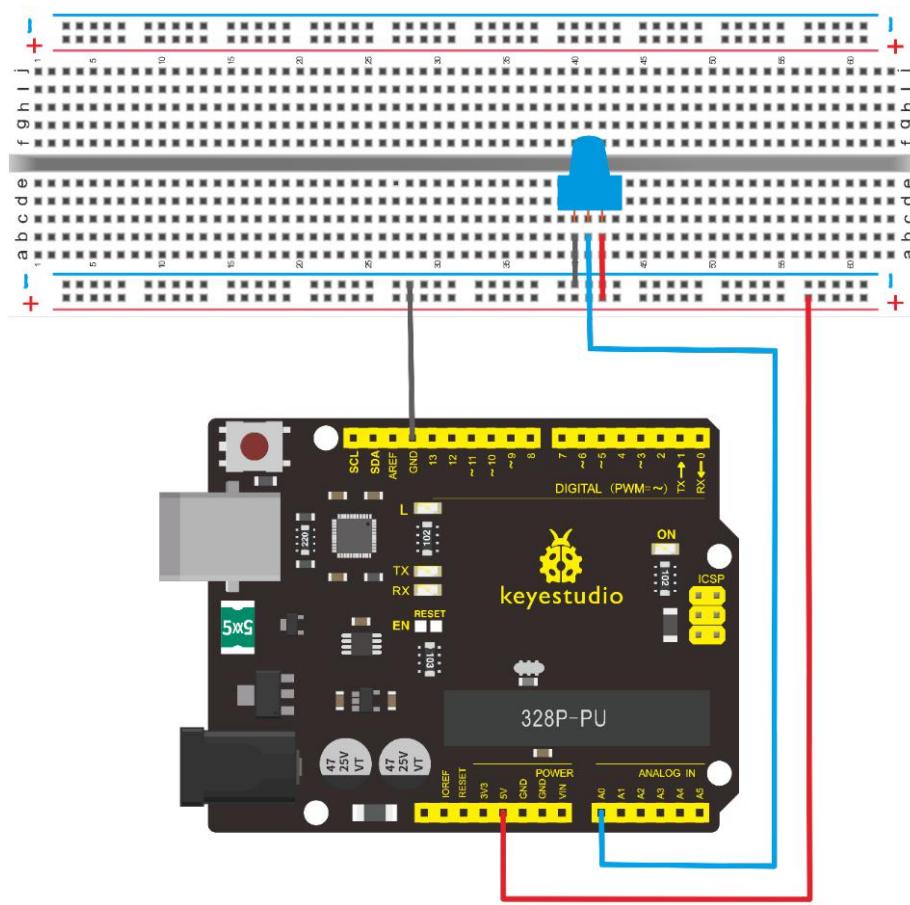
4.Circuit Connection

In this experiment, we will convert the resistance value of the potentiometer to analog ones and display it on the screen.

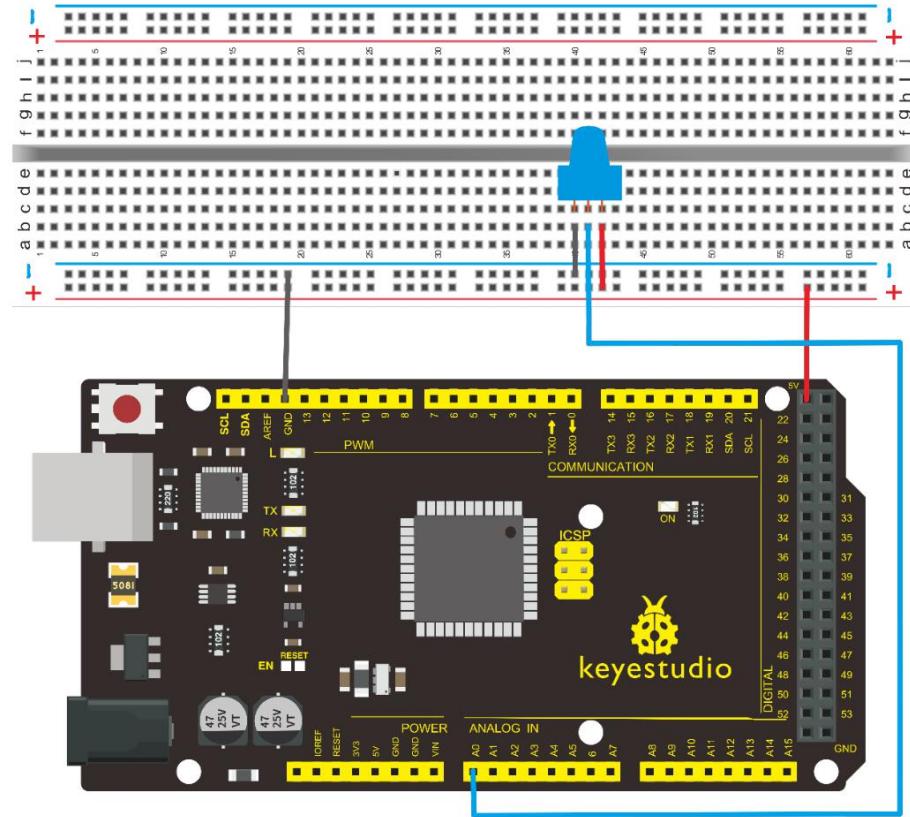
This is an application you need to master well for our future experiments.



Connection for V4.0



Connection for 2560



5.Sample Code

The program compiling is simple. An `analogRead()` Statement can read the value of the interface. The A/D acquisition of Arduino 328 is in 10 bits, so the value it reads is among 0 to 1023.

One difficulty in this project is to display the value on the screen, which is actually easy to learn.

First, you need to set the baud rate in `voidsetup()`. Displaying the value is a communication between Arduino and PC, so the baud rate of the Arduino should match the one in the PC's software set up. Otherwise, the display will be messy codes or no display at all.

In the lower right corner of the Arduino software monitor window, there is a button for baud rate set up. The set up here needs to match the one in the program. The statement in the program is `Serial.begin();` enclosed is the baud rate value, followed by statement for displaying. You can either use `Serial.print()` or `Serial.println()` statement.

```
/*
keyestudio super learning kit
Project 15
```

Potentiometer

<http://www.keyestudio.com>

* /

```
int potpin=0;// initialize analog pin 0
```

```
int ledpin=13;// initialize digital pin 13
```

```
int val=0;// define val, assign initial value 0
```

```
void setup()
```

{

```
pinMode(ledpin,OUTPUT);// set digital pin as "output"
```

```
Serial.begin(9600); // set baud rate at 9600
```

}

void loop()

{

```
digitalWrite(ledpin,HIGH); // turn on the LED on pin 13
```

```
delay(50); // wait for 0.05 second
```

```
digitalWrite(ledpin,LOW);// turn off the LED on pin 13
```

```
delay(50); // wait for 0.05 second
```

```
val=analogRead(potpins); // read the analog value of analog pin 0, and
```

assign it to val

```
Serial.println(val); // display val's value
```

}

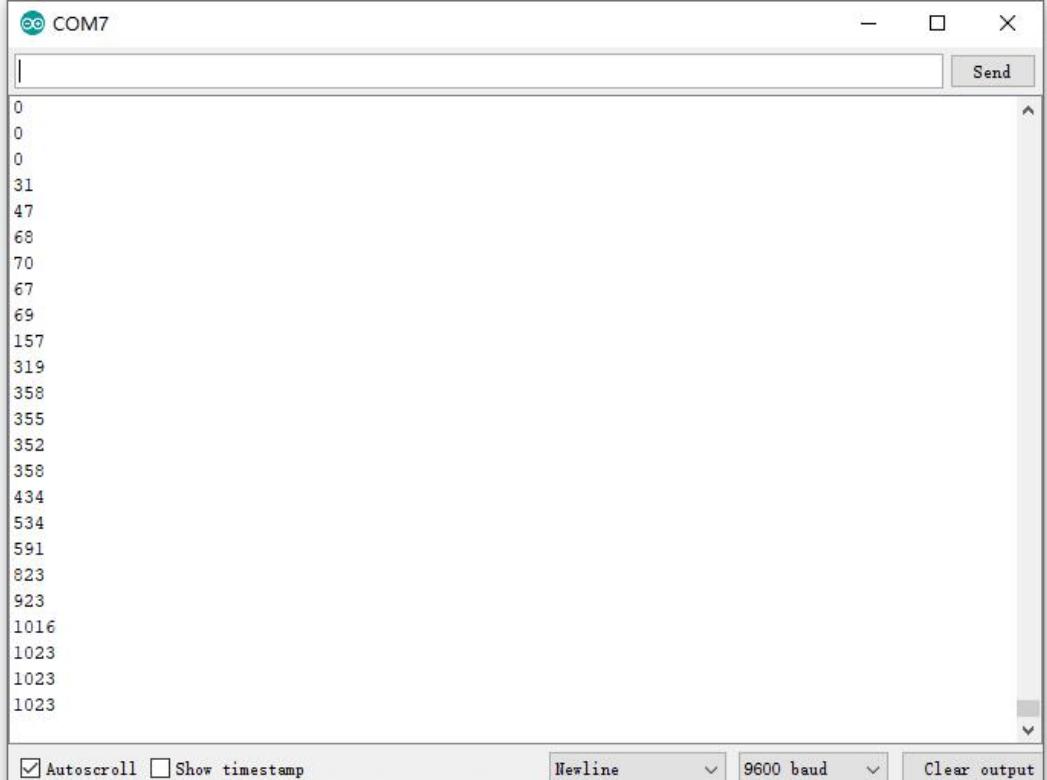
|||||

6. Test Result

The Sample Code uses the built-in LED connected to pin 13.

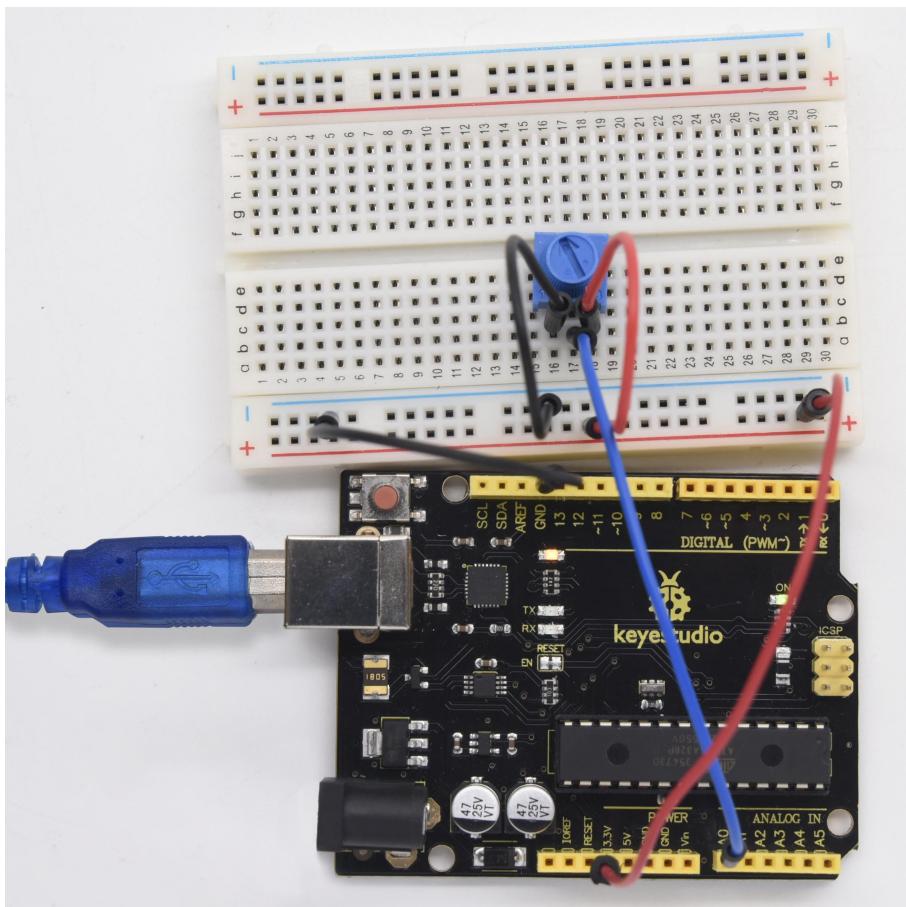
Each time the device reads a value, the LED blinks. When you rotate the potentiometer knob, you can see the displayed value change. The reading of analog value is a very common function since most sensors output analog value. After calculation, you can get the corresponding value you need.

Below figure shows the analog value it reads.

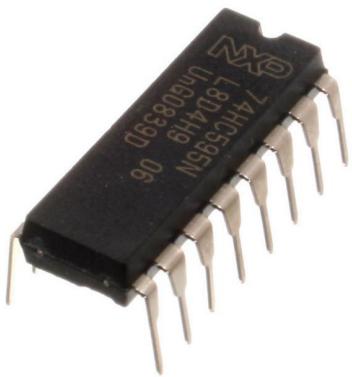


```
0
0
0
31
47
68
70
67
69
157
319
358
355
352
358
434
534
591
823
923
1016
1023
1023
1023
```

The experiment is now completed. Thank you!



Project 16: 74HC595 chip



1. Introduction

74HC595 chip is a serial output and parallel output device. To

put it simply, 74HC595 chip is a combination of 8-digit shifting register, memorizer and equipped with tri-state output.

Here, we use it to control 8 LEDs. You may wonder why use a 74HC595 to control LED? Well, think about how many I/O it takes for an Arduino to control 8 LEDs? Yes, 8.

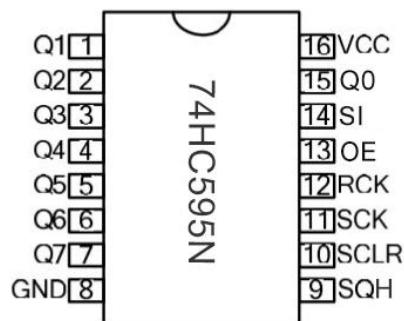
For an Arduino 328, it has only 20 I/O including analog ports. To save port resources, using 74HC595 enables us to use 3 digital I/O ports to control 8 LEDs!

2.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- 74HC595 chip*1
- Red M5 LED*4
- Green M5 LED*4
- 220 Ω Resistor*8
- Breadboard*1
- Breadboard Jumper Wires
- USB cable*1



3. Pins Description :



Pins No	Name	Function
1-7 , 15	Q0-Q7	Parallel Output
8	GND	GND
9	MR	Serial Output
10		Master Reserve , connect 5V
11	SH_CP	Shift Register Clock Output

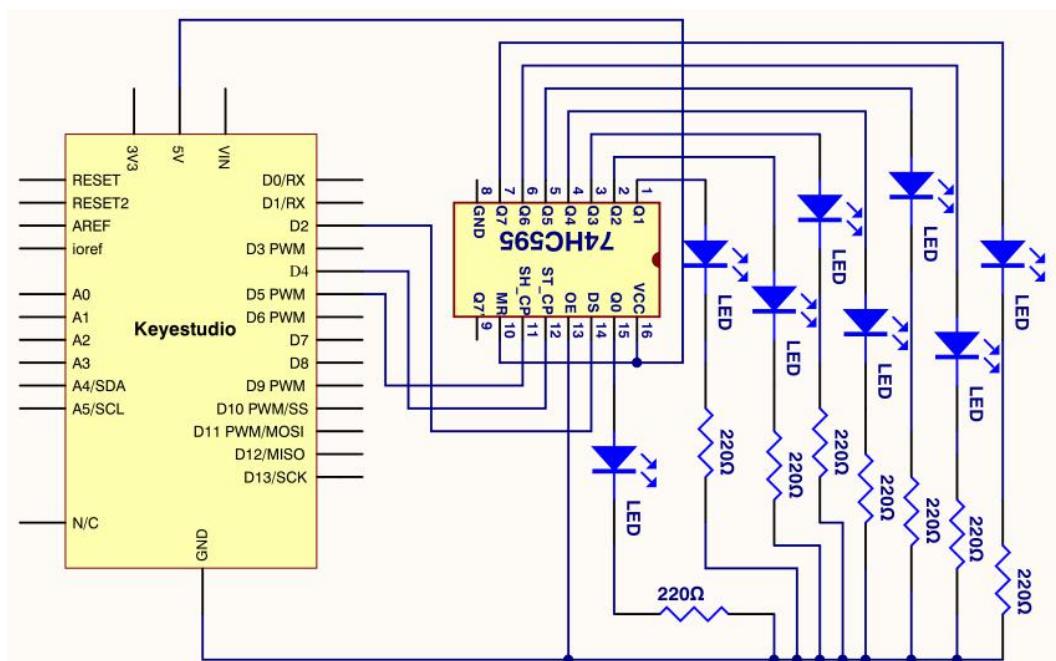
12	ST_CP	Storage Register Clock Input
13	OE	Output Enable (active LOW)
14	DS	Serial data input
16	Vcc	5V working voltage

- VCC and GND are used to supply power for chip, the working voltage is 5V.
- Q0~Q7: This eight pins are output pins.
- DS pin is serial input pin, we need to write data into this pin by bit.
- STCP is a latch pin. The data can be copied to latch and output in parallel after 8-digit data of latch is all transmitted.
- SHCP is a clock pin. The data can be written into storage register.
- OE is an output enable pin, which is used to make sure if the data of latch is input into Q0-Q7 pins. When in low level, high level is not output. In this experiment, we directly connect to GND to keep low level output data.
- MR is a pin to initialize the pin of storage register. Initialize the internal storage register when low level. In this

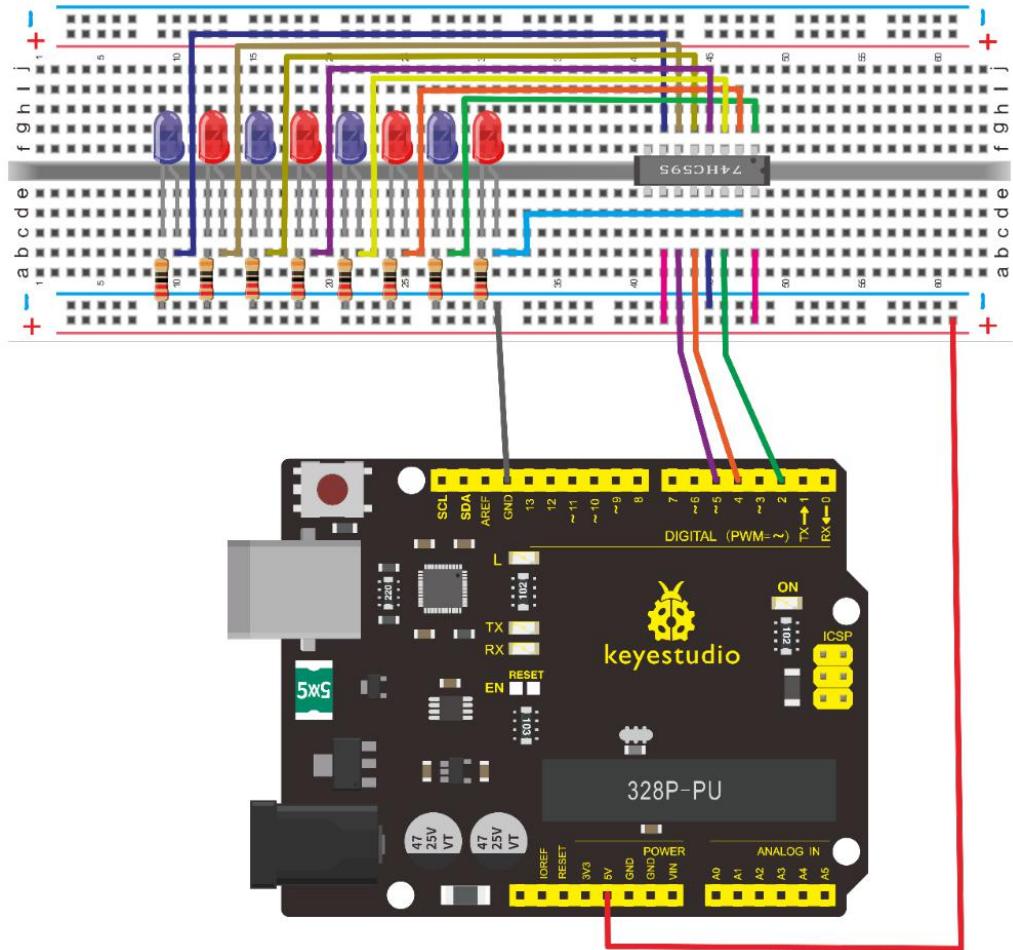
experiment, we connect to VCC to keep high level.

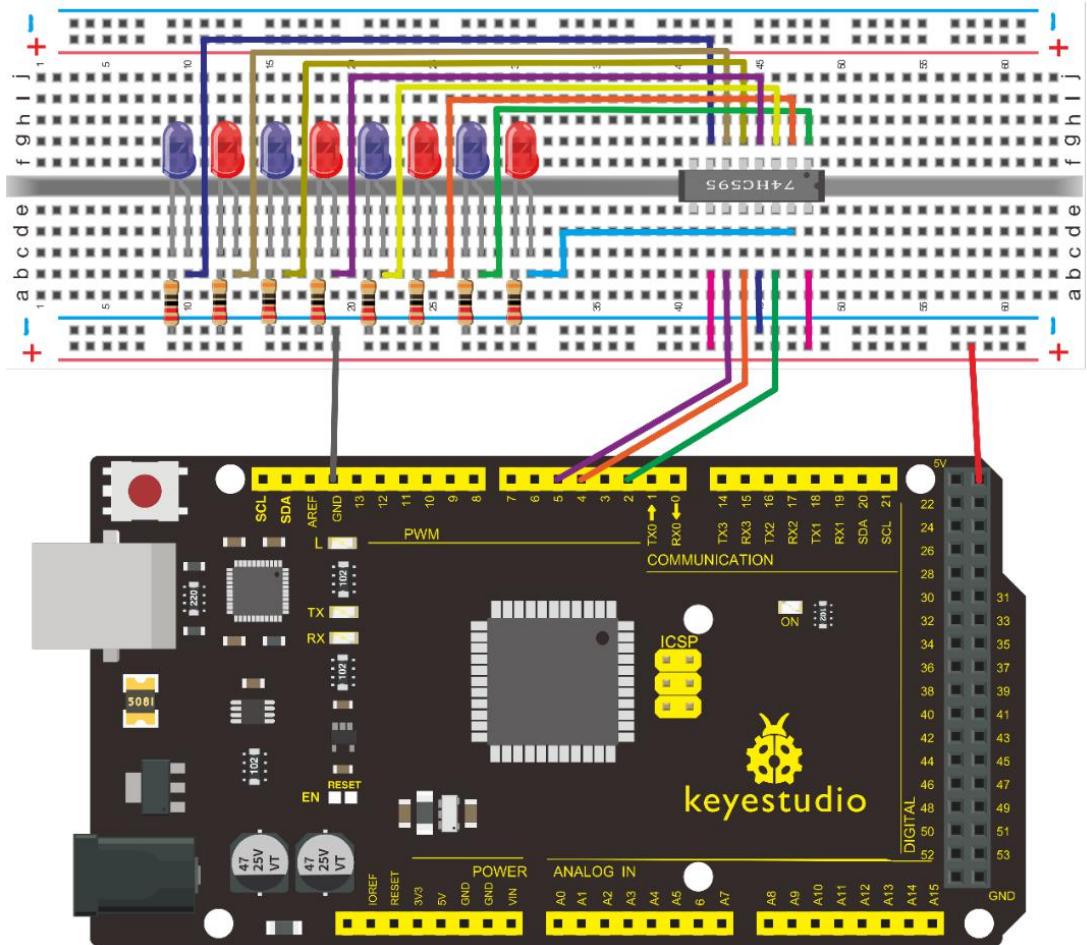
- Q7S pin is a serial output pin, which is specially used for chip cascade.

4. Circuit Connection



Connection for V4.0





The circuit may seem complicated, but once you wire it in order, you will find it more easier!

5. Sample Code

```
/*
keyestudio super learning kit
Project 16
74hc595
http://www.keyestudio.com
```

```
*/  
  
int data = 2;// set pin 14 of 74HC595as data input pin SI  
  
int clock = 5;// set pin 11 of 74hc595 as clock pin SCK  
  
int latch = 4;// set pin 12 of 74hc595 as output latch RCK  
  
int ledState = 0;  
  
const int ON = HIGH;  
  
const int OFF = LOW;  
  
void setup()  
  
{  
  
pinMode(data, OUTPUT);  
  
pinMode(clock, OUTPUT);  
  
pinMode(latch, OUTPUT);  
  
}  
  
void loop()  
  
{  
  
for(int i = 0; i < 256; i++)  
  
{  
  
updateLEDs(i);  
  
delay(500);  
  
}  
  
}  
  
void updateLEDs(int value)
```

```

{

digitalWrite(latch, LOW);//

shiftOut(data, clock, MSBFIRST, ~value);// serial data "output", high

level first

digitalWrite(latch, HIGH);// latch

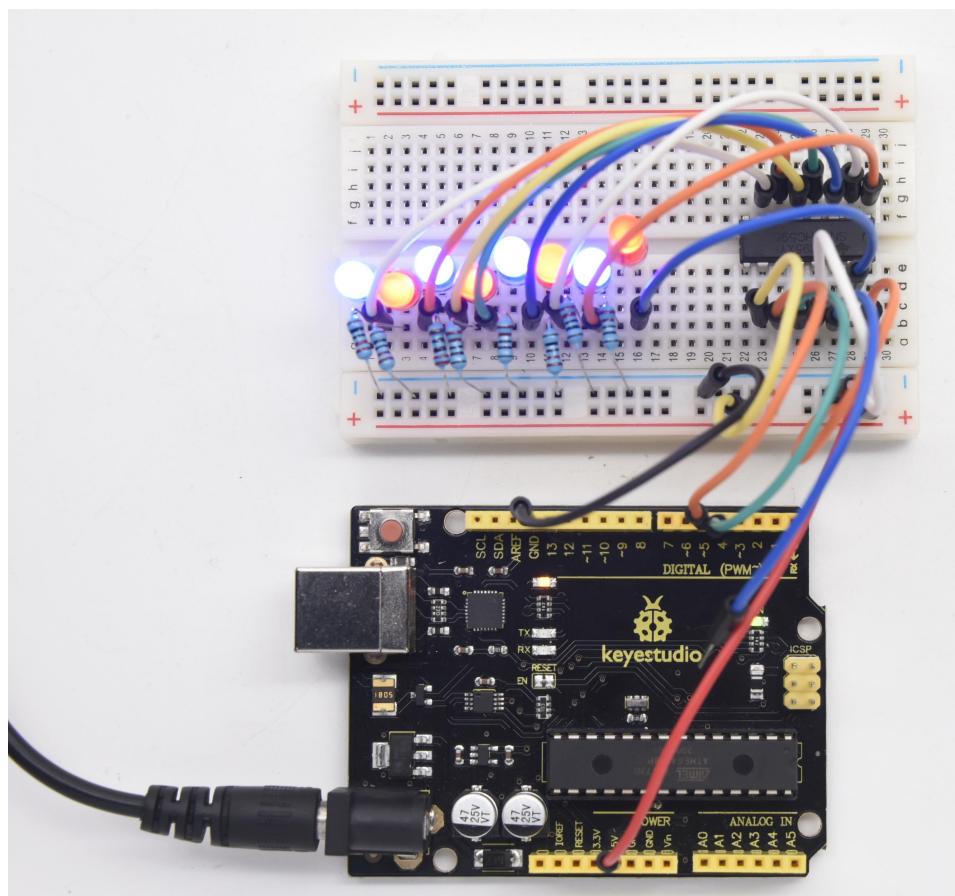
}

///////////////////////////////

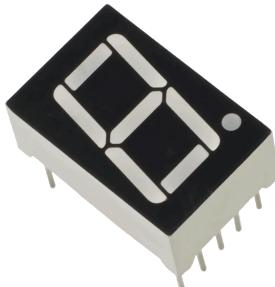
```

6. Test Result

After downloading the program, you can see 8 LEDs display 8-bit binary number.



Project 17: 1-digit LED Segment Display



1. Introduction

LED segment displays are common for displaying numerical information. It's widely applied on displays of electromagnetic oven, full automatic washing machine, water temperature display, electronic clock, etc. It is necessary for us to learn how it works.

2. Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- 1-digit LED Segment Display*1
- 220 Ω Resistor*8
- Breadboard*1
- Breadboard Jumper Wires *several
- USB cable*1

3.The display principle of the 1-digit LED Segment Display

LED segment display is a semiconductor light-emitting device.

Its basic unit is a light-emitting diode (LED).

LED segment display can be divided into 7-segment display

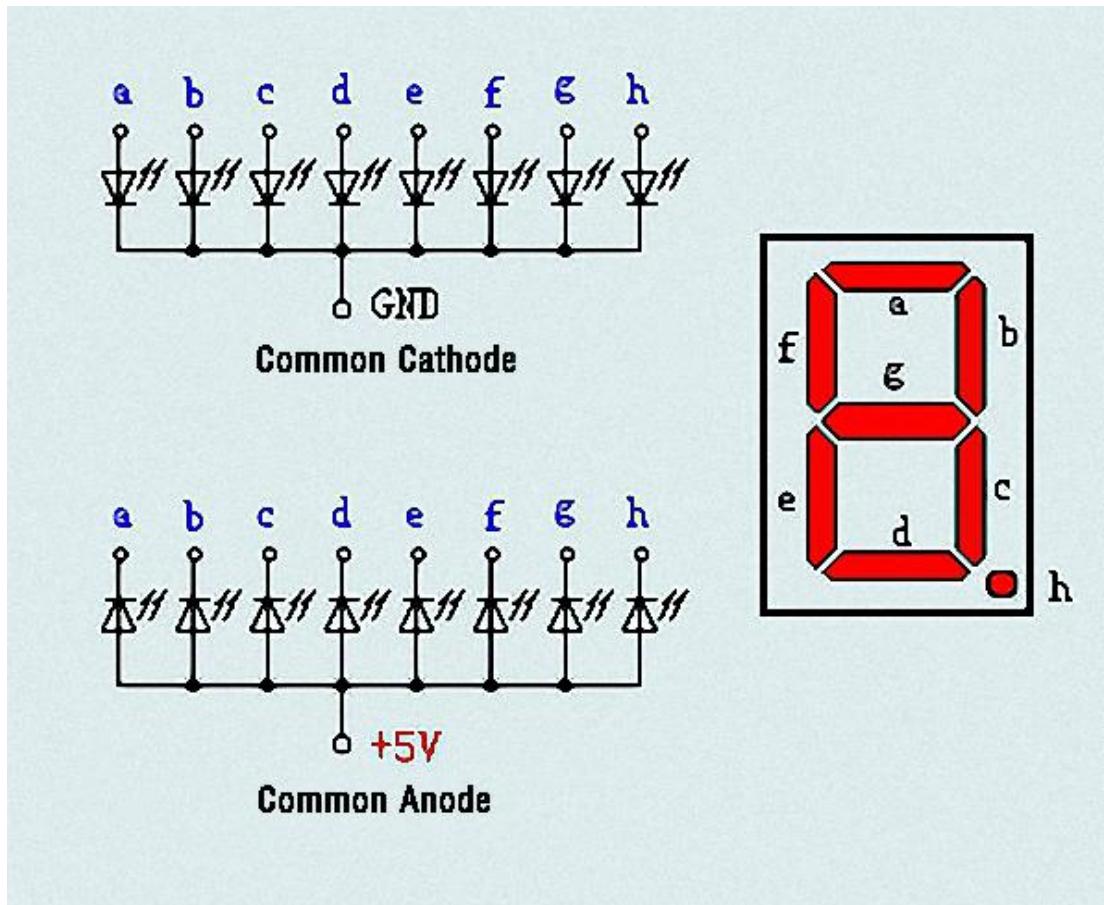
and 8-segment display according to the number of segments.

8-segment display has one more LED unit (for decimal point display) than 7-segment one.

According to the wiring method of LED units, LED segment display can be divided into common anode display and common cathode display. Common anode display refers to the one that combine all the anodes of LED units into one common anode (COM).

For the common anode display, connect the common anode (COM) to +5V. When the cathode level of a certain segment is low, the segment is on; when the cathode level of a certain segment is high, the segment is off.

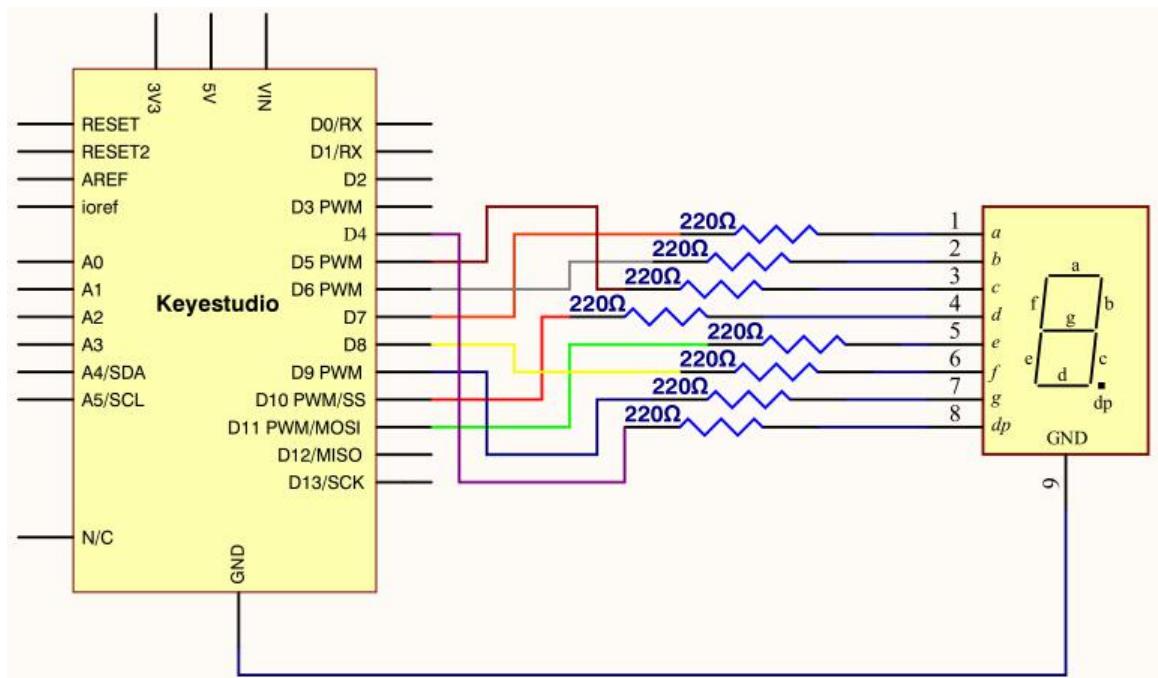
For the common cathode display, connect the common cathode (COM) to GND. When the anode level of a certain segment is high, the segment is on; when the anode level of a certain segment is low, the segment is off.



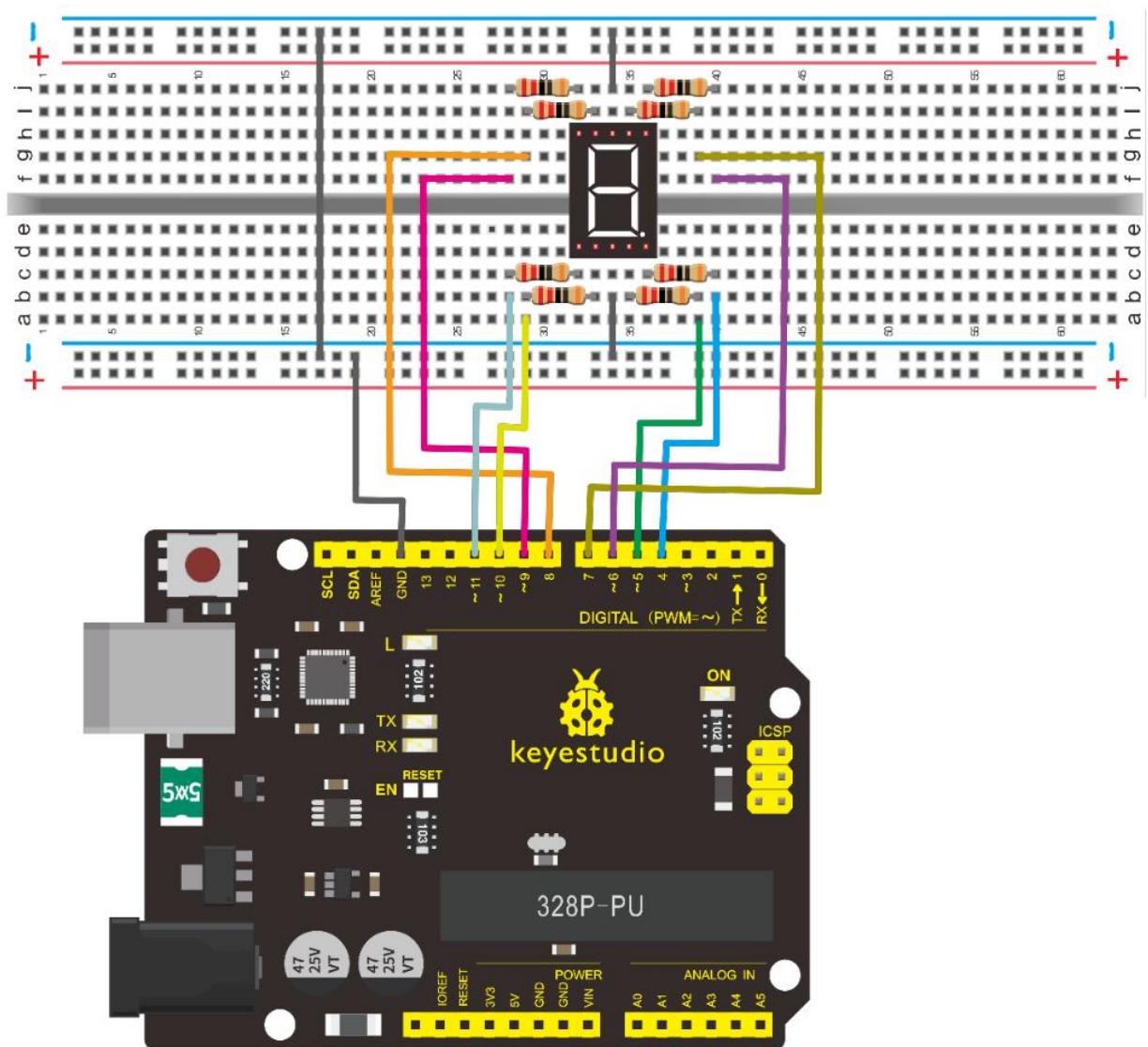
Each segment of the display consists of an LED. So when you use it, you also need to use a current-limiting resistor. Otherwise, LED will be burnt out.

In this experiment, we use a common cathode display. As we mentioned above, for common cathode display, connect the common cathode (COM) to GND. When the anode level of a certain segment is high, the segment is on; when the anode level of a certain segment is low, the segment is off.

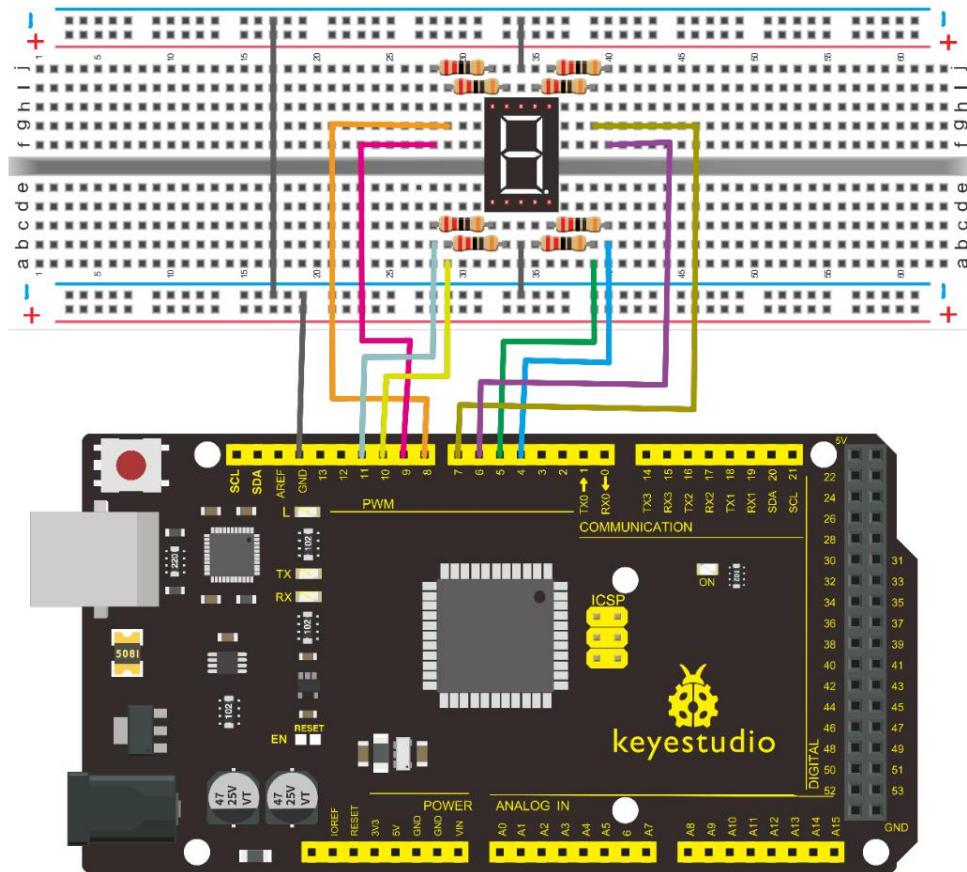
4.Circuit Connection



Connection for V4.0



Connection for 2560



5. Sample Code

There are seven segments for numerical display, one for decimal point display. Corresponding segments will be turned on when displaying certain numbers.

For example, when displaying number 1, b and c segments will be turned on. We compile a subprogram for each number, and compile the main program to display one number every 2 seconds, cycling display number 0 ~ 9.

The displaying time for each number is subject to the delay time, the longer the delay time, the longer the displaying

time.

```
/*
keyestudio super learning kit
Project 17
1-digit LED Segment Display
http://www.keyestudio.com
*/
// set the IO pin for each segment
int a=7;// set digital pin 7 for segment a
int b=6;// set digital pin 6 for segment b
int c=5;// set digital pin 5 for segment c
int d=10;// set digital pin 10 for segment d
int e=11;// set digital pin 11 for segment e
int f=8;// set digital pin 8 for segment f
int g=9;// set digital pin 9 for segment g
int dp=4;// set digital pin 4 for segment dp
void digital_0(void) // display number 5
{
    unsigned char j;
    digitalWrite(a,HIGH);
    digitalWrite(b,HIGH);
```

```
digitalWrite(c,HIGH);

digitalWrite(d,HIGH);

digitalWrite(e,HIGH);

digitalWrite(f,HIGH);

digitalWrite(g,LOW);

digitalWrite(dp,LOW);

}

void digital_1(void) // display number 1

{

unsigned char j;

digitalWrite(c,HIGH);// set level as "high" for pin 5, turn on segment c

digitalWrite(b,HIGH);// turn on segment b

for(j=7;j<=11;j++)// turn off other segments

digitalWrite(j,LOW);

digitalWrite(dp,LOW);// turn off segment dp

}

void digital_2(void) // display number 2

{

unsigned char j;

digitalWrite(b,HIGH);

digitalWrite(a,HIGH);

for(j=9;j<=11;j++)
```

```
digitalWrite(j,HIGH);

digitalWrite(dp,LOW);

digitalWrite(c,LOW);

digitalWrite(f,LOW);

}

void digital_3(void) // display number 3

{digitalWrite(g,HIGH);

digitalWrite(a,HIGH);

digitalWrite(b,HIGH);

digitalWrite(c,HIGH);

digitalWrite(d,HIGH);

digitalWrite(dp,LOW);

digitalWrite(f,LOW);

digitalWrite(e,LOW);

}

void digital_4(void) // display number 4

{digitalWrite(c,HIGH);

digitalWrite(b,HIGH);

digitalWrite(f,HIGH);

digitalWrite(g,HIGH);

digitalWrite(dp,LOW);

digitalWrite(a,LOW);
```

```
digitalWrite(e,LOW);

digitalWrite(d,LOW);

}

void digital_5(void) // display number 5

{

unsigned char j;

digitalWrite(a,HIGH);

digitalWrite(b, LOW);

digitalWrite(c,HIGH);

digitalWrite(d,HIGH);

digitalWrite(e, LOW);

digitalWrite(f,HIGH);

digitalWrite(g,HIGH);

digitalWrite(dp,LOW);

}

void digital_6(void) // display number 6

{

unsigned char j;

for(j=7;j<=11;j++)

digitalWrite(j,HIGH);

digitalWrite(c,HIGH);

digitalWrite(dp,LOW);
```

```
digitalWrite(b,LOW);

}

void digital_7(void) // display number 7

{

unsigned char j;

for(j=5;j<=7;j++)

digitalWrite(j,HIGH);

digitalWrite(dp,LOW);

for(j=8;j<=11;j++)

digitalWrite(j,LOW);

}

void digital_8(void) // display number 8

{

unsigned char j;

for(j=5;j<=11;j++)

digitalWrite(j,HIGH);

digitalWrite(dp,LOW);

}

void digital_9(void) // display number 5

{

unsigned char j;

digitalWrite(a,HIGH);
```

```
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e, LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}

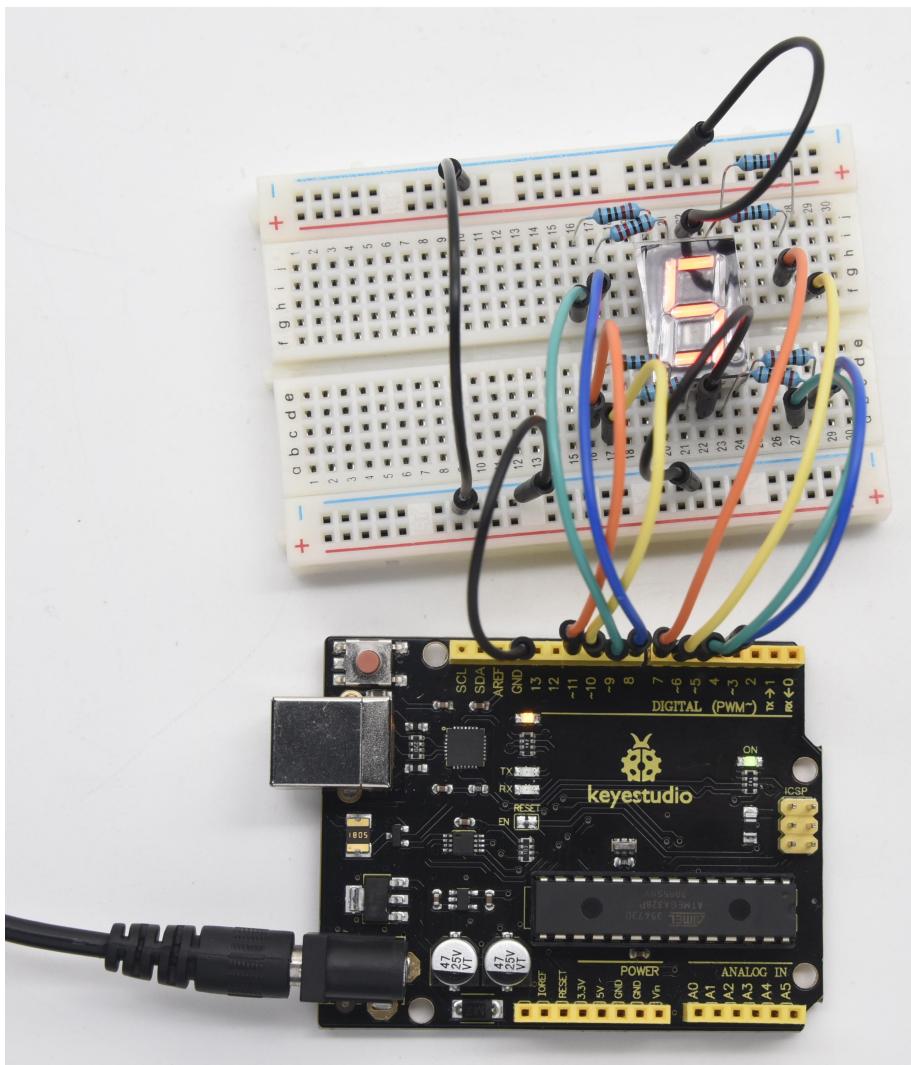
void setup()
{
int i;// set variable
for(i=4;i<=11;i++)
pinMode(i,OUTPUT);// set pin 4-11as "output"
}

void loop()
{
while(1)
{
digital_0();// display number 0
delay(1000);// wait for 1s
digital_1();// display number 1
delay(1000);// wait for 1s
}
```

```
digital_2();// display number 2  
delay(1000); // wait for 1s  
  
digital_3();// display number 3  
delay(1000); // wait for 1s  
  
digital_4();// display number 4  
delay(1000); // wait for 1s  
  
digital_5();// display number 5  
delay(1000); // wait for 1s  
  
digital_6();// display number 6  
delay(1000); // wait for 1s  
  
digital_7();// display number 7  
delay(1000); // wait for 1s  
  
digital_8();// display number 8  
delay(1000); // wait for 1s  
  
digital_9();// display number 9  
delay(1000); // wait for 1s  
  
}  
//////////
```

6. Test Result

LED segment display will show the number from 0 to 9.



Project 18: 4-digit LED Segment Display



1. Introduction

In this experiment, we use an Arduino to drive a common cathode, 4-digit, 7-segment LED display.

For LED display, current-limiting resistors are indispensable.

There are two wiring methods for Current-limiting resistor.

One is to connect one resistor for each cathode end, 4 in total for d1-d4 cathode. An advantage for this method is that it requires fewer resistors, only 4. But it cannot maintain consistent brightness, 1, the brightest; 8, the least bright.

Another method is to connect one resistor to each pin. It guarantees consistent brightness, but requires more resistors.

In this experiment, we use 8 Resistors (220Ω). We use 220Ω Resistors because of no 100Ω resistor available. If you use 100Ω , the displaying is more brighter.

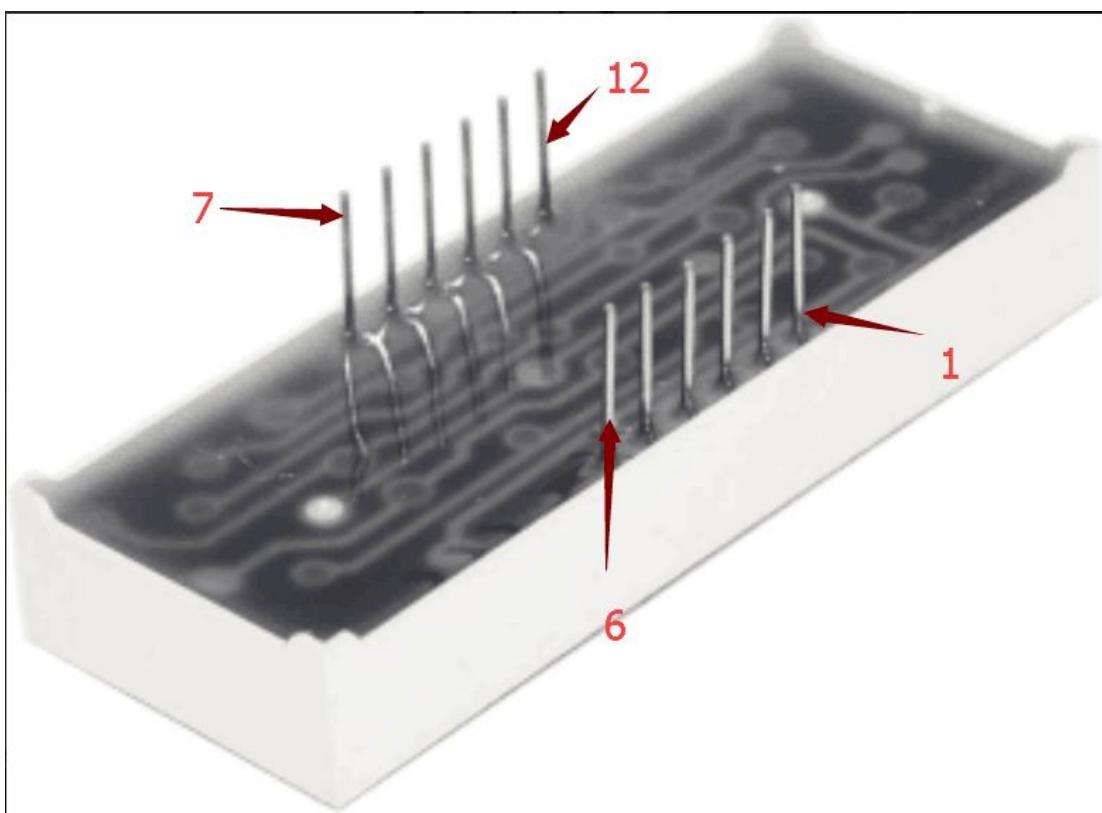
2.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- 4-digit LED Segment Display*1
- 220Ω Resistor*8
- Breadboard*1
- Breadboard Jumper Wires *several

- USB cable*1

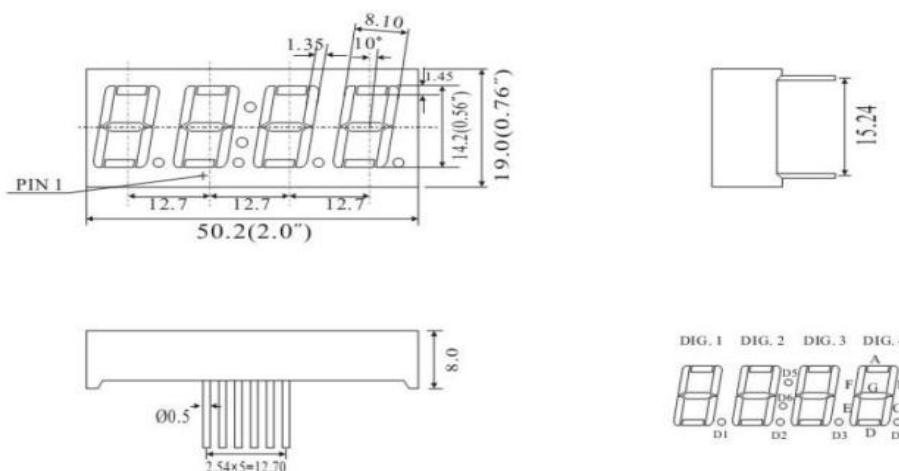
3.The display principle of the 4-digit display

For 4-digit display, there are 12 pins in total. When you place the decimal point downward, the pin on the lower left part is refer to as 1, the upper left part 12. Shown below.



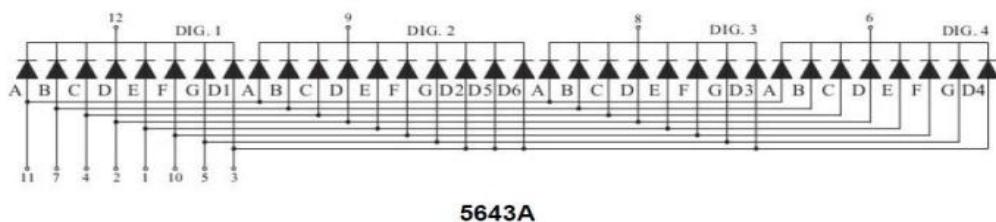
Package Dimensions

CPS05643AB

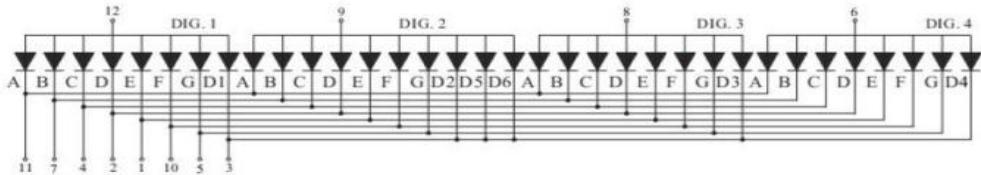


UNIT: MM(INCH) TOLERANCE: $\pm 0.25(0.01")$

Internal Circuit Diagram



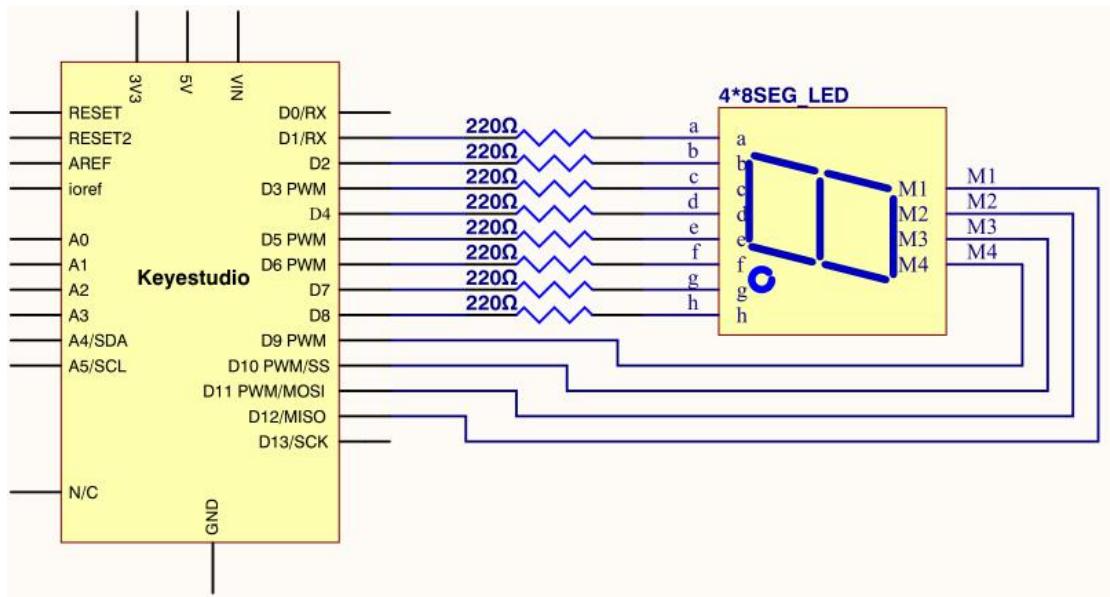
5643A



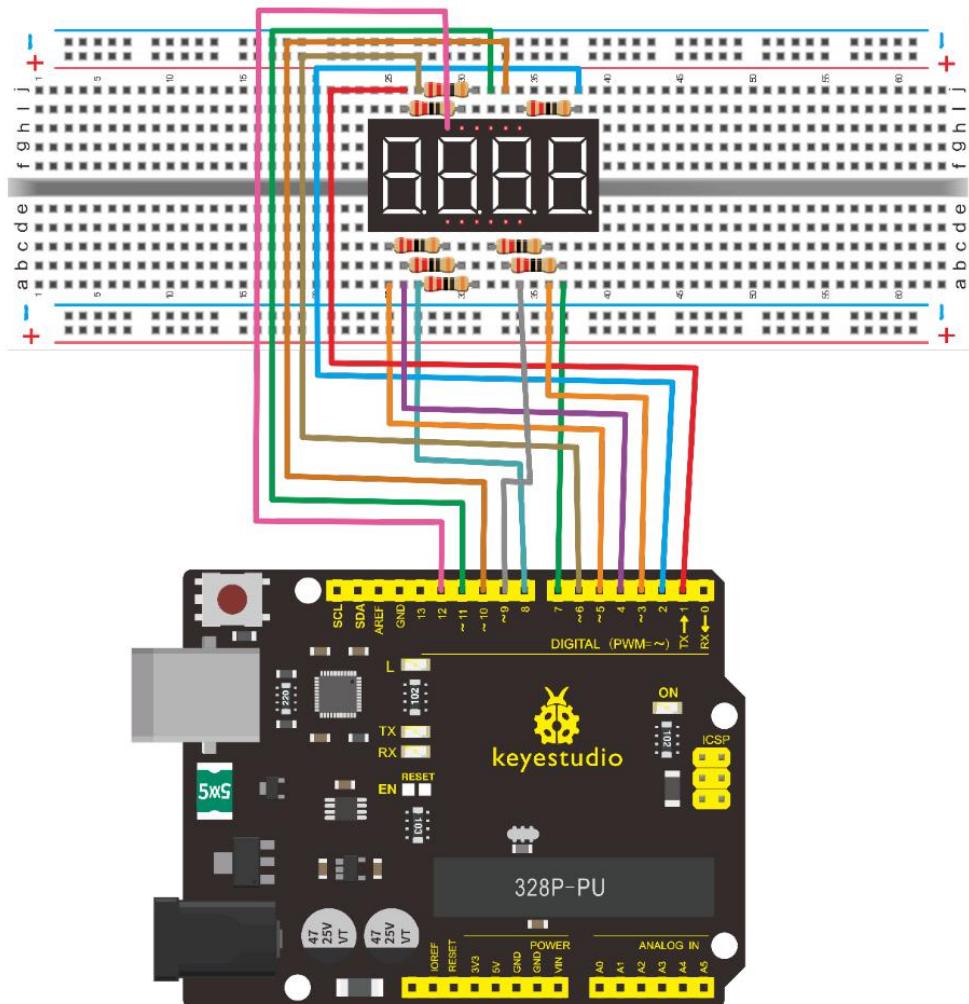
5643B

Four Digits Displays Series

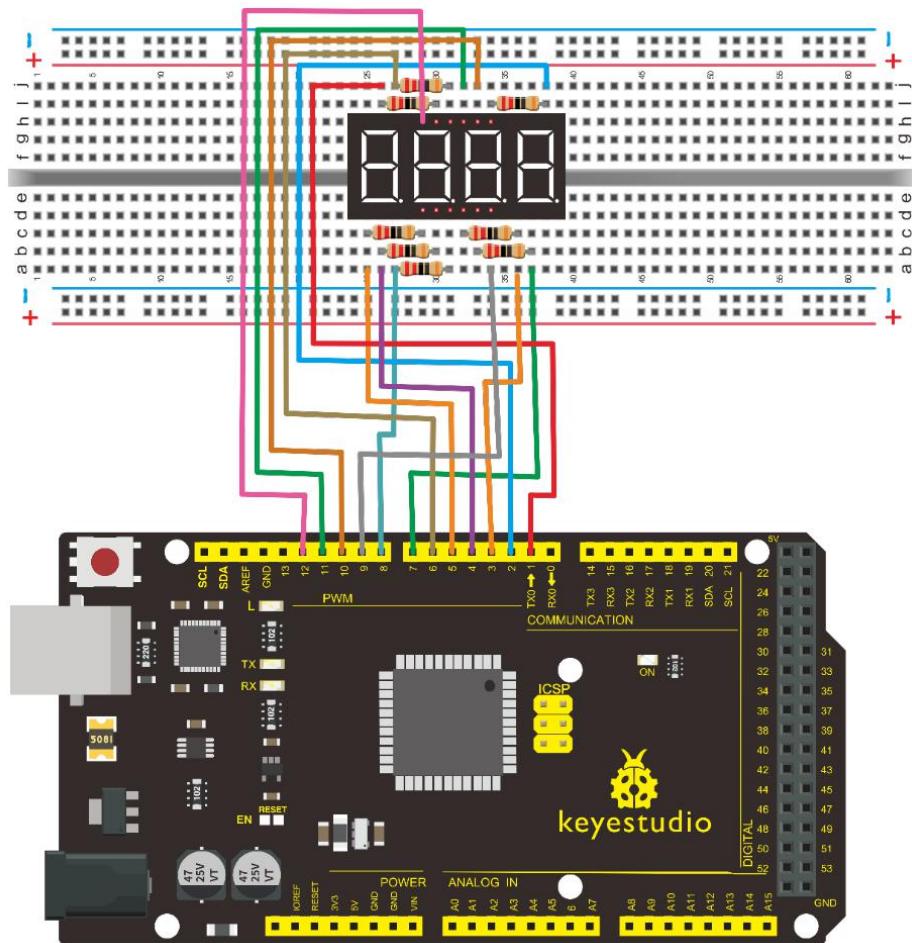
4.Circuit Connection



Connection for V4.0



Connection for 2560 R3



5. Sample Code

```
/*
```

keyestudio super learning kit

Project 18

4-digit LED Segment Display

<http://www.keyestudio.com>

```
*/  
  
// display 1234  
  
// select pin for cathode  
  
int a = 1;  
  
int b = 2;  
  
int c = 3;  
  
int d = 4;  
  
int e = 5;  
  
int f = 6;  
  
int g = 7;  
  
int dp = 8;  
  
// select pin for anode  
  
int d4 = 9;  
  
int d3 = 10;  
  
int d2 = 11;  
  
int d1 = 12;  
  
// set variable  
  
long n = 1230;  
  
int x = 100;  
  
int del = 55; // fine adjustment for clock  
  
  
  
void setup()
```

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

```
}

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

{

    switch(n)

    {

        case 1:

            digitalWrite(d1,LOW);

            digitalWrite(d2, HIGH);

            digitalWrite(d3, HIGH);

            digitalWrite(d4, HIGH);

            break;

        case 2:

            digitalWrite(d1, HIGH);

            digitalWrite(d2, LOW);

            digitalWrite(d3, HIGH);

            digitalWrite(d4, HIGH);

            break;

        case 3:

            digitalWrite(d1,HIGH);

            digitalWrite(d2, HIGH);

            digitalWrite(d3, LOW);
```

```
    digitalWrite(d4, HIGH);

    break;

case 4:

    digitalWrite(d1, HIGH);

    digitalWrite(d2, HIGH);

    digitalWrite(d3, HIGH);

    digitalWrite(d4, LOW);

    break;

default :

    digitalWrite(d1, HIGH);

    digitalWrite(d2, HIGH);

    digitalWrite(d3, HIGH);

    digitalWrite(d4, HIGH);

    break;

}

}

void Num_0()

{

    digitalWrite(a, HIGH);

    digitalWrite(b, HIGH);

    digitalWrite(c, HIGH);

    digitalWrite(d, HIGH);
```

```
    digitalWrite(e, HIGH);  
    digitalWrite(f, HIGH);  
    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, HIGH);  
    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, HIGH);  
    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, HIGH);  
    digitalWrite(g, HIGH);  
    digitalWrite(dp,LOW);  
}
```

```
void Num_5()
```

```
{  
    digitalWrite(a, HIGH);  
    digitalWrite(b, LOW);  
    digitalWrite(c, HIGH);  
    digitalWrite(d, HIGH);  
    digitalWrite(e, LOW);  
    digitalWrite(f, HIGH);  
    digitalWrite(g, HIGH);  
    digitalWrite(dp,LOW);
```

```
}
```

```
void Num_6()
```

```
{  
    digitalWrite(a, HIGH);  
    digitalWrite(b, LOW);  
    digitalWrite(c, HIGH);  
    digitalWrite(d, HIGH);
```

```
    digitalWrite(e, HIGH);  
    digitalWrite(f, HIGH);  
    digitalWrite(g, HIGH);  
    digitalWrite(dp,LOW);  
}
```

```
void Num_7()  
{
```

```
    digitalWrite(a, HIGH);  
    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, HIGH);  
    digitalWrite(b, HIGH);  
    digitalWrite(c, HIGH);  
    digitalWrite(d, HIGH);
```

```
digitalWrite(e, HIGH);
digitalWrite(f, HIGH);
digitalWrite(g, HIGH);
digitalWrite(dp,LOW);

}

void Num_9()

{
    digitalWrite(a, HIGH);
    digitalWrite(b, HIGH);
    digitalWrite(c, HIGH);
    digitalWrite(d, HIGH);
    digitalWrite(e, LOW);
    digitalWrite(f, HIGH);
    digitalWrite(g, HIGH);
    digitalWrite(dp,LOW);

}

void Clear() // clear the screen

{
    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)// select number

{
switch(n)

{
case 0:Num_0();
break;
case 1:Num_1();
break;
case 2:Num_2();
break;
case 3:Num_3();
break;
case 4:Num_4();
break;
case 5:Num_5();
break;
case 6:Num_6();
```

```
        break;

    case 7:Num_7();

    break;

    case 8:Num_8();

    break;

    case 9:Num_9();

    break;

    default:Clear();

    break;

}

}

void Display(unsigned char x, unsigned char Number)// take x as
coordinate and display number

{

WeiXuan(x);

pickNumber(Number);

delay(1);

Clear() ; // clear the screen

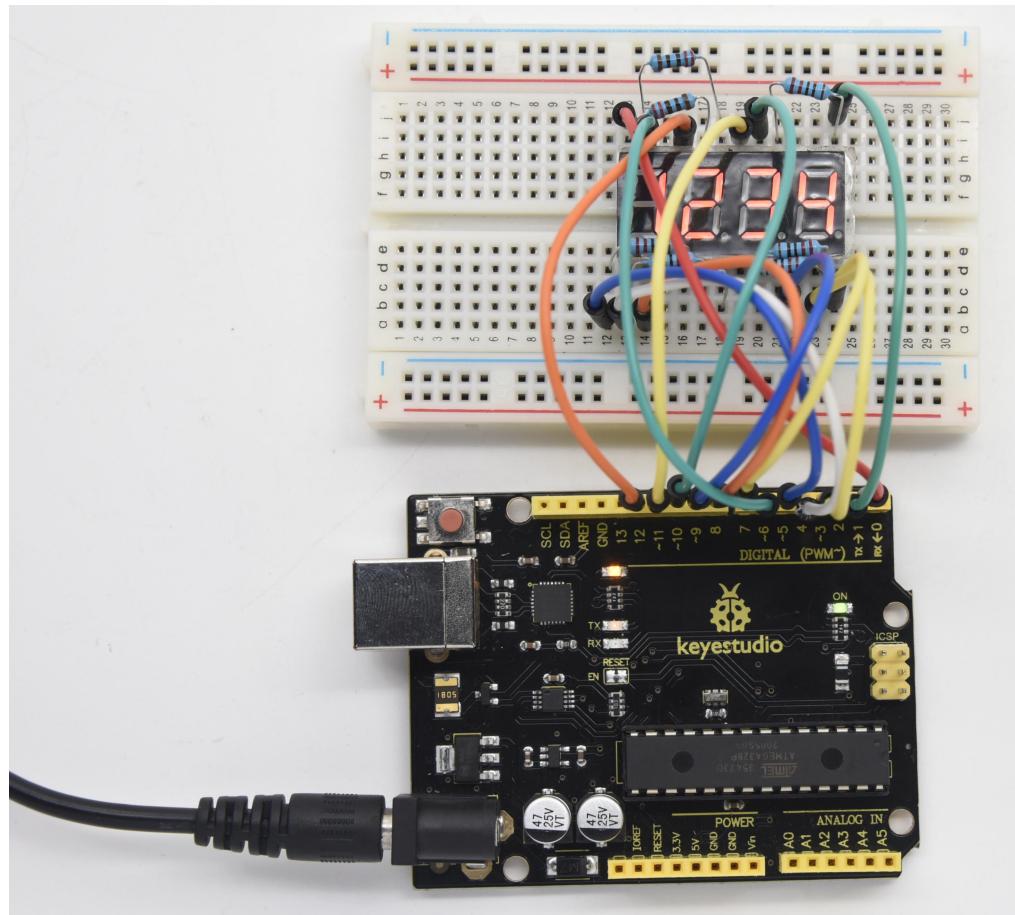
}

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

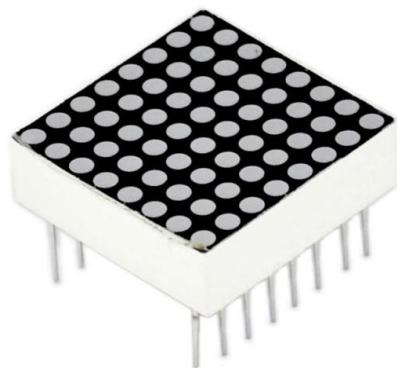
6.Test Result

Download the above code to the controller board, you can see the LED display shows the number 1234.

Note if it's not displaying correctly, check the wiring.



Project 19: 8*8 LED Matrix



1.Introduction

LED dot-matrix display can meet the needs of different applications, thus has a broad development prospect. With low-voltage scanning, LED dot-matrix has some advantages such as power saving, long service life, low cost, high brightness, wide angle of view, long visual range, waterproof, and numerous specifications.

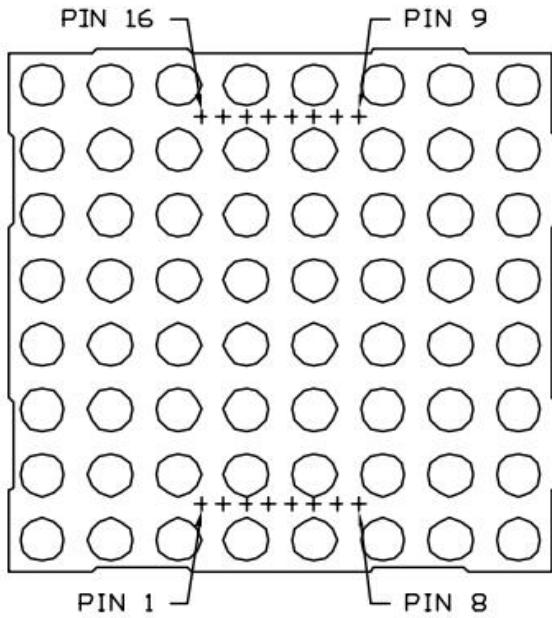
This project, we will conduct an LED dot-matrix experiment to experience its charm firsthand.

2.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- 1 * 8*8 Dot Matrix
- 8 * Resistor (220Ω)
- 1 * Breadboard
- 1 * USB Cable
- Several* Jumper Wires

3. The display principle of the 8*8 dot-matrix

The external view of a dot-matrix is shown as follows



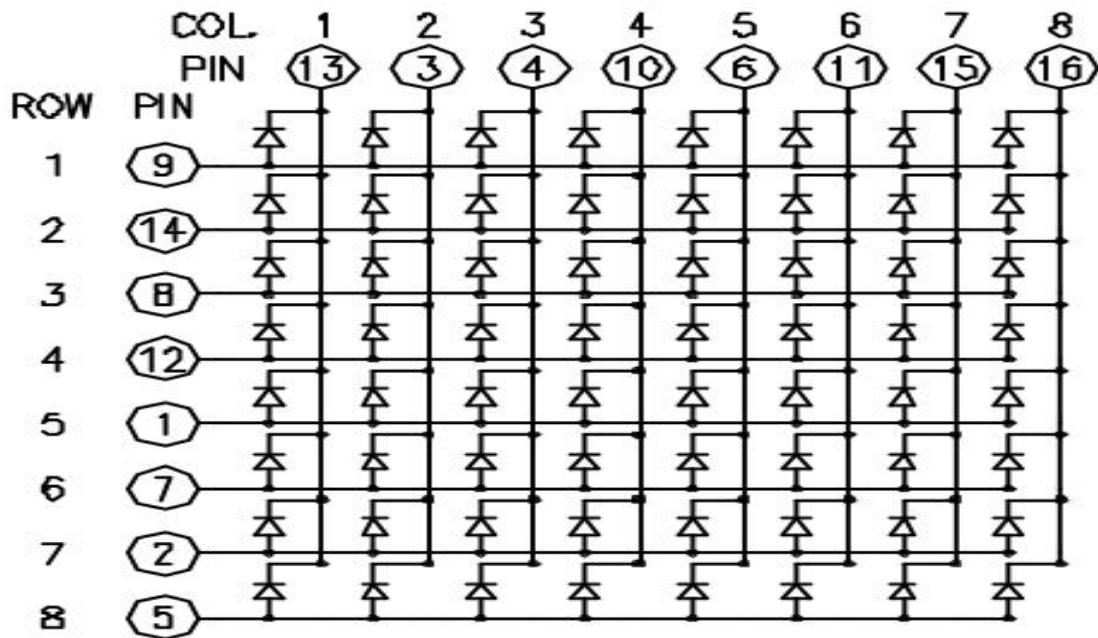
The 8*8 dot-matrix is made up of sixty-four LEDs, and each LED is placed at the cross point of a row and a column.

When the electrical level of a certain row is 1 and the electrical level of a certain column is 0, the corresponding LED will lighten. If you want to light the LED on the first dot, you should set pin 9 to high level and pin 13 to low level.

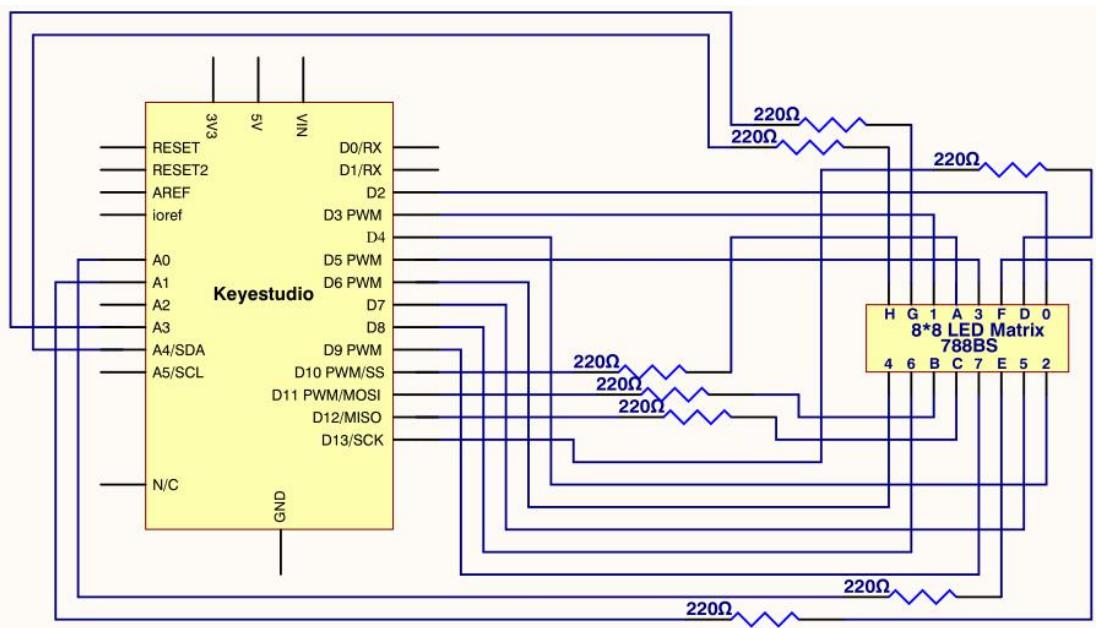
If you want to light LEDs on the first row, you should set pin 9 to high level and pins 13, 3, 4, 10, 6, 11, 15 and 16 to low level.

If you want to light the LEDs on the first column, set pin 13 to low level and pins 9, 14, 8, 12, 1, 7, 2 and 5 to high level.

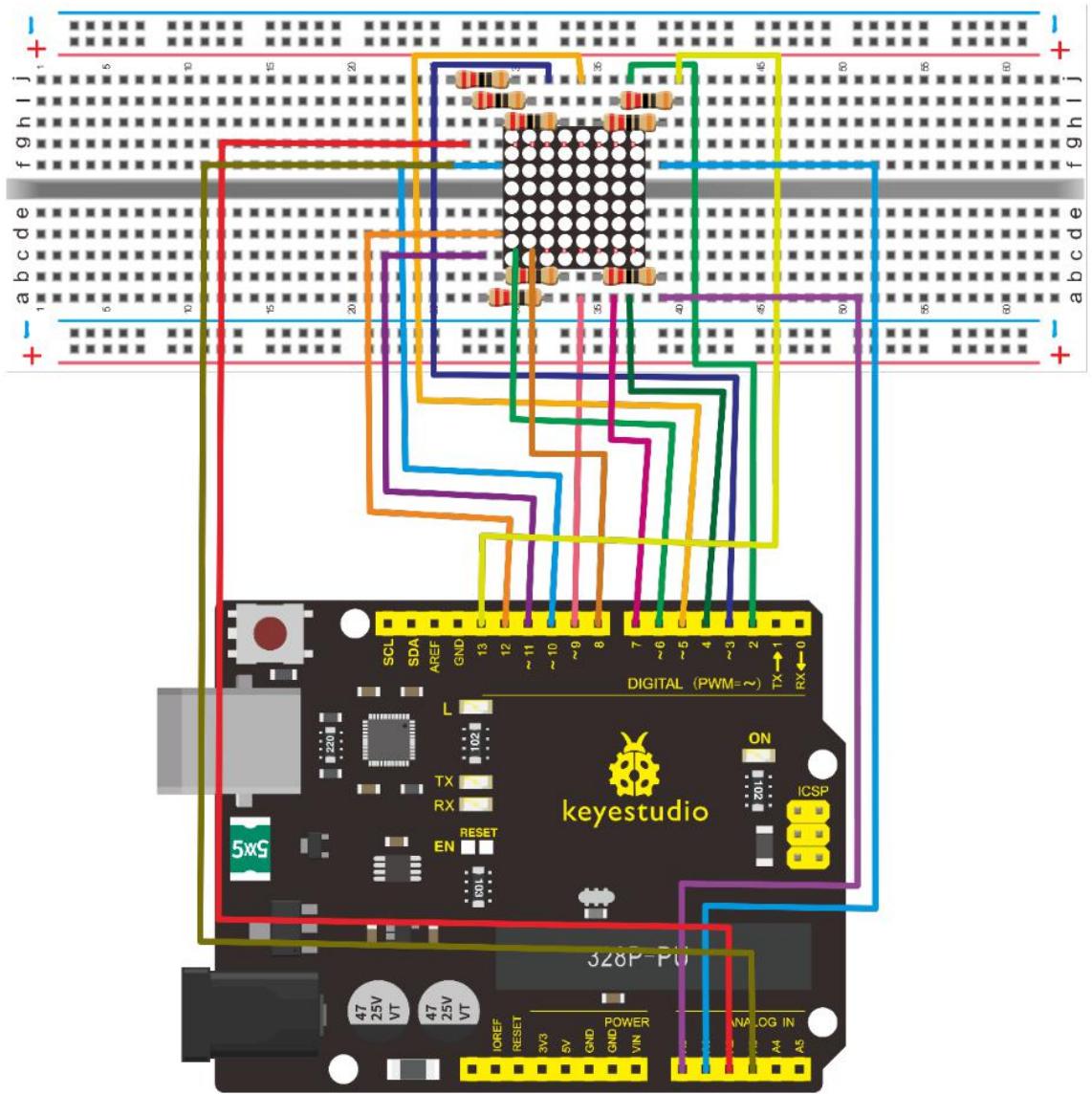
The internal view of a dot-matrix is shown as follows



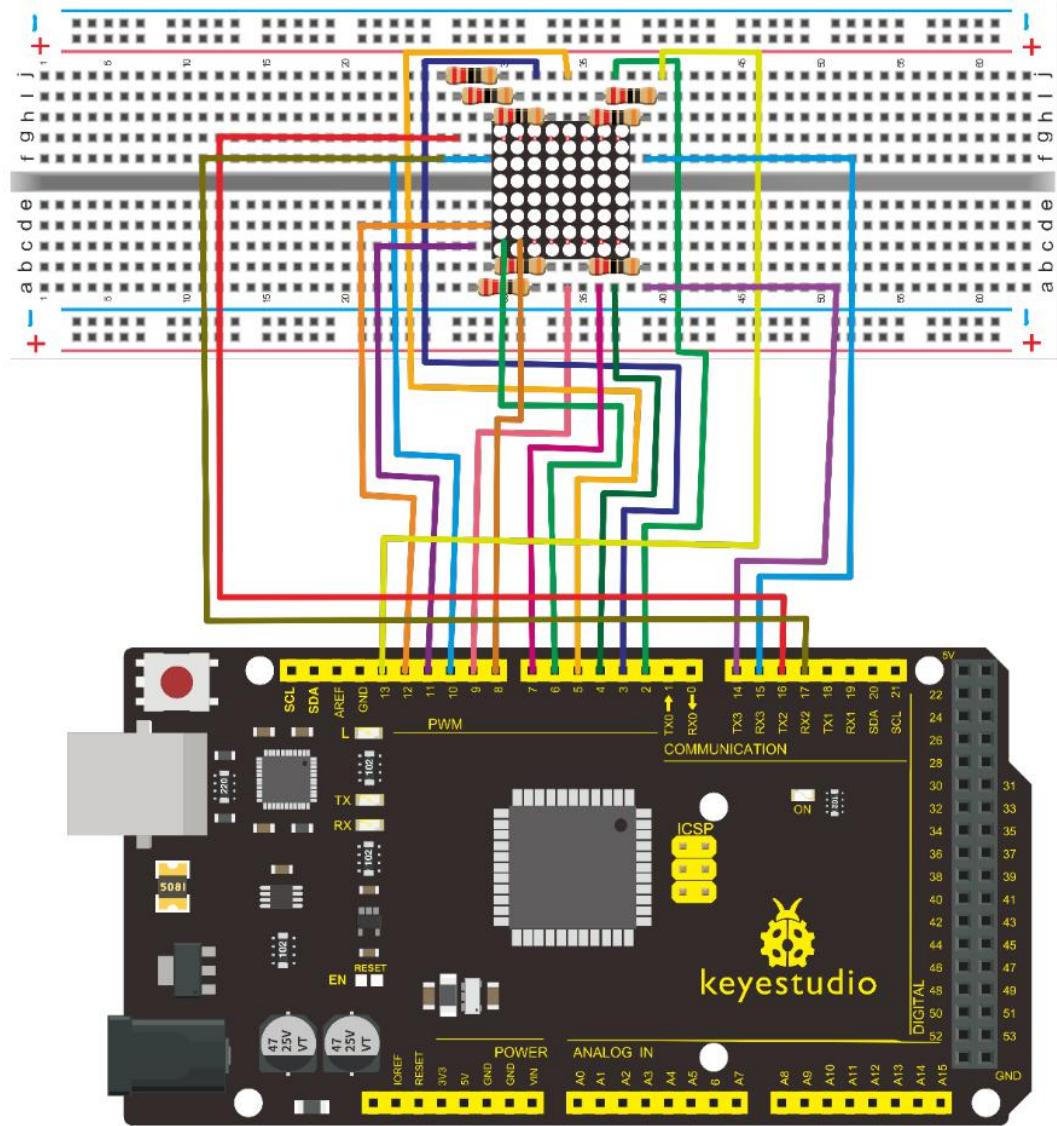
4. Circuit Connection



Connection for V4.0



Connection for 2560 R3



5. Sample Code for displaying “0”

```
/*
keyestudio super learning kit
Project 19
8*8 dot-matrix
```

<http://www.keyestudio.com>

```
*/  
// set an array to store character of "0"  
unsigned char  
Text[]={0x00,0x1c,0x22,0x22,0x22,0x22,0x22,0x1c};  
void Draw_point(unsigned char x,unsigned char y)// point  
drawing function  
{ clear_();  
    digitalWrite(x+2, HIGH);  
    digitalWrite(y+10, LOW);  
    delay(1);  
}  
void show_num(void)// display function, call point drawing  
function  
{  
    unsigned char i,j,data;  
    for(i=0;i<8;i++)  
    {  
        data=Text[i];  
        for(j=0;j<8;j++)  
        {  
            if(data & 0x01)Draw_point(j,i);  
            data>>=1;  
        }  
    }  
}
```

```
        }
    }
}

void setup(){
int i = 0 ;
for(i=2;i<18;i++)
{
    pinMode(i, OUTPUT);
}
clear_();
}

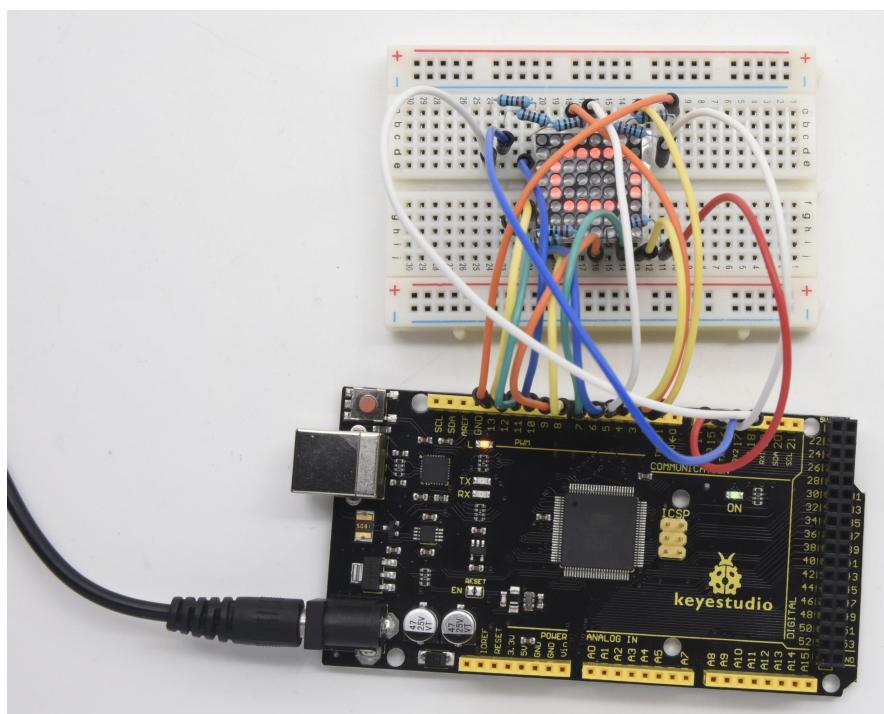
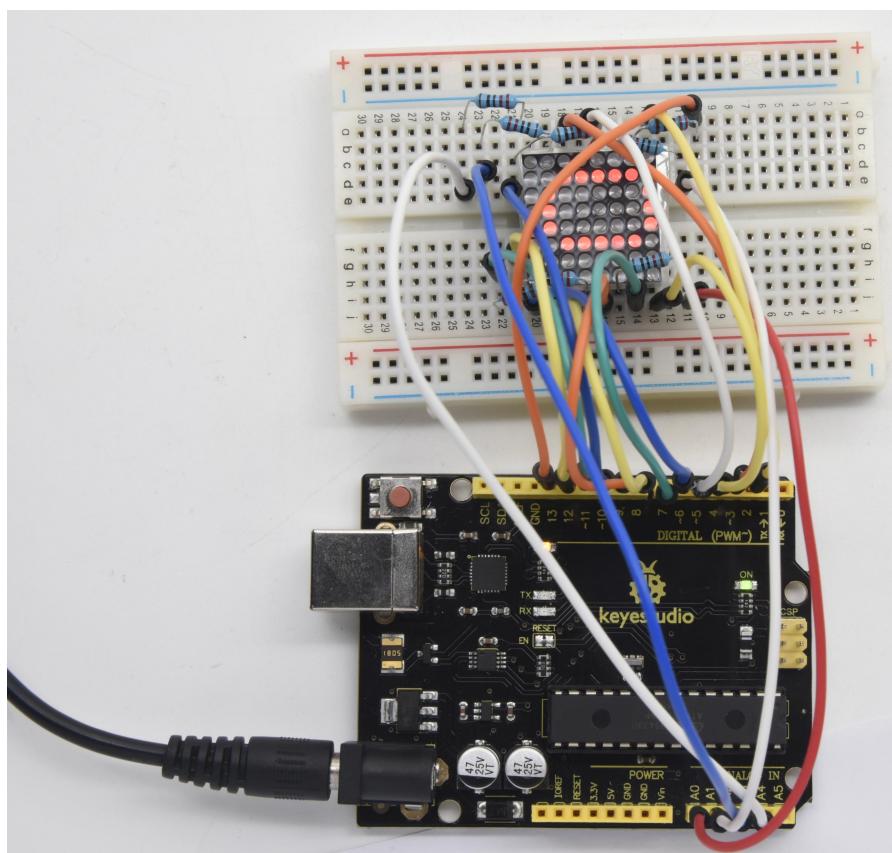
void loop()
{ show_num();
}

void clear_(void)// clear screen
{for(int i=2;i<10;i++)
    digitalWrite(i, LOW);
for(int i=0;i<8;i++)
    digitalWrite(i+10, HIGH);
}

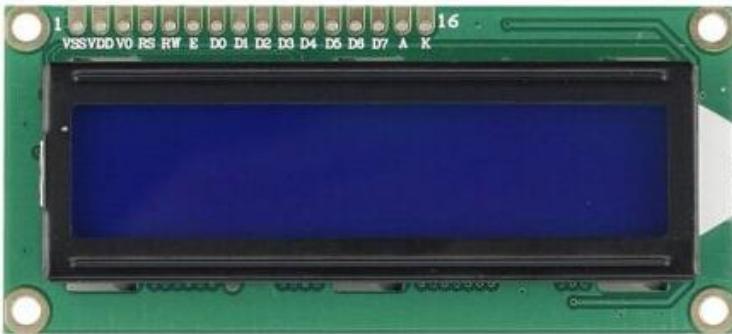
///////////////////////////////
```

6. Test Result

Burn the program into V4.0 board, the dot-matrix will display
0.



Project 20: 1602 LCD



1. Introduction

In this experiment, we use an Arduino to drive the 1602 LCD.

1602 LCD has wide applications. In the beginning, 1602 LCD uses a HD44780 controller.

Now, almost all 1602 LCD module uses a compatible IC, so their features are basically the same.

2. 1602LCD Main Parameters

- V4.0 Board or MEGA 2650 Board*1
- Display capacity 16 * 2 characters
- Chip operating voltage 4.5 ~ 5.5V
- Working current 2.0mA (5.0V)
- Optimum working voltage of the module is 5.0V
- Character size 2.95 * 4.35 (W * H) mm

3.Pin description of 1602 LCD

No .	Mark	Pin description	No .	Mark	Pin description
1	VSS	Power GND	9	D2	Date I/O
2	VDD	Power positive	10	D3	Date I/O
3	VL	LCD voltage bias signal	11	D4	Date I/O
4	RS	Select data/command (V/L)	12	D5	Date I/O
5	R/W	Select read/write(H/L)	13	D6	Date I/O
6	E	Enable signal	14	D7	Date I/O
7	D0	Date I/O	15	BLA	Back light power positive
8	D1	Date I/O	16	BLK	Back light power negative

4.Interface Description

1. Two power pins, one for module power, another one for back light, generally use 5V. In this project, we use 3.3V for backlight.
2. **VL** is the pin for adjusting contrast ratio. It usually connects a potentiometer(no more than $5K\Omega$) in series for its adjustment.

In this experiment, we use a $1K\Omega$ resistor. For its connection, it has two methods, namely high potential and low potential. Here, we use low potential method; connect the resistor and then the GND.
3. **RS** is a very common pin in LCD. It's a selecting pin for command/data. When the pin is in high level, it's in data mode; when it's in low level, it's in command mode.
4. **RW** pin is also very common in LCD. It's a selecting pin for read/write. When the pin is in high level, it's in read operation; if in low level, it's in write operation.
5. **E** pin is also very common in LCD. Usually, when the signal in the bus is stabilized, it sends out a positive pulse requiring read operation. When this pin is in high level, the bus is not allowed to have any change.

6. D0-D7 is 8-bit bidirectional parallel bus, used for command and data transmission.
7. BLA is anode for back light; BLK, cathode for back light.

5.Four Basic Operations for 1602LCD

Read status	inp ut	RS=L, R/W=H, E=H	outp ut	D0-D7=status word
Write command	inp ut	RS=L, R/W=H, D0-D7=command code, E=high pulse	outp ut	none
Read data	inp ut	RS=H, R/W=H, E=H	outp ut	D0-D7=data
Write data	inp ut	RS=H, R/W=L, D0-D7=data, E=high pulse	outp ut	none

6.Hardware Required

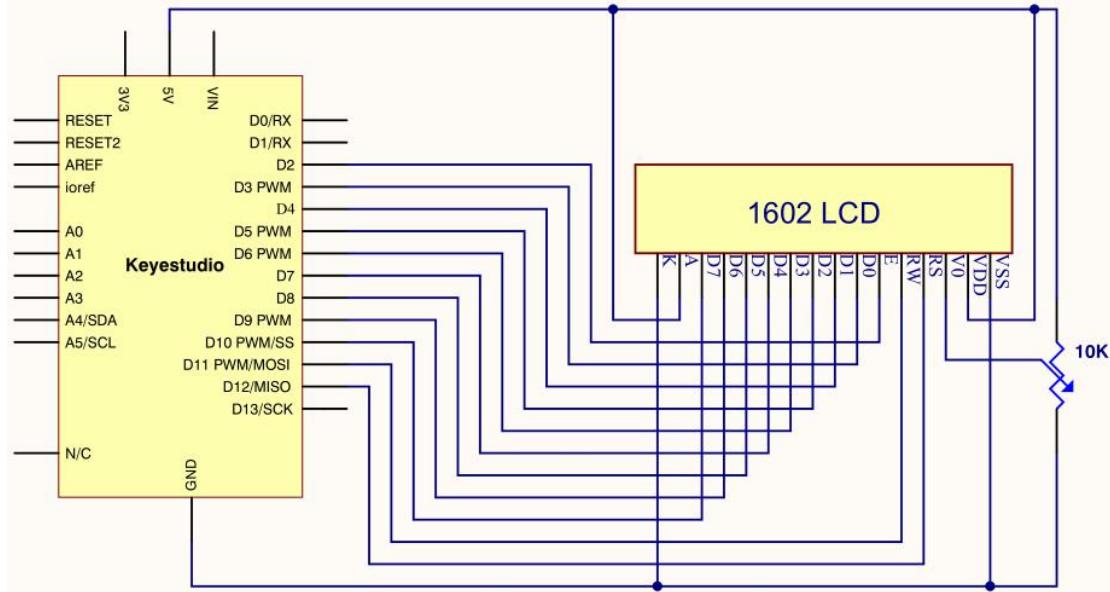
- V4.0 Board or MEGA 2650 Board*1
- 1 * 1602 LCD
- 1 * Potentiometer
- 1 * Breadboard

- 1 * USB Cable
- Several* Jumper Wires

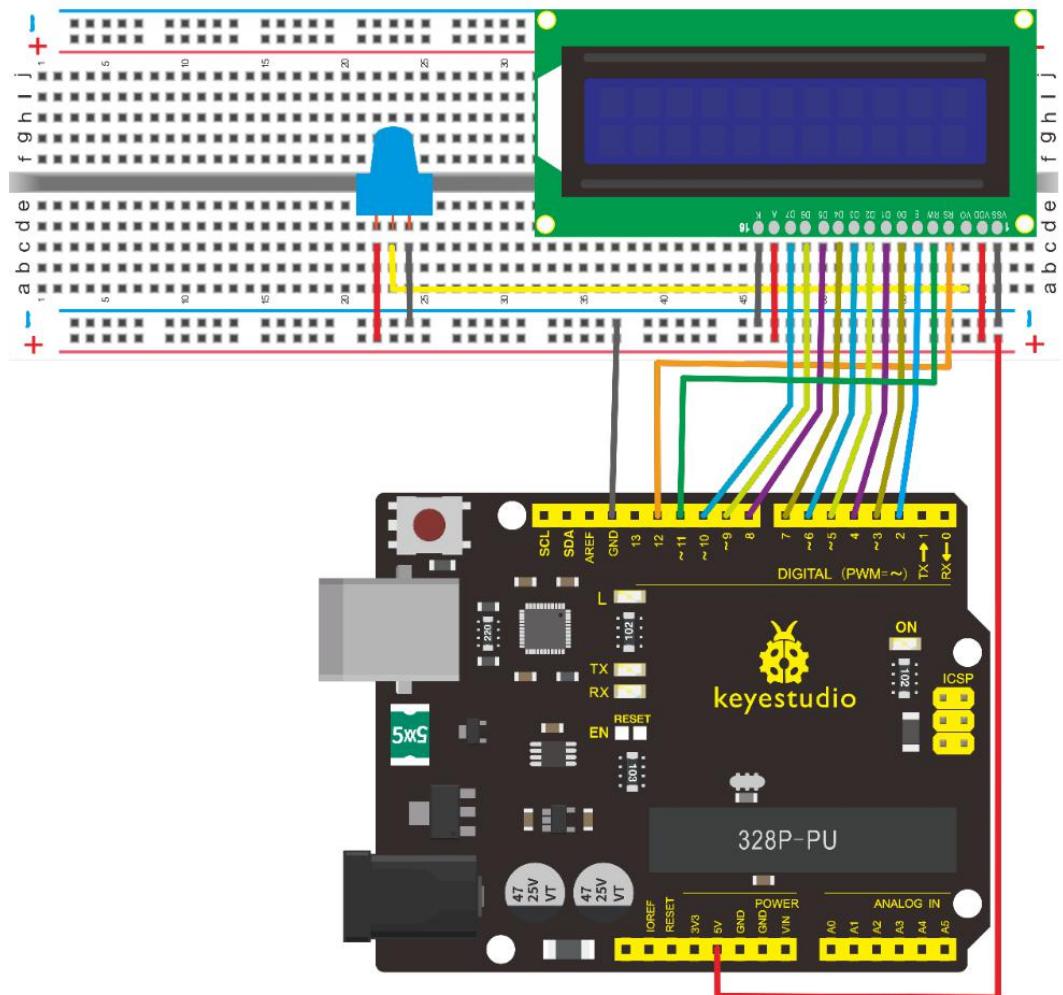
7.Connection

1602 can directly communicate with Arduino. According to the product manual, it has two connection methods, namely 8-bit connection and 4-bit connection.

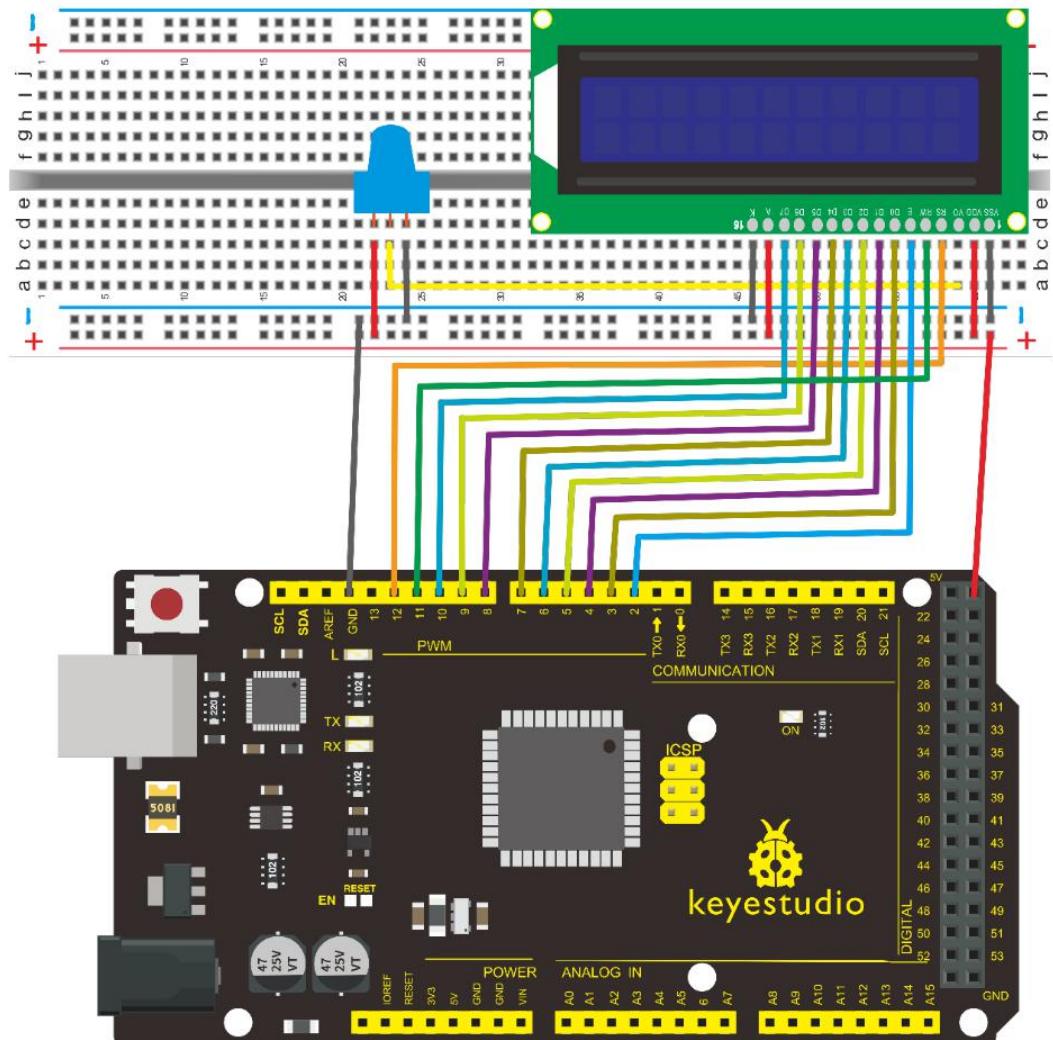
8-bit Connection Method



Connection for V4.0



Connection for 2560 R3



8. Sample Code A

/*

keyestudio super learning kit

Project 20.1

LCD1602

<http://www.keyestudio.com>

* /

```
int DI = 12;  
int RW = 11;  
int DB[] = {3, 4, 5, 6, 7, 8, 9, 10}// use array to select pin  
for bus  
int Enable = 2;  
  
  
void LcdCommandWrite(int value) {  
// define all pins  
int i = 0;  
for (i=DB[0]; i <= DI; i++) // assign value for bus  
{  
    digitalWrite(i,value & 01);// for 1602 LCD, it uses  
D7-D0( not D0-D7) for signal identification; here, it's used for  
signal inversion.  
    value >>= 1;  
}  
digitalWrite(Enable,LOW);  
delayMicroseconds(1);  
digitalWrite(Enable,HIGH);  
delayMicroseconds(1); // wait for 1ms  
digitalWrite(Enable,LOW);  
delayMicroseconds(1); // wait for 1ms
```

```
}
```

```
void LcdDataWrite(int value) {  
    // initialize all pins  
    int i = 0;  
    digitalWrite(DI, HIGH);  
    digitalWrite(RW, LOW);  
    for (i=DB[0]; i <= DB[7]; i++) {  
        digitalWrite(i,value & 01);  
        value >>= 1;  
    }  
    digitalWrite(Enable,LOW);  
    delayMicroseconds(1);  
    digitalWrite(Enable,HIGH);  
    delayMicroseconds(1);  
    digitalWrite(Enable,LOW);  
    delayMicroseconds(1); // wait for 1ms  
}  
  
void setup (void) {  
    int i = 0;  
    for (i=Enable; i <= DI; i++) {
```

```
pinMode(i,OUTPUT);

}

delay(100);

// initialize LCD after a brief pause

// for LCD control

LcdCommandWrite(0x38); // select as 8-bit interface, 2-line
display, 5x7 character size

delay(64);

LcdCommandWrite(0x38); // select as 8-bit interface, 2-line
display, 5x7 character size

delay(50);

LcdCommandWrite(0x38); // select as 8-bit interface, 2-line
display, 5x7 character size

delay(20);

LcdCommandWrite(0x06); // set input mode

                // auto-increment, no display of shifti
ng

delay(20);

LcdCommandWrite(0x0E); // display setup

                // turn on the monitor, cursor on, no
flickering

delay(20);
```

```
LcdCommandWrite(0x01); // clear the scree, cursor position
returns to 0
delay(100);

LcdCommandWrite(0x80); // display setup
                      // turn on the monitor, cursor on, no
flickering

delay(20);
}

void loop (void) {
    LcdCommandWrite(0x01); // clear the scree, cursor
position returns to 0
    delay(10);
    LcdCommandWrite(0x80+3);
    delay(10);
    // write in welcome message
    LcdDataWrite('W');
    LcdDataWrite('e');
    LcdDataWrite('l');
    LcdDataWrite('c');
    LcdDataWrite('o');
```

```
LcdDataWrite('m');
LcdDataWrite('e');
LcdDataWrite(' ');
LcdDataWrite('t');
LcdDataWrite('o');
delay(10);

LcdCommandWrite(0xc0+1); // set cursor position at
second line, second position

delay(10);

LcdDataWrite('k');
LcdDataWrite('e');
LcdDataWrite('y');
LcdDataWrite('e');
LcdDataWrite('s');
LcdDataWrite('t');
LcdDataWrite('u');
LcdDataWrite('d');
LcdDataWrite('i');
LcdDataWrite('o');
delay(5000);

LcdCommandWrite(0x01); // clear the screen, cursor
returns to 0
```

```
delay(10);

LcdDataWrite('I');

LcdDataWrite(' ');

LcdDataWrite('a');

LcdDataWrite('m');

LcdDataWrite(' ');

LcdDataWrite('h');

LcdDataWrite('u');

LcdDataWrite('n');

LcdDataWrite('t');

LcdDataWrite('e');

LcdDataWrite('r');

delay(3000);

LcdCommandWrite(0x02); // set mode as new characters
replay old ones, where there is no new ones remain the same

delay(10);

LcdCommandWrite(0x80+5); // set cursor position at first
line, sixth position

delay(10);

LcdDataWrite('t');

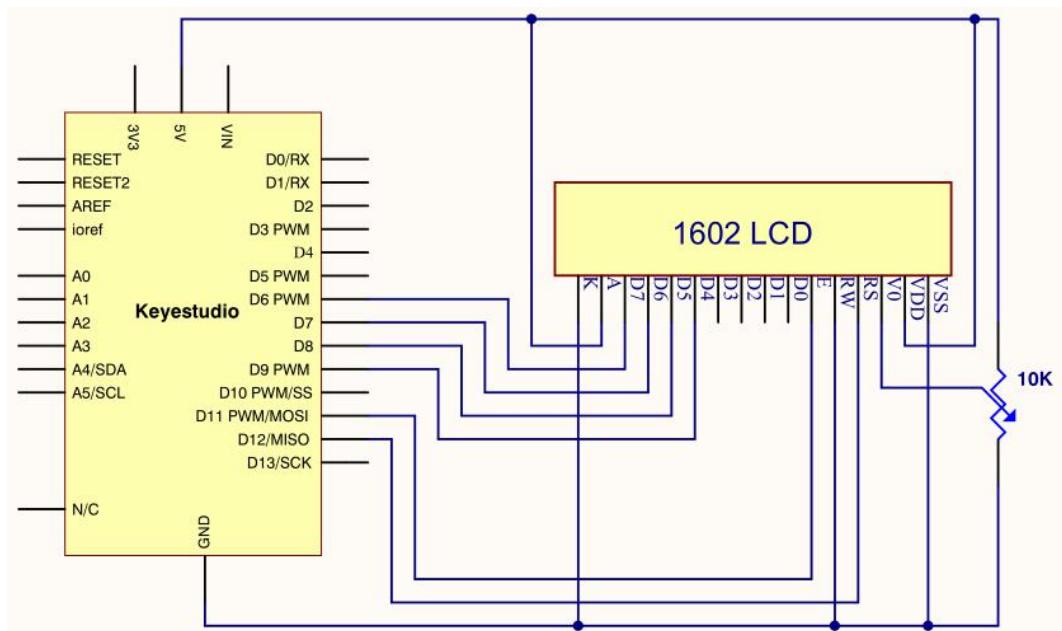
LcdDataWrite('h');

LcdDataWrite('e');
```

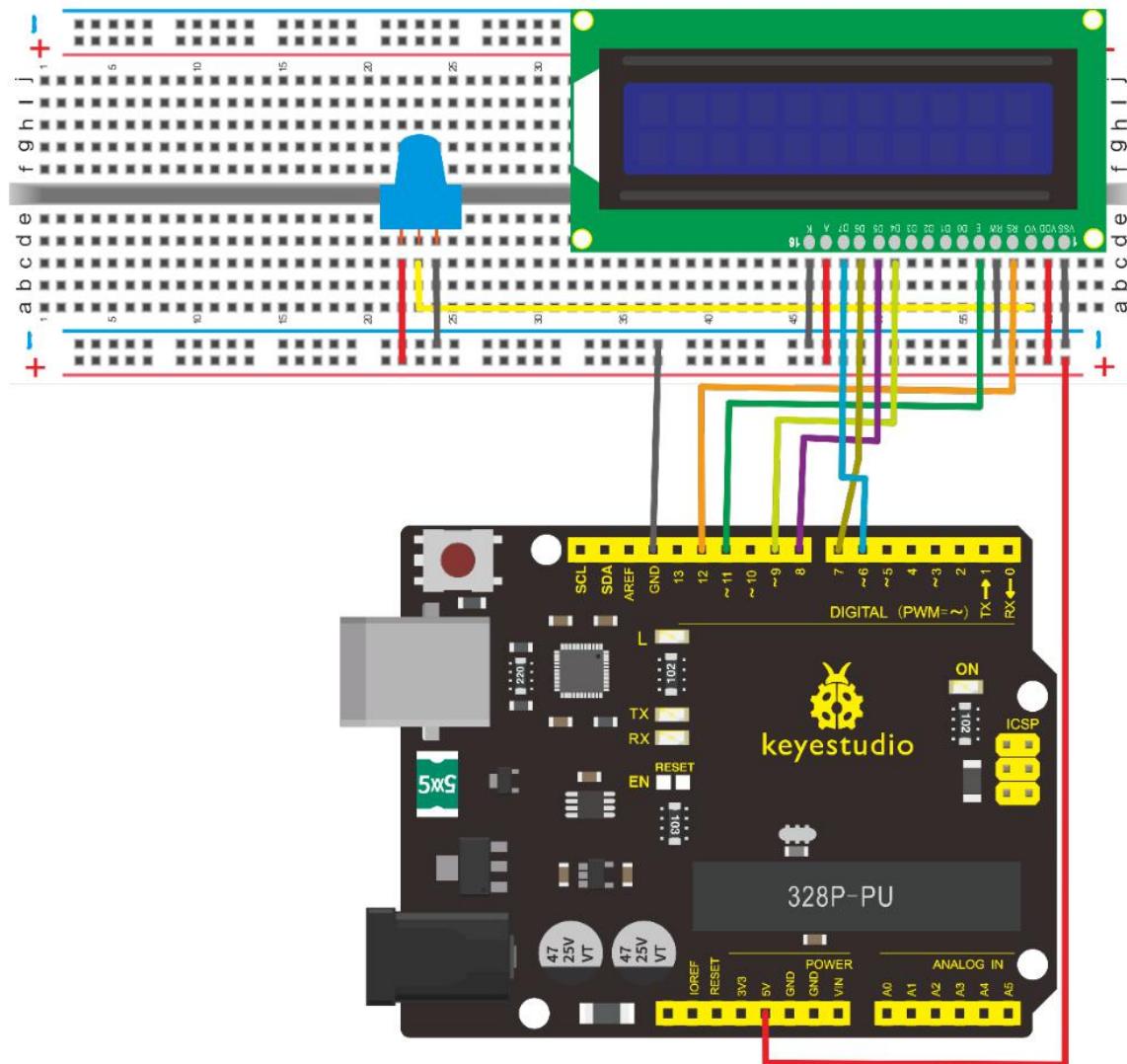
```
LcdDataWrite(' ');\nLcdDataWrite('w');\nLcdDataWrite('o');\nLcdDataWrite('r');\nLcdDataWrite('l');\nLcdDataWrite('d');\ndelay(5000);\n}\n//////////
```

4-bit Connection Method

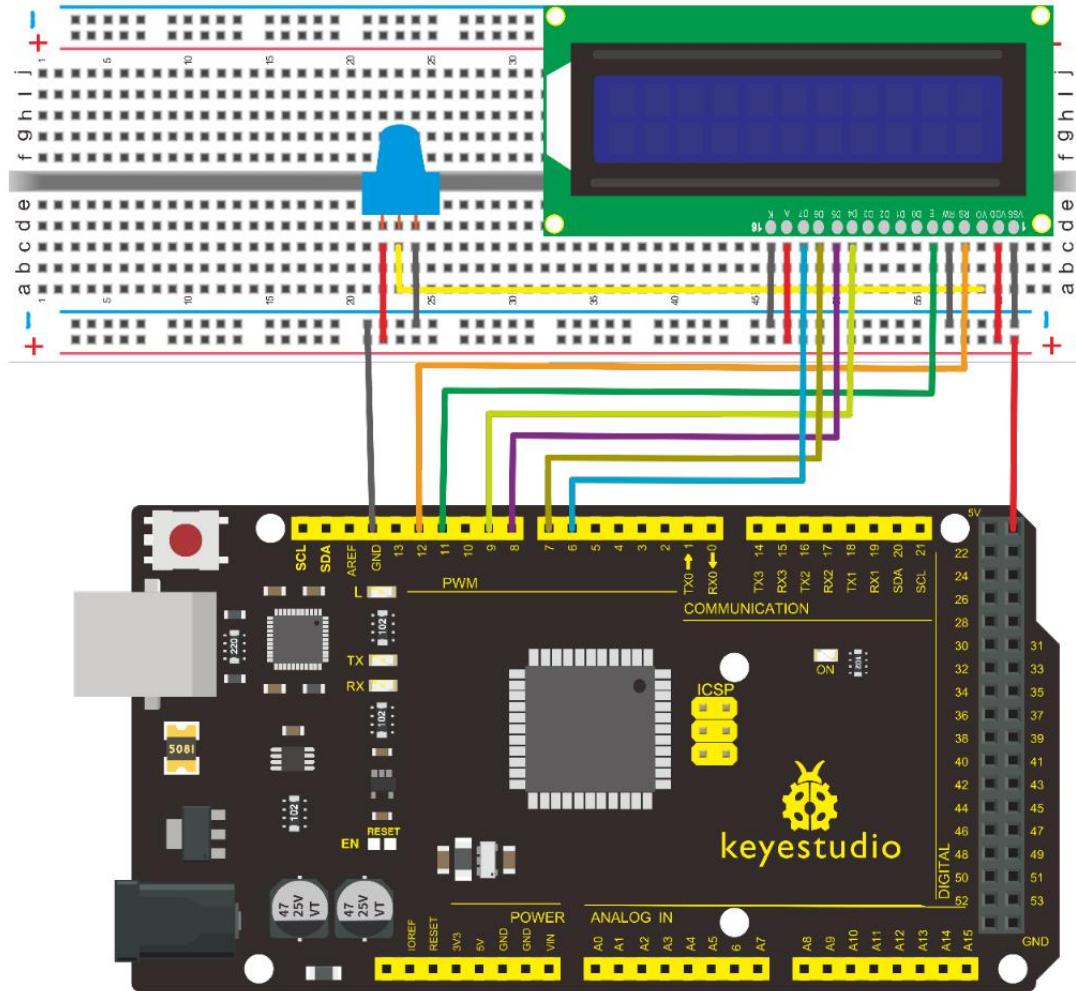
When using this module, 8-bit connection uses all the digital pins of the Arduino, leaving no pin for sensors. What then? You can use 4-bit connection.



Connection for V4.0



Connection for 2560 R3



After the connection, upload below code to the controller board and see how it goes.

9. Sample Code B

```
/*
keyestudio super learning kit
Project 20.2
LCD1602
```

<http://www.keyestudio.com>

```
*/  
/* LCD RS pin to digital pin 12  
 * LCD Enable pin to digital pin 11  
 * LCD D4 pin to digital pin 9  
 * LCD D5 pin to digital pin 8  
 * LCD D6 pin to digital pin 7  
 * LCD D7 pin to digital pin 6  
 * LCD R/W pin to ground  
 * LCD VSS pin to ground  
 * LCD VCC pin to 5V  
 * 10K resistor  
 * ends to +5V and ground  
 * wiper to LCD VO pin (pin 3)
```

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/LiquidCrystal>

```
*/  
// include the library code  
#include <LiquidCrystal.h>  
// initialize the library with the numbers of the interface pins  
LiquidCrystal lcd(12, 11, 9, 8, 7, 6);  
void setup() {
```

```
// set up the LCD's number of columns and rows
lcd.begin(16, 2);

// Print a message to the LCD.
lcd.print("hello, world!");

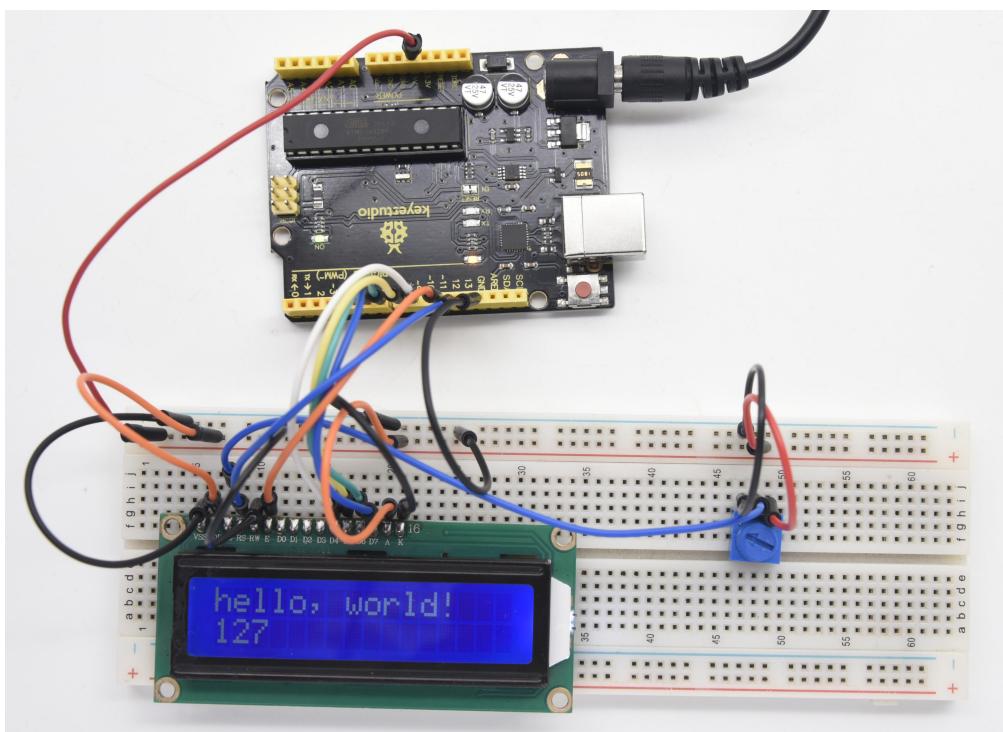
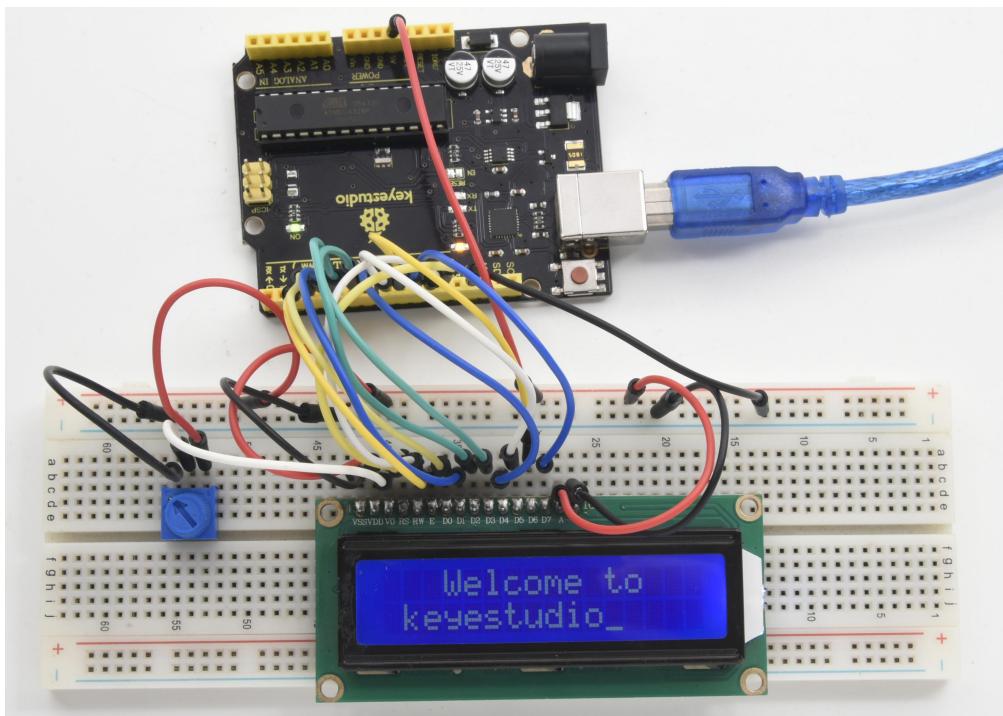
}

void loop() {
    // set the cursor to column 0, line 1
    // (note line 1 is the second row, since counting begins with
    0)
    lcd.setCursor(0, 1);
    // print the number of seconds since reset
    lcd.print(millis() / 1000);

}
///////////

```

10. Test Result



Project 21: Servo Control



1. Introduction

Servomotor is a position control rotary actuator. It mainly consists of housing, circuit board, core-less motor, gear and position sensor. For this lesson, we will introduce servo to you.

2. Working Principle

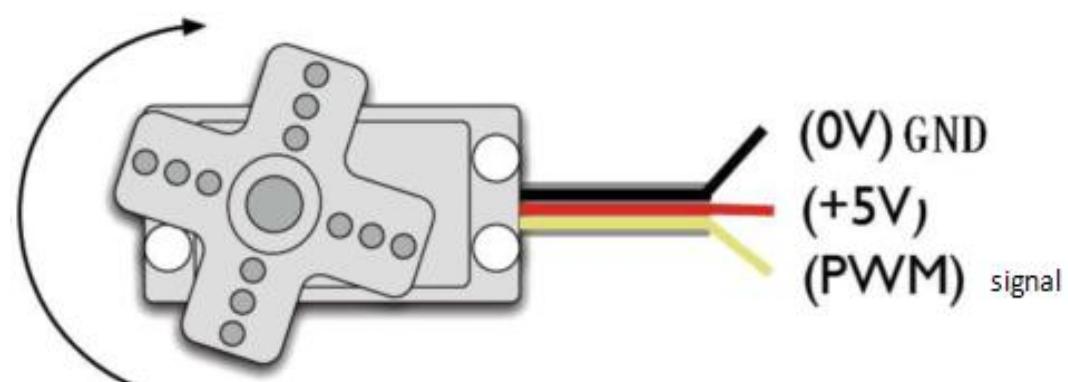
The MCU outputs a signal to the servomotor. The motor has a built-in reference circuit that gives out reference signal, cycle of 20ms and width of 1.5ms. The motor compares the acquired DC bias voltage to the voltage of the potentiometer and outputs a voltage difference. The IC on the circuit board will decide the rotate direction accordingly and drive the core-less motor. The gear then pass the force to the shaft. The sensor will determine whether it

has reached the commanded position according to the feedback signal.

Servomotors are used in control systems that require to have and maintain different angles. When the motor speed is definite, the gear will drive the potentiometer to rotate. When the voltage difference reduces to zero, the motor stops. Normally, the rotation angle range is among 0-180 degrees.

Servomotor comes with many specifications. But all of them have three connection wires, distinguished by brown, red, orange (different brand may have different color).

Brown one is for GND, red one for power positive, orange one for signal line.



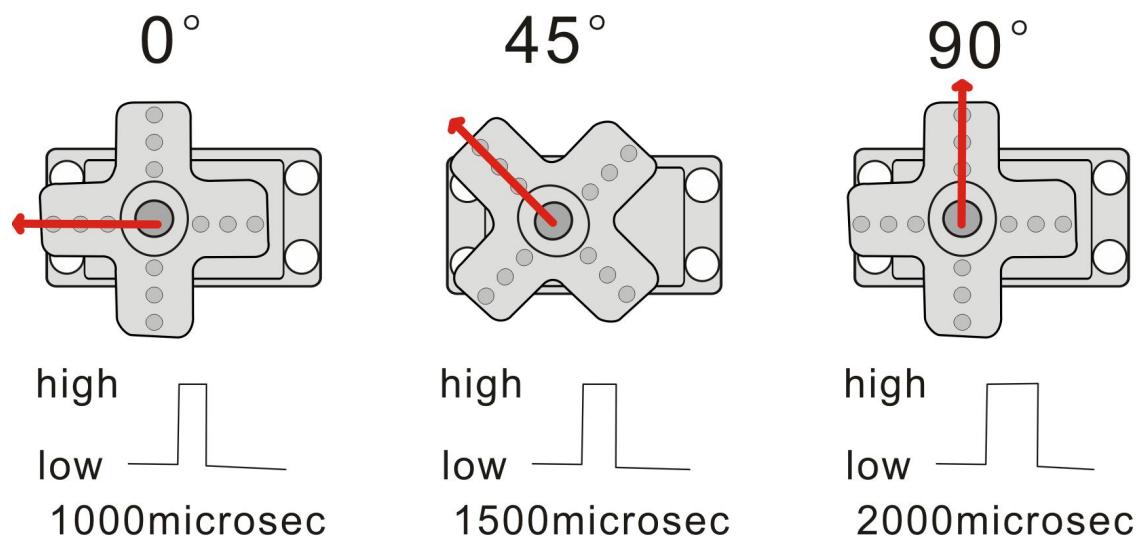
The rotation angle of the servo motor is controlled by regulating the duty cycle of the PWM(Pulse-Width Modulation)

signal.

The standard cycle of the PWM signal is 20ms (50Hz) .

Theoretically, the width is distributed between 1ms-2ms. The width corresponds the rotation angle from 0° to 90°.

But note that for different brand motor, the same signal may have different rotating angle.



With some basic knowledge, let's learn how to control a servomotor. In this experiment, you only need a servo motor and several jumper wires.

3.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- 9G Servo Motor*1
- Breadboard Jumper Wire*3

- USB cable*1

Connection & Sample Code

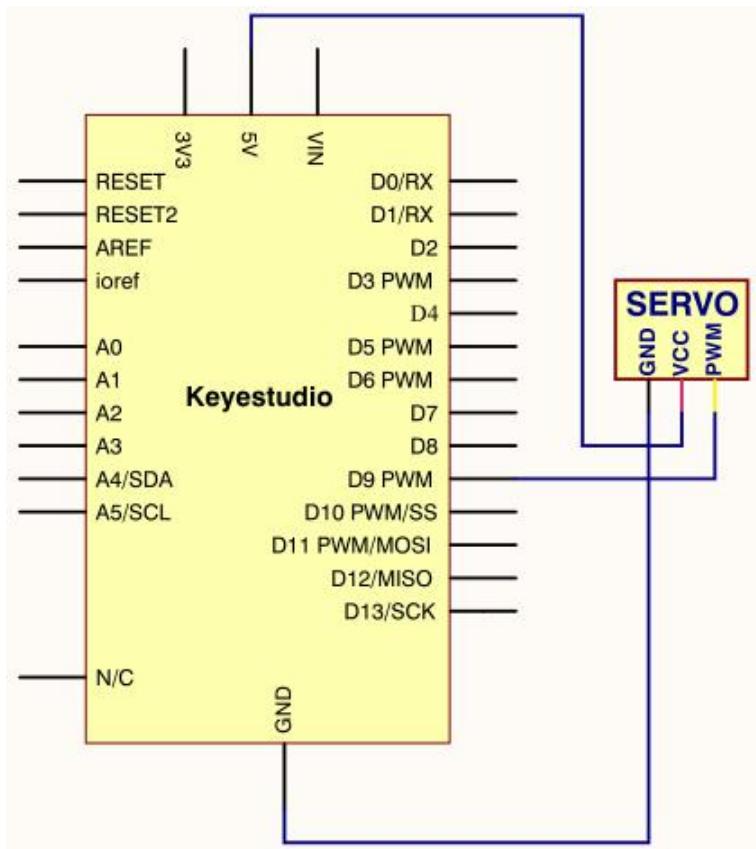
There are two ways to control a servomotor with Arduino.

One is to use a common digital sensor port of Arduino to produce square wave with different duty cycle to simulate PWM signal and use that signal to control the positioning of the motor.

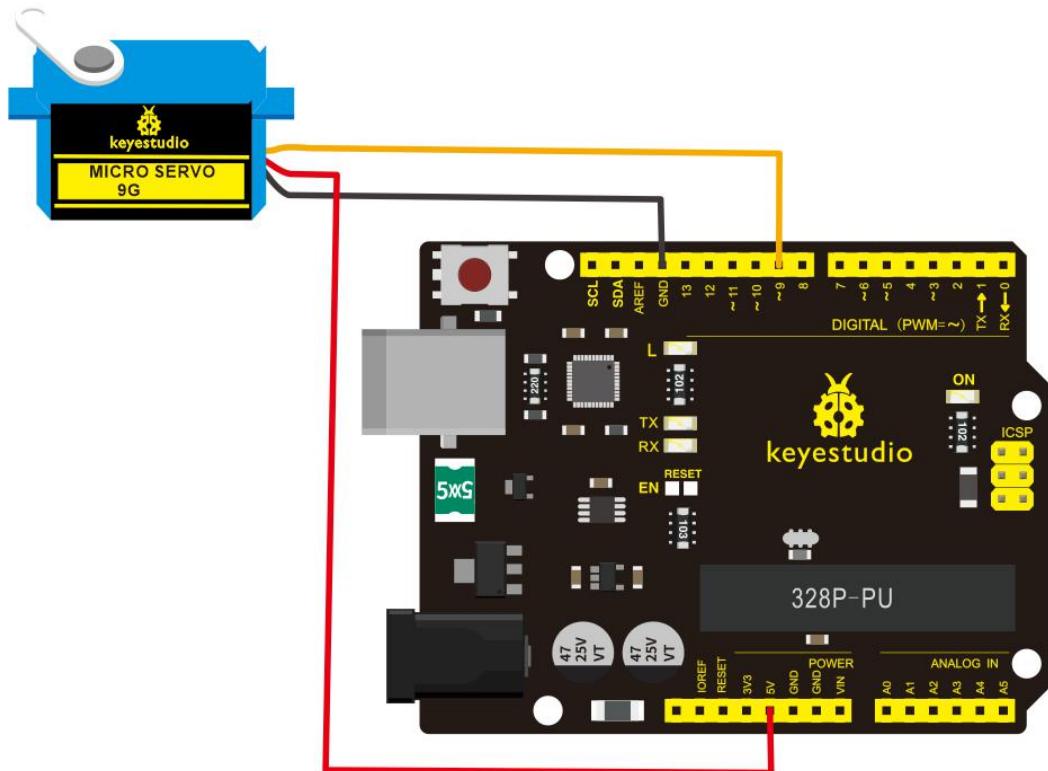
Another way is to directly use the Servo function of the Arduino to control the motor. In this way, the program will be more easier but it can only control two-contact motor because of the servo function, only digital pin 9 and 10 can be used.

The Arduino drive capacity is limited. So if you need to control more than one motor, you will need external power.

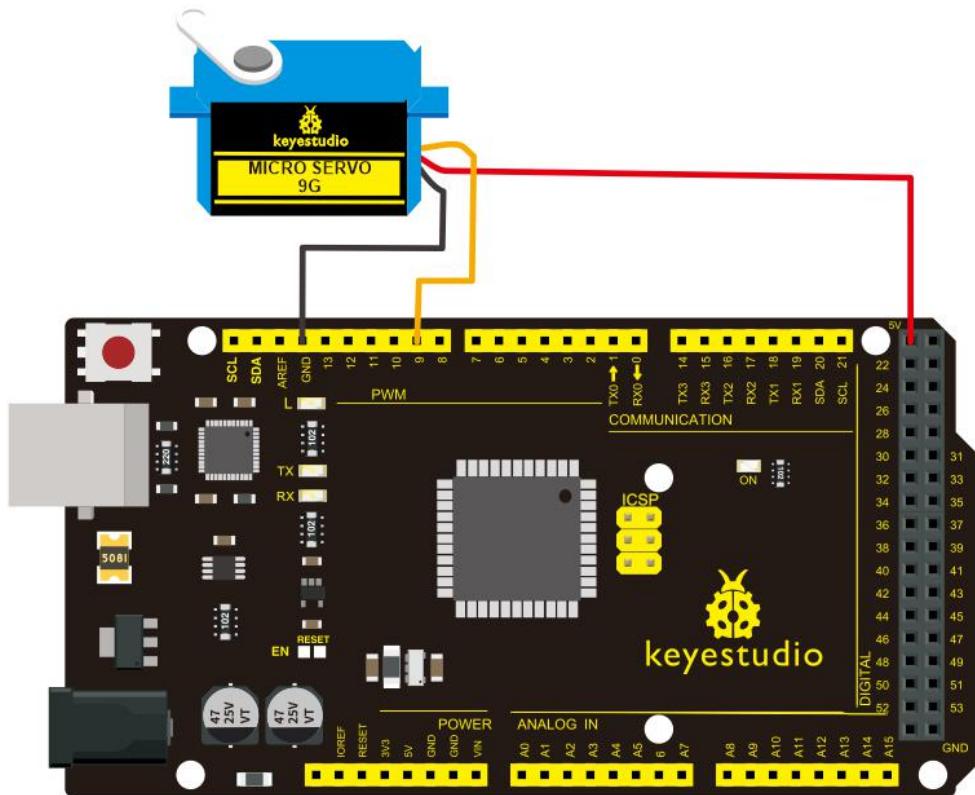
4.Method 1



Connection for V4.0



Connection for 2560 R3



Connect the motor to digital pin 9.

Compile a program to control the motor to rotate in the commanded angle, and display the angle on the screen.

Sample Code A

```
/*
keyestudio super learning kit
Project 21.1
servo
```

<http://www.keyestudio.com>

```
*/  
int servopin=9;// select digital pin 9 for servomotor signal line  
int myangle;// initialize angle variable  
int pulsewidth;// initialize width variable  
int val;  
void servopulse(int servopin,int myangle)// define a servo  
pulse function  
{  
pulsewidth=(myangle*11)+500;// convert angle to  
500-2480 pulse width  
digitalWrite(servopin,HIGH);// set the level of servo pin as  
"high"  
delayMicroseconds(pulsewidth);// delay microsecond of  
pulse width  
digitalWrite(servopin,LOW);// set the level of servo pin as  
"low"  
delay(20-pulsewidth/1000);  
}  
void setup()  
{  
pinMode(servopin,OUTPUT);// set servo pin as "output"
```

```

Serial.begin(9600);// connect to serial port, set baud rate at
"9600"

Serial.println("servo=o_serl_simple ready" ) ;

}

void loop()// convert number 0 to 9 to corresponding 0-180
degree angle, LED blinks corresponding number of time
{

val=Serial.read();// read serial port value
if(val>='0'&&val<='9')

{
val=val-'0';// convert characteristic quantity to numerical
variable
val=val*(180/9);// convert number to angle
Serial.print("moving servo to ");
Serial.print(val,DEC);
Serial.println();
for(int i=0;i<=50;i++) // giving the servo time to rotate to
commanded position
{
servopulse(servopin,val);// use the pulse function
}}}
///////////

```

5.Method 2

Let's first take a look at the Arduino built-in servo function and some common statements.

1. **attach (interface)** ——select pin for servo, can only use pin 9 or 10.
2. **write (angle)** ——used to control the rotate angle of the servo, can set the angle among 0 degree to 180 degree.
3. **read ()** ——used to read the angle of the servo, consider it a function to read the value in the write() function.
4. **attached ()** ——determine whether the parameter of the servo is sent to the servo pin.
5. **detach ()** —— disconnect the servo and the pin, and the pin (digital pin 9 or 10) can be used for PWM port.

Note the written form of the above statements are " servo variable name. specific statement ()", e.g. myservo. Attach (9).

Still, connect the servo to pin 9.

Sample Code B

```
/*
keyestudio super learning kit
```

Project 21.2

servo

<http://www.keyestudio.com>

*/

```
#include <Servo.h>
```

```
Servo myservo;// define servo variable name
```

```
void setup()
```

```
{
```

```
myservo.attach(9);// select servo pin(9 or 10)
```

```
}
```

```
void loop()
```

```
{
```

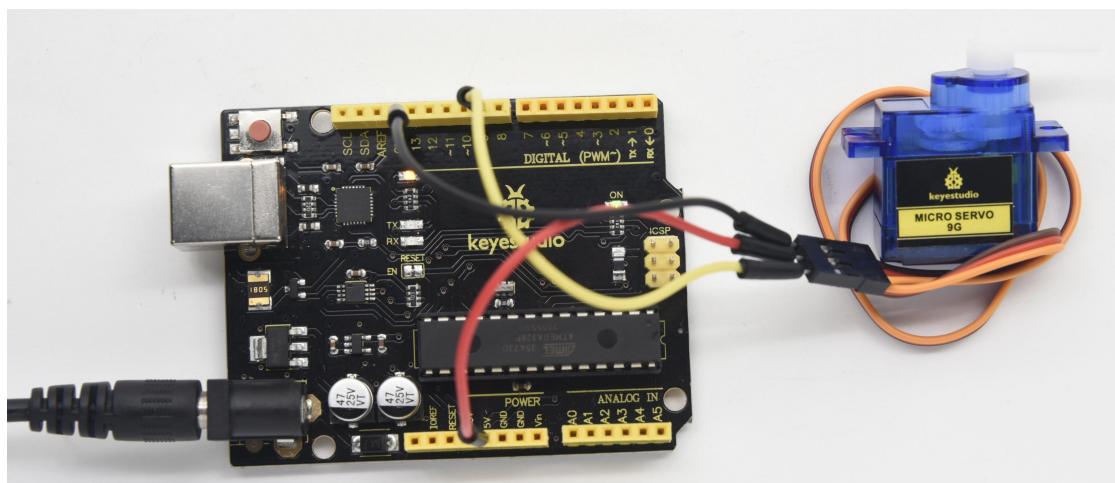
```
myservo.write(90);// set rotate angle of the motor
```

```
}
```

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

Above are the two methods to control the servo. You can choose either one according to your liking or actual need.

6. Test Result



Project 22: Stepper Motor



1. Introduction

A stepper motor is an electromechanical device which can convert electrical pulses into discrete mechanical movements. The shaft or spindle of a stepper motor rotates in discrete step increments when electrical command pulses are applied

to it in the proper sequence.

The motors rotation has several direct relationships to these applied input pulses. The sequence of the applied pulses is directly related to the direction of motor shafts rotation. The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation is directly related to the number of input pulses applied.

One of the most significant advantages of a stepper motor is its ability to be accurately controlled in an open loop system.

Open loop control means no feedback information about position is needed. This type of control eliminates the need for expensive sensing and feedback devices such as optical encoders. Your position is known simply by keeping track of the input step pulses.

2.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- Stepper Motor*1
- Stepper Motor driver*1
- Jumper Wire *1
- USB cable*1

3.Features

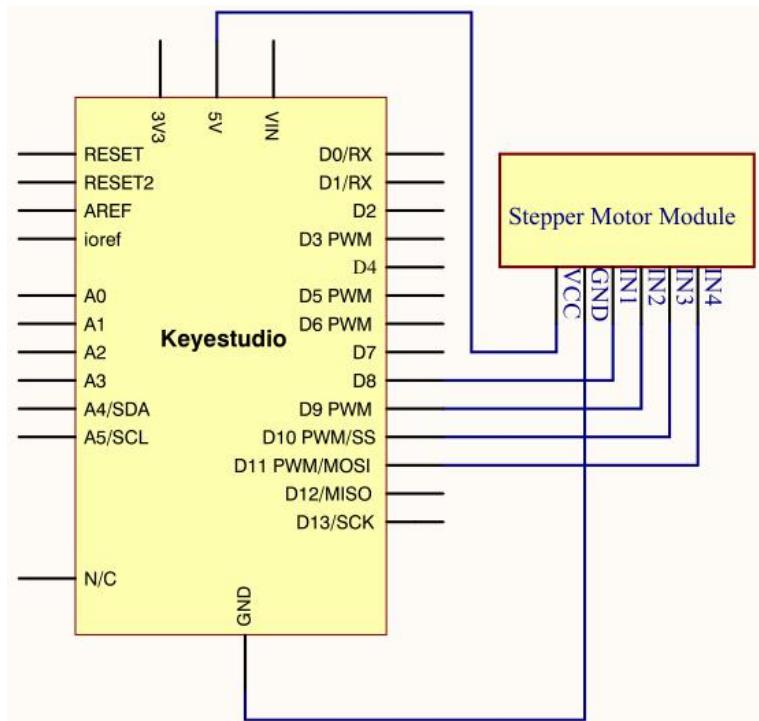
- The rotation angle of the motor is proportional to the input pulse.
- The motor has full torque at standstill(if the windings are energized)
- Precise positioning and repeatability of movement since good stepper motors have an accuracy of – 5% of a step and this error is non cumulative from one step to the next.
- Excellent response to starting/stopping/reversing.
- Very reliable since there are no contact brushes in the motor. Therefore the life of the motor is simply dependant on the life of the bearing.
- The motors response to digital input pulses provides open-loop control, making the motor simpler and less costly to control.
- It is possible to achieve very low speed synchronous rotation with a load that is directly coupled to the shaft.
- A wide range of rotational speeds can be realized as the speed is proportional to the frequency of the input pulses.

4.Parameters of Stepper Motor 28BYJ-48

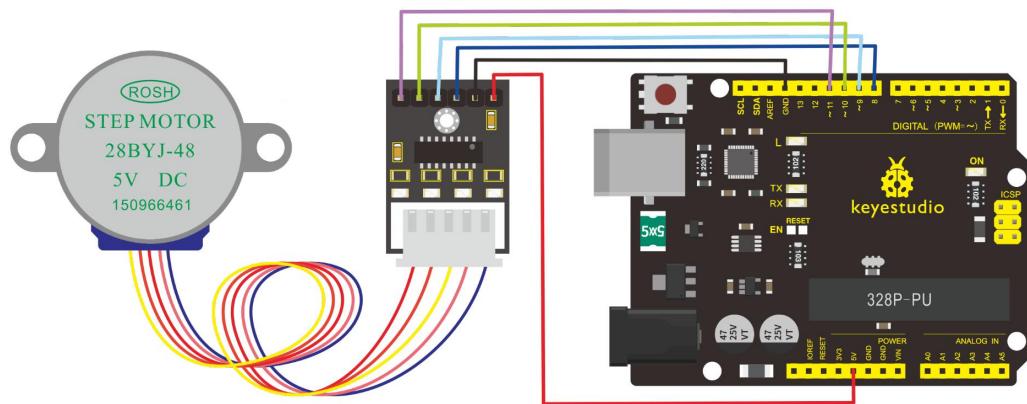
- Model 28BYJ-48

- Rated Voltage 5VDC
- Number of Phase 4
- Speed Variation Ratio 1/64
- Stride Angle 5.625° /64
- Frequency 100Hz
- DC Resistance $50\Omega \pm 7\%$ (25°C)
- Idle In-traction Frequency > 600Hz
- Idle Out-traction Frequency > 1000Hz
- In-traction Torque > 34.3mN.m(120Hz)
- Self-positioning Torque > 34.3mN.m
- Friction Torque 600-1200 gf.cm
- Pull in Torque 300 gf.cm
- Insulated Resistance > $10M\Omega$ (500V)
- Insulated Electricity Power 600VAC/1mA/1s
- Insulation Grade A
- Rise in Temperature < 40K(120Hz)
- Noise < 35dB(120Hz, No load, 10cm)

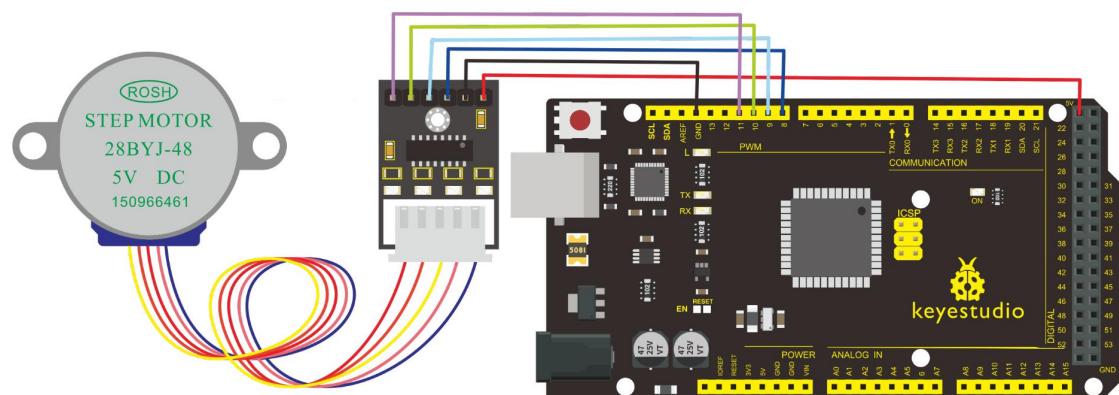
5.Circuit Connection



Connection for V4.0



Connection for 2560 R3

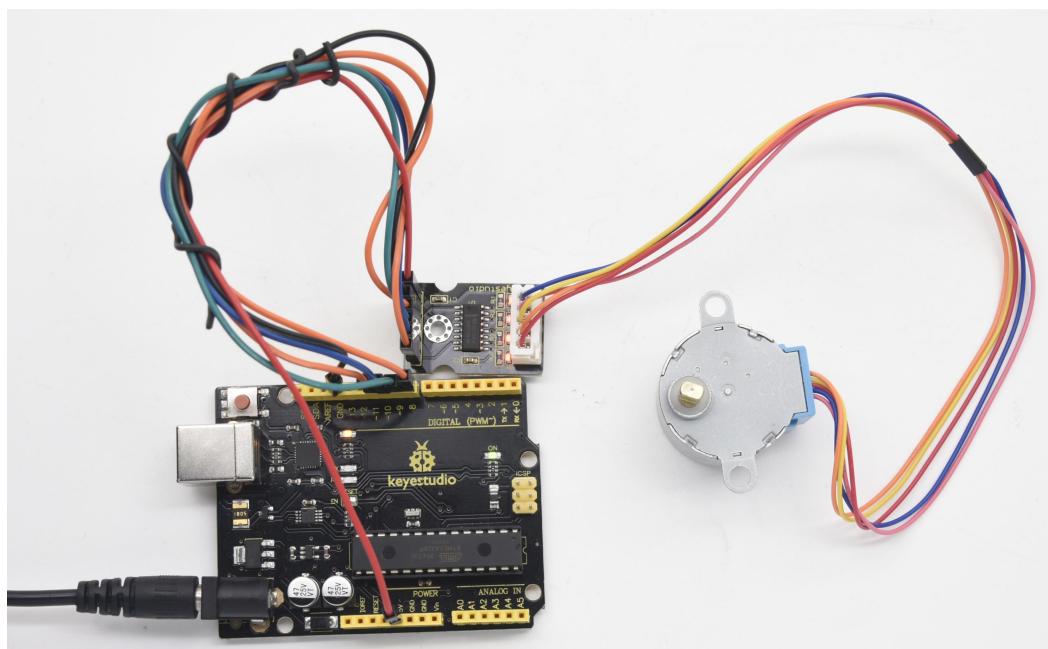


6.Sample Code

```
/*
keyestudio super learning kit
Project 22
Stepper Motor
http://www.keyestudio.com
*/
#include <Stepper.h>
#define STEPS 100
Stepper stepper(STEPS, 8, 9, 10, 11);
int previous = 0;
void setup()
{
    stepper.setSpeed(90);
}
void loop()
{
    int val = analogRead(0);
    stepper.step(val - previous);
    previous = val;
}
```



7. Test Result



Project 23: PIR Motion Sensor



1. Introduction

Pyroelectric infrared motion sensor can detect infrared signals

from a moving person or moving animal, and output switching signals.

It can be applied to a variety of occasions to detect the movement of human body.

Conventional pyroelectric infrared sensors require body pyroelectric infrared detector, professional chip, complex peripheral circuit, so it is more bigger with complex circuit, and lower reliability.

Now we launch this new pyroelectric infrared motion sensor, which is specially designed for Arduino.

It uses an integrated digital body pyroelectric infrared sensor, and has smaller size, higher reliability, lower power consumption as well as simpler peripheral circuit.

2.Hardware Required

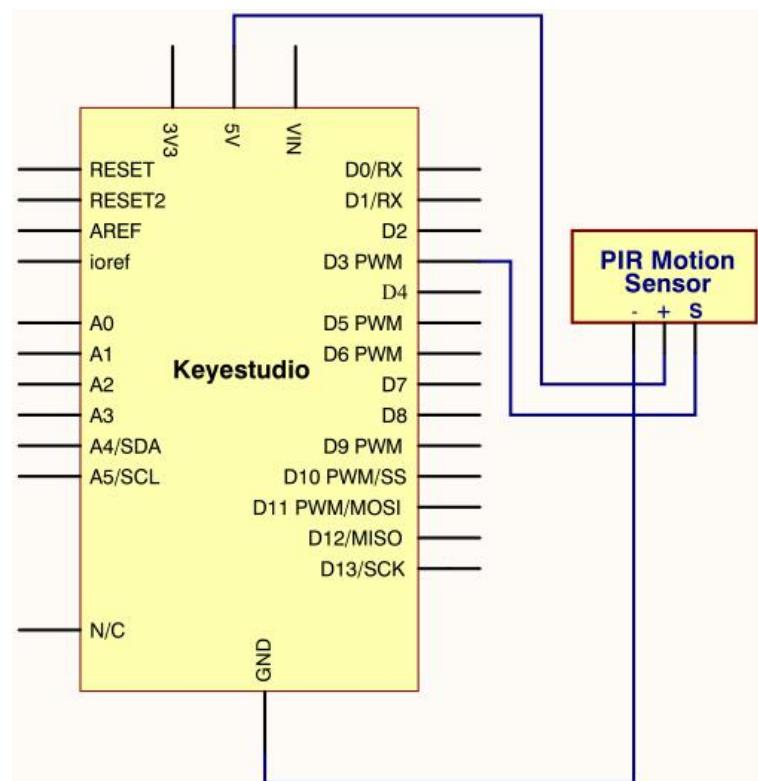
- V4.0 Board or MEGA 2650 Board*1
- PIR Motion Sensor*1
- Jumper Wire *1
- USB cable*1

3.Specification

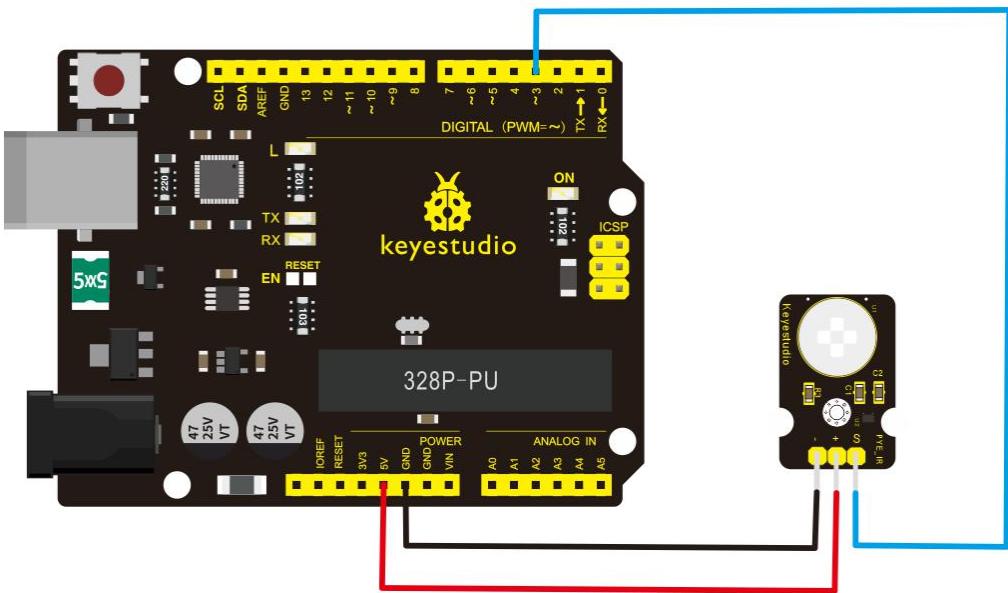
- Input Voltage 3.3 ~ 5V (6V Maximum)
- Working Current 15uA

- Working Temperature -20 ~ 85 °C
- Output Voltage High 3V, Low 0V
- Output Delay Time (High Level) About 2.3 to 3 Seconds
- Detection Angle 100 °
- Detection Distance 7 meters
- Output Indicator LED (When output HIGH, it will be ON)
- Pin limit Current 100mA

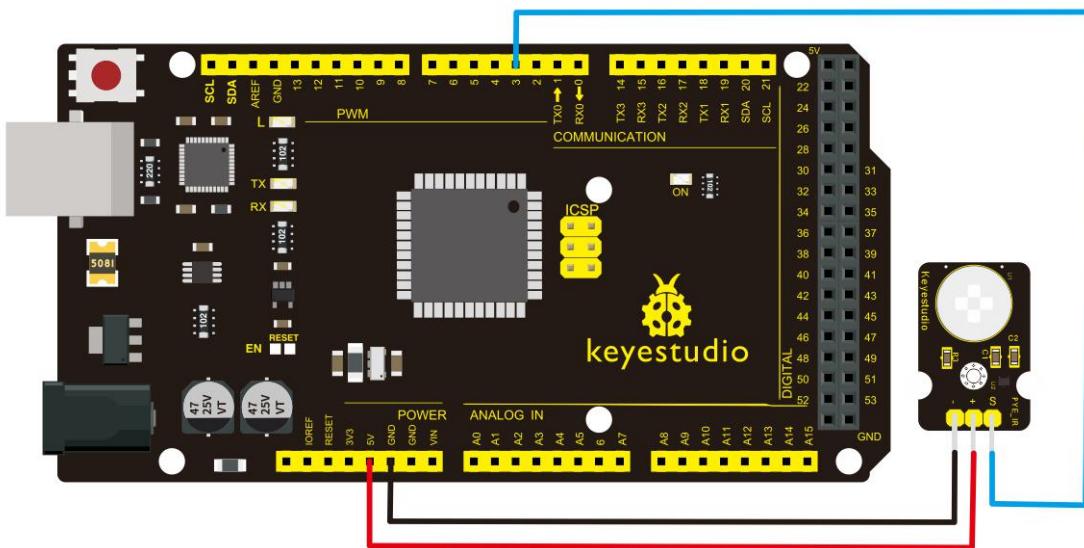
4.Circuit Connection



Connection for V4.0



Connection for MEGA 2560 R3



5. Sample Code

```
/*
keyestudio super learning kit
Project 23
```

PIR

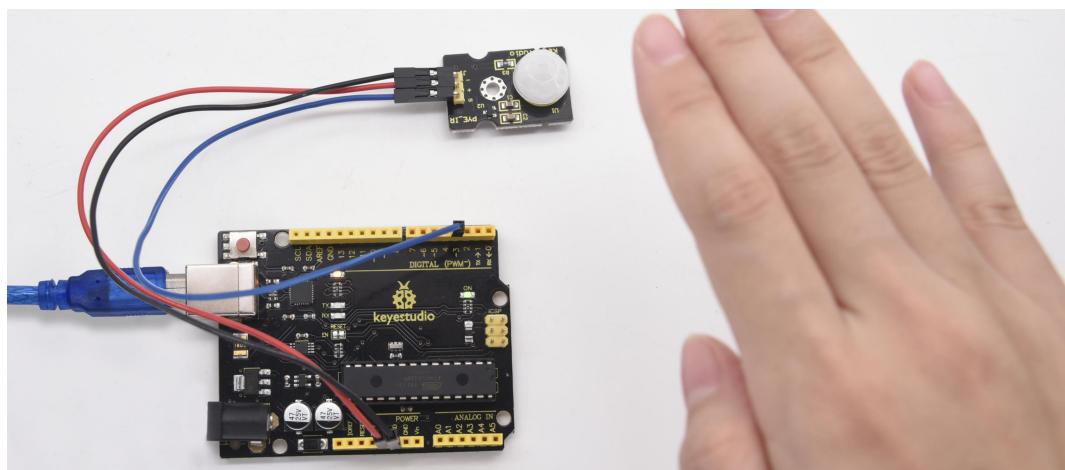
<http://www.keyestudio.com>

```
*/  
byte sensorPin = 3;  
byte indicator = 13;  
void setup()  
{  
    pinMode(sensorPin,INPUT);  
    pinMode(indicator,OUTPUT);  
    Serial.begin(9600);  
}  
void loop()  
{  
    byte state = digitalRead(sensorPin);  
    digitalWrite(indicator,state);  
    if(state == 1)Serial.println("Somebody is in this area!");  
    else if(state == 0)Serial.println("No one!");  
    delay(500);  
}  
//////////
```

6. Test Result

If the sensor detects someone moving nearby, D13 indicator on V4.0 board will light up, and "Somebody is in this area!" is displayed on the serial monitor.

If no detecting the movement, D13 indicator on V4.0 board will be off, and "No one!" is displayed on the serial monitor.



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

Autoscroll Show timestamp Newline 9600 baud Clear output

Project 24: Analog Gas Sensor



1. Introduction

This analog gas sensor - MQ2 is used in gas leakage detecting equipment in consumer electronics and industrial markets.

This sensor is suitable for LPG, I-butane, propane, methane, alcohol, Hydrogen and smoke detection. It has high sensitivity and quick response. In addition, the sensitivity can be adjusted by the potentiometer.

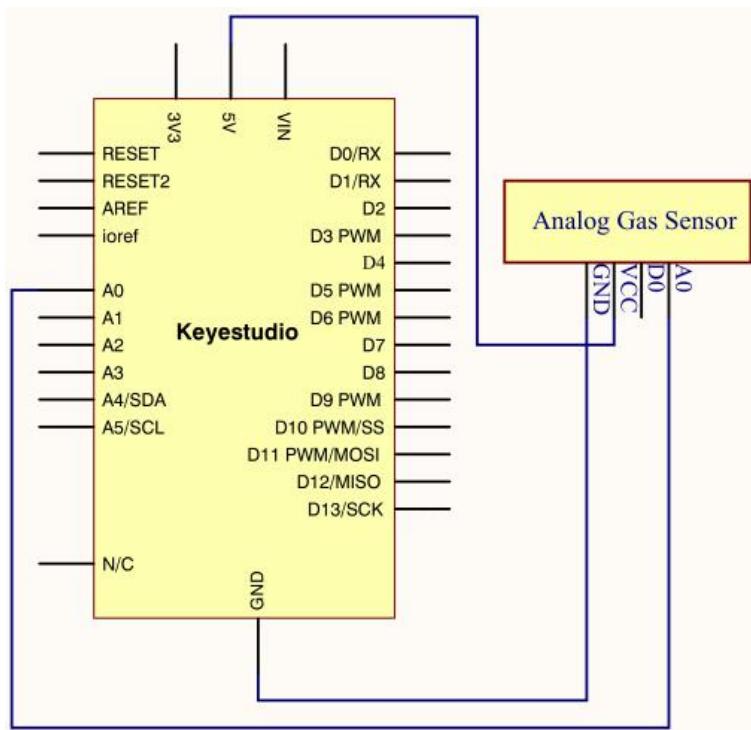
2.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- Analog Gas Sensor*1
- Jumper Wire *1
- USB cable*1

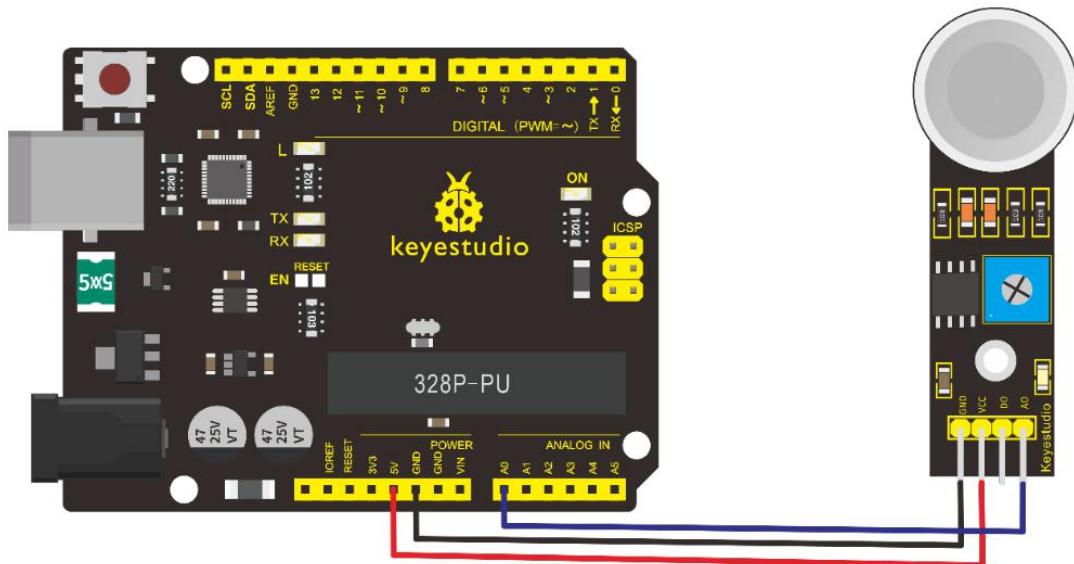
3.Specification

- Power Supply 5V
- Interface Type Analog
- Wide detecting scope
- Quick response and high sensitivity
- Simple drive circuit
- Stable and long lifespan

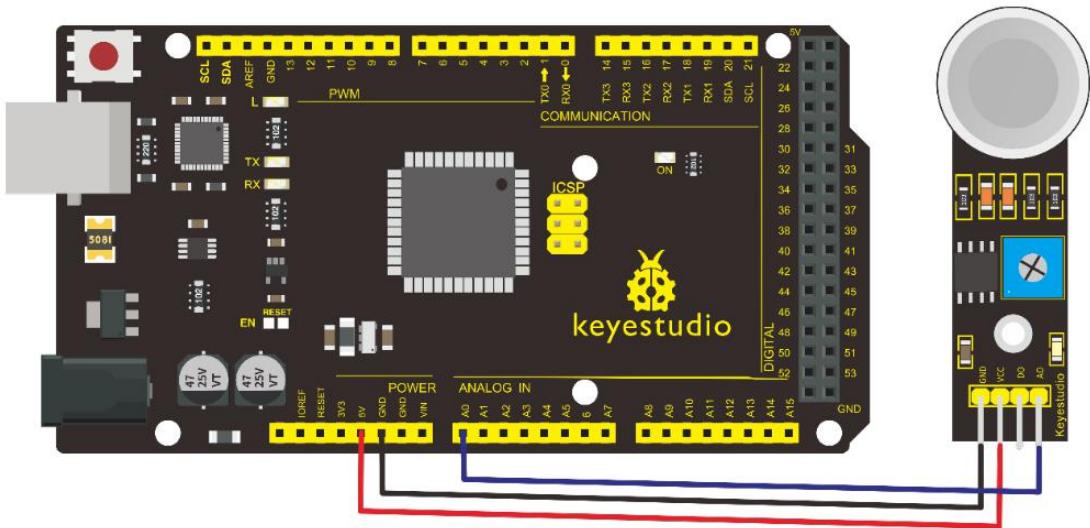
4.Circuit Connection



Connection for V4.0



Connection for MEGA 2560 R3

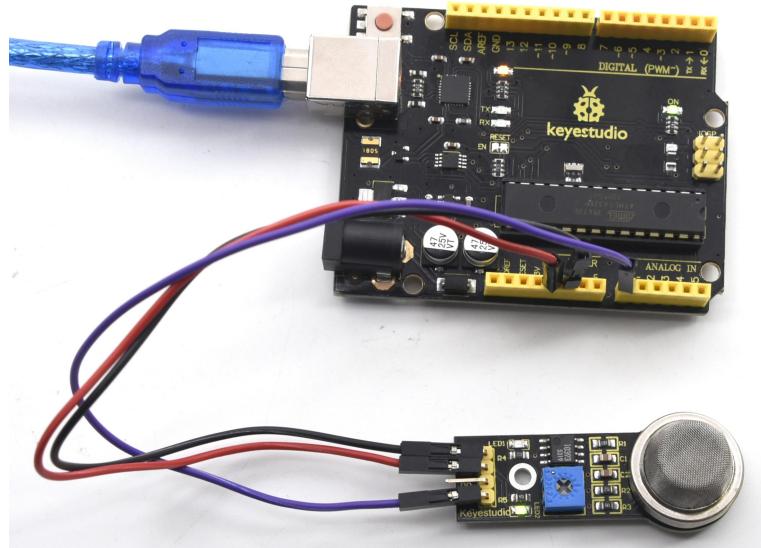


5. Sample Code

```
/*
keyestudio super learning kit
Project 24
Gas
http://www.keyestudio.com
*/
void setup()
{
    Serial.begin(9600); //Set serial baud rate to 9600 bps
}
void loop()
{int val;
val=analogRead(0);//Read Gas value from analog 0
Serial.println(val,DEC);//Print the value to serial port
}
```

```
delay(100);  
}  
//////////
```

6. Test Result



Project 25: ADXL345 Three Axis Acceleration



1. Introduction

The ADXL345 is a small, thin, low power, 3-axis MEMS accelerometer with high resolution (13-bit) measurement at

up to +16 g. Digital output data is formatted as 16-bit two's complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface.

The ADXL345 is well suited to measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables measurement of inclination change less than 1.0 degrees.

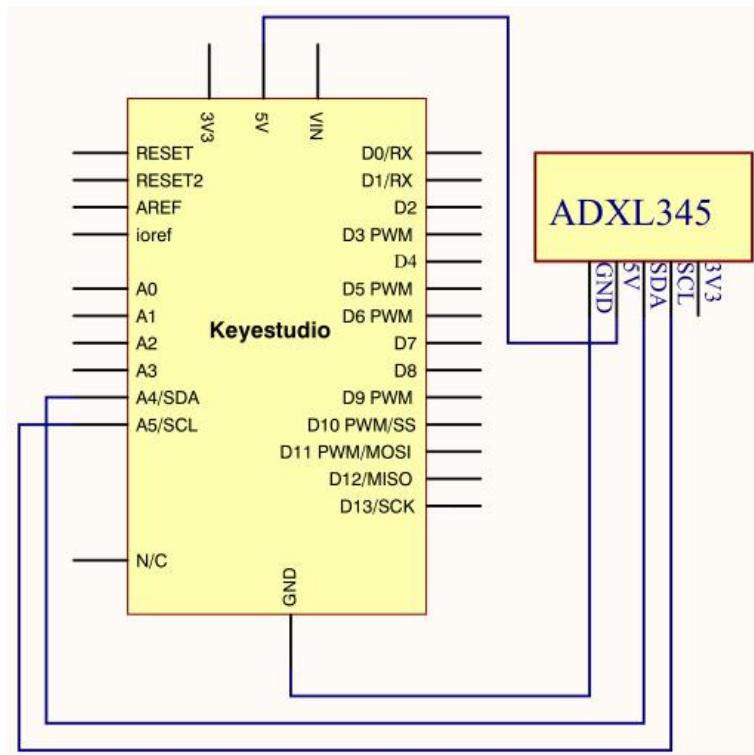
2.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- ADXL345 Sensor*1
- Jumper Wire *1
- USB cable*1

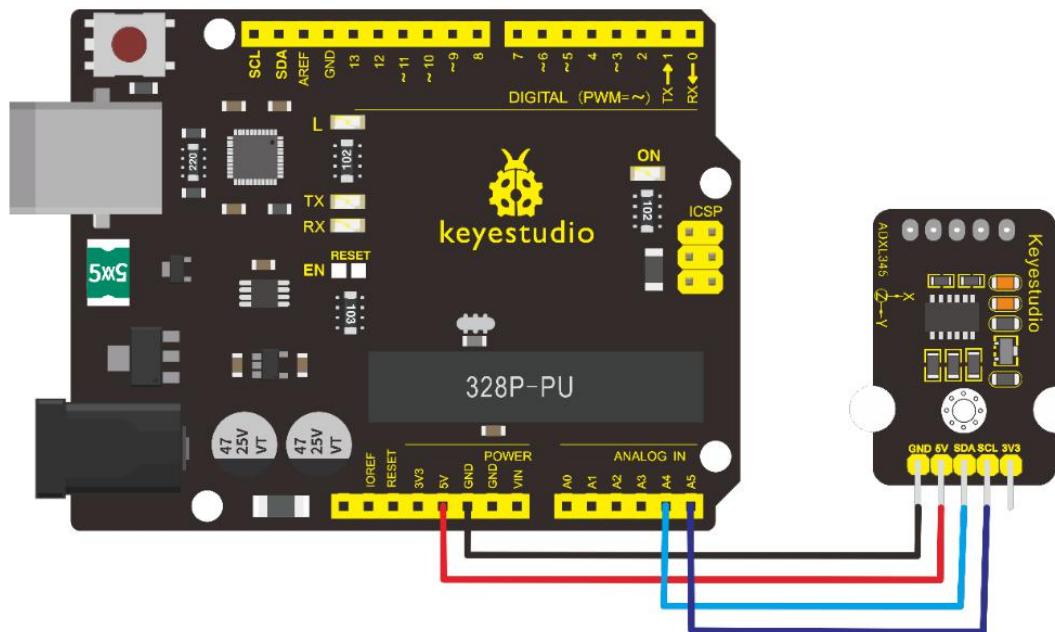
3.Specification

- 2.0-3.6V DC Supply Voltage
- Ultra Low Power 40uA in measurement mode, 0.1uA in standby@ 2.5V
- Tap/Double Tap Detection
- Free-Fall Detection
- SPI and I2C interfaces

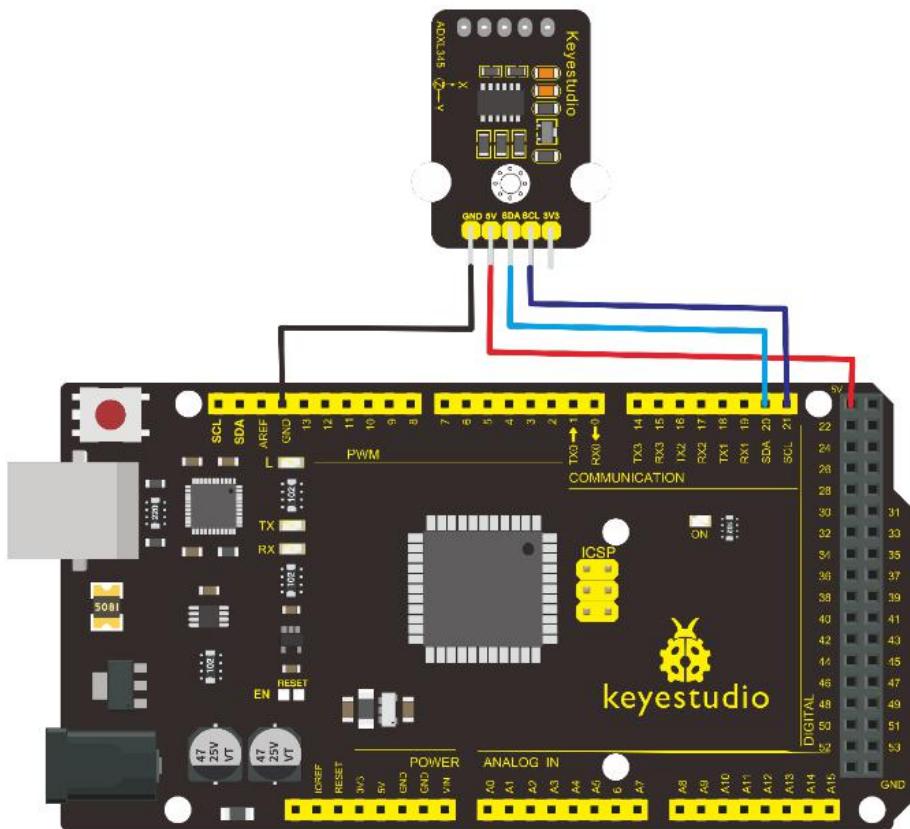
4. Circuit Connection



Connection for V4.0



Connection for MEGA 2560 R3



5. Sample Code

```
/*
keyestudio super learning kit
Project 25
ADXL345
http://www.keyestudio.com
*/
#include <Wire.h>

// Registers for ADXL345
#define ADXL345_ADDRESS (0xA6 >> 1) // address for
```

device is 8 bit but shift to the

// right by 1 bit to

make it 7 bit because the

// wire library only

takes in 7 bit addresses

```
#define ADXL345_REGISTER_XLSB (0x32)
```

```
int accelerometer_data[3];
```

// void because this only tells the cip to send data to its output
register

// writes data to the slave's buffer

```
void i2c_write(int address, byte reg, byte data) {
```

// Send output register address

```
Wire.beginTransmission(address);
```

// Connect to device

```
Wire.write(reg);
```

// Send data

```
Wire.write(data); //low byte
```

```
Wire.endTransmission();
```

```
}
```

// void because using pointers

```
// microcontroller reads data from the sensor's input register
void i2c_read(int address, byte reg, int count, byte* data) {
    // Used to read the number of data received
    int i = 0;
    // Send input register address
    Wire.beginTransmission(address);
    // Connect to device
    Wire.write(reg);
    Wire.endTransmission();

    // Connect to device
    Wire.beginTransmission(address);
    // Request data from slave
    // Count stands for number of bytes to request
    Wire.requestFrom(address, count);
    while(Wire.available()) // slave may send less than
requested
{
    char c = Wire.read(); // receive a byte as character
    data[i] = c;
    i++;
}
```

```
    Wire.endTransmission();

}

void init_adxl345() {
    byte data = 0;

    i2c_write(ADXL345_ADDRESS, 0x31, 0x0B); // 13-bit
mode +_ 16g

    i2c_write(ADXL345_ADDRESS, 0x2D, 0x08); // Power
register

    i2c_write(ADXL345_ADDRESS, 0x1E, 0x00); // x
    i2c_write(ADXL345_ADDRESS, 0x1F, 0x00); // Y
    i2c_write(ADXL345_ADDRESS, 0x20, 0x05); // Z

// Check to see if it worked!
    i2c_read(ADXL345_ADDRESS, 0X00, 1, &data);
    if(data==0xE5)
        Serial.println("it work Success");
    else
        Serial.println("it work Fail");
}
```

```
void read_adxl345() {  
    byte bytes[6];  
  
    memset(bytes,0,6);  
  
    // Read 6 bytes from the ADXL345  
    i2c_read(ADXL345_ADDRESS, ADXL345_REGISTER_XLSB,  
6, bytes);  
  
    // Unpack data  
    for (int i=0;i<3;++i) {  
        accelerometer_data[i] = (int)bytes[2*i] +  
        (((int)bytes[2*i + 1]) << 8);  
    }  
}  
  
// initialise and start everything  
void setup() {  
    Wire.begin();  
    Serial.begin(9600);  
    for(int i=0; i<3; ++i) {  
        accelerometer_data[i] = 0;  
    }  
    init_adxl345();
```

```
}

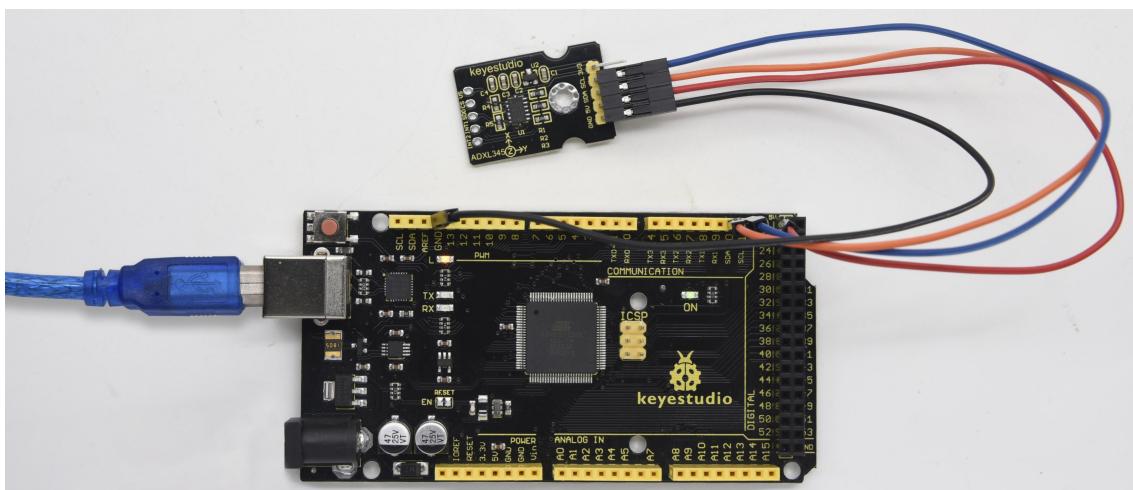
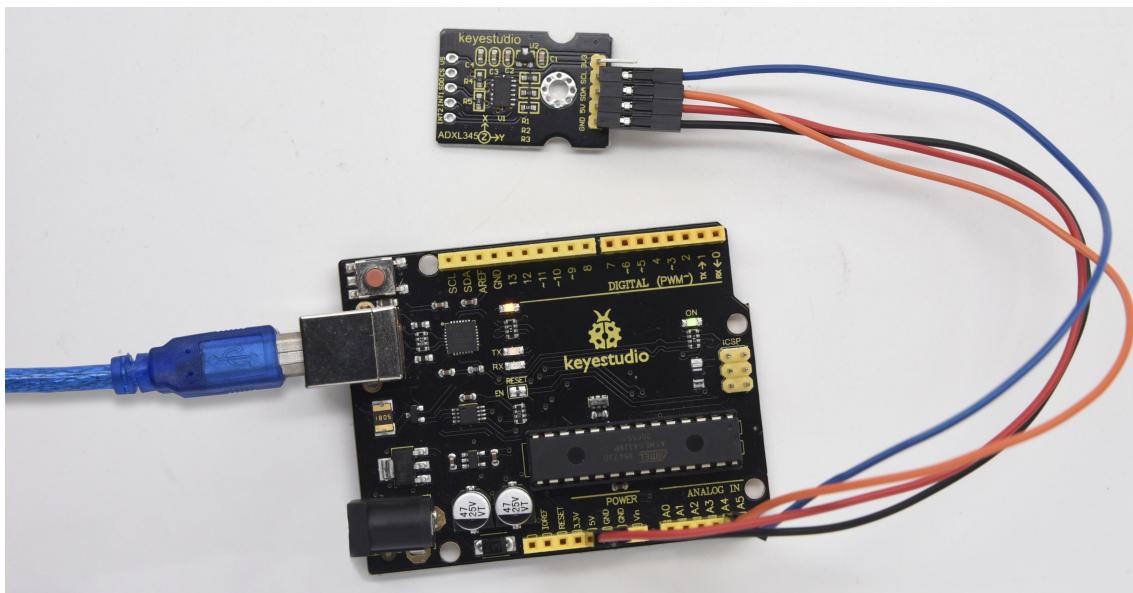
void loop() {
    read_adxl345();
    Serial.print("ACCEL ");
    Serial.print(float(accelerometer_data[0])*3.9/1000); //3.9m
    g/LSB scale factor in 13-bit mode
    Serial.print("t");
    Serial.print(float(accelerometer_data[1])*3.9/1000);
    Serial.print("t");
    Serial.print(float(accelerometer_data[2])*3.9/1000);
    Serial.print("n");
    delay(100);
}
```

//////////

6. Test Result

Wiring as the above diagram and power on, then upload the code and open the serial monitor, it will display the triaxial acceleration of sensor and its status, as the graph shown

below.



COM7

			Send
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

Project 26: HC-SR04 Ultrasonic Sensor



1. Introduction

The HC-SR04 Ultrasonic Sensor is a very affordable proximity/distance sensor that is mainly used for object avoidance in various robotics projects.

It essentially gives your Arduino eyes / spacial awareness and can prevent your robot from crashing or falling off a table. It has also been used in turret applications, water level sensing, and even as a parking sensor.

This simple project will use the HC-SR04 sensor with an Arduino and a Processing sketch to provide a more interactive display on your computer screen.

2. Hardware Required

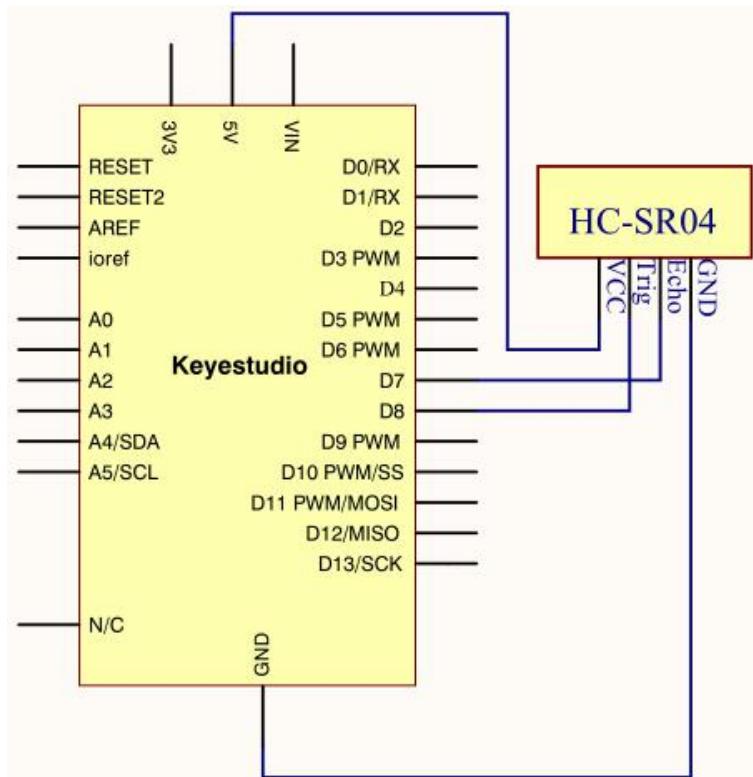
- V4.0 Board or MEGA 2650 Board*1

- Ultrasonic Sensor*1
- Jumper Wire *1
- USB cable*1

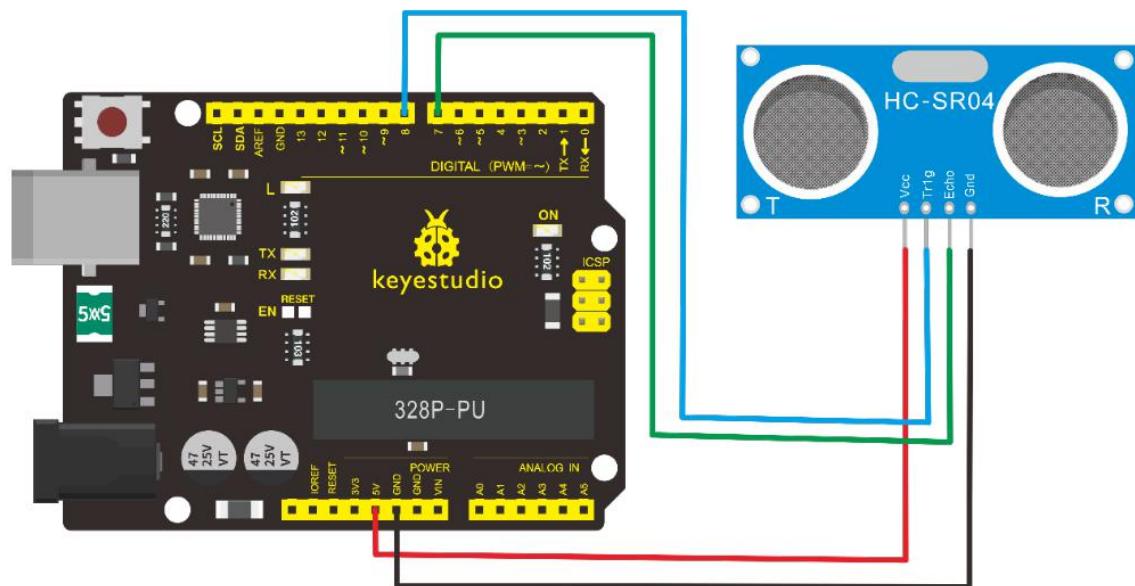
3.Specification

- Working Voltage DC 5V
- Working Current 15mA
- Working Frequency 40KHz
- Max Range 4m
- Min Range 2cm
- Measuring Angle 15 degree
- Trigger Input Signal 10 μ S TTL pulse
- Echo Output Signal Input TTL lever signal and the range in proportion

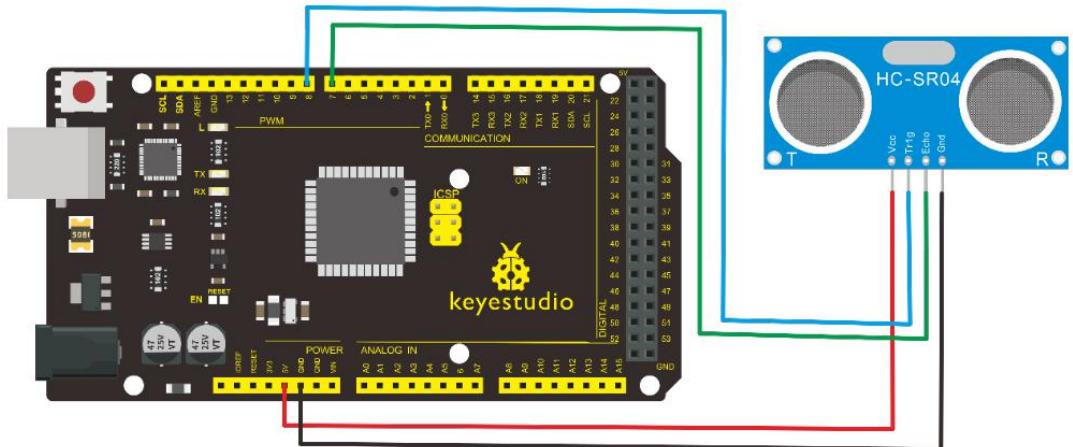
4.Circuit Connection



Connection for V4.0



Connection for MEGA 2560 R3



5.Sample Code

```
/*
keyestudio super learning kit
Project 26
Ultrasonic
http://www.keyestudio.com
*/
#define echoPin 7 // Echo Pin
#define trigPin 8 // Trigger Pin
#define LEDPin 13 // Onboard LED
int maximumRange = 200; // Maximum range needed
int minimumRange = 0; // Minimum range needed
long duration, distance; // Duration used to calculate distance
void setup() {
    Serial.begin (9600);
    pinMode(trigPin, OUTPUT);
```

```
pinMode(echoPin, INPUT);
pinMode(LEDPin, OUTPUT); // Use LED indicator (if
required)
}
void loop() {
/* The following trigPin/echoPin cycle is used to determine
the
distance of the nearest object by bouncing soundwaves off
of it. */
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
//Calculate the distance (in cm) based on the speed of
sound.
distance = duration/58.2;

if (distance >= maximumRange || distance <=
minimumRange){
/* Send a negative number to computer and Turn LED ON
```

```

        to indicate "out of range" */

    Serial.println("-1");

    digitalWrite(LEDPin, HIGH);

}

else {

/* Send the distance to the computer using Serial protocol,
and

turn LED OFF to indicate successful reading. */

Serial.println(distance);

digitalWrite(LEDPin, LOW);

}

//Delay 50ms before next reading.

delay(50);

}

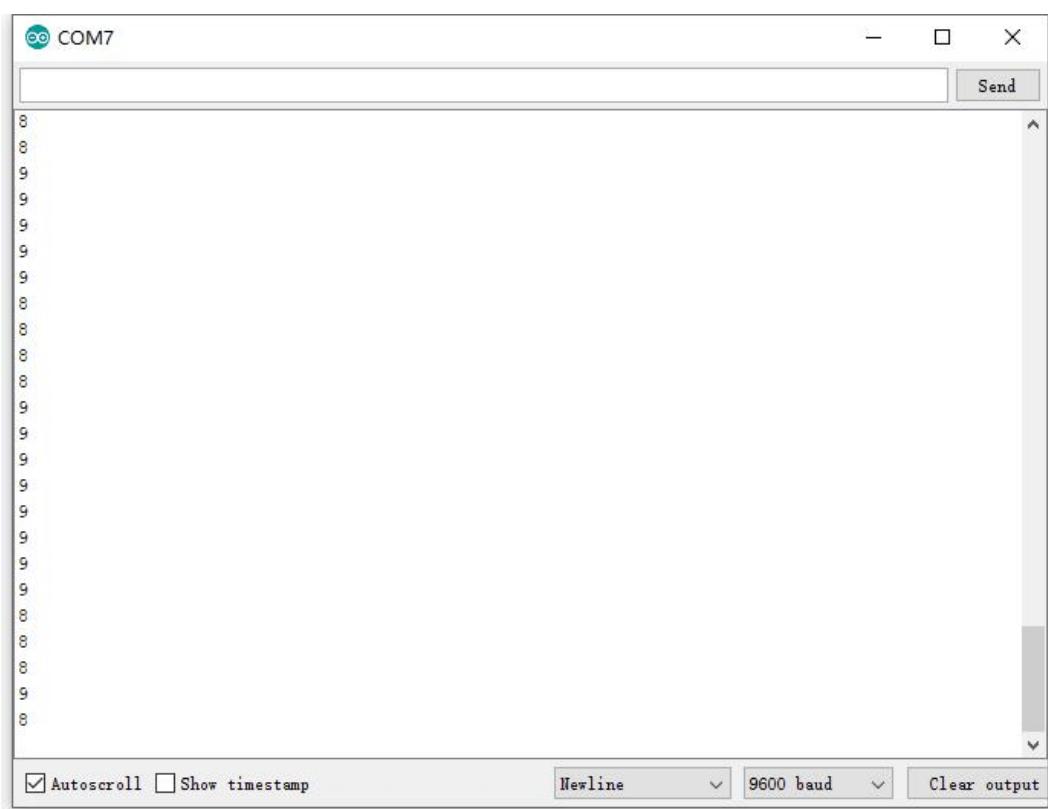
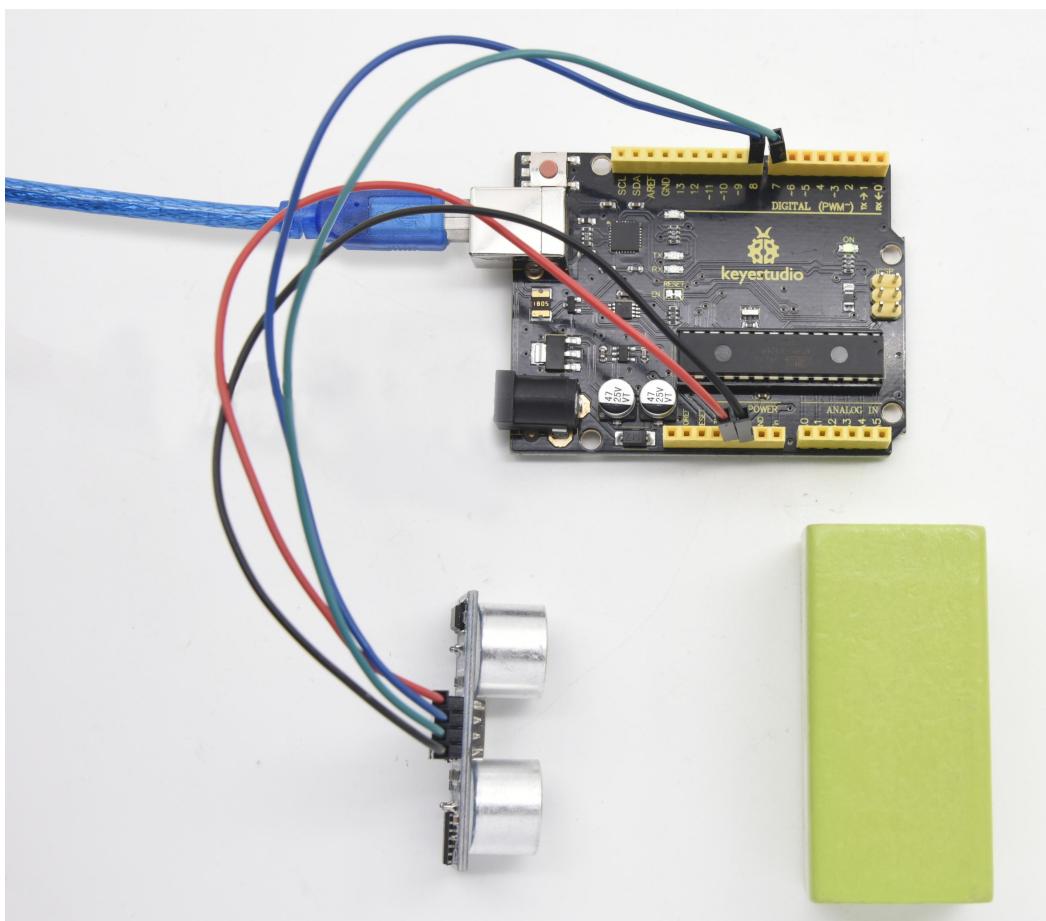
///////////

```

6. Test Result

After upload well the code to V4.0 board, then open the serial monitor.

When place an object in front of the ultrasonic sensor (from near and far), it will detect the distance of object. The value will be displayed on the monitor.



Project 27: Joystick Module



1. Introduction

Lots of robot projects need joystick. This module provides an affordable solution. By simply connecting to two analog inputs, the robot is at your commands with X, Y control. It also has a switch that is connected to a digital pin.

This joystick module can be easily connected to Arduino by IO Shield. This module is for Arduino(V5) with cable supplied.

2. Hardware Required

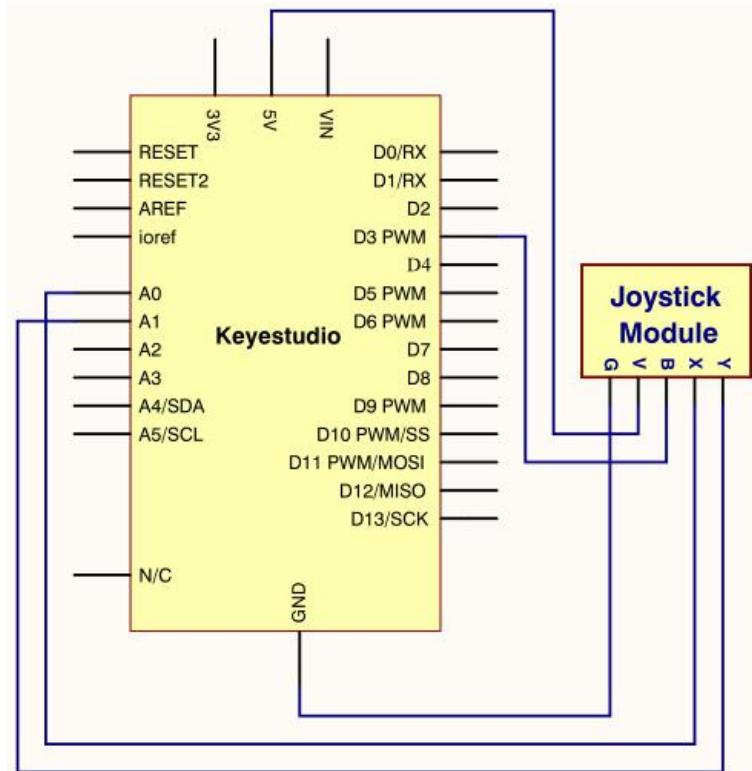
- V4.0 Board or MEGA 2650 Board*1
- Joystick Module*1
- Jumper Wire *1
- USB cable*1

3. Specification

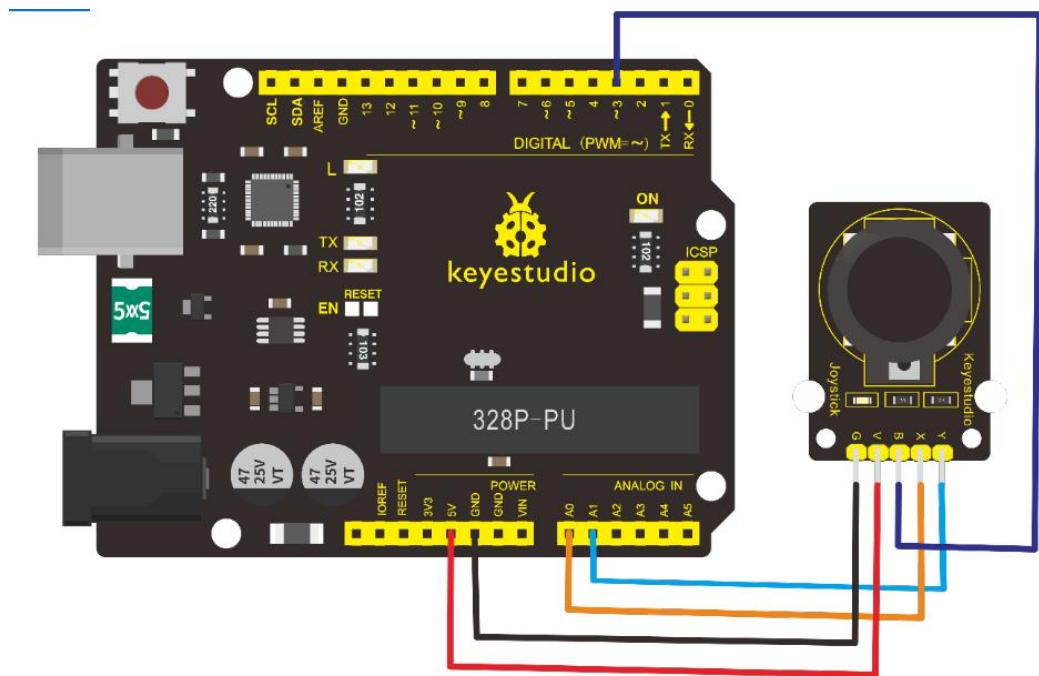
- Supply Voltage 3.3V to 5V

- Interface Analog x2, Digital x1

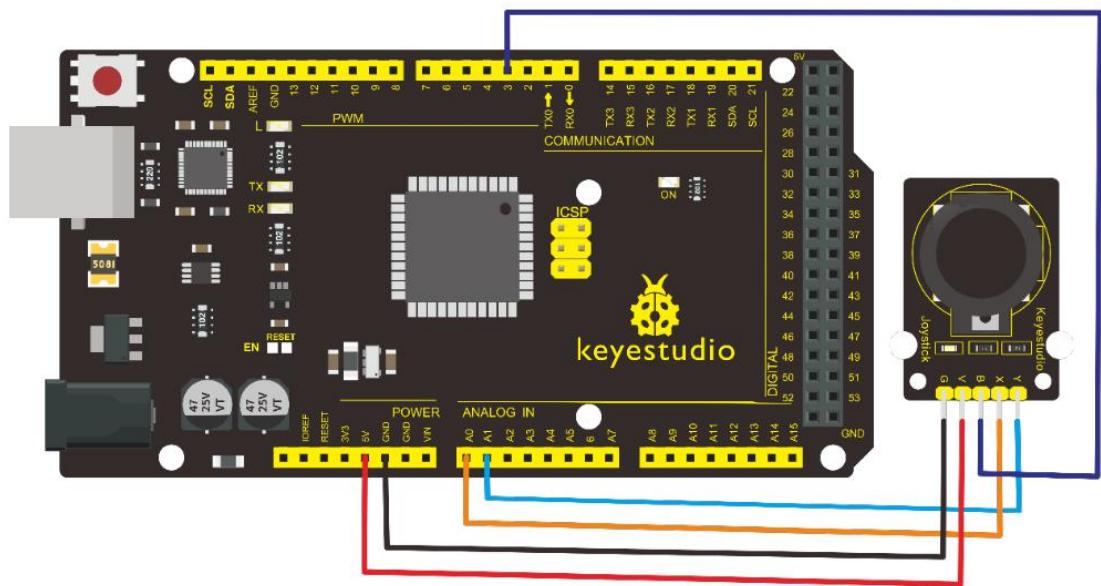
4.Circuit Connection



Connection for V4.0



Connection for MEGA 2560 R3



5. Sample Code

```
/*
keyestudio super learning kit
Project 27
joystick
http://www.keyestudio.com
*/
int JoyStick_X = 0; //x
int JoyStick_Y = 1; //y
int JoyStick_Z = 3; //key
void setup()
{
```

```

pinMode(JoyStick_Z, INPUT);

Serial.begin(9600); // 9600 bps

}

void loop()
{
    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);

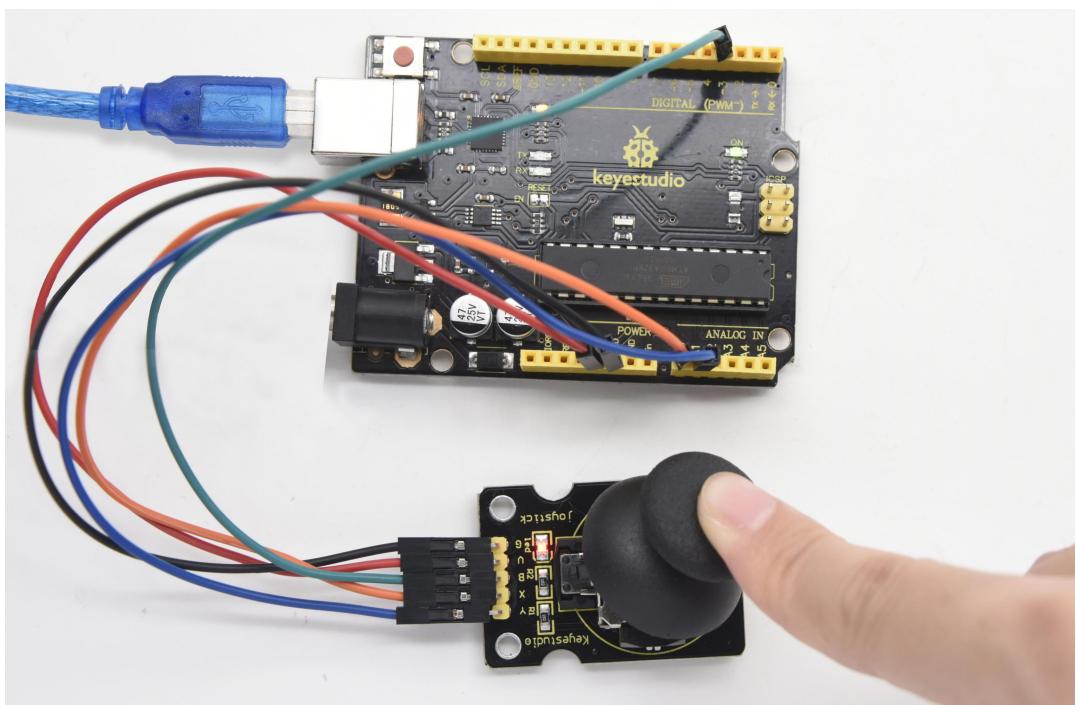
    Serial.print(",");
    Serial.println(z ,DEC);

    delay(100);
}
///////////

```

6. Test Result

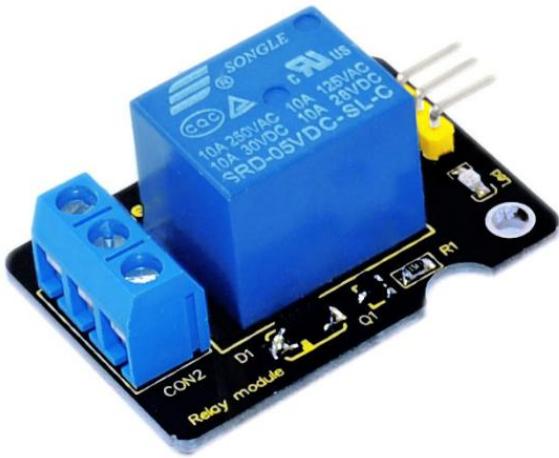
Wiring well and uploading the code, open the serial monitor and set the baud rate to 9600, push the joystick, you will see the value shown below.



```
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

Project 28: 5V Relay Module



1. Introduction

This single relay module can be used in interactive projects.

This module uses SONGLE 5v high-quality relay.

It can also be used to control the lighting, electrical and other equipment. The modular design makes it easy to expand with the Arduino Board (not included).

The Relay output is by a light-emitting diode. It can be controlled through digital IO port, such as solenoid valves, lamps, motors and other high current or high voltage devices.

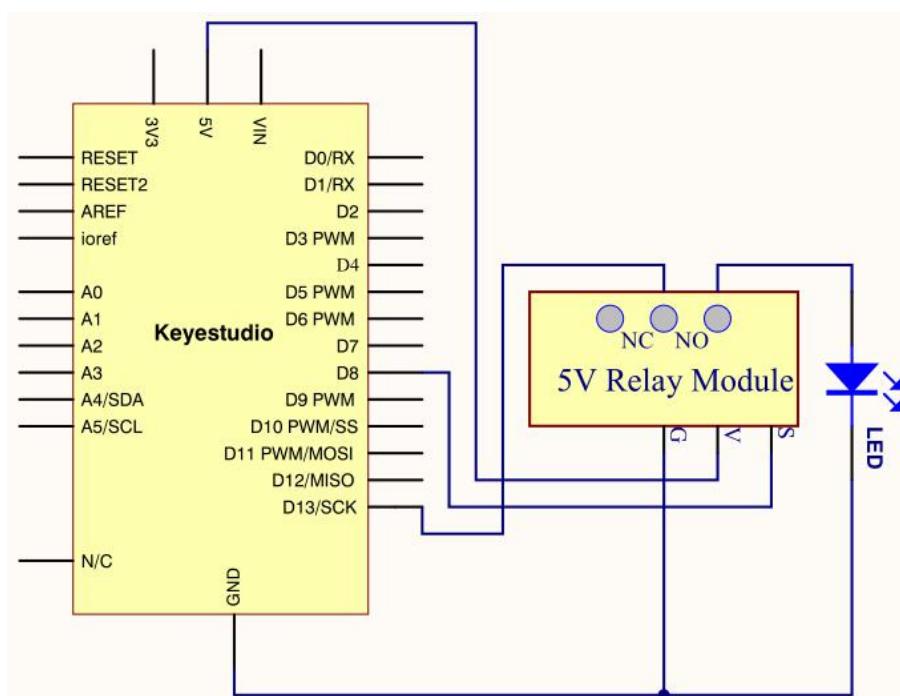
2. Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- Relay Module*1
- Jumper Wire *1
- USB cable*1

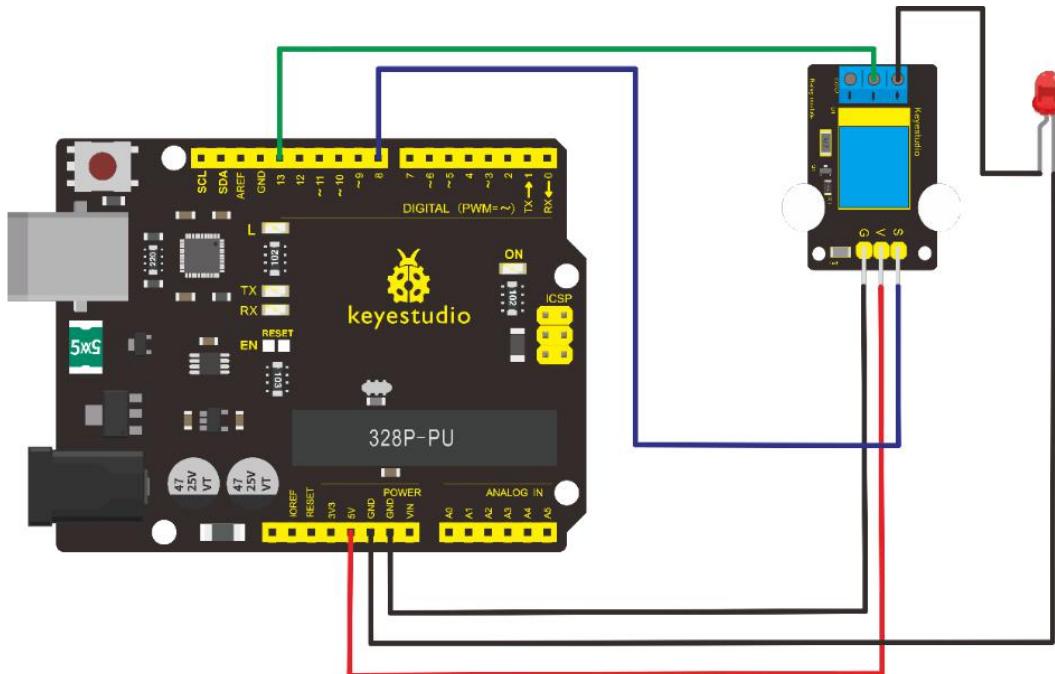
3.Specification

- Type Digital
- Rated Current 10A (NO) 5A (NC)
- Maximum Switching Voltage 150VAC 24VDC
- Digital Interface
- Control Signal TTL level
- Rated Load 8A 150VAC (NO) 10A 24VDC (NO), 5A 250VAC (NO/NC) 5A 24VDC (NO/NC)
- Maximum Switching Power AC1200VA DC240W (NO)
AC625VA DC120W (NC)
- Contact Action Time 10ms

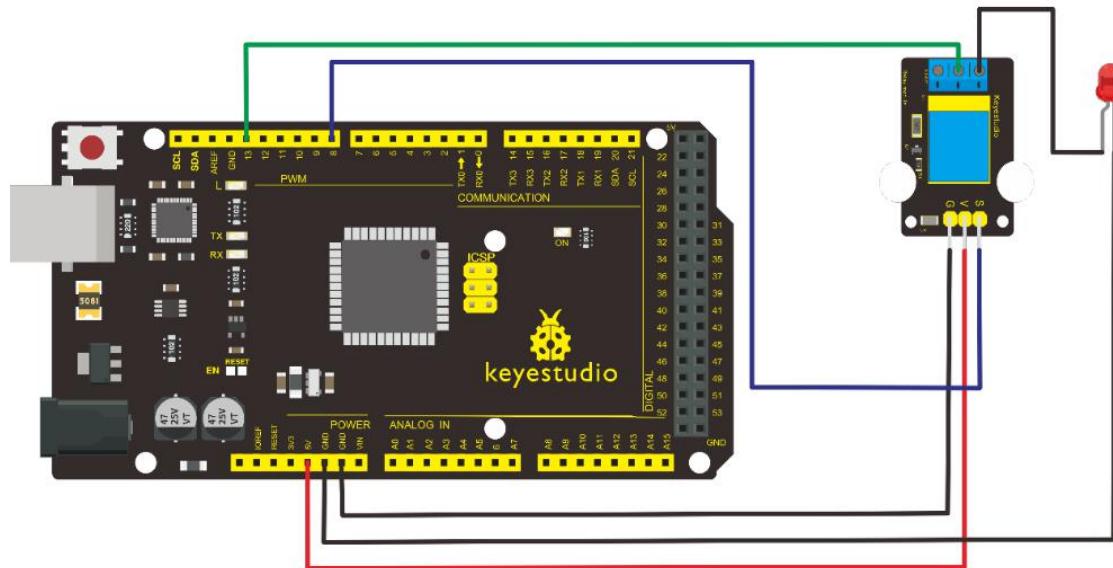
4.Circuit Connection



Connection for V4.0



Connection for MEGA 2560 R3



5. Sample Code

```
/*
```

keyestudio super learning kit

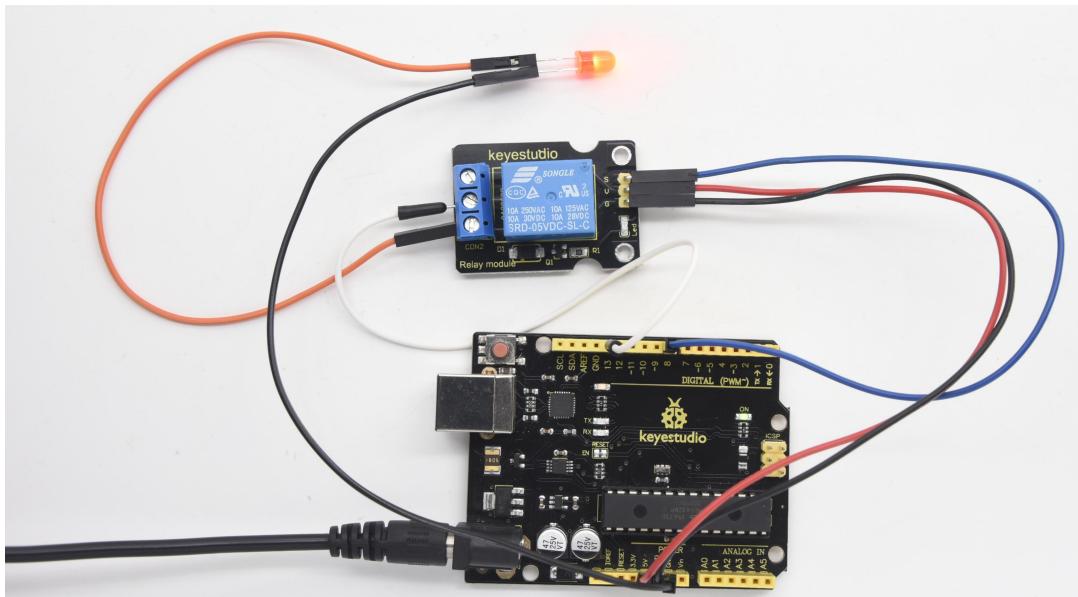
Project 28

Relay

<http://www.keyestudio.com>

```
*/  
int Relay = 8;  
  
void setup()  
{  
    pinMode(13, OUTPUT);      //Set Pin13 as output  
    digitalWrite(13, HIGH);    //Set Pin13 High  
    pinMode(Relay, OUTPUT);   //Set Pin8 as output  
}  
  
void loop()  
{  
    digitalWrite(Relay, HIGH); //Turn off relay  
    delay(2000);  
    digitalWrite(Relay, LOW); //Turn on relay  
    delay(2000);  
}  
//////////
```

6. Test Result



Project 29: DS3231 Clock Module



1. Introduction

DS3231 is equipped with integrated TCXO and crystal, which makes it a cost-effective I2C real time clock with high precision.

The device carries a battery input, so even if you disconnect the main power supply, it can still maintain accurate

timing. The integrated oscillator ensures the long-term accuracy of the device and reduces the number of components.

DS3231 provides both commercial and industrial temperature range and supports 16 pins small-outline package (300mil).

The module itself can adapt to the system of 3.3V and 5V without level switch, which is quite convenient!

2.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- DS3231 Clock Module*1
- Jumper Wire *1
- USB cable*1

3.Specification

- Temperature Range -40 to +85;
- Timing Accuracy \pm 5ppm (± 0.432 seconds / day)
- Provide battery backup for continuous timing
- Low power consumption
- Device package and function compatible with DS3231
- Complete clock calendar function contains seconds and minutes, hour, week, date, month, and year timing and

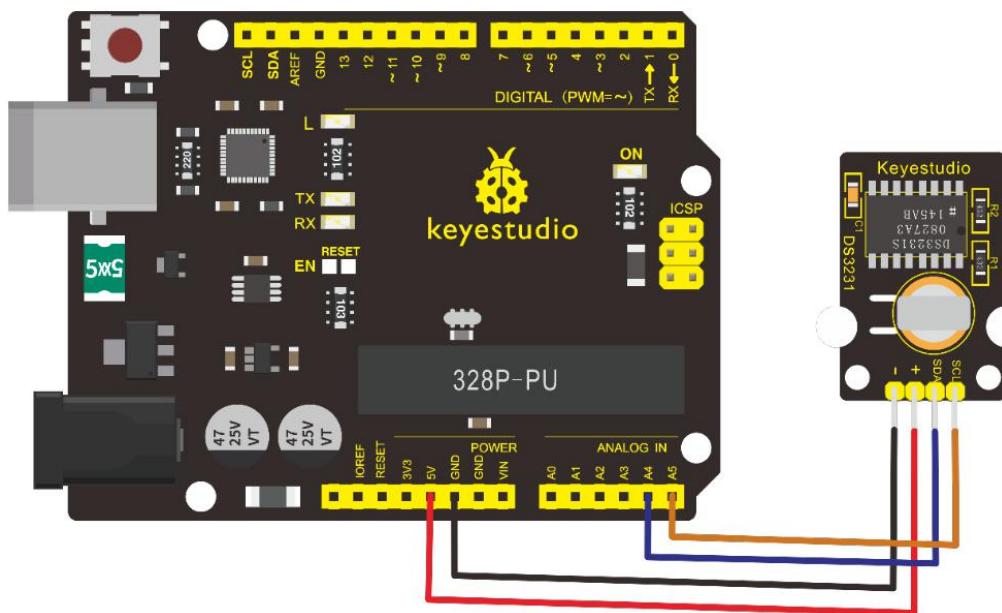
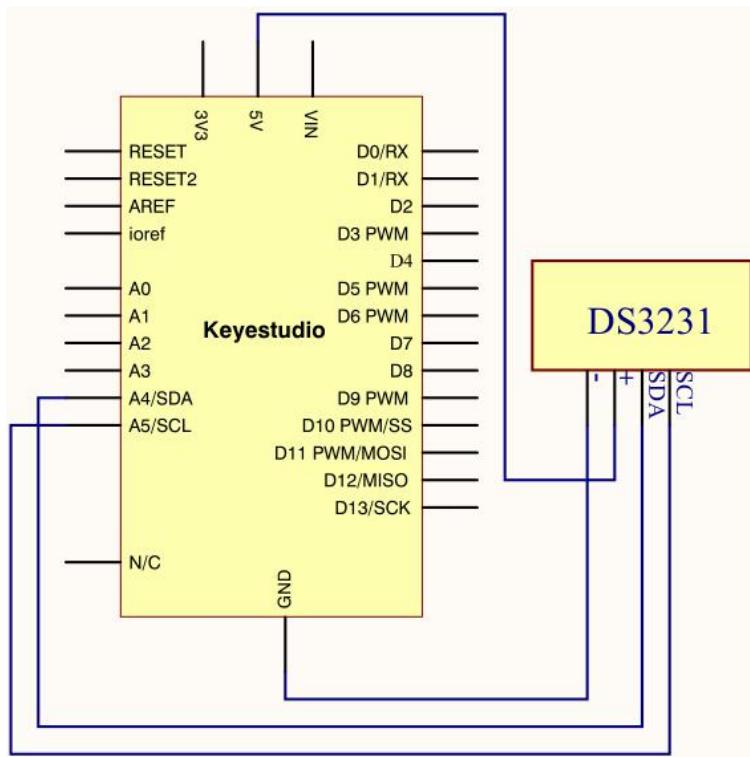
provides leap year compensation until 2100.

- Two calendar clock
- Output 1Hz and 32.768kHz
- Reset output and Input Debounce of Pushbutton
- High speed (400kHz), I2C serial bus
- Supply voltage +3.3V to +5.5V
- Digital temperature sensor with a precision of $\pm 3^\circ\text{C}$
- Working temperature -40 ~ C to +85 ~ C
- 16 pins Small Outline Package (300mil)

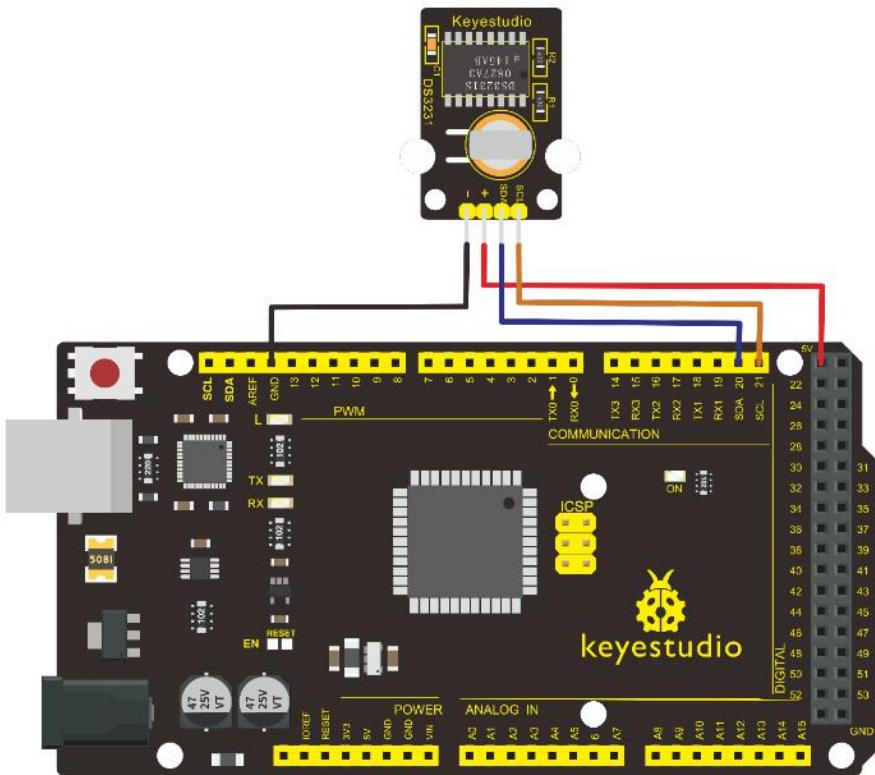
4.Circuit Connection

This module adopts the IIC test method, so only need to connect 'SDA' to Arduino A4, 'SCL' to A5, '+' to VCC and '-' to GND as follows

Connection for V4.0



Connection for MEGA 2560 R3



5. Sample Code

Before compiling the code, you'd better put [DS3231 library](#) under file into Arduino catalogue.

```
/*
keyestudio super learning kit
```

Project 29

clock

<http://www.keyestudio.com>

```
*/
#include <Wire.h>
#include "DS3231.h"
DS3231 RTC; //Create the DS3231 object
```

```

char weekDay[][4] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri",
"Sat" };

//year, month, date, hour, min, sec and week-day(starts from
0 and goes to 6)

//writing any non-existent time-data may interfere with
normal operation of the RTC.

//Take care of week-day also.

DateTime dt(2011, 11, 10, 15, 18, 0, 5);//open the series
port and you can check time here or make a change to the
time as needed.

void setup ()
{
    Serial.begin(57600);//set baud rate to 57600
    Wire.begin();
    RTC.begin();
    RTC.adjust(dt); //Adjust date-time as defined 'dt' above
}

void loop ()
{
    DateTime now = RTC.now(); //get the current date-time
    Serial.print(now.year(), DEC);
    Serial.print("/");
    Serial.print(now.month(), DEC);

```

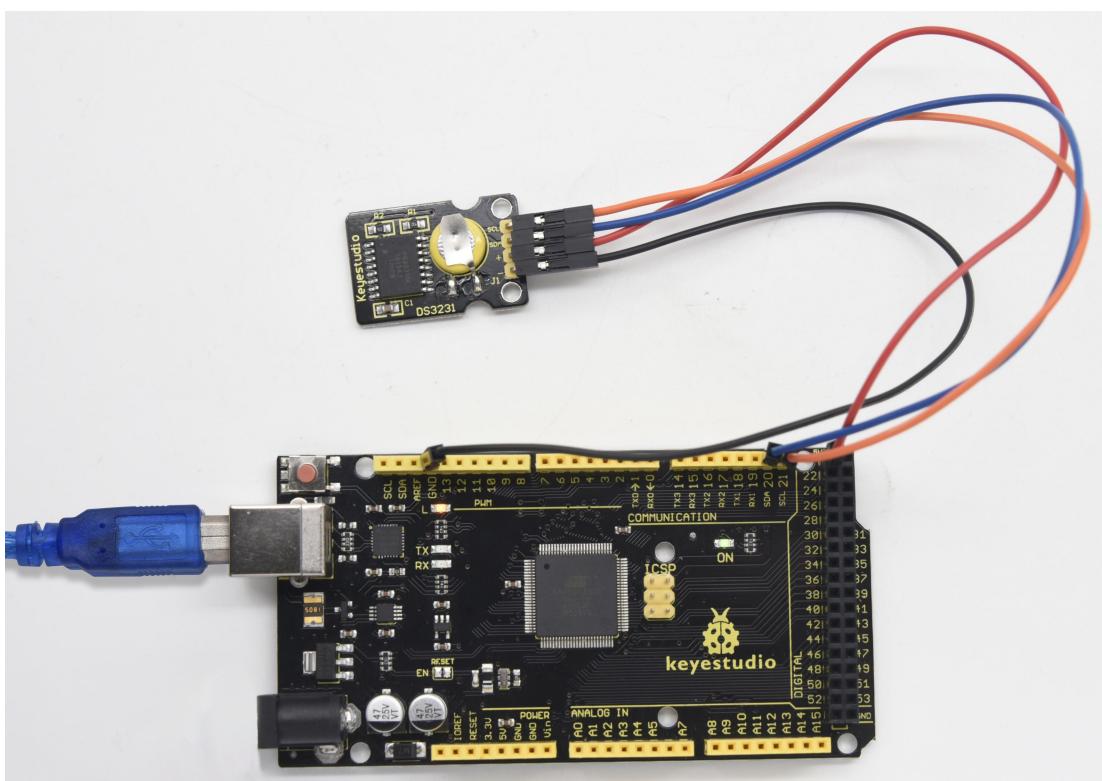
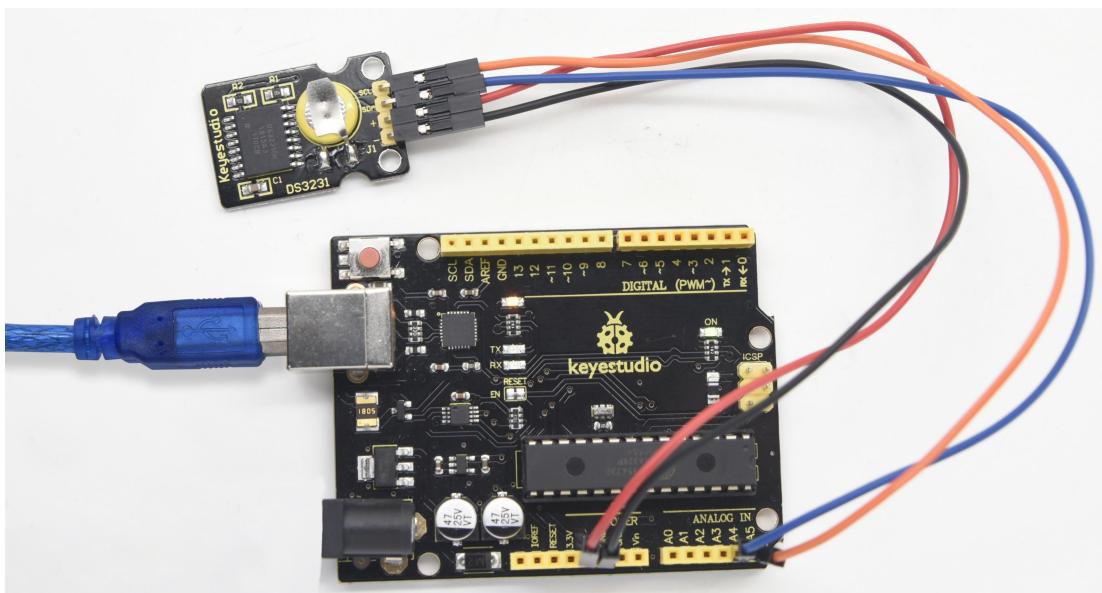
```
Serial.print("/");
Serial.print(now.date(), DEC);
Serial.print(" ");
Serial.print(now.hour(), DEC);
Serial.print(":");
Serial.print(now.minute(), DEC);
Serial.print(":");
Serial.print(now.second(), DEC);
Serial.println();
Serial.print(weekDay[now.dayOfWeek()]);
Serial.println();
delay(1000);

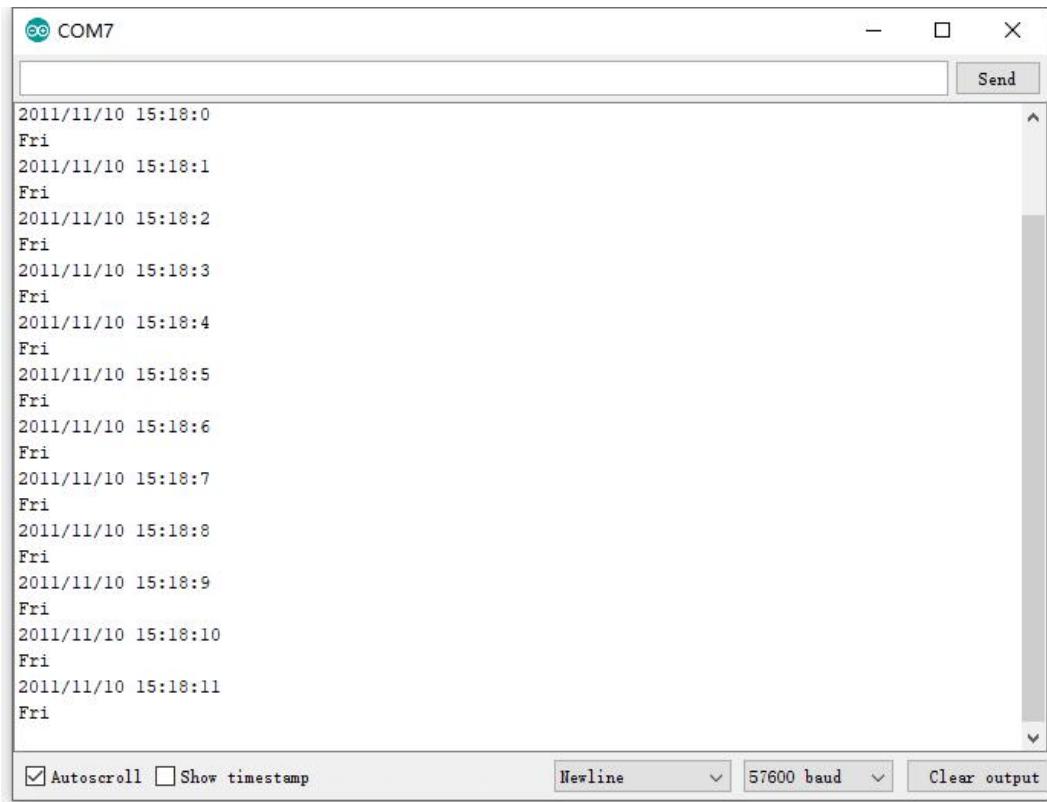
}
```

//////////

6. Test Result

Done uploading the code to arduino, open the serial monitor and get the following results

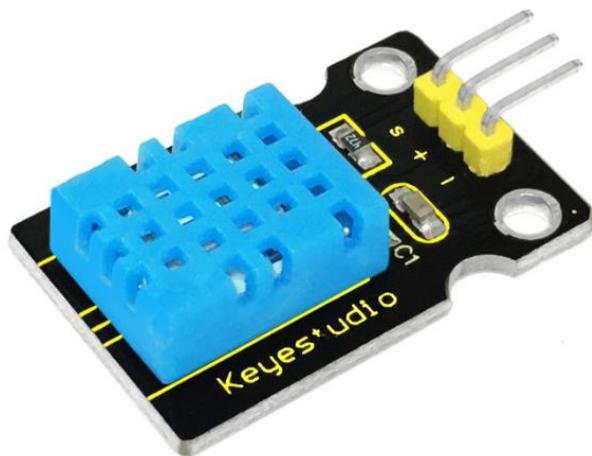




```
COM7
Send
2011/11/10 15:18:0
Fri
2011/11/10 15:18:1
Fri
2011/11/10 15:18:2
Fri
2011/11/10 15:18:3
Fri
2011/11/10 15:18:4
Fri
2011/11/10 15:18:5
Fri
2011/11/10 15:18:6
Fri
2011/11/10 15:18:7
Fri
2011/11/10 15:18:8
Fri
2011/11/10 15:18:9
Fri
2011/11/10 15:18:10
Fri
2011/11/10 15:18:11
Fri
```

Autoscroll Show timestamp Newline 57600 baud Clear output

Project 30: DHT11 Temperature and Humidity Sensor



1. Introduction

This DHT11 Temperature and Humidity Sensor features

calibrated digital signal output with the temperature and humidity sensor complex. Its technology ensures high reliability and excellent long-term stability. A high-performance 8-bit microcontroller is connected.

This sensor includes a resistive element and a sense of wet NTC temperature measuring devices. It has the advantages of excellent quality, fast response, anti-interference ability and high cost performance.

Each DHT11 sensor features extremely accurate calibration data of humidity calibration chamber. The calibration coefficients stored in the OTP program memory, internal sensors detect signals in the process, and we should call these calibration coefficients.

The single-wire serial interface system is integrated to make it quick and easy. Qualities of small size, low power, and 20-meter signal transmission distance make it a widely applied application and even the most demanding one. Convenient connection and special package can be provided according to your need.

2.Hardware Required

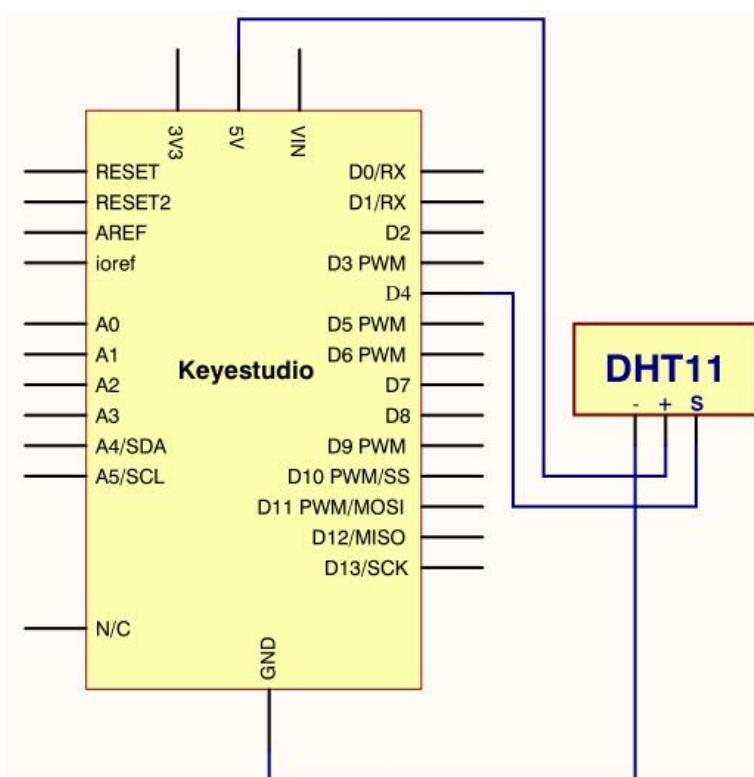
- V4.0 Board or MEGA 2650 Board*1
- DHT11 sensor*1

- Jumper Wire *1
- USB cable*1

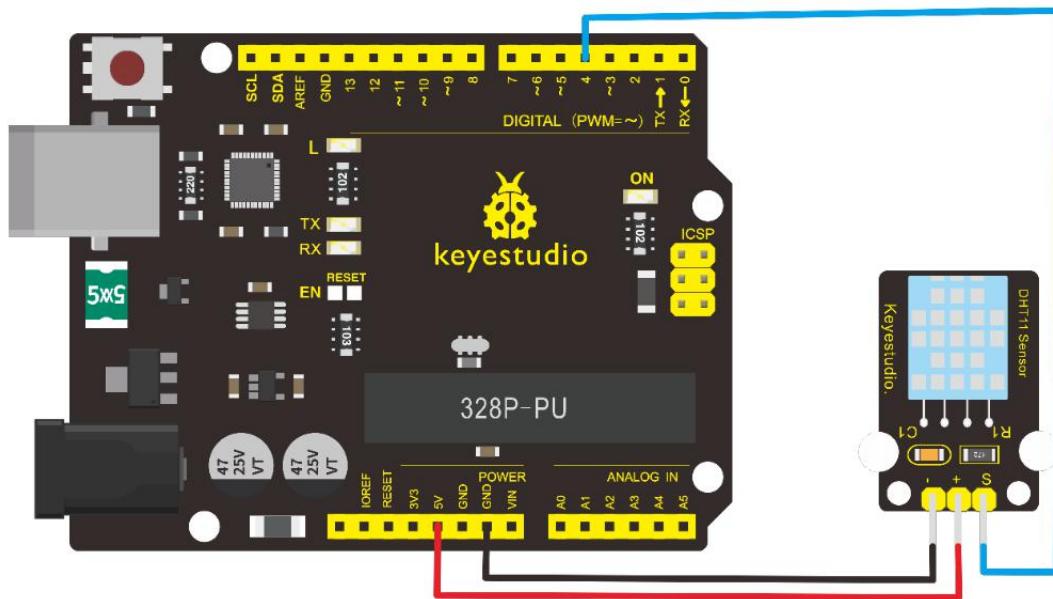
3.Specification

- Supply Voltage +5 V
- Temperature Range 0-50 °C error of ± 2 °C
- Humidity 20-90% RH ± 5% RH error
- Interface digital

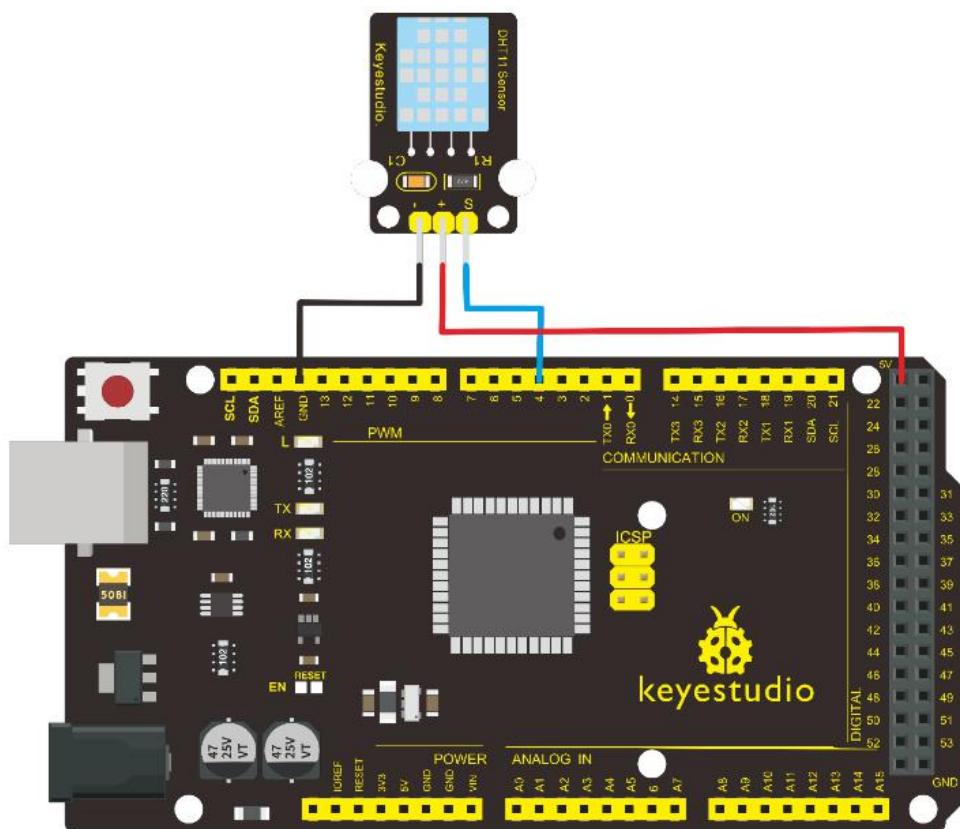
4.Circuit Connection



Connection for V4.0



Connection for MEGA 2560 R3



5. Sample Code

Please download the [DHT11Lib](#) firstly. Or [see the website](#)

```
/*
keyestudio super learning kit
Project 30
DHT11
http://www.keyestudio.com
*/
#include <dht11.h>
dht11 DHT;
#define DHT11_PIN 4

void setup(){
    Serial.begin(9600);
    Serial.println("DHT TEST PROGRAM ");
    Serial.print("LIBRARY VERSION: ");
    Serial.println(DHT11LIB_VERSION);
    Serial.println();
    Serial.println("Type,¥tstatus,¥tHumidity");
    Serial.println("(%),¥tTemperature (C)");
}

void loop(){
    int chk;
```

```
Serial.print("DHT11, \t");
chk = DHT.read(DHT11_PIN);      // READ DATA
switch (chk){
    case DHTLIB_OK:
        Serial.print("OK,\t");
        break;
    case DHTLIB_ERROR_CHECKSUM:
        Serial.print("Checksum error,\t");
        break;
    case DHTLIB_ERROR_TIMEOUT:
        Serial.print("Time out error,\t");
        break;
    default:
        Serial.print("Unknown error,\t");
        break;
}
// DISPLAY DATA
Serial.print(DHT.humidity,1);
Serial.print(",\t");
Serial.println(DHT.temperature,1);

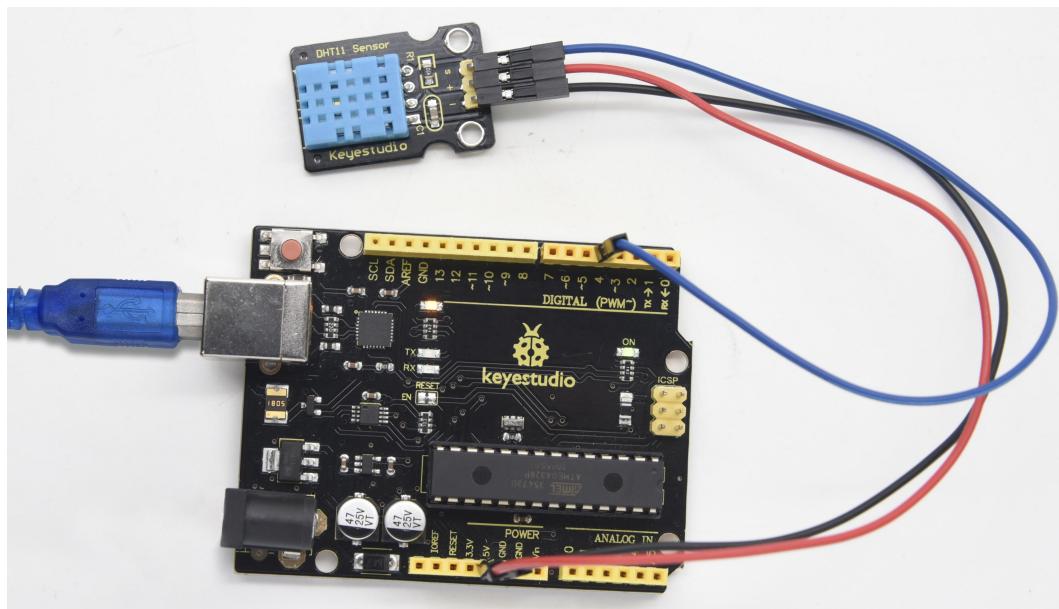
delay(1000);
```

}

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

6. Test Result

Wire it up well and upload the above code to V4.0 board.



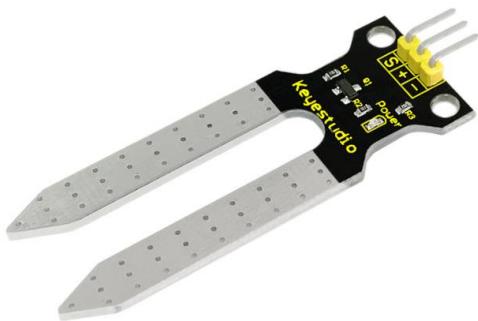
Then open the serial monitor and set the baud rate to 9600, you will see the current temperature and humidity value.

```
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
```

Autoscroll Show timestamp Newline 9600 baud Clear output

Project 31: Soil Humidity Sensor



1. Introduction

This is a simple soil humidity sensor aims to detect the soil humidity. If the soil is lack of water, the analog value output

by the sensor will decrease; otherwise, it will increase.

If you use this sensor to make an automatic watering device, it can detect whether your botany is thirsty so as to prevent it from withering when you go out. Using the sensor with Arduino controller makes your plant more comfortable and your garden smarter.

The soil humidity sensor module is not as complicated as you might think, and if you need to detect the soil in your project, it will be your best choice.

The sensor is set with two probes inserted into the soil, then with the current go through the soil, the sensor will get resistance value by reading the current changes between the two probes and convert such resistance value into moisture content.

The higher the moisture (less resistance) is, the higher conductivity the soil has.

The surface of the sensor have undergone metallization process to prolong its service life. Insert it into the soil and then use the AD converter to read it. With the help of this sensor, the plant can remind of you I need water.

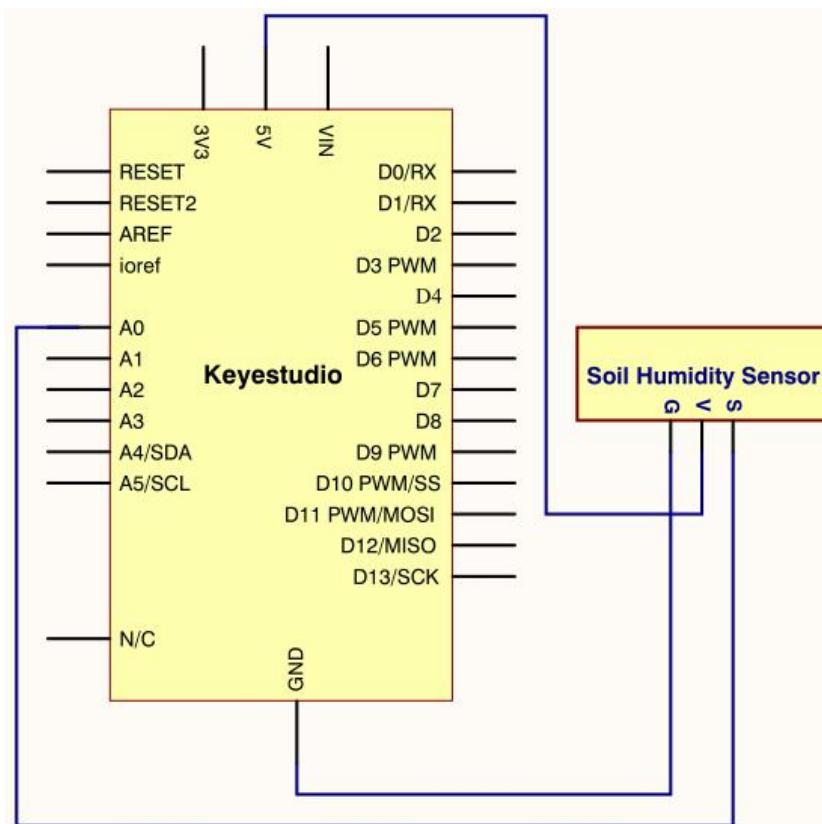
2.Hardware Required

- V4.0 Board or MEGA 2650 Board*1
- Soil Humidity Sensor*1
- Jumper Wire *1
- USB cable*1

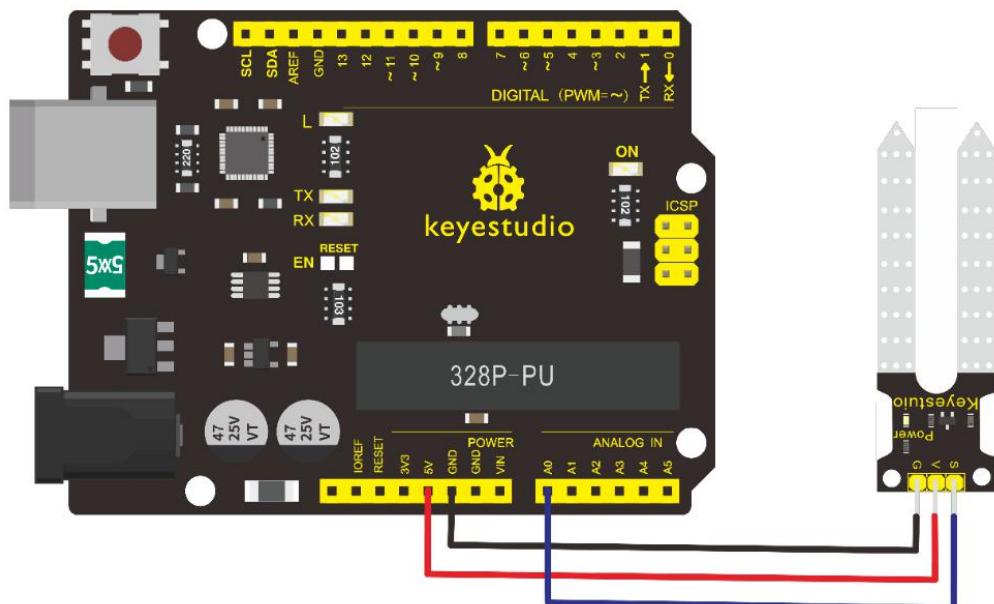
3.Specification

- Power Supply Voltage 3.3V or 5V
- Working Current \leq 20mA
- Output Voltage 0-2.3V (When the sensor is totally immersed in water, the voltage will be 2.3V)
- 5V power supply (the higher humidity, the higher the output voltage)
- Sensor Type Analog output
- Interface Definition Pin1- signal, pin2- GND, pin3 - VCC

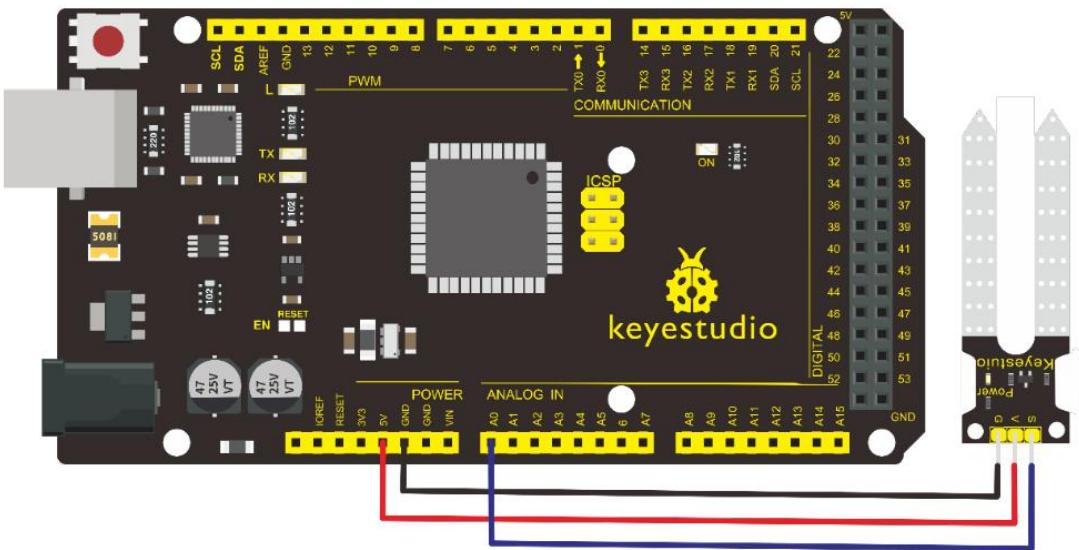
4.Circuit Connection



Connection for V4.0



Connection for MEGA 2560 R3



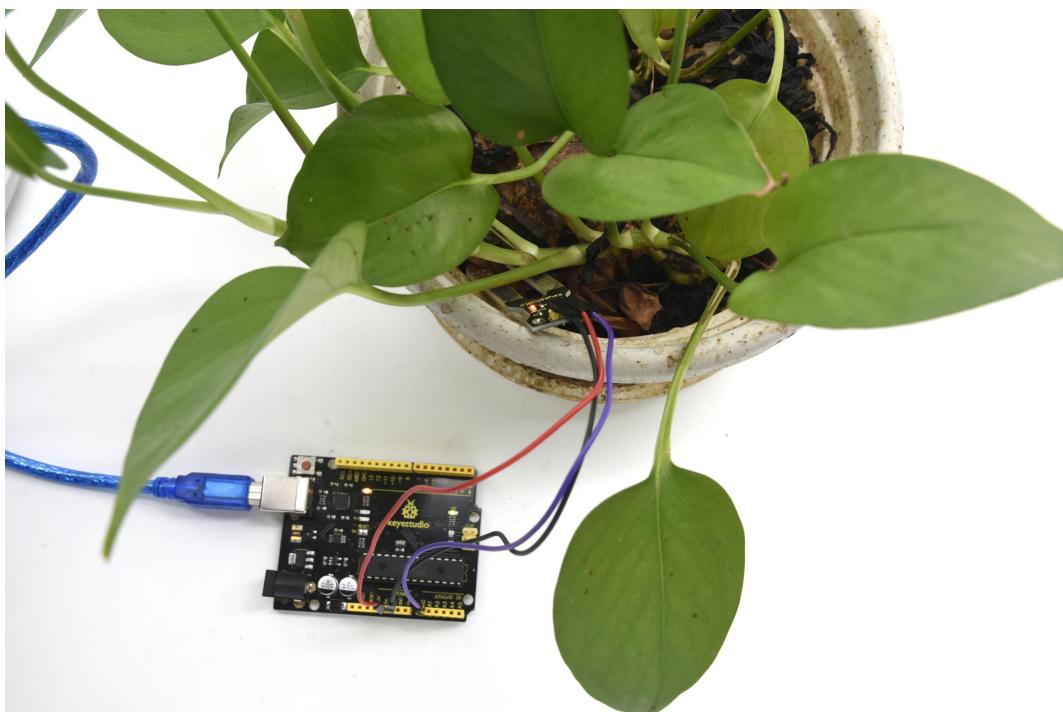
5. Sample Code

```
/*
keyestudio super learning kit
Project 31
soil humidity sensor
http://www.keyestudio.com
*/
void setup(){
Serial.begin(57600);
}
void loop(){
Serial.print("Moisture Sensor Value");
Serial.println(analogRead(0));
delay(100);
}
```

}

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

6. Test Result



Project 32: RC522 RFID Module



1. Introduction

MF522-AN module adopts Philips MFRC522 original reader circuit chip design, easy to use, low cost, suitable for

equipment development, development of advanced applications, the need for the user of RF card terminal design / production. It can be loaded directly into a variety of readers molds.

Module uses voltage of 3.3V, through the SPI interface using simple few lines, it can be directly connected to any CPU board communication modules to guarantee stable and reliable work and reader distance.

2.Hardware Required

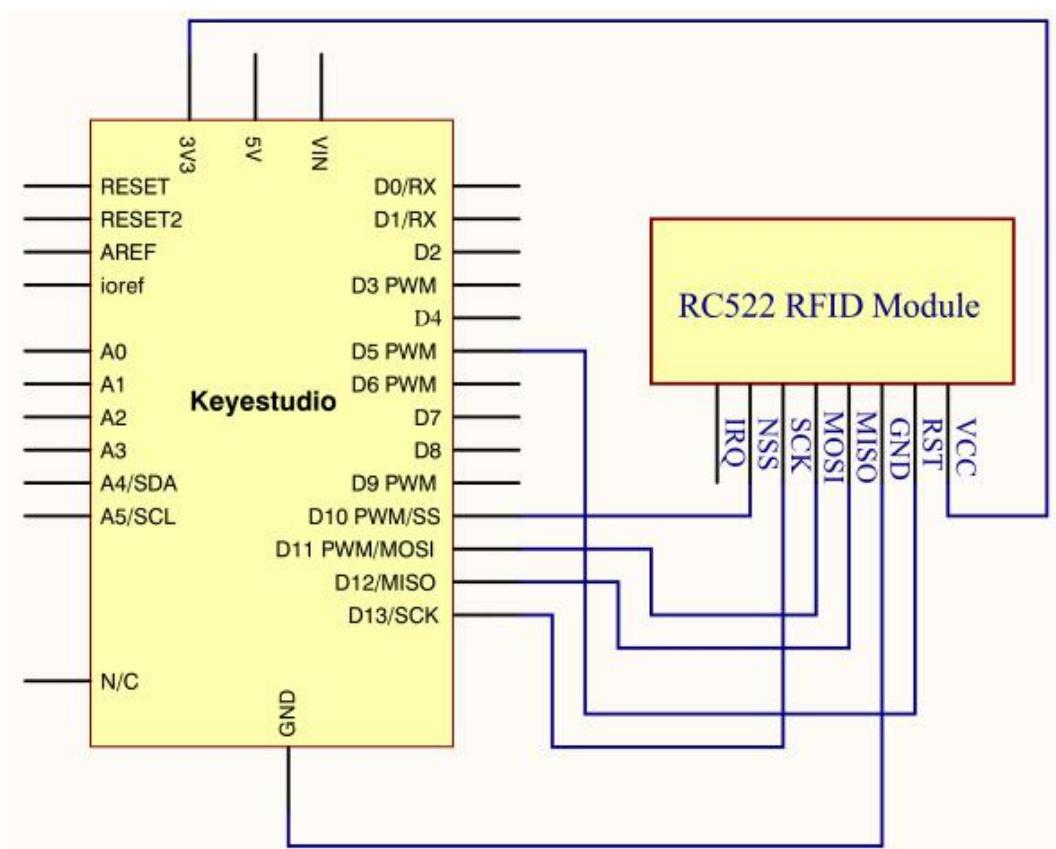
- V4.0 Board or MEGA 2650 Board*1
- RFID Module*1
- Jumper Wire *1
- USB cable*1

3.Specification

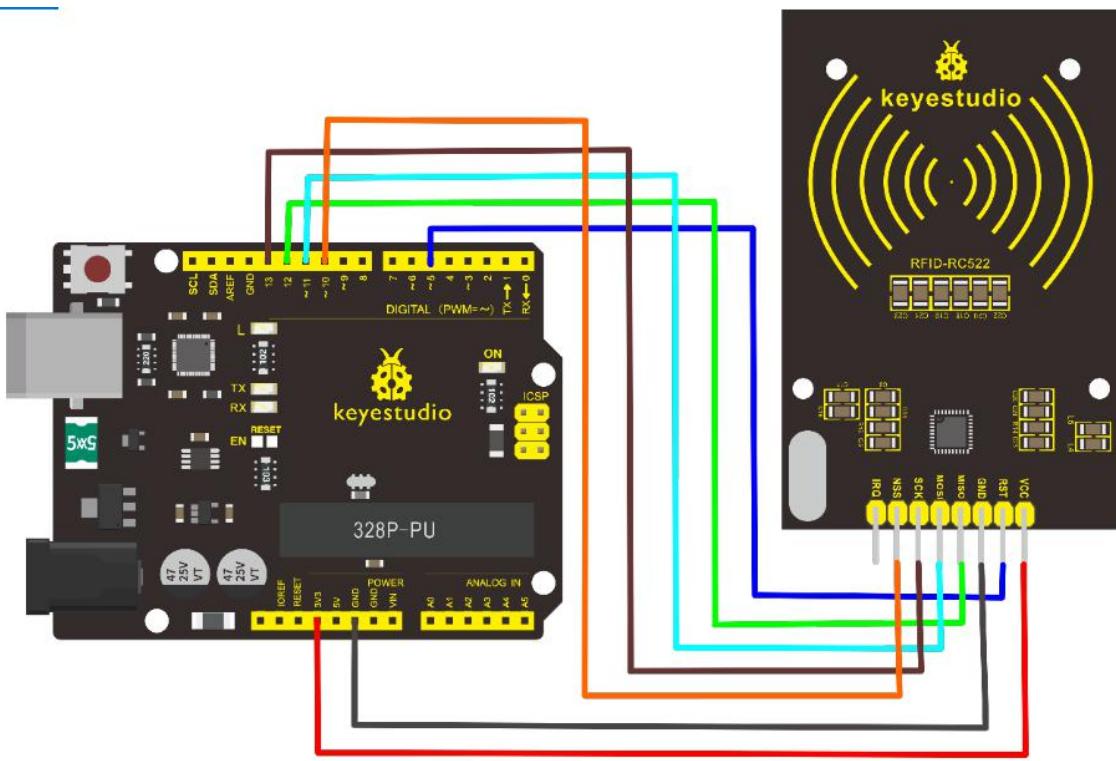
1. Current 13-26mA / DC 3.3V
2. Idle Current 10-13mA / DC 3.3V
3. Sleep Current <80uA
4. Peak Current <30mA
5. Operating Frequency 13.56MHz
6. Supported card types mifare1 S50, mifare1 S70, mifare UltraLight, mifare Pro, mifare Desfire

7. Environmental Operating Temperature -20-80 degrees Celsius
8. Environment Storage Temperature -40-85 degrees Celsius
9. Relative Humidity 5% -95%

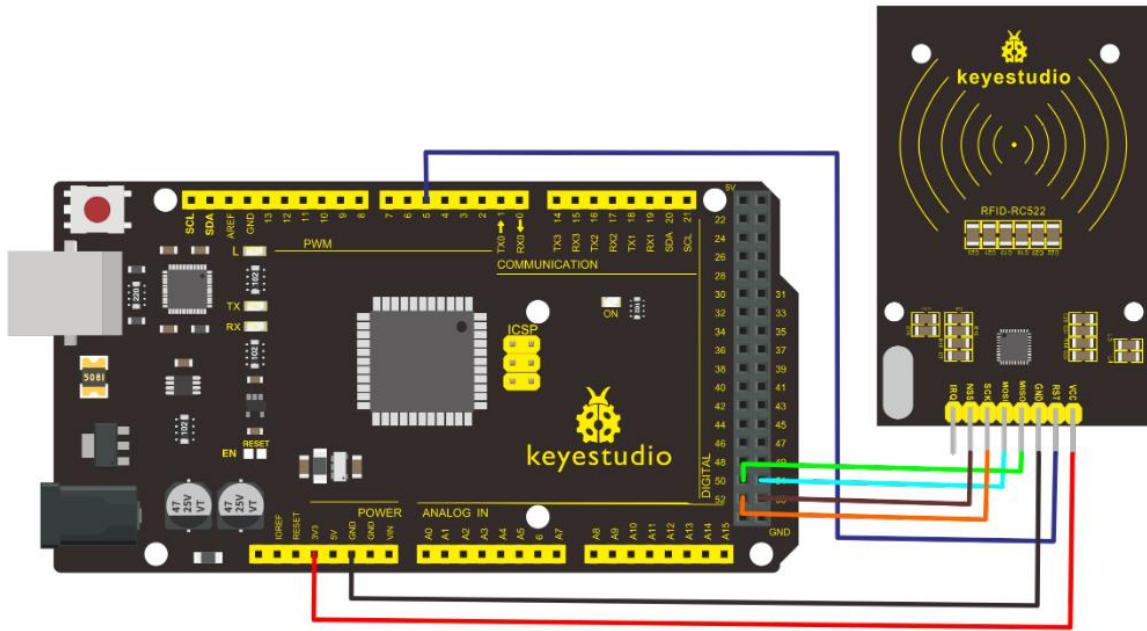
4. Circuit Connection



Connection for V4.0



Connection for 2560 R3



5. Sample Code

```
/*
```

keyestudio super learning kit

Project 32

RFID

<http://www.keyestudio.com>

```
*/  
  
#include <SPI.h>  
  
#define uchar unsigned char  
  
#define uint unsigned int  
  
#define MAX_LEN 16  
  
const int chipSelectPin = 10;//if the controller is UNO,328,168  
  
const int NRSTPD = 5;  
  
  
//MF522command word  
  
#define PCD_IDLE 0x00 //NO action;cancel current  
command  
  
#define PCD_AUTHENT 0x0E //verify key  
  
#define PCD_RECEIVE 0x08 //receive data  
  
#define PCD_TRANSMIT 0x04 //send data  
  
#define PCD_TRANSCEIVE 0x0C //receive and send data  
  
#define PCD_RESETPHASE 0x0F //reset  
  
#define PCD_CALCCRC 0x03 //CRC calculation
```

```
//Mifare_One Card command word

#define PICC_REQIDL          0x26          // line-tracking area is
dormant #define PICC_REQALL          0x52

//line-tracking area is interfered

#define PICC_ANTICOLL         0x93          //Anti collision
#define PICC_SEIECTTAG         0x93          //choose cards
#define PICC_AUTHENT1A         0x60          //Verify A key
#define PICC_AUTHENT1B         0x61          //Verify B key
#define PICC_READ               0x30          // Reader Module
#define PICC_WRITE              0xA0          // letter block
#define PICC_DECREMENT          0xC0
#define PICC_INCREMENT           0xC1
#define PICC_RESTORE             0xC2          //Transfer data to buffer
#define PICC_TRANSFER            0xB0          //Save buffer data
#define PICC_HALT                0x50          //Dormancy
```

//MF522 Error code returned when communication

```
#define MI_OK                 0
#define MI_NOTAGERR            1
#define MI_ERR                 2
```

```
//-----MFRC522 Register-----
```

```
//Page 0:Command and Status
```

#define	Reserved00	0x00
#define	CommandReg	0x01
#define	CommIEnReg	0x02
#define	DivIEnReg	0x03
#define	CommIrqReg	0x04
#define	DivIrqReg	0x05
#define	ErrorReg	0x06
#define	Status1Reg	0x07
#define	Status2Reg	0x08
#define	FIFODataReg	0x09
#define	FIFOLevelReg	0x0A
#define	WaterLevelReg	0x0B
#define	ControlReg	0x0C
#define	BitFramingReg	0x0D
#define	CollReg	0x0E
#define	Reserved01	0x0F

```
//Page 1:Command
```

#define	Reserved10	0x10
#define	ModeReg	0x11

```
#define TxModeReg 0x12
#define RxModeReg 0x13
#define TxControlReg 0x14
#define TxAutoReg 0x15
#define TxSelReg 0x16
#define RxSelReg 0x17
#define RxThresholdReg 0x18
#define DemodReg 0x19
#define Reserved11 0x1A
#define Reserved12 0x1B
#define MifareReg 0x1C
#define Reserved13 0x1D
#define Reserved14 0x1E
#define SerialSpeedReg 0x1F
```

//Page 2:CFG

```
#define Reserved20 0x20
#define CRCResultRegM 0x21
#define CRCResultRegL 0x22
#define Reserved21 0x23
#define ModWidthReg 0x24
#define Reserved22 0x25
#define RFCfgReg 0x26
```

```
#define GsNReg 0x27  
  
#define CWGsPReg 0x28  
  
#define ModGsPReg 0x29  
  
#define TModeReg 0x2A  
  
#define TPrescalerReg 0x2B  
  
#define TReloadRegH 0x2C  
  
#define TReloadRegL 0x2D  
  
#define TCounterValueRegH 0x2E  
  
#define TCounterValueRegL 0x2F
```

//Page 3:TestRegister

```
#define Reserved30 0x30  
  
#define TestSel1Reg 0x31  
  
#define TestSel2Reg 0x32  
  
#define TestPinEnReg 0x33  
  
#define TestPinValueReg 0x34  
  
#define TestBusReg 0x35  
  
#define AutoTestReg 0x36  
  
#define VersionReg 0x37  
  
#define AnalogTestReg 0x38  
  
#define TestDAC1Reg 0x39  
  
#define TestDAC2Reg 0x3A  
  
#define TestADCReg 0x3B
```

```

#define Reserved31 0x3C
#define Reserved32 0x3D
#define Reserved33 0x3E
#define Reserved34 0x3F

uchar serNum[5];

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

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

uchar sectorNewKeyA[16][16] = {{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                               {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                               {0xFF, 0x07, 0x80, 0x69, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                               {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                               {0xFF, 0x07, 0x80, 0x69, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                               };

void setup() {
    Serial.begin(9600); // RFID reader SOUT pin connected
    to Serial RX pin at 2400bps
    // start the SPI library:
    SPI.begin();
}

```

```

pinMode(chipSelectPin,OUTPUT);           // Set digital pin 10 as OUTPUT
to connect it to the RFID /ENABLE pin

digitalWrite(chipSelectPin, LOW);        // Activate the RFID reader

pinMode(NRSTPD,OUTPUT);                // Set digital pin 10 , Not Reset and
Power-down

digitalWrite(NRSTPD, HIGH);

MFRC522_Init();

}

void loop()
{
    uchar i,tmp;
    uchar status;

    uchar str[MAX_LEN];

    uchar RC_size;

    uchar blockAddr; //Select the address of the operation 0~63

    // searching card, return card type

    status = MFRC522_Request(PICC_REQIDL, str);

```

```
if (status == MI_OK)

{

}

status = MFRC522_Anticoll(str);

memcpy(serNum, str, 5);

if (status == MI_OK)

{

    Serial.println("The card's number is : ");

    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(" ");

}

// select card, return card capacity

RC_size = MFRC522_SelectTag(serNum);

if (RC_size != 0)

{}
```

```

// write data card

blockAddr = 7;      // data block 7

status = MFRC522_Auth(PICC_AUTHENT1A, blockAddr,
sectorKeyA[blockAddr/4], serNum); // authentication

if (status == MI_OK)

{

// write data

status = MFRC522_Write(blockAddr, sectorNewKeyA[blockAddr/4]);

Serial.print("set the new card password, and can modify

the data of the Sector: ");

Serial.print(blockAddr/4,DEC);

// write data

blockAddr = blockAddr - 3 ;

status = MFRC522_Write(blockAddr, writeDate);

if(status == MI_OK)

{

Serial.println("OK!");

}

}

```

```

// read card

blockAddr = 7;      // data block 7

status = MFRC522_Auth(PICC_AUTHENT1A, blockAddr,
sectorNewKeyA[blockAddr/4], serNum); // authentication

if (status == MI_OK)

{

    // read data

    blockAddr = blockAddr - 3 ;

    status = MFRC522_Read(blockAddr, str);

    if (status == MI_OK)

    {

        Serial.println("Read from the card ,the data is :

");

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

        {

            Serial.print(str[i]);

        }

        Serial.println(" ");

    }

    Serial.println(" ");

    MFRC522_Halt();      // command card to enter standby mode

```

```
}

void Write_MFRC522(uchar addr, uchar val)

{

    digitalWrite(chipSelectPin, LOW);

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

    SPI.transfer(val);

    digitalWrite(chipSelectPin, HIGH);

}

uchar Read_MFRC522(uchar addr)

{

    uchar val;

    digitalWrite(chipSelectPin, LOW);

    // address format: 1XXXXXX0

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

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

digitalWrite(chipSelectPin, HIGH);

return val;

}

void SetBitMask(uchar reg, uchar mask)

{

uchar tmp;

tmp = Read_MFRC522(reg);

Write_MFRC522(reg, tmp | mask); // set bit mask

}

void ClearBitMask(uchar reg, uchar mask)

{

uchar tmp;

tmp = Read_MFRC522(reg);

Write_MFRC522(reg, tmp & (~mask)); // clear bit mask
```

```
}

void AntennaOn(void)

{

uchar temp;

temp = Read_MFRC522(TxControlReg);

if (!(temp & 0x03))

{

    SetBitMask(TxControlReg, 0x03);

}

}

void AntennaOff(void)

{

    ClearBitMask(TxControlReg, 0x03);

}

void MFRC522_Reset(void)

{
```

```

        Write_MFRC522(CommandReg, PCD_RESETPHASE);

    }

void MFRC522_Init(void)

{

    digitalWrite(NRSTPD,HIGH);

    MFRC522_Reset();




//Timer: TPrescaler*TreloadVal/6.78MHz = 24ms

    Write_MFRC522(TModeReg, 0x8D);      //Tauto=1; f(Timer) =

6.78MHz/TPreScaler

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

    Write_MFRC522(TReloadRegL, 30);

    Write_MFRC522(TReloadRegH, 0);




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

    Write_MFRC522(ModeReg, 0x3D);    //CRC initial value


    AntennaOn();      // open antenna

}

```

```

uchar MFRC522_Request(uchar reqMode, uchar *TagType)

{

    uchar status;

    uint backBits;      // received data bits

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

    TagType[0] = reqMode;

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

    if ((status != MI_OK) || (backBits != 0x10))

    {

        status = MI_ERR;

    }

    return status;

}

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

```

```

{

uchar status = MI_ERR;

uchar irqEn = 0x00;

uchar waitIRq = 0x00;

uchar lastBits;

uchar n;

uint i;

switch (command)

{

case PCD_AUTHENT: // card key authentication

{

irqEn = 0x12;

waitIRq = 0x10;

break;

}

case PCD_TRANSCEIVE: // send data in FIFO

{

irqEn = 0x77;

waitIRq = 0x30;

break;

}

```

default:

 break;

}

 Write_MFRC522(CommIEnReg, irqEn|0x80); // Allow interrupt request

 ClearBitMask(CommIrqReg, 0x80); // clear bits of all interrupt request

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

 Write_MFRC522(CommandReg, PCD_IDLE); //NO action; cancel current
 command ???

// write data into FIFO

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

 {

 Write_MFRC522(FIFODataReg, sendData[i]);

 }

// execute command

 Write_MFRC522(CommandReg, command);

 if (command == PCD_TRANSCEIVE)

 {

```

SetBitMask(BitFramingReg, 0x80);      //StartSend=1,transmission of data

starts

}

// wait for the completion of data receiving

i = 2000; // adjust i according to clock frequency, maximum waiting time of
operating M1 is 25ms  ???

do

{

//CommIrqReg[7..0]

//Set1 TxIRq RxIRq IdleIRq HiAlerIRq LoAlertIRq ErrIRq TimerIRq

n = Read_MFRC522(CommIrqReg);

i--;

}

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




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

if (i != 0)

{

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

ProtecolErr

```

```

{

    status = MI_OK;

    if (n & irqEn & 0x01)

    {

        status = MI_NOTAGERR;      //??

    }

}

if (command == PCD_TRANSCEIVE)

{

    n = Read_MFRC522(FIFOLevelReg);

    lastBits = Read_MFRC522(ControlReg) & 0x07;

    if (lastBits)

    {

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

    }

    else

    {

        *backLen = n*8;

    }

}

if (n == 0)

{

```

```

n = 1;

}

if (n > MAX_LEN)

{

n = MAX_LEN;

}

// read data which FIFO received

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

{

backData[i] = Read_MFRC522(FIFODataReg);

}

}

}

else

{

status = MI_ERR;

}

}

//SetBitMask(ControlReg,0x80);           //timer stops

```

```

//Write_MFRC522(CommandReg, PCD_IDLE);

return status;

}

uchar MFRC522_Anticoll(uchar *serNum)
{
    uchar status;
    uchar i;
    uchar serNumCheck=0;
    uint unLen;

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

    serNum[0] = PICC_ANTICOLL;
    serNum[1] = 0x20;
    status = MFRC522_ToCard(PCD_TRANSCEIVE, serNum, 2, serNum, &unLen);

    if (status == MI_OK)
    {
        // verify card sequence number
        for (i=0; i<4; i++)

```

```

{

    serNumCheck ^= serNum[i];

}

if (serNumCheck != serNum[i])

{

    status = MI_ERR;

}

}

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


return status;

}

void CalulateCRC(uchar *pIndata, uchar len, uchar *pOutData)

{

    uchar i, n;

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

    SetBitMask(FIFOLevelReg, 0x80); // clear FIFO pointer

    //Write_MFRC522(CommandReg, PCD_IDLE);
}

```

```

// write data into FIFO

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

{

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

}

Write_MFRC522(CommandReg, PCD_CALCCRC);

// wait for the completion of CRC calculation

i = 0xFF;

do

{

    n = Read_MFRC522(DivIrqReg);

    i--;

}

while ((i!=0) && !(n&0x04));      //CRCIrq = 1

// read CRC calculation result

pOutData[0] = Read_MFRC522(CRCResultRegL);

pOutData[1] = Read_MFRC522(CRCResultRegM);

}

uchar MFRC522_SelectTag(uchar *serNum)

```

```

{

    uchar i;

    uchar status;

    uchar size;

    uint recvBits;

    uchar buffer[9];

//ClearBitMask(Status2Reg, 0x08);      //MF Crypto1 On=0

    buffer[0] = PICC_SELECTTAG;

    buffer[1] = 0x70;

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

    {

        buffer[i+2] = *(serNum+i);

    }

CalulateCRC(buffer, 7, &buffer[7]);    //??

status = MFRC522_ToCard(PCD_TRANSCEIVE, buffer, 9, buffer, &recvBits);

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

{

    size = buffer[0];

}

```

```

else

{

size = 0;

}

return size;

}

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

{

uchar status;

uint recvBits;

uchar i;

uchar buff[12];

// Verification commands + block address + sector password + card sequence
number

buff[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);

}

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

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

{

status = MI_ERR;

}

return status;

}

uchar MFRC522_Read(uchar blockAddr, uchar *recvData)

{

uchar status;

uint unLen;

recvData[0] = PICC_READ;

```

```

recvData[1] = blockAddr;

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

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

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

{

    status = MI_ERR;

}

return status;

}

uchar MFRC522_Write(uchar blockAddr, uchar *writeData)

{

    uchar status;

    uint recvBits;

    uchar i;

    uchar buff[18];

    buff[0] = PICC_WRITE;

    buff[1] = blockAddr;

```

```

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

status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 4, buff, &recvBits);

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

{

    status = MI_ERR;

}

if (status == MI_OK)

{

    for (i=0; i<16; i++) // write 16Byte data into FIFO

    {

        buff[i] = *(writeData+i);

    }

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

    status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 18, buff, &recvBits);

}

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

{

    status = MI_ERR;

}

```

```

    return status;

}

void MFRC522_Halt(void)

{
    uchar status;

    uint unLen;

    uchar buff[4];

    buff[0] = PICC_HALT;

    buff[1] = 0;

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

    status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 4, buff,&unLen);

}
/////////////////////////////////////////////////////////////////

```

Note if you want to use MEGA 2560 R3, please in the code
change

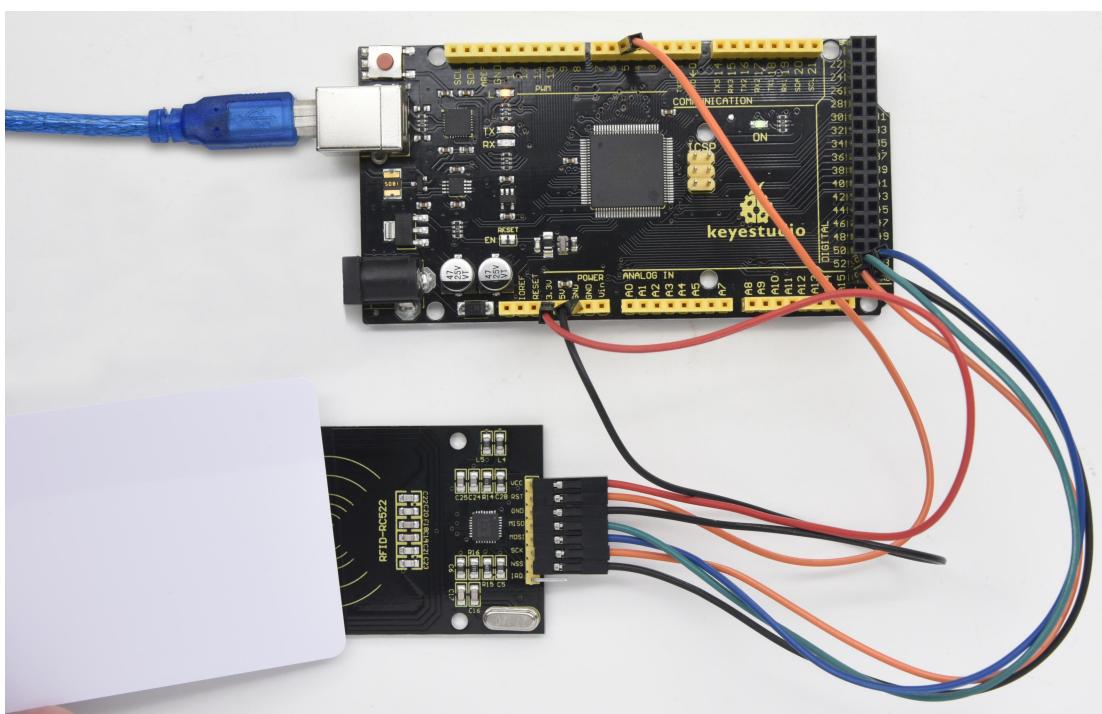
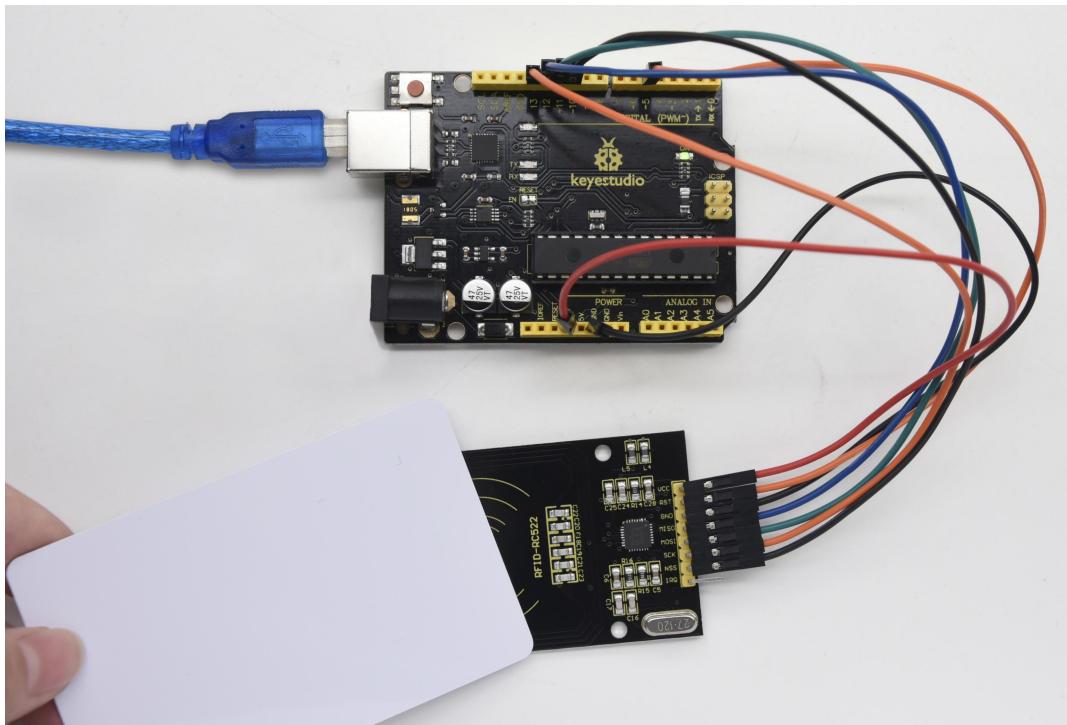
```

const int chipSelectPin = 10;//if the controller is V4.0
const int chipSelectPin = 53;//if the controller is MEGA2560

```

6. Test Result

In this experiment, when the IC card approaches, RFID module will write data to the IC card and read the card's data, you can see the data on the monitor window. Shown below.



The card's number is :
10110011000001010100010011000101001
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. Resources

Download code, driver and library:

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