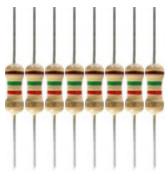
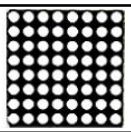
Keyes Ultimate Starter Kit for Arduino

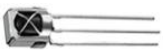
CONTENT

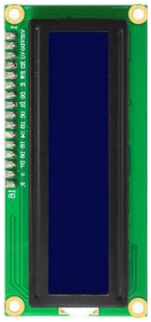
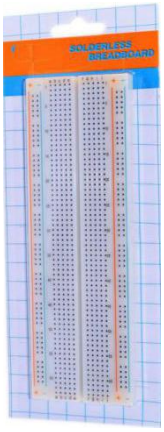
1. Kit Description

Keyes ultimate starter kit includes various sensor components, modules, control board and so on, which is very popular for electronic hobbyists to learn the microcontroller. We also provide you with more detailed learning projects of this kit based on the control board, including wiring methods, test code and more. Helping you to have a preliminary understanding of those electronic components and control board.

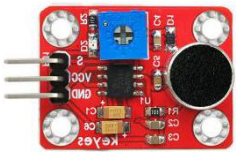
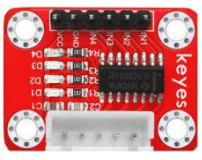
2. Component List

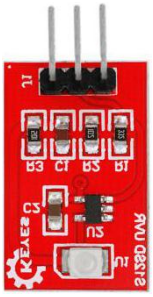
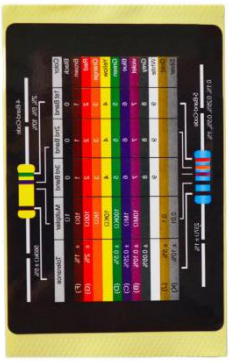
No.	Item	Specification	Quantity	Picture
1	LED	F5-white-to-red	5	
2	LED	F5-white-to-yellow	5	
3	LED	F5-white-to-blue	5	
4	Resistor	carbon film color band 1/4W 1% 220R	8	
5	Resistor	carbon film color band 1/4W 1% 1K	5	
6	Resistor	carbon film color band 1/4W 1% 10K	5	
7	Dot matrix	20*20MM 1.9MM red common cathode	1	

8	LED display	1-digit 0.56inch common cathode red	1	
9	LED display	4-digit 0.36inch common cathode red	1	
10	IC	74HC595 DIP	1	
11	potentiometer	3386 MU 103 （3Pin straight）	1	
12	Buzzer	passive 12*8.5MM 5V ordinary fission 2K	1	
13	Buzzer	active 12*9.5MM 5V ordinary fission 2300Hz	1	
14	Tactile button	12*12*7.3MM plugin	4	
15	Button cap	A24 yellow cap (12*12*7.3) round	4	
16	Sensor component	LM35DZ	1	
17	Sensor component	5MM photocell	3	
18	Sensor component	IR receiver 5MM flame	1	
19	Sensor component	IR receiver VS1838B	1	
20	Ball switch	HDX-2801 same pin	2	

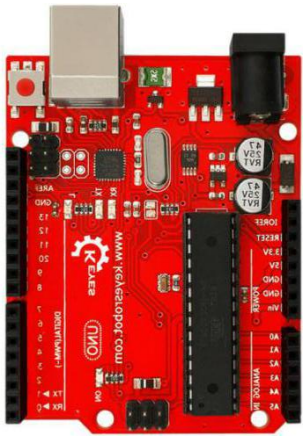
21	LCD	1602 I2C blue screen	1	
22	Module	4*4 thin-film keypad	1	
23	Breadboard	ZY-102 830-hole white (packed in paper card)	1	
24	Module	5V stepper motor	1	
25	USB cable	AM/BM transparent blue OD:5.0 L=50cm	1	

26	Remote control	JMP-1 17-button 86*40*6.5MM black	1	
27	Breadboard jumper wire	Pack of 65	1	
28	Dupont line	M-F 20CM/40P/2.54/10 strands of copper clad aluminum NO.24 wire BL	0.5	
29	Servo motor	SG90 9G 23*12.2*29mm (blue and eco-friendly)	1	
30	Battery clip lead	premium 9V battery clip lead to power connector	1	
31	IC card	White card 85.5*54*0.80MM	1	
32	Key chain	TAG-03 41*33*403mm ABS blue	1	

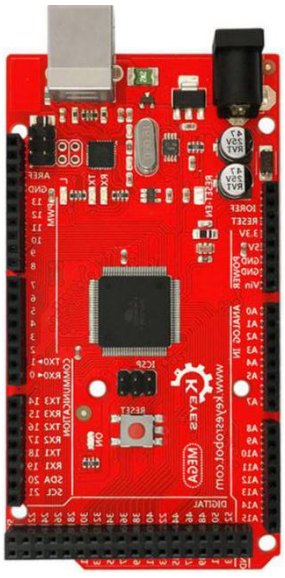
33	Keyes module	Keyes RFID－RC522 module (red and eco-friendly)	1	
34	Keyes sensor	keyes sound sensor (red and eco-friendly)	1	
35	Keyes sensor	keyes ultrasonic sensor	1	
36	Keyes module	keyes RGB module module (red and eco-friendly)	1	
37	Keyes module	keyes 5V single relay module (red and eco-friendly)	1	
38	Keyes sensor	keyes DHT11 temperature and humidity sensor (red and eco-friendly)	1	
39	Keyes module	keyes 1302 clock module (red and eco-friendly)	1	
40	Keyes sensor	keyes PIR Motion Sensor (red and eco-friendly)	1	
41	Keyes drive board	Keyes ULN2003 stepper motor driver board (red and eco-friendly)	1	

42	Keyes sensor	keyes MQ-2 gas sensor (red and eco-friendly)	1	
43	Keyes sensor	keyes joystick module sensor (red and eco-friendly)	1	
44	Sensor module	keyes TMD27713 distance sensor	1	
45	Sensor module	keyes MMA8452Q three-axis digital acceleration sensor	1	
46	Sensor module	keyes GUVA-S12SD 3528 Ultraviolet sensor	1	
47	Resistor reference card	100*70MM	1	

KE0084

48	Development Board	Keyes UNO R3 development board for arduino (red and eco-friendly)	1	

KE0085

48	Development Board	Keyes 2560 R3 development board for arduino (red and eco-friendly)	1	

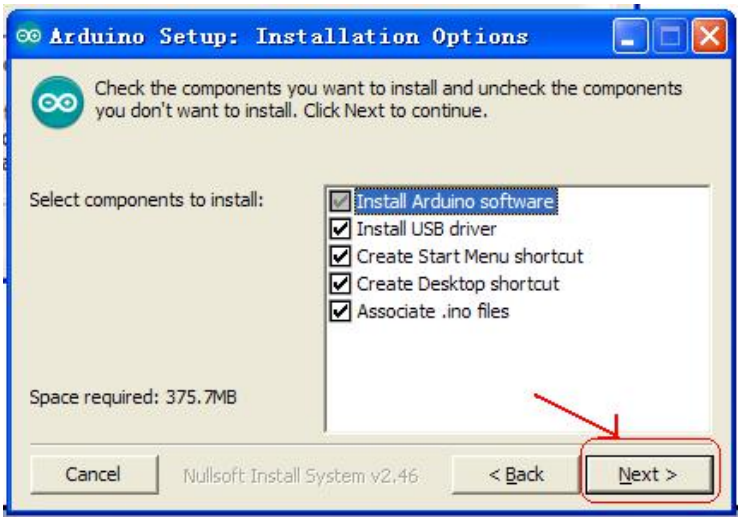
3. Arduino IDE and Driver Installation

When getting the keyes development board, first of all you have to install the Arduino IDE and the driver, and all relevant files can be found on the official website.

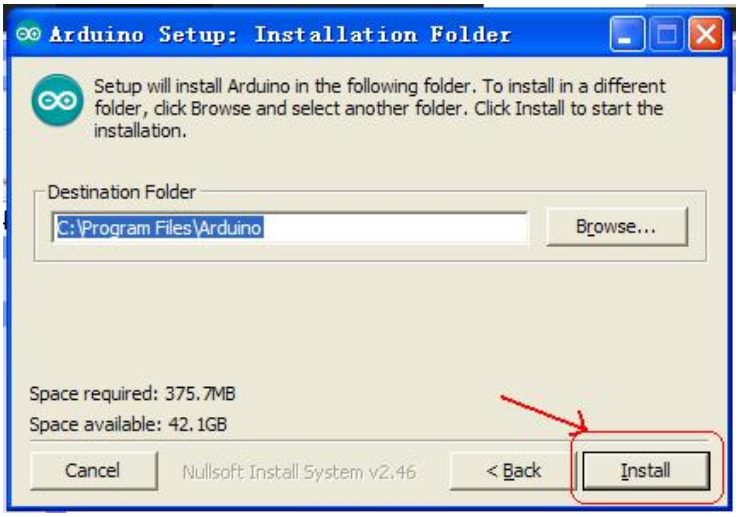
The following link includes various systems, various versions of the Arduino IDE and drivers whatever you choose.

https://www.arduino.cc/en/Main/OldSoftwareReleases#1.5.x

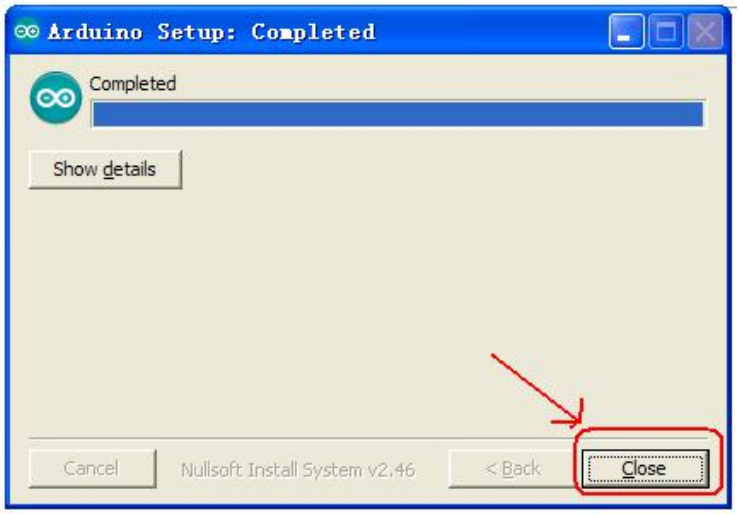
Click “I Agree”. Then, click “Next”



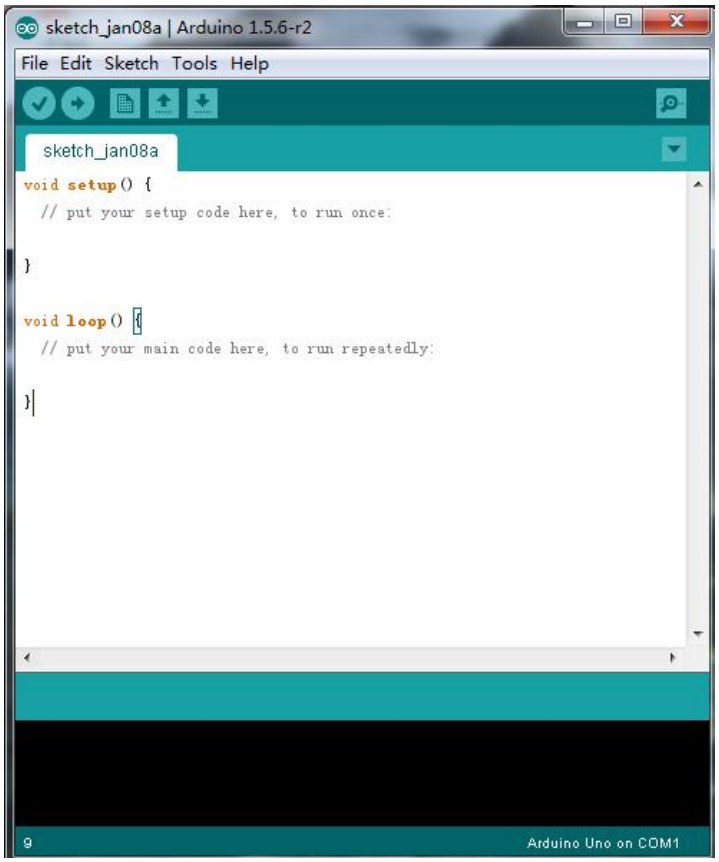
Next, click “Install”.



Finally, click “Close” after completing the installation.



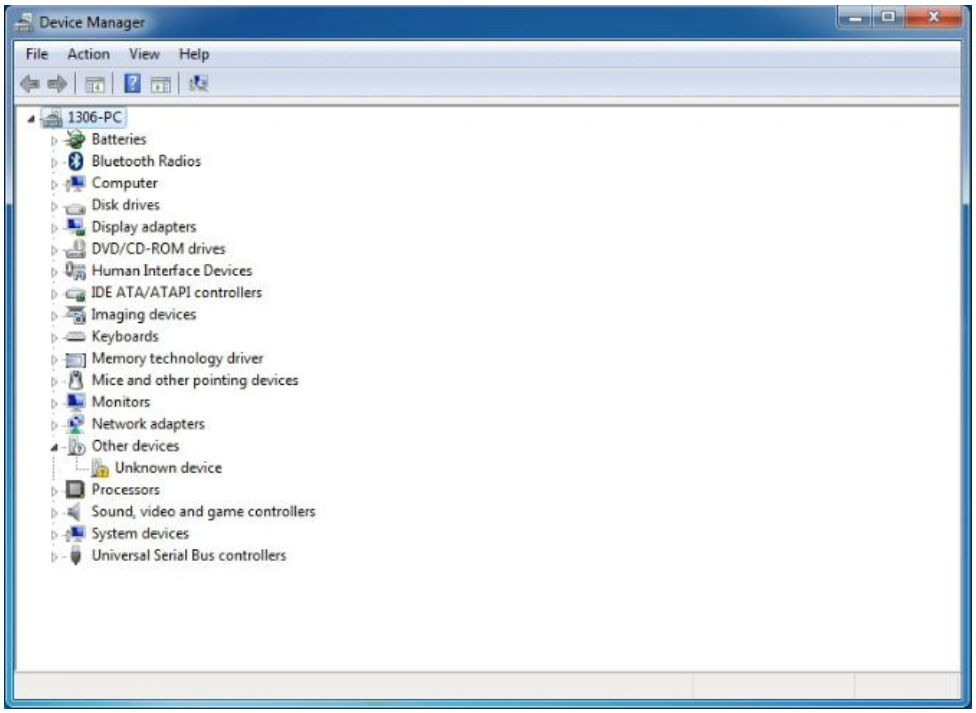
The figure below shows the successful installation of Arduino1.5.6 version:



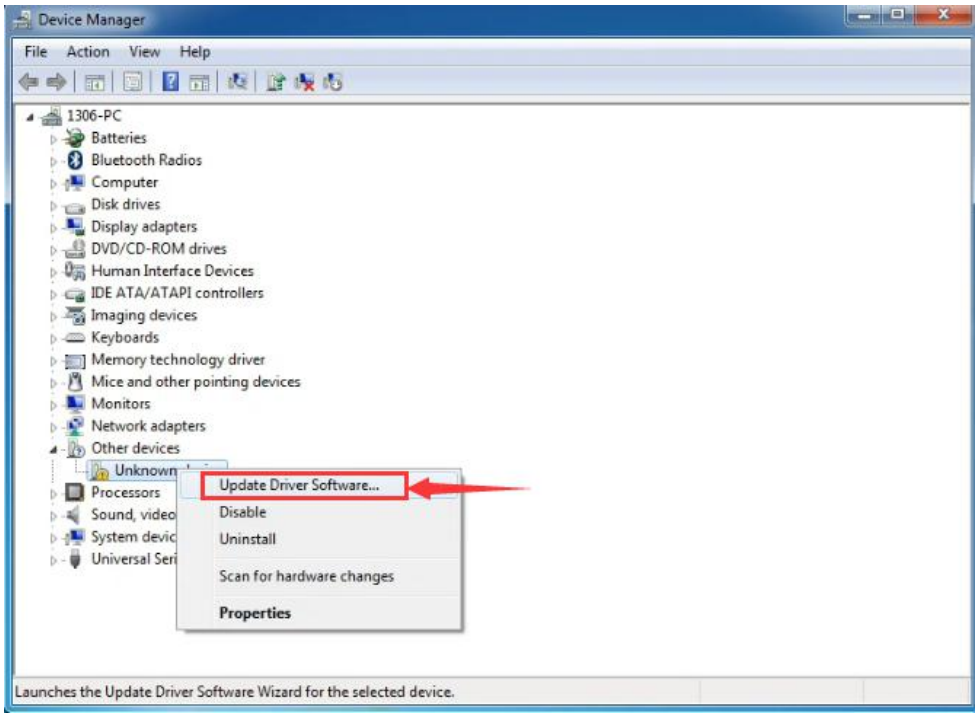
Next, we will introduce the driver installation of Keyes UNO R3 development board. Actually you can use the same method to install the driver of Keyes 2560 R3 development board.

Let's move on to the driver installation in the WIN 7 system.

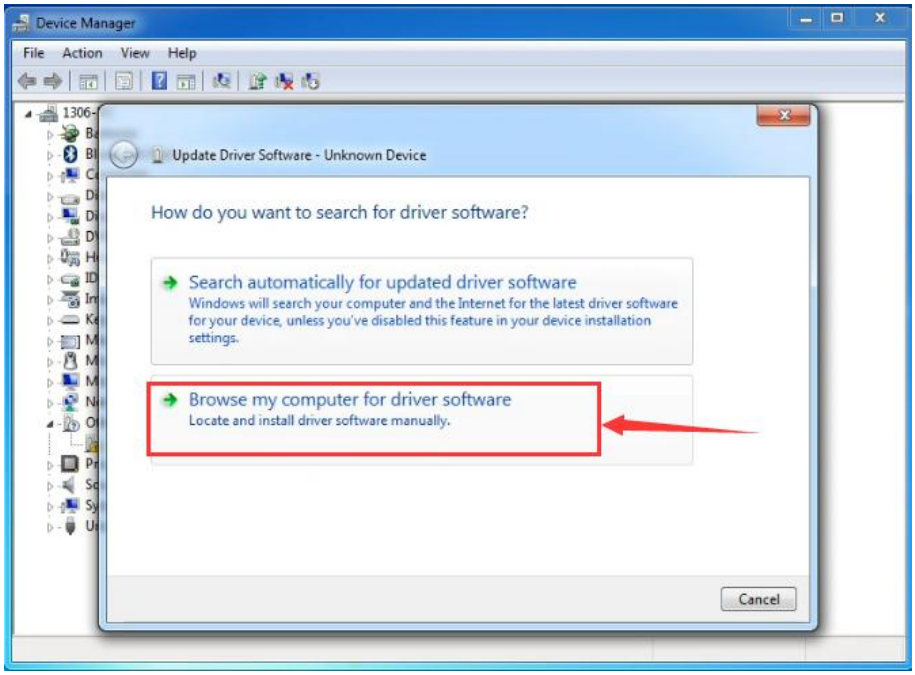
A. When you connect Keyes UNO R3 to your computer at the first time, right click "Computer" —>"Properties"—> "Device manager", you can see "Unknown devices".



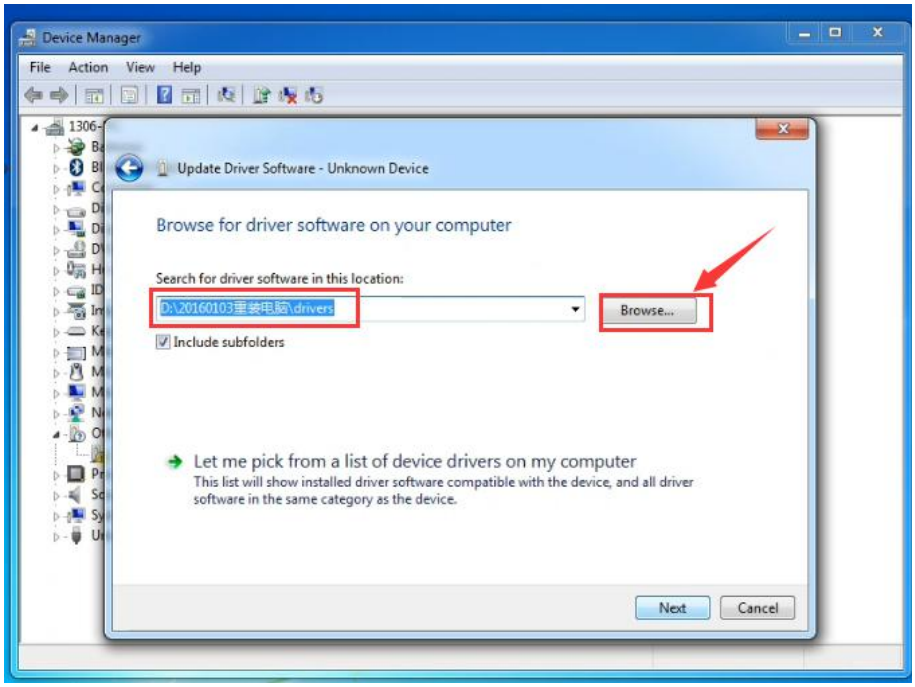
B. Click "Unknown devices", select "Update Driver software".



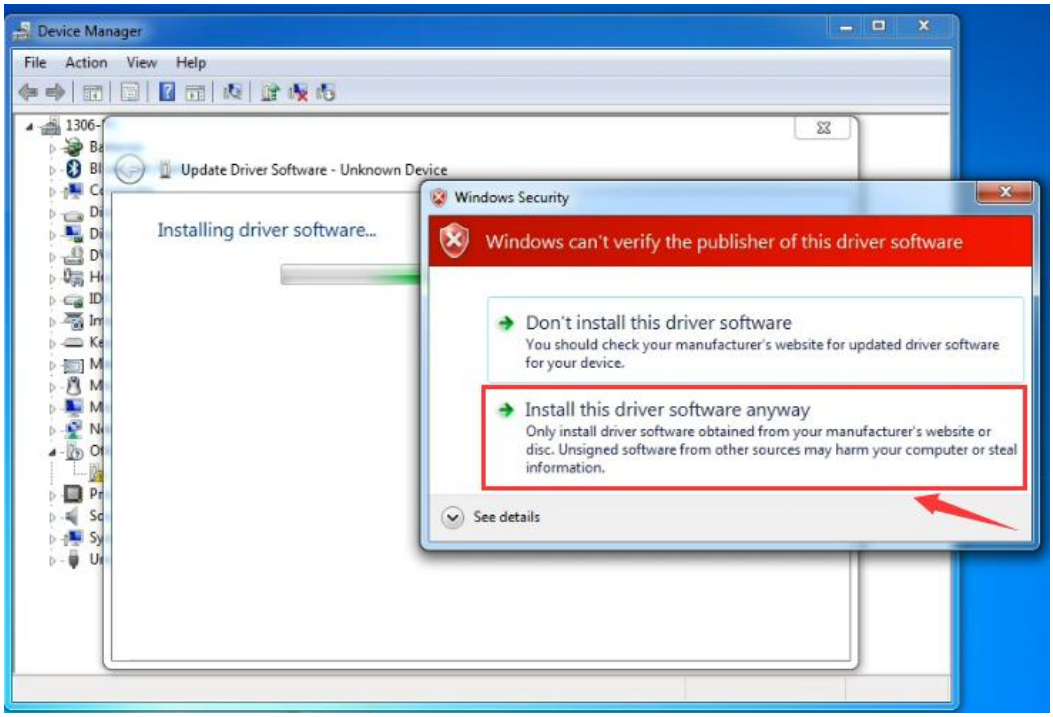
C. In this page, click “Browse my computer for driver software”.



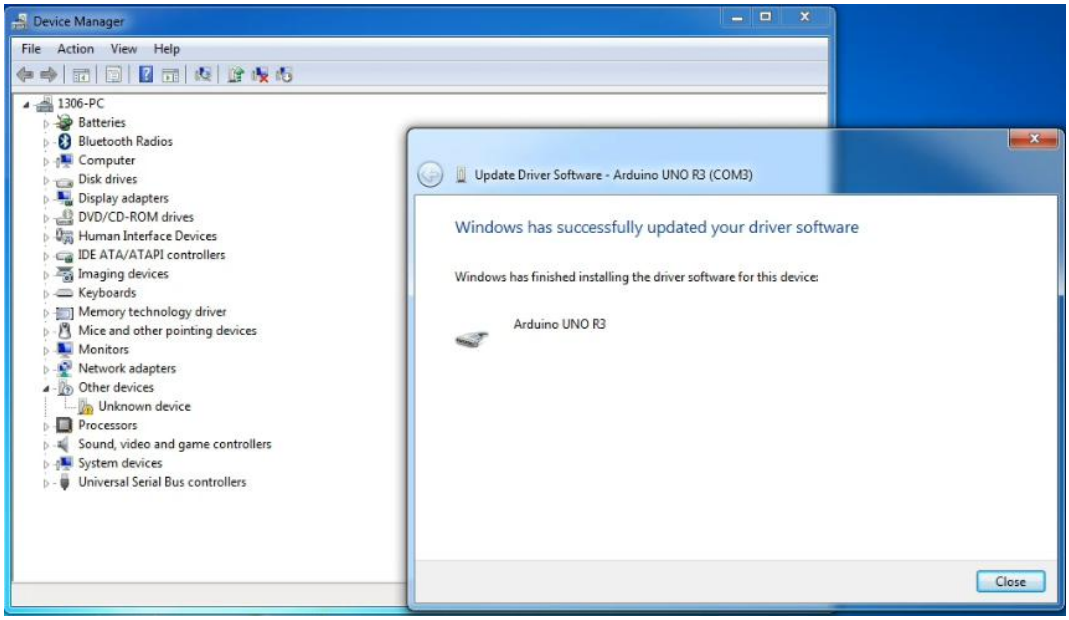
D. Find the “drivers” folder.



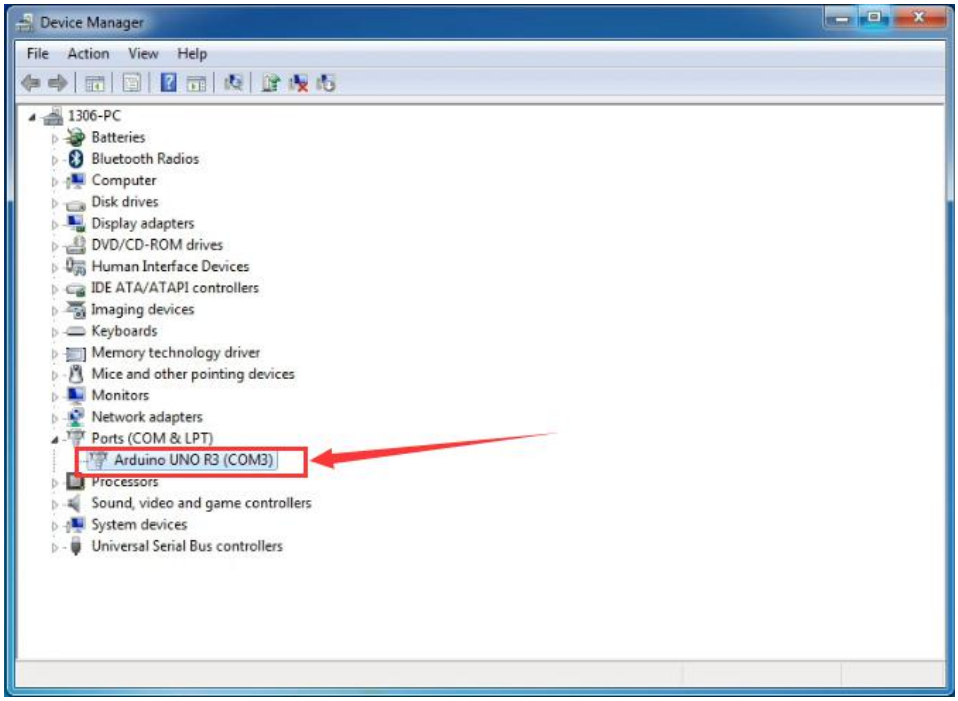
E. Click “Next”; select “Install this driver software anyway” to begin the installation.



F. Installation completed, click “Close”.



G. After installation, go to see the “Device manager” again. right click “Computer” —> “Properties”—> “Device manager”, you can see the device as below figure shown.



4. Using Method of Arduino IDE

When successfully installing the USB driver of Keyes UNO R3 development board, you can find the corresponding serial port in Windows Device Manager.
Next, we will show you the first program to show the “Hello World!” on the serial monitor.

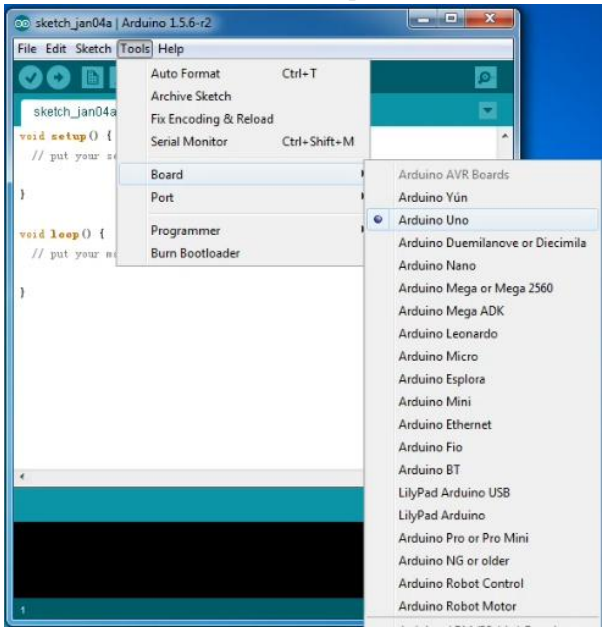
Sample Code as below:

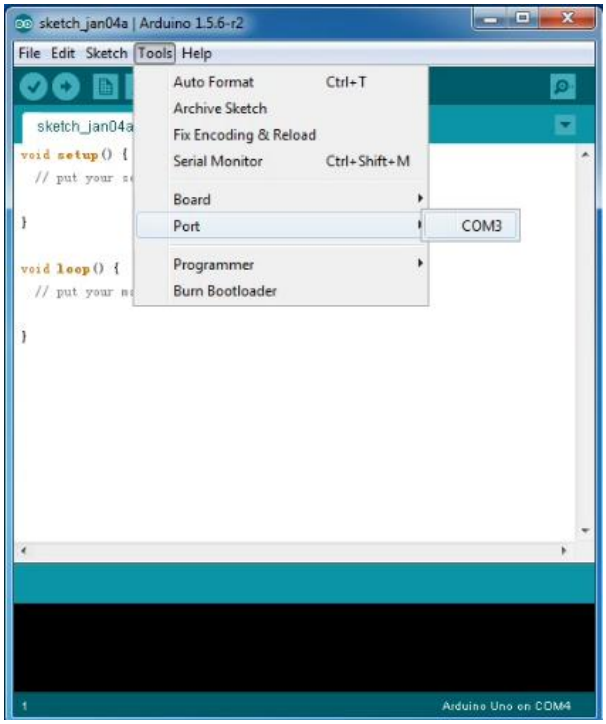
```
//////////////////////////////////////////
int val;
int ledpin=13;
void setup()
{
Serial.begin(9600);
pinMode(ledpin,OUTPUT);
}
void loop()
{
val=Serial.read();
if(val=='R')
{
digitalWrite(ledpin,HIGH);
delay(500);
digitalWrite(ledpin,LOW);
delay(500);
```

```
Serial.println("Hello World!");
}
}
//////////////////////////////////////////
```

Open Arduino software, print the program to make keyes UNO R3 development board display the character “Hello World！”. When the board receives the instruction, D13 indicator light on the board blinks and “Hello World！ ” is displayed on the monitor.

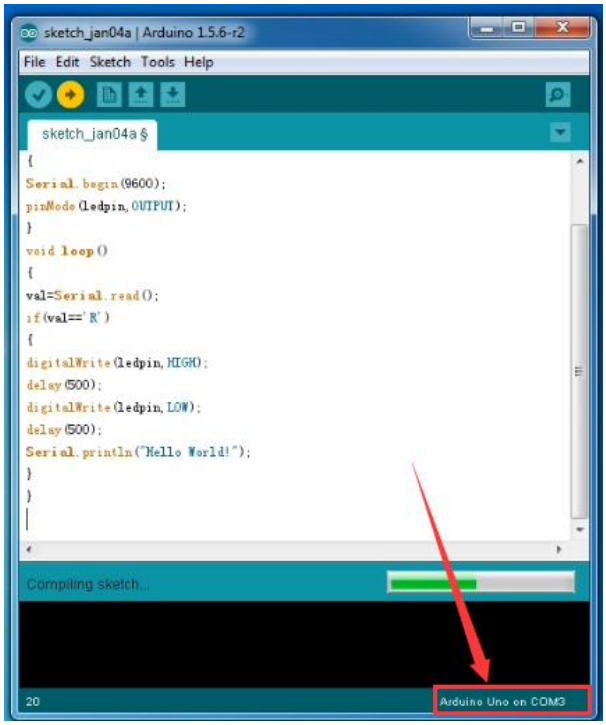
First set the Board and COM port, shown below.





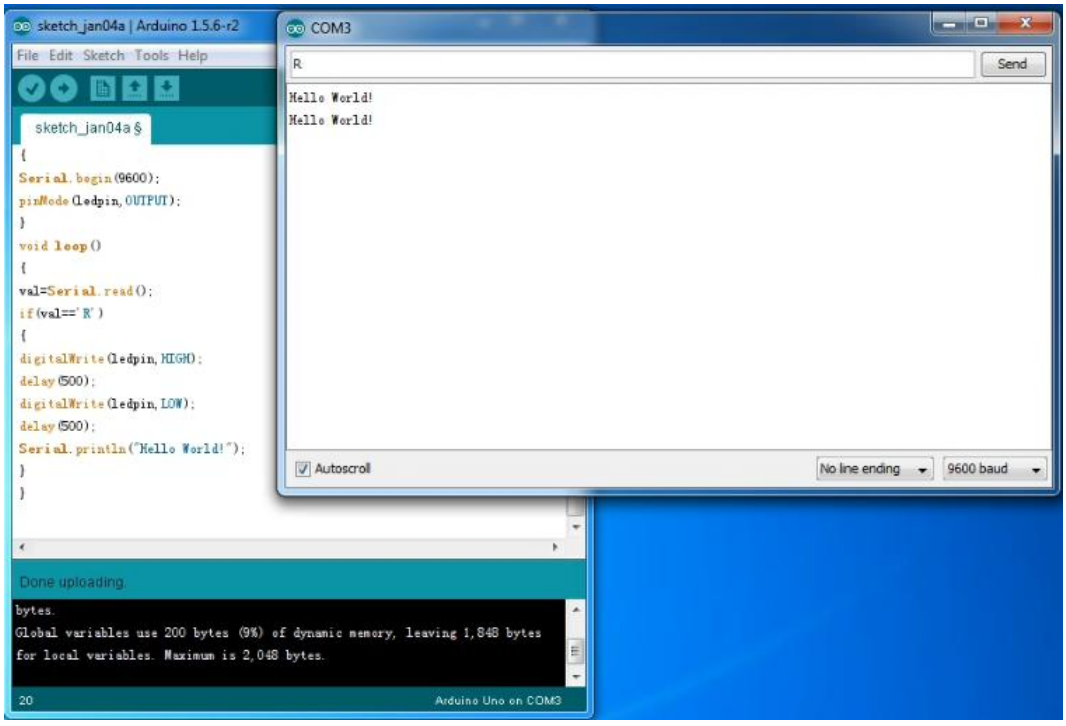
If setting well the board and port, you can see the display on the bottom right corner, which is the

same as the Device Manager display.



Then, click the verify  to compile the sketch, if no mistake, click upload  to upload the program code.

Done uploading, open the serial monitor, print an “R” and click “Send”, you can see the D13 indicator on the Keyes UNO R3 board blinks once, and the “Hello World!” is displayed on the serial monitor. Shown below.



Congrats. Your first programming is done well!

5. Project Details

Project 1: Blinking an LED

Description

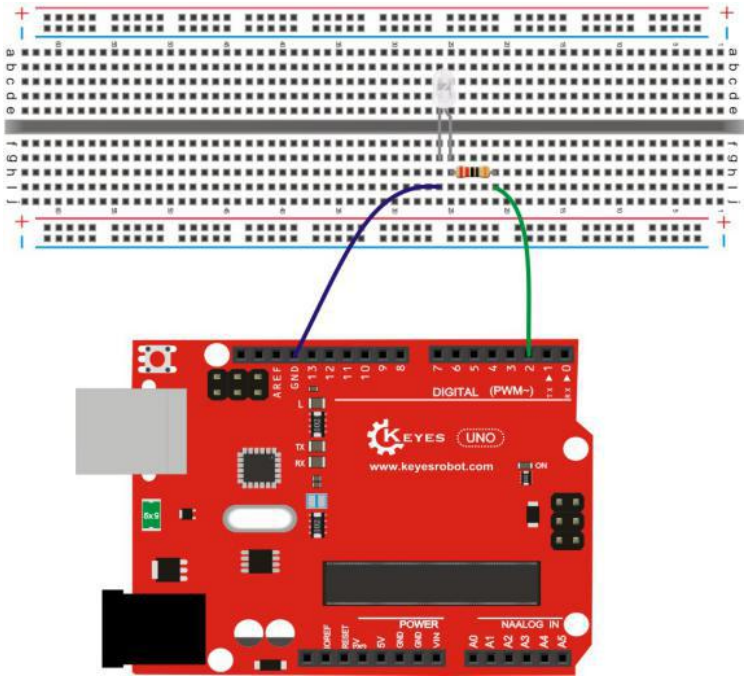
LED blinking is one of the more basic experiments. In displaying the "Hello World!", actually it makes use of the built-in LED. This project, we are going to use other I/O port and to connect an external LED to finish the LED blinking experiment.

Part List

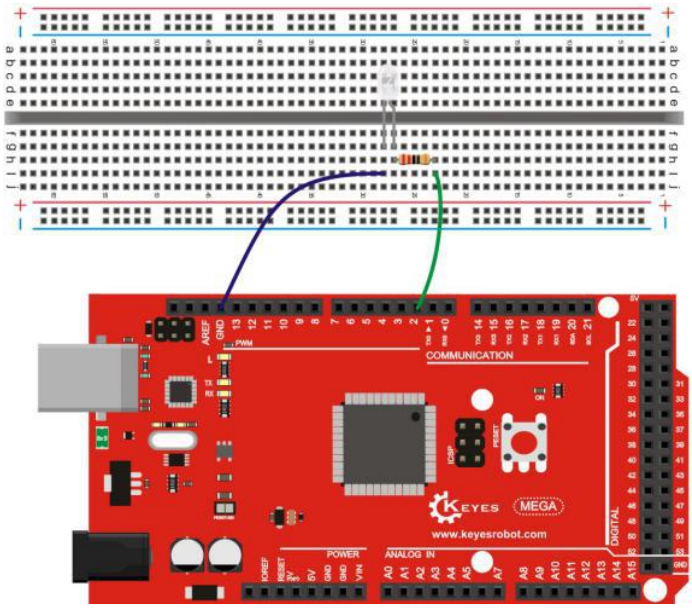
Development Board*1
USB Cable*1
LED*1
220Ω resistor*1
Breadboard*1
Jumper wire*2

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////////
int led = 2;                              //define the digital 2
void setup()
{
 pinMode(led, OUTPUT);      //set led as output
}
void loop()
{
 digitalWrite(led, HIGH);    // led on
    delay(2000); //delay 2S
 digitalWrite(led, LOW);     //led off
 delay(2000);//delay 2S
}
//////////////////////////////////////////////////////////////
```

Test Result

Done uploading the code, you can see the external LED continue to blink with an interval of about two seconds.

Project 2: Breathing an LED

Description

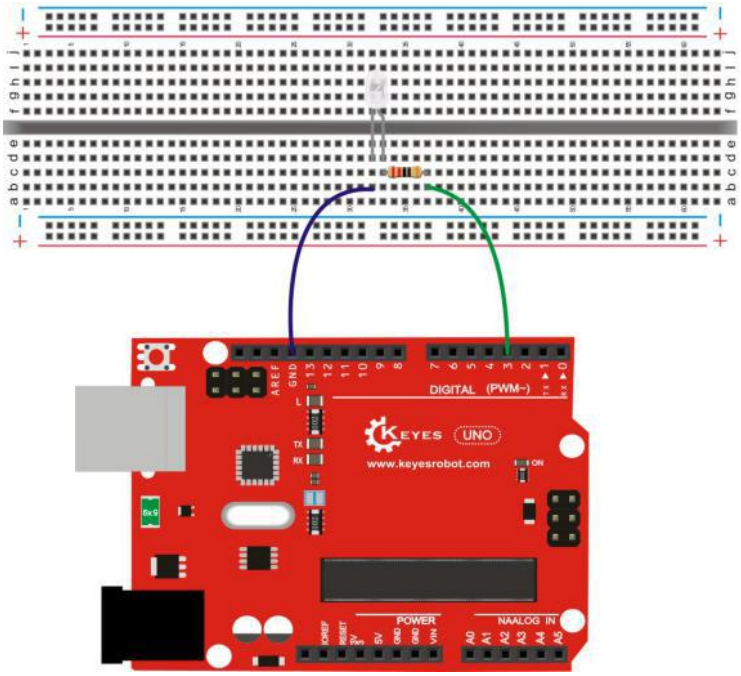
Last project, we have controlled the LED on and off. So how to control the brightness of an LED? In this project, we are going to connect the LED to PWM port, then to adjust the brightness of an LED via changing the PWM value, finally you can see the result that LED is gradually getting brighter and gradually dimming.

Part List

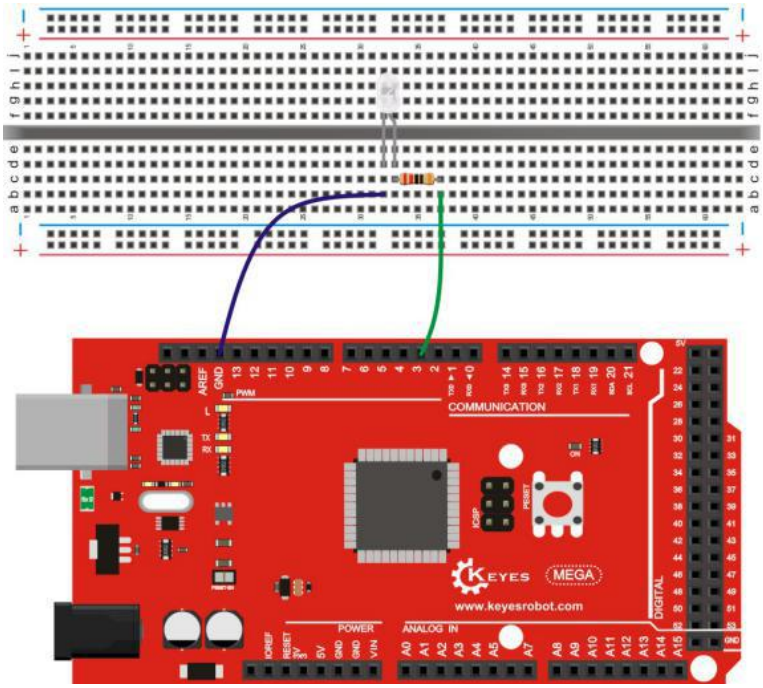
Development Board*1
USB Cable*1
LED*1
220Ω resistor*1
Breadboard*1
Jumper wire*2

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
int ledPin = 3; // define the digital 3
void setup()
{
pinMode(ledPin, OUTPUT);// set ledPin as output
}
void loop()
{
for (int a=0; a<=255;a++)// set the LED gradually brighten
{
analogWrite(ledPin,a); // led is on, to adjust the brightness, with the rang of 0-255, LED is the brightest in 255
delay(10); // delay 0.01S
}
for (int a=255; a>=0;a--) // set the LED gradually dimming
{
analogWrite(ledPin,a); // led is on, to adjust the brightness, with the range of 0-255, LED is the brightest in 255
delay(10); // delay 0.01S
}
delay(1000);// delay 1S
}
//////////////////////////////////////////////////////////
```

Test Result

Done uploading the code, you can see the external LED is gradually brightening, then gradually dimming, circularly. The effect seems like breathing LED.

Project 3: LED Chasing

Description

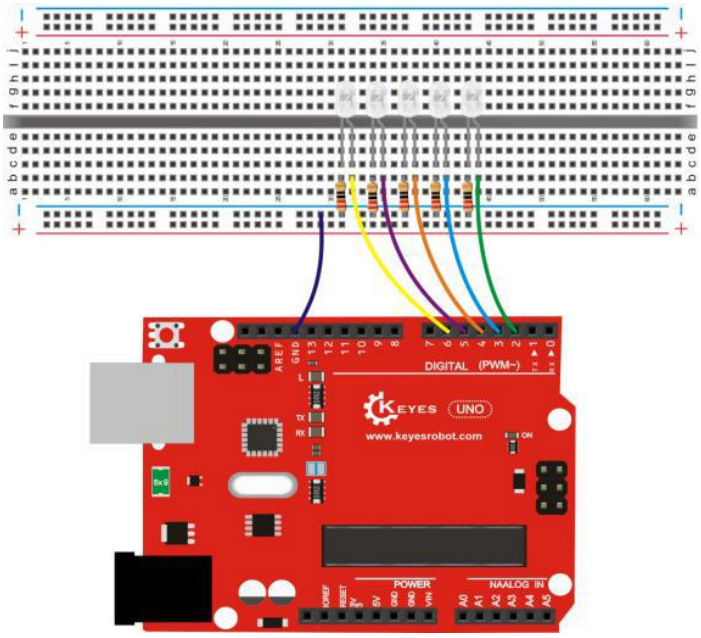
In life we can often see some billboards composed of LED lights with various colors. If LED lights constantly change, it can form various effects. This experiment we will use the LED lights to simulate the effect of advertising light, you can see the LEDs chasing one by one.

Part List

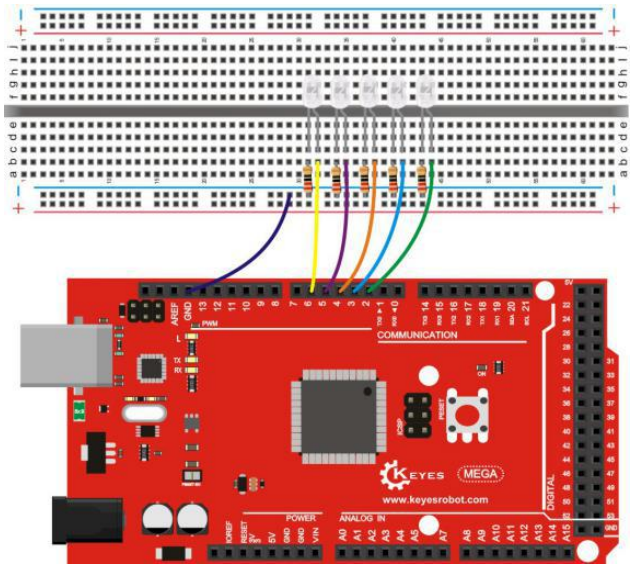
Development Board*1
USB Cable*1
LED*5
220Ω resistor*5
Breadboard*1
Jumper wire*6

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
int BASE = 2 ;    //    I/O port that the first LED connected
int NUM = 5;      //the number of LED
void setup()
{
    for (int i = BASE; i < BASE + NUM; i ++)
    {
      pinMode(i, OUTPUT);      /set the digital I/O 口 as output
    }
}

void loop()
{
    for (int i = BASE; i < BASE + NUM; i ++)
    {
      digitalWrite(i, HIGH);          //set the digital I/O output as “HIGH”, LED gradually becomes on
      delay(200);              //delay
    }
    for (int i = BASE; i < BASE + NUM; i ++)
    {
      digitalWrite(i, LOW);          //set the digital I/O output as “LOW”, LED gradually becomes off
      delay(200);              //delay
    }
}
//////////////////////////////////////////////////////////
```

Test Result

Done uploading the code, you can see the external LED connected to IO port is gradually on and off one by one, circularly.

Project 4: Button-Controlled LED

Description

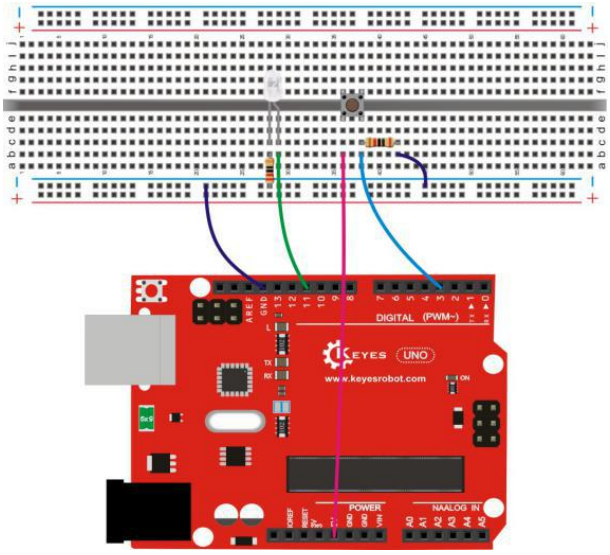
I/O port means INPUT and OUTPUT interface. Until now, we only apply the output function of I/O port in the LED experiments. In this project, we try to apply the input function of I/O port, namely reading the output value of external devices. Using only a button and an LED to finish the experiment combined input and output functions.

Part List

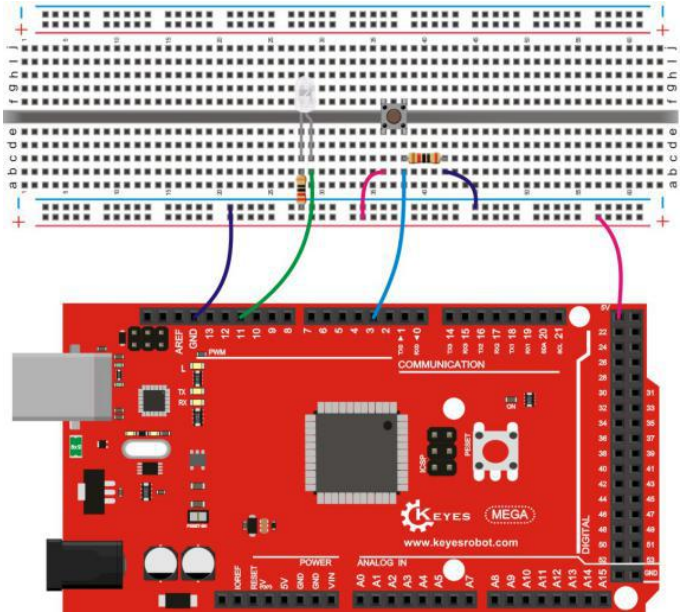
Development Board*1
USB Cable*1
LED*1
Tactile button*1
220Ω resistor*1
10KΩ resistor*1
Breadboard*1
Jumper wire* several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
int ledPin = 11;    //define the digital 11
int inputPin = 3; //define the digital 3
void setup()
{
pinMode(ledPin, OUTPUT);      //set ledPin as output
pinMode(inputPin, INPUT); //set inputPin as input
}
void loop()
{
int val = digitalRead(inputPin);
//set the digital variable val, read the value of digital 3 and assign it to val
if (val == LOW) //if val is low level, LED dims.
{
digitalWrite(ledPin, LOW); // LED dims
}
  else
{
digitalWrite(ledPin, HIGH); // LED brightens
}
}
//////////////////////////////////////////////////////////
```

Test Result

Done uploading the code and powered up, when the button is pressed down, LED is on; otherwise LED is off.

Project 5: Making a Responder

Description

You can make a further study based on the button-controlled LED experiment, replacing the LED with keyes RGB module.

Keyes RGB LED module is made from a full-color LED. It can achieve the mixed full-color effect through using the PWM voltage input of R, G, B pin to adjust the intensity of three primary colors (namely red, blue, green).

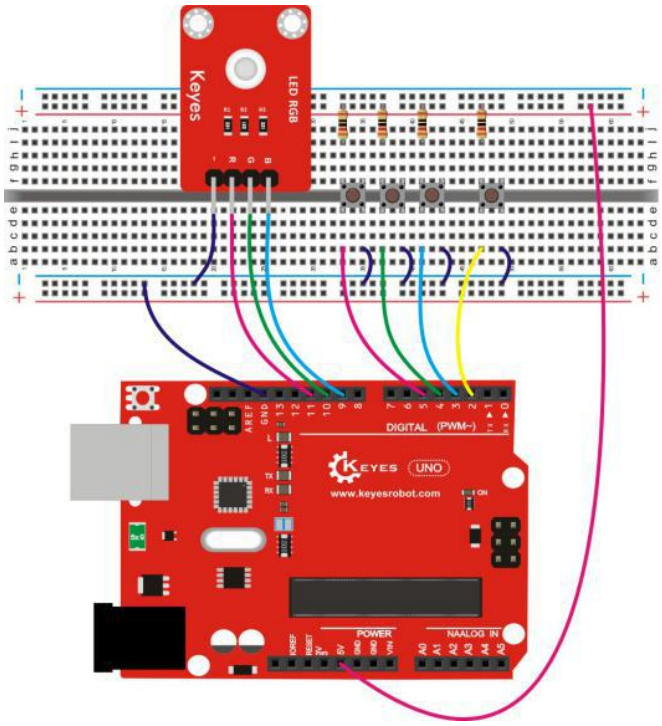
This project, we are going to use four buttons to control three PWM ports, so as to control the emitting color of RGB module. Based on that, you can simulate the responder.

Part List

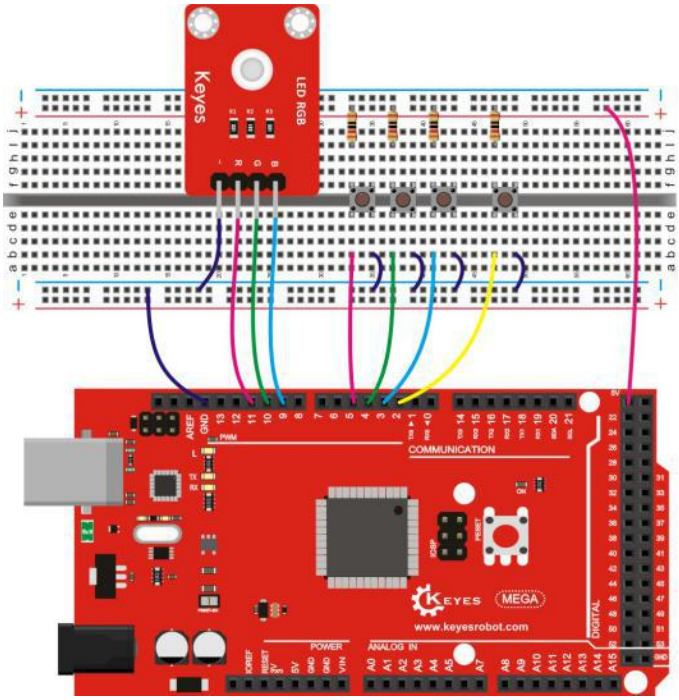
Development Board*1
USB Cable*1
keyes RGB module*1
Tactile button*4
10KΩ resistor*4
Breadboard*1
Jumper wire* several
Dupont line* several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
int redled=11;
int greenled=10;
int blueled=9;
int redpin=5;
int greenpin=4;
int bluepin=3;
int restpin=2;
int red;
int green;
int blue;
void setup()
{
pinMode(redled,OUTPUT);
pinMode(greenled,OUTPUT);
pinMode( blueled,OUTPUT);
pinMode(redpin,INPUT);
pinMode(greenpin,INPUT);
pinMode(bluepin,INPUT);
}
void loop()
{
red=digitalRead(redpin);
green=digitalRead(greenpin);
```

```
blue=digitalRead(bluepin);
if(red==LOW)RED_YES();
if(green==LOW)GREEN_YES();
if(blue==LOW)BLUE_YES();
}

void RED_YES()
{
   while(digitalRead(restpin)==1)
   {
      color(255, 0, 0);
    }
   clear_led();
}
void GREEN_YES()
{
   while(digitalRead(restpin)==1)
   {
      color(0, 255, 0);
   }
   clear_led();
}
void BLUE_YES()
{
   while(digitalRead(restpin)==1)
   {
        color(0, 0, 255);

   }
   clear_led();
}
void clear_led()
{
 color(0, 0, 0);
}
void color (unsigned char red, unsigned char green, unsigned char blue)    //color control
function
{
   analogWrite(redled, red);
   analogWrite(greenled,green);
   analogWrite(blueled, blue);
}
////////////////////////////////////////////////////////////
```

Test Result

Done uploading the code and powered up, the responder experiment is done well. You can judge who has responded first according to the emitting color of RGB LED. After that, just need to press the reset button to turn off the RGB LED, and turn into next round.

Project 6: Controlling LED Brightness

Description

In the project 2, we have done the effect of LED breathing through controlling the brightness of LED via PWM port.

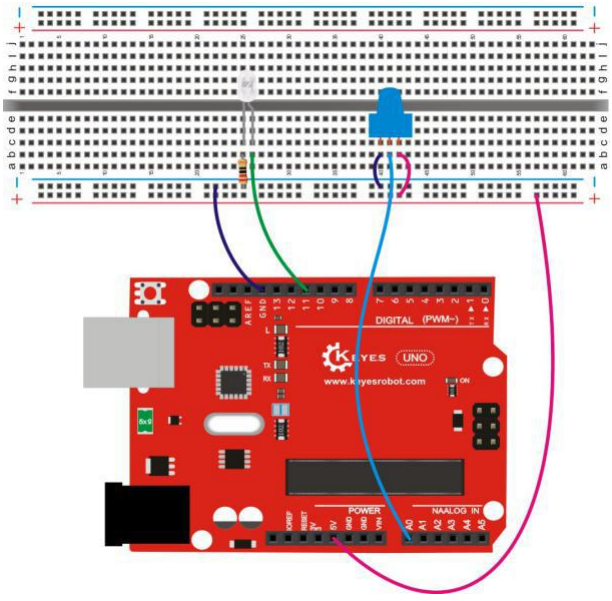
In this project, we are going to adjust the PWM value using a potentiometer, so as to control the brightness of LED.

Part List

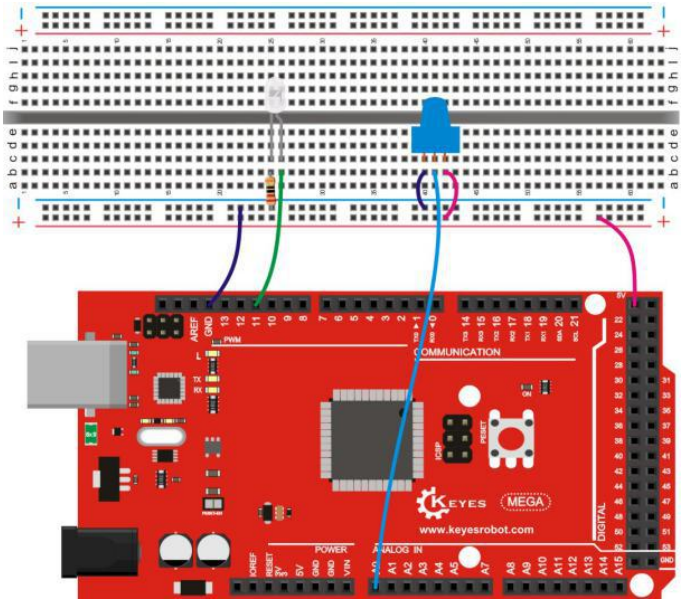
Development Board*1
USB Cable*1
LED *1
220Ω resistor*1
Potentiometer*1
Breadboard*1
Breadboard jumper wire* 6

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
int ledpin=11;//define the digital 11（PWM output）
void setup()
{
pinMode(ledpin,OUTPUT);//define the digital 11 as output
Serial.begin(9600);//set the baud rate as 9600
}
void loop()
{
int val=analogRead(0);//read the value of analog port A0
val = map(val, 0, 1023, 0, 255);//Mapping from 0-1023 to 0-255
Serial.println(val);//display val variable
analogWrite(ledpin,val);// turn on LED and set the brightness
delay(100);//delay 0.1 seconds
}
//////////////////////////////////////////////////////////
```

Test Result

Done uploading the code, you can control the brightness of LED via rotating the knob of potentiometer. Then open the serial monitor, and set the baud rate as 9600, you can see the PWM value of LED brightness.

Project 7: Photosensitive Light

Description

Mastering the basic digital input and output, analog input and PWM generation, you can start to learn some sensor applications.

Next, we will do a rather simple experiment of photocell. Since the resistance value of photo-resistor component can vary from the intensity of light, it naturally need the analog port to read the analog value.

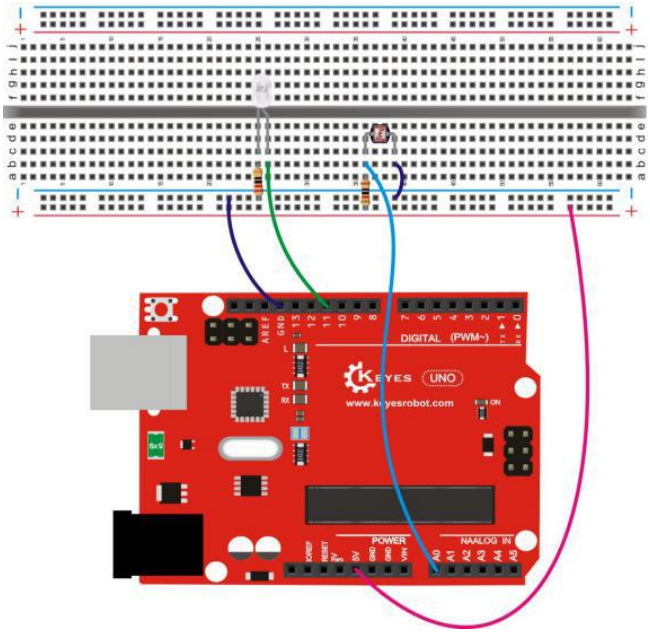
You can learn from the last project 6, just using photo-resistor instead of potentiometer to achieve that LED brightness will be changed when the light intensity is different.

Part List

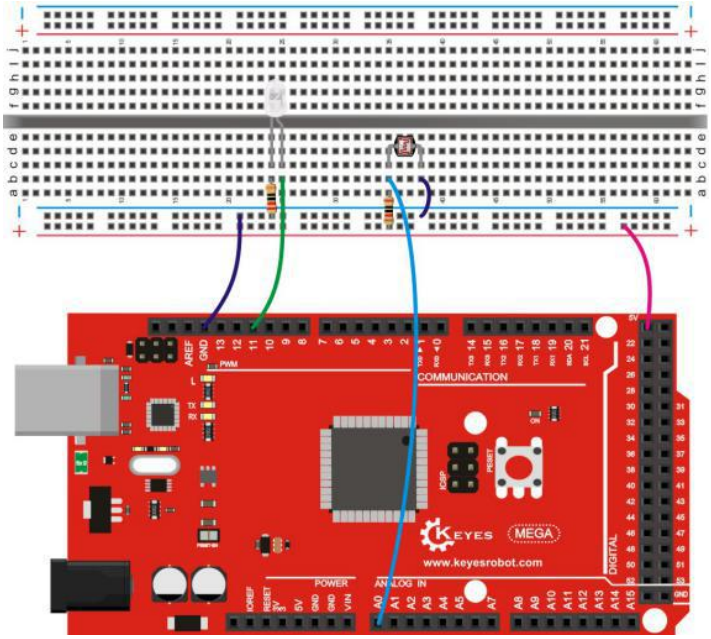
Development Board*1
USB Cable*1
LED *1
220Ω resistor*1
10KΩ resistor*1
Photocell*1
Breadboard*1
Jumper wire* 5

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
int ledpin=11;//define the digital 11（PWM output）
void setup()
{
pinMode(ledpin,OUTPUT);//define the digital 11 as output
Serial.begin(9600);//set the baud rate as 9600
}
void loop()
{
int val=analogRead(0);//read the value of analog port A0
Serial.println(val);//display val variable
val = map(val, 0, 1023, 0, 255);//mapping from 0-1023 to 0-255
analogWrite(ledpin,255-val);// turn on LED and set its brightness
delay(10);//delay 0.01 seconds
}
//////////////////////////////////////////////////////////
```

Test Result

Uploading the code, the brightest light the photocell senses, the darkest the LED. On the contrary, the darkest the photocell senses, the brightest the LED.

Open the serial monitor, set the baud rate as 9600, you can see the analog value representing the external light intensity the photocell senses.

Project 8: Active Buzzer Beeping

Description

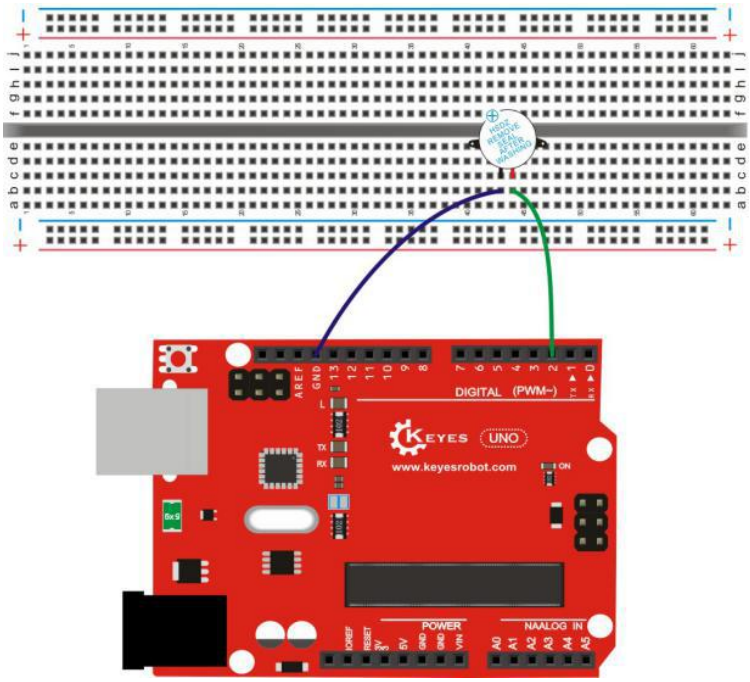
In this project, we will use an active buzzer. The active buzzer inside has a simple oscillator circuit which can convert constant direct current into a certain frequency pulse signal. When an active buzzer receives a high level, it will produce an audible beep.

Part List

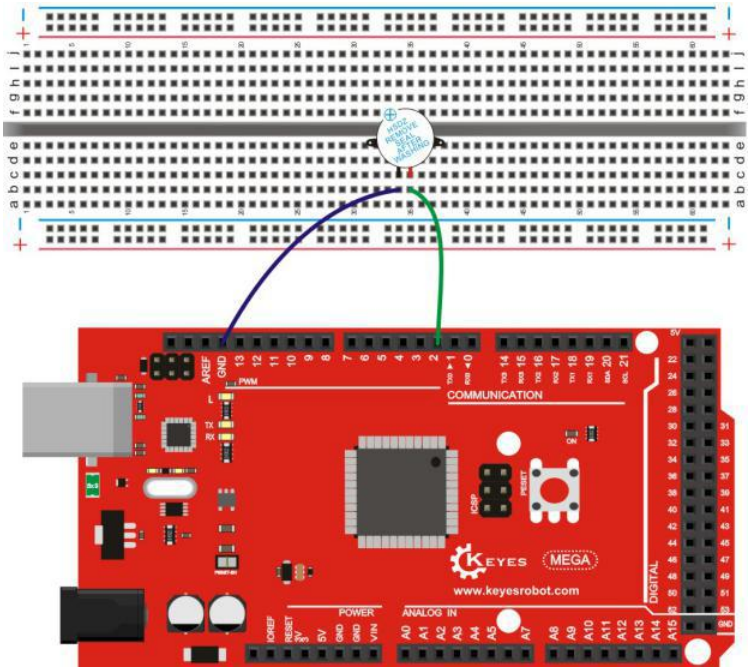
Development Board*1
USB Cable*1
Active buzzer *1
Breadboard*1
Jumper wire*2

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

Copy and paste the below code into your Arduino IDE, then upload!

```
//////////////////////////////////////////////////////////
int buzzer = 2;                          //define the digital 2
void setup()
{
  pinMode(buzzer, OUTPUT);      //set buzzer as output
}
void loop()
{
  digitalWrite(buzzer, HIGH);    //turn on buzzer
   delay(1000); //delay1S
  digitalWrite(buzzer, LOW);     //turn off buzzer
  delay(1000);//delay1S
}
//////////////////////////////////////////////////////////
```

Test Result

Uploading the code, you can hear the buzzer beep for 1 seconds, then stop and beep for 1 seconds, circularly.

Project 9: Passive Buzzer Singing

Description

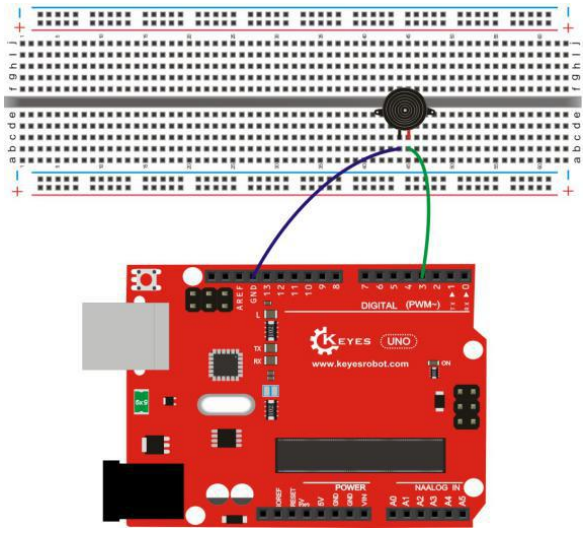
You have learned about the active buzzer last project. So this project we will use a passive buzzer. Since the passive buzzer has no oscillator inside, DC signal can't make it beep, so need to use square-wave to drive it.

Part List

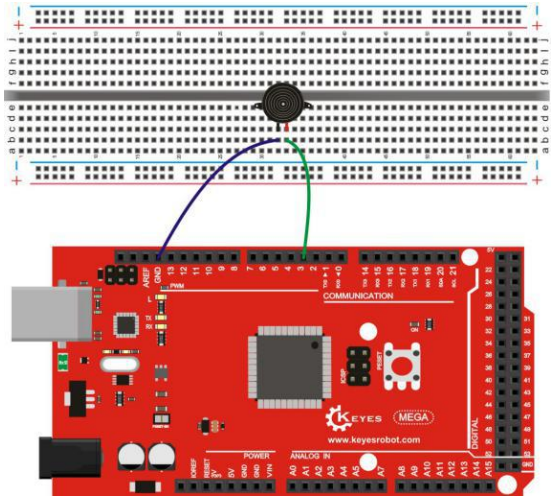
Development Board*1
USB Cable*1
Passive buzzer *1
Breadboard*1
Jumper wire*2

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

Copy and paste the below code into your Arduino IDE, then upload!

Code 1:

```
//////////////////////////////////////////////////////////////
int buzzer=3;                    //define the digital 3
void setup()
{
pinMode(buzzer,OUTPUT);//set buzzer as output
}
void loop()
{
unsigned char i,j;//define the variable i，j
while(1)
{
for(i=0;i<80;i++)// output the sound of a frequency
{
digitalWrite(buzzer,HIGH);
delay(1);//delay 1ms
digitalWrite(buzzer,LOW);
delay(1);//delay 1ms
}
for(i=0;i<100;i++)//    output the sound of another frequency
{
digitalWrite(buzzer,HIGH);
delay(2);// delay 2ms
digitalWrite(buzzer,LOW);
delay(2);//delay 2ms
}
}
}
//////////////////////////////////////////////////////////////
```

Code 2:

```
//////////////////////////////////////////////////////////////
#define D0 -1
#define D1 262
#define D2 293
#define D3 329
#define D4 349
#define D5 392
#define D6 440
#define D7 494
```

```
#define M1 523
#define M2 586
#define M3 658
#define M4 697
#define M5 783
#define M6 879
#define M7 987
#define H1 1045
#define H2 1171
#define H3 1316
#define H4 1393
#define H5 1563
#define H6 1755
#define H7 1971
//list out all the frequency of D tone
#define WHOLE 1
#define HALF 0.5
#define QUARTER 0.25
#define EIGHTH 0.25
#define SIXTEENTH 0.625
///list out all the beat
int tune[]=             //list out the frequency according to numbered musical notation
{
  M3,M3,M4,M5,
  M5,M4,M3,M2,
  M1,M1,M2,M3,
  M3,M2,M2,
  M3,M3,M4,M5,
  M5,M4,M3,M2,
  M1,M1,M2,M3,
  M2,M1,M1,
  M2,M2,M3,M1,
  M2,M3,M4,M3,M1,
  M2,M3,M4,M3,M2,
  M1,M2,D5,D0,
  M3,M3,M4,M5,
  M5,M4,M3,M4,M2,
  M1,M1,M2,M3,
  M2,M1,M1
};
float durt[]=           //list out the beats according to numbered musical notation
{
  1,1,1,1,
  1,1,1,1,
```

```
  1,1,1,1,
  1+0.5,0.5,1+1,
  1,1,1,1,
  1,1,1,1,
  1,1,1,1,
  1+0.5,0.5,1+1,
  1,1,1,1,
  1,0.5,0.5,1,1,
  1,0.5,0.5,1,1,
  1,1,1,1,
  1,1,1,1,
  1,1,1,0.5,0.5,
  1,1,1,1,
  1+0.5,0.5,1+1,
};
int length;
int tonepin=3;     //use pin3
void setup()
{
  pinMode(tonepin,OUTPUT);
  length=sizeof(tune)/sizeof(tune[0]);     //calculate the length
}
void loop()
{
  for(int x=0;x<length;x++)
  {
    tone(tonepin,tune[x]);
    delay(500*durt[x]);      //here you can use the beat to adjust the delay time, and in this
  music 500 might be rather suitable.
    noTone(tonepin);
  }
  delay(2000);
}
////////////////////////////////////////////////////////////
```

Test Result

We have offered you two codes above. If uploading the code1, you can hear the buzzer make two different sounds, alternately cycle. While uploading the code2, the buzzer will play the song of Ode To Joy. Sounds fun! Try to do more interesting projects by yourself.

Project 10: Fire Alarm

Description

Flame sensor is specially used on robots to find the fire source. This sensor has high sensitivity to flame. Flame sensor is made based on the principle that infrared ray is highly sensitive to flame. It has a special infrared receiving triode designed to detect fire, and then convert the flame brightness to fluctuating level signal.

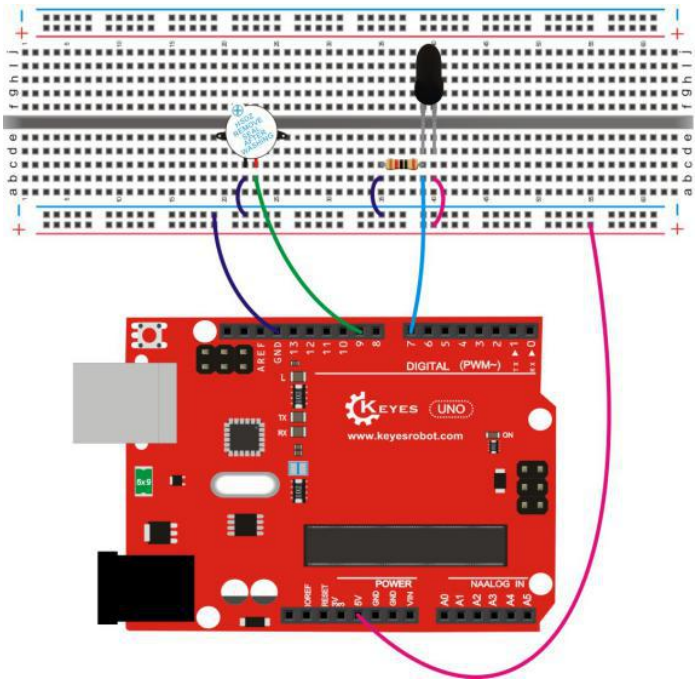
In this project, just need to convert the brightness of LED into fluctuating level signal, and then input it into UNO board, so as to control the buzzer beeping.

Part List

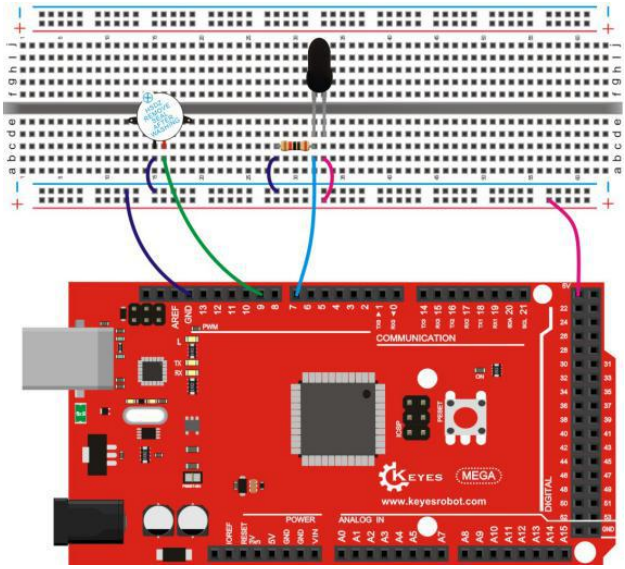
Development Board*1
USB Cable*1
Active buzzer *1
Flame sensor*1
10KΩ resistor*1
Breadboard*1
Jumper wire*7

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
int flame=7;//define the flame interface as the digital 7
int Beep=9;//define the buzzer interface as the digital 9

void setup()
{
 pinMode(Beep,OUTPUT);//set Beep as output interface
 pinMode(flame,INPUT);//set flame as output interface

 }
void loop()
{
   int val=digitalRead(flame);//read the flame sensor
   if(val==HIGH)//as digital 7 is high level, buzzer beeps.
   {
    digitalWrite(Beep,HIGH);
    }else
    {
       digitalWrite(Beep,LOW);
     }
    delay(500);
}
//////////////////////////////////////////////////////////
```

Test Result

Done uploading the code, it can simulate the fire alarm. When there is no fire, everything is normal; while flame sensor detects the fire, the buzzer will make an alarm.

Project 11: A Cup with Temperature Indicator

Description

LM35 is a common and easy to use temperature sensor component. You can connect the sensor to the development board, then through calculating to convert the analog value it reads into actual temperature.

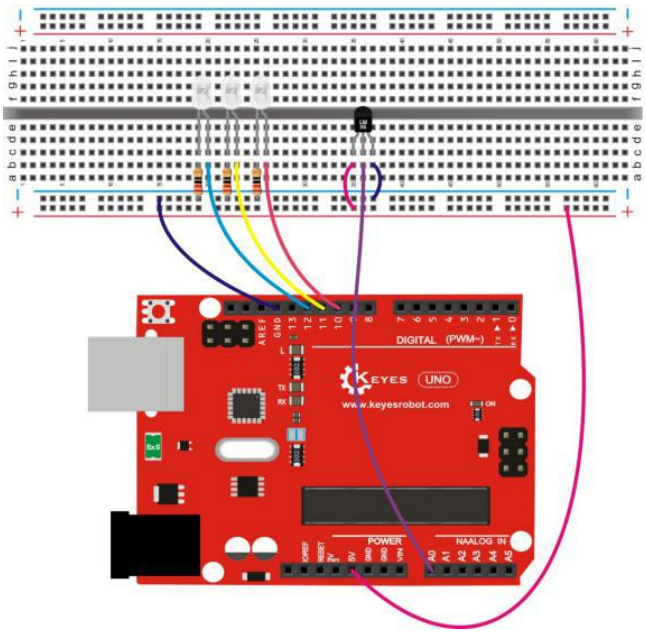
In this project, we will connect three external LEDs, then in the code you can set that in the different temperature range, the LED will make different color change. Based on that, you can try to make a temperature-indicated cup. This way, you can know the hot or cool condition of your water.

Part List

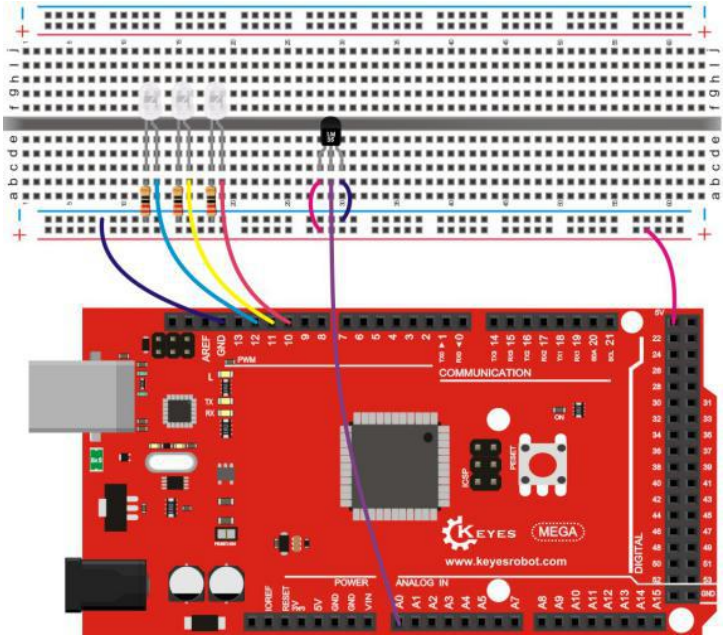
Development Board*1
USB Cable*1
LM35DZ*1
LED*3
200Ω resistor*3
Breadboard*1
Jumper wire*8

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
void setup() {
   Serial.begin(9600);
   pinMode(12, OUTPUT);
   pinMode(11, OUTPUT);
   pinMode(10, OUTPUT);
}
void loop() {
  int vol = analogRead(A0) * (5.0 / 1023.0*100);
  Serial.print("Tep:");
  Serial.print(vol);
  Serial.println("C");
if (vol<28)
{
   digitalWrite(12, HIGH);
   digitalWrite(11, LOW);
   digitalWrite(10, LOW);
}
else if (vol>=28 && vol<=30)
  {
   digitalWrite(12, LOW);
   digitalWrite(11, HIGH);
   digitalWrite(10, LOW);
}
else if (vol>30)
```

```
{
   digitalWrite(12, LOW);
   digitalWrite(11, LOW);
   digitalWrite(10, HIGH);
}
  }
////////////////////////////////////////////////////////////
```

Test Result

Done uploading the code, open the serial monitor, set the baud rate as 9600, you can see the current temperature.

When the temperature is higher than 30°, red LED on. When the temperature is higher or equal to 28° and less than or equal to 30 degrees Celsius, red LED off and yellow LED on. While the temperature is lower than 28 degrees Celsius, yellow LED off and blue LED on.

Project 12: Magical Light Cup

Description

The working principle of the tilt switch is that when one end of the switch is tilted below the horizontal position, the switch is turned on; when the other end is tilted below the horizontal position, the switch is turned off.

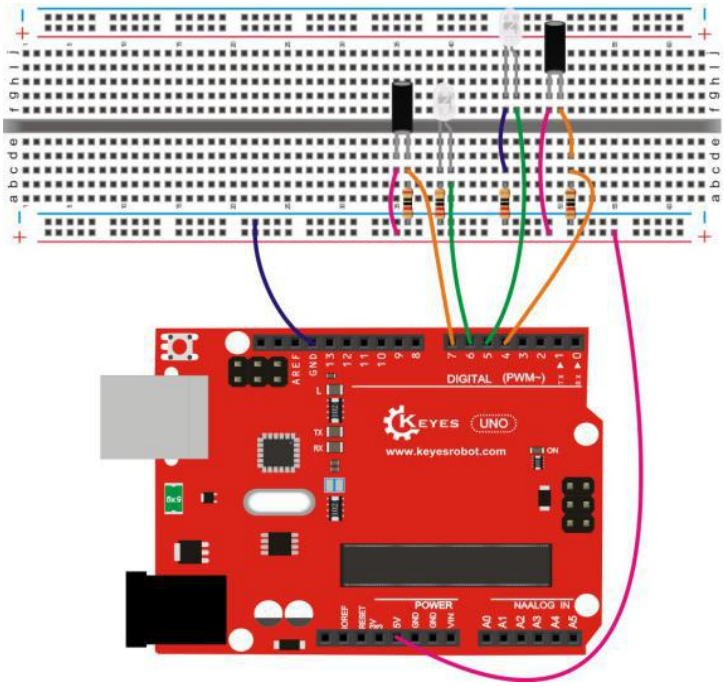
The principle of this experiment is to use PWM light regulation theory to change two LEDs brightness. The tilt switch provides digital signal to trigger PWM regulation, through the program design, you can see the test effect like two cups pouring light to one another.

Part List

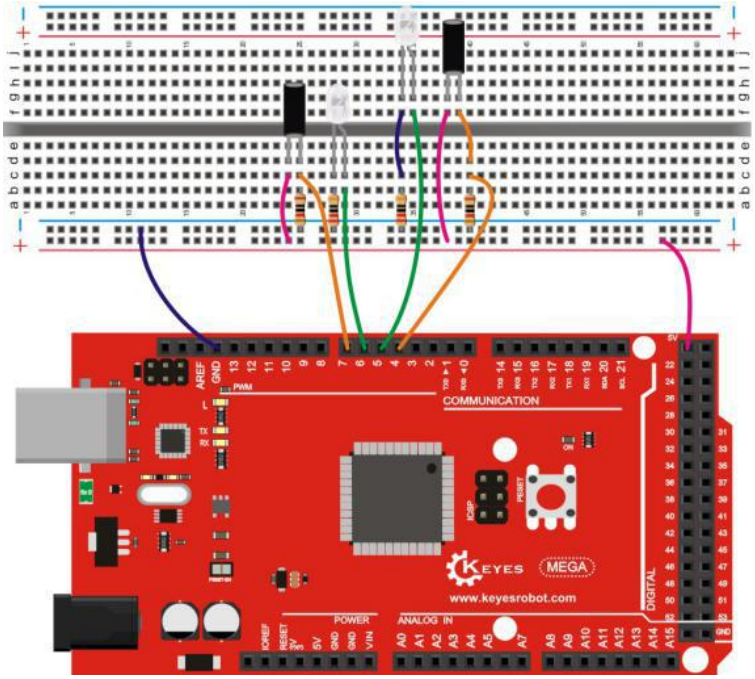
Development Board*1
USB Cable*1
LED*2
Ball tilt switch*2
200Ω resistor*2
10KΩ resistor*2
Breadboard*1
Jumper wire*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
int LedPinA = 5; //define the digital 5
int LedPinB = 6; //define the digital 6
int ButtonPinA = 7;//define the digital 7
int ButtonPinB = 4;//define the digital 4
int buttonStateA = 0;
```

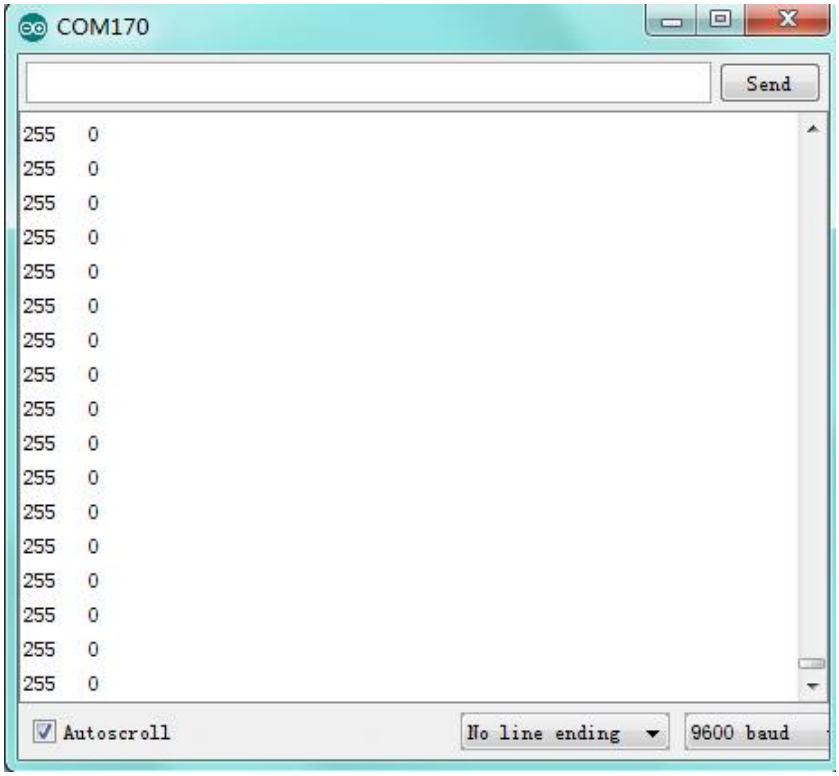
```
int buttonStateB = 0;
int brightnessA = 0;
int brightnessB= 255;
void setup()
{
Serial.begin(9600);//set the baud rate
pinMode(LedPinA, OUTPUT);//set the digital 5 as output
pinMode(LedPinB, OUTPUT);//set the digital 6 as output
pinMode(ButtonPinA, INPUT);//set the digital 7 as input
pinMode(ButtonPinB, INPUT);//set the digital 4 as input
}
void loop()
{
buttonStateA = digitalRead(ButtonPinA);//read the value of digital 7 and assign it to buttonStateA
if (buttonStateA == HIGH && brightnessA != 255)
//if buttonStateA is high level and brightnessA is not 255
{
brightnessA ++;//brightnessA plus 1
delay(10);//delay 0.01S
}
if (buttonStateA == LOW && brightnessA != 0)
//if buttonStateA is low level and brightnessA is not 0
{
brightnessA --;//brightnessA subtracts 1
delay(10);//delay 0.01S
}
analogWrite(LedPinB, brightnessA);//assign the brightnessA to PWM port 6
Serial.print(brightnessA);//display brightnessA value
Serial.print("      ");
buttonStateB = digitalRead(ButtonPinB);//read the value of digital 4 and assign it to buttonStateB
if (buttonStateB == HIGH && brightnessB != 0)
//if buttonStateB is high level and brightnessA is not 0
{
brightnessB --;//brightnessB subtracts 1
delay(10);//delay 0.01S
}
if (buttonStateB == LOW && brightnessB != 255)
//if buttonStateB is low level and brightnessA is not 255
{
brightnessB++;//brightnessB plus 1
delay(10);//delay 0.01S
}
```

```
analogWrite(LedPinA, brightnessB); //assign brightnessB to PWM port 5
Serial.println(brightnessB);//display the value of brightnessB, and line wrap
delay(5);
}
//////////////////////////////////////////////////////////////
```

Test Result

Wire it up as the above diagram, then upload well the code, tilt the two switches to the same side, one LED gradually dims, and another LED gradually brightens, finally you can see one LED totally goes out, another one is the brightest.

Open the serial monitor, set the baud rate as 9600, you can see the corresponding data changes shown below.



Project 13: IR Remote Control Decoding

Description

The general infrared remote control system consists of two major parts, transmitter and receiver. In this experiment, the transmitter is the remote control, and the receiver is the infrared receiver VS1838B.

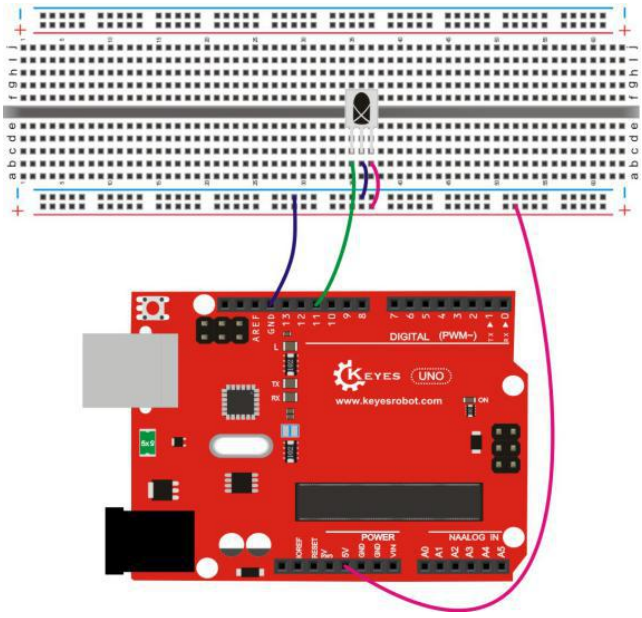
Infrared receiver VS1838B is a device that integrates reception, amplification and demodulation. Its internal IC has been demodulated and the output is a digital signal.

Part List

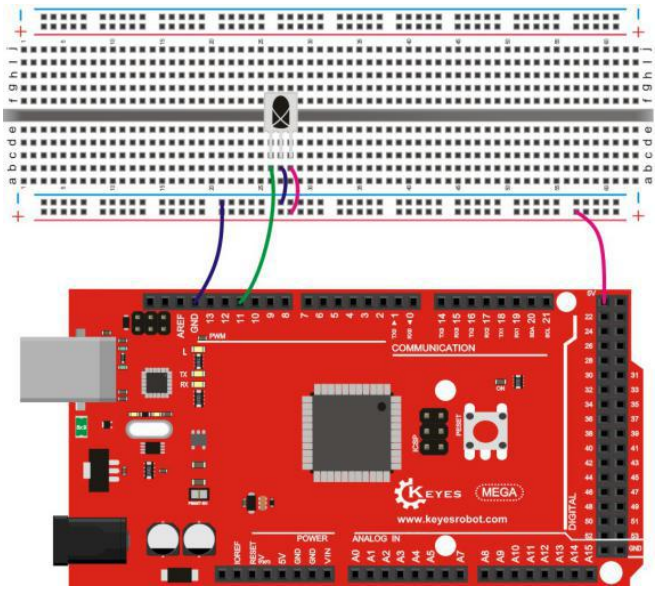
Development Board*1
USB Cable*1
IR Remote Control*1
IR Receiver VS1838B *1
Breadboard*1
Jumper wire*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3

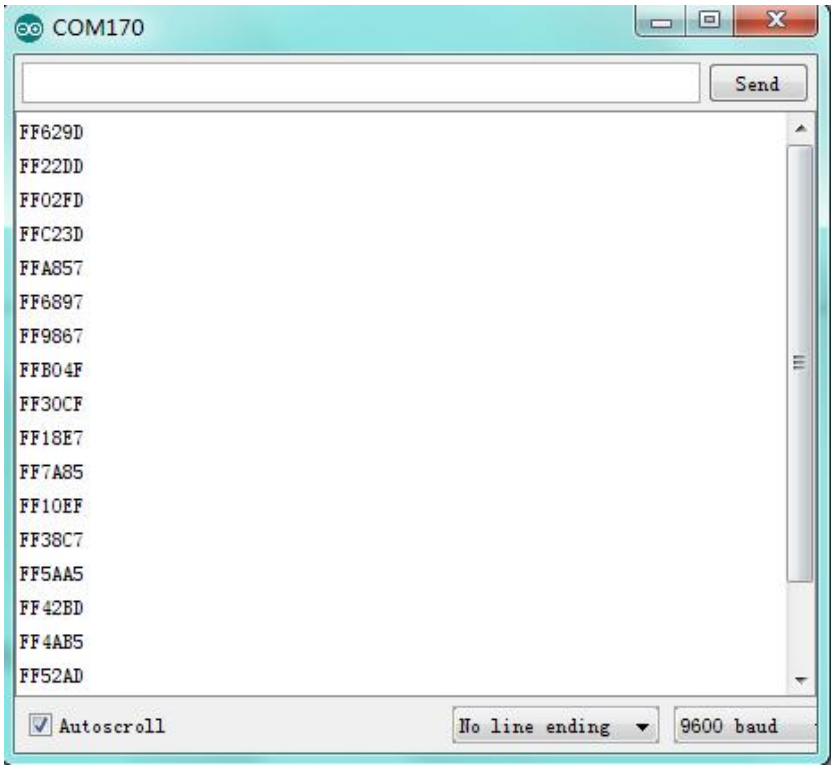


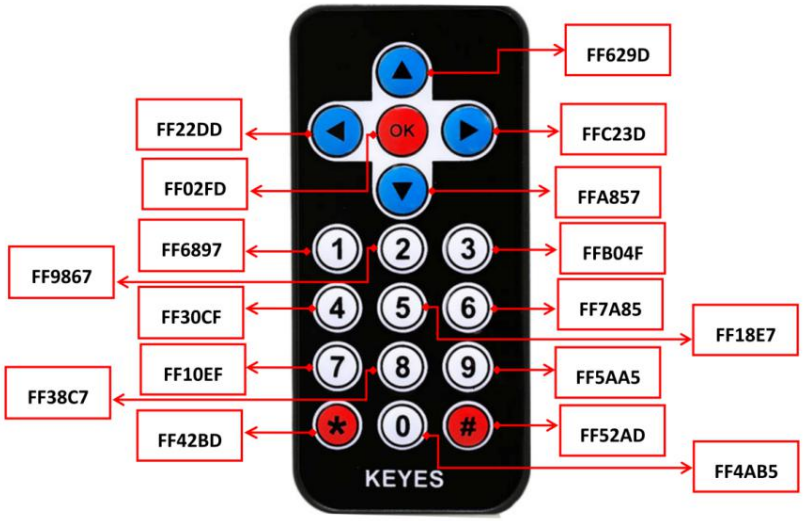
Sample Code

```
//////////////////////////////////////////////////////////////
#include <IRremote.h>
int RECV_PIN = 11; //define the digital 11
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
Serial.begin(9600);//set the baud rate
irrecv.enableIRIn(); // enable IR receiving
}
void loop() {
if (irrecv.decode(&results))
{
Serial.println(results.value, HEX);//display the data
irrecv.resume(); // receive the next data
}
}
//////////////////////////////////////////////////////////////
```

Test Result

Powered up and upload the code, send the signal to IR receiver sensor using IR remote control. Then open the serial monitor, you can see the decoding of corresponding button. Shown below.


COM170
Send
FF629D
FF22DD
FF02FD
FFC23D
FFA857
FF6897
FF9867
FFB04F
FF30CF
FF18E7
FF7A85
FF10EF
FF38C7
FF5AA5
FF42BD
FF52AD
FF4AB5
Autoscroll
No line ending
9600 baud


FF629D
FF22DD
OK
FFC23D
FF02FD
FFA857
FF6897
1
2
3
FFB04F
FF9867
FF30CF
4
5
6
FF7A85
FF18E7
FF10EF
7
8
9
FF5AA5
FF38C7
FF42BD
*
0
#
FF52AD
FF4AB5
KEYES

Project 14: 1-digit LED Display

Description

LED segment display is a semiconductor display component, and its basic unit is a light emitting diode. It can be divided into seven-segment display and eight-segment display. Compared with 7-seg display, 8-segment display just has one more light-emitting diode unit (more than a decimal point display).

If you want to display the number, just need to light up the corresponding segment.

Part List

Development Board*1

USB Cable*1

1-digit LED Display*1

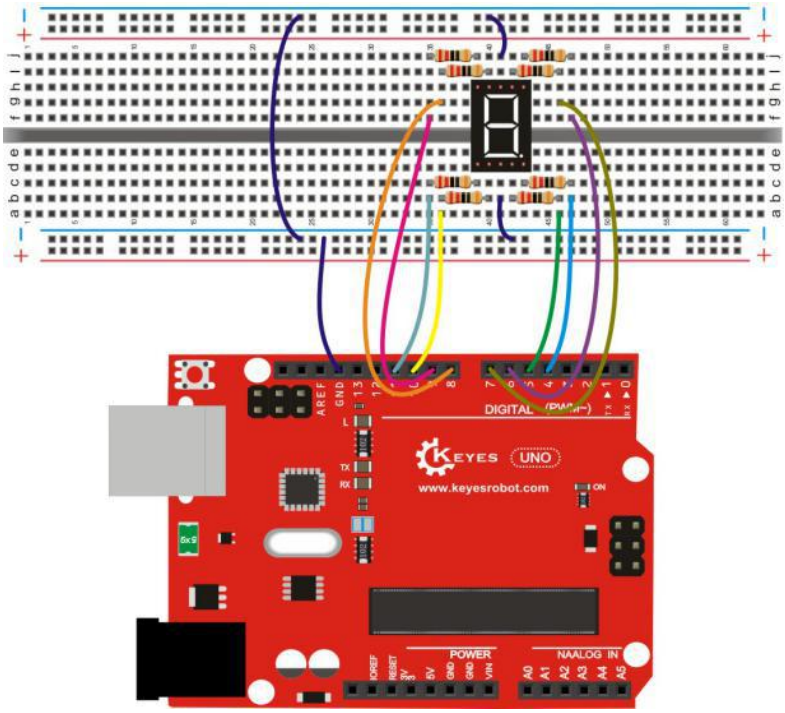
220Ω resistor*8

Breadboard*1

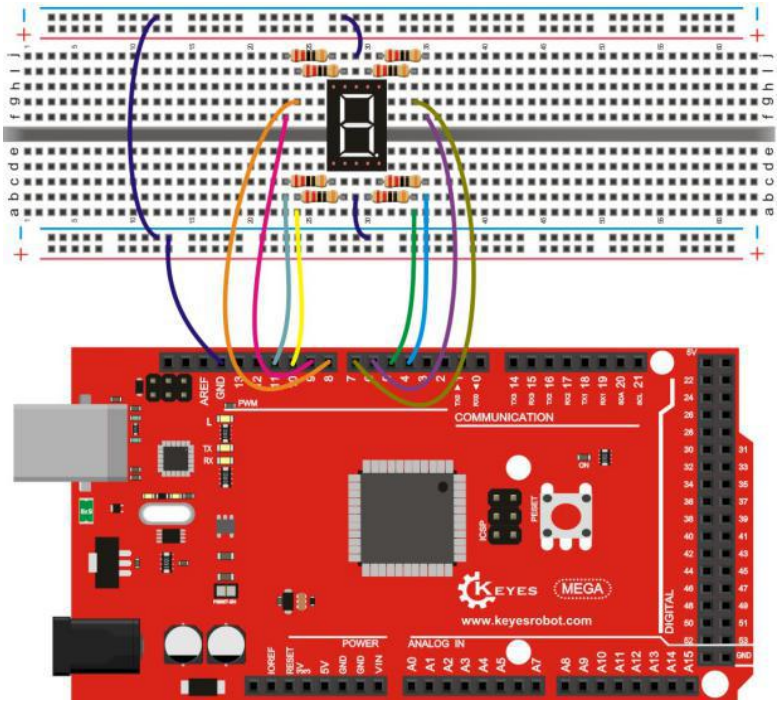
Jumper wire*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////////
//set the digital IO pin of each segment
int a=7;//digital 7 is connected to a segment of LED display
int b=6;// digital 6 is connected to b segment of LED display
int c=5;// digital 5 is connected to c segment of LED display
int d=10;// digital 11 is connected to d segment of LED display
int e=11;// digital 10 is connected to e segment of LED display
int f=8;// digital 8 is connected to f segment of LED display
int g=9;// digital 9 is connected to g segment of LED display
int dp=4;// digital 4 is connected to dp segment of LED display
void digital_1(void) //display number 1
{
unsigned char j;
digitalWrite(c,HIGH);//set the digital 5 as HIGH level, light up c segment
digitalWrite(b,HIGH);//light up b segment
for(j=7;j<=11;j++)//go out other segments
digitalWrite(j,LOW);
digitalWrite(dp,LOW);//go out the decimal point DP segment
}
void digital_2(void) //display the 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 the number 3
{
unsigned char j;
digitalWrite(g,HIGH);
digitalWrite(d,HIGH);
for(j=5;j<=7;j++)
digitalWrite(j,HIGH);
digitalWrite(dp,LOW);
digitalWrite(f,LOW);
digitalWrite(e,LOW);
}
void digital_4(void) //display the 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 the number 5
{
unsigned char j;
for(j=7;j<=9;j++)
digitalWrite(j,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(dp,LOW);
digitalWrite(b,LOW);
digitalWrite(e,LOW);
}
void digital_6(void) //display the 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 the 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 the number 8
{
unsigned char j;
for(j=5;j<=11;j++)
digitalWrite(j,HIGH);
digitalWrite(dp,LOW);
}
void setup()
{
int i;//define the variable
for(i=4;i<=11;i++)
pinMode(i,OUTPUT);//set 4～11pins as output mode
}
void loop()
{
while(1)
{
digital_1();//display the number1
delay(2000);//delay 2s
digital_2();//display the number2
delay(1000); //delay 1s
digital_3();//display the number3
delay(1000); //delay 1s
digital_4();//display the number4
delay(1000); //delay 1s
digital_5();//display the number5
delay(1000); //delay 1s
digital_6();//display the number6
delay(1000); //delay 1s
```

```
digital_7();//display the number7
delay(1000); //delay 1s
digital_8();//display the number8
delay(1000); //delay 1s
}
}
//////////////////////////////////////////////////////////
```

Test Result

Upload well the code, the LED segment display will show the number from 1 to 8.

Project 15: 74HC595 Controlling LED Display

Description

In the previous project, we had used the development board to control the 1-digit LED display, occupying more digital interfaces. So this project, we add a 74HC595 chip to control the 1-digit LED display, only using three digital interface to control 8 LEDs.

You can refer to the specific setting method as the below table shown.

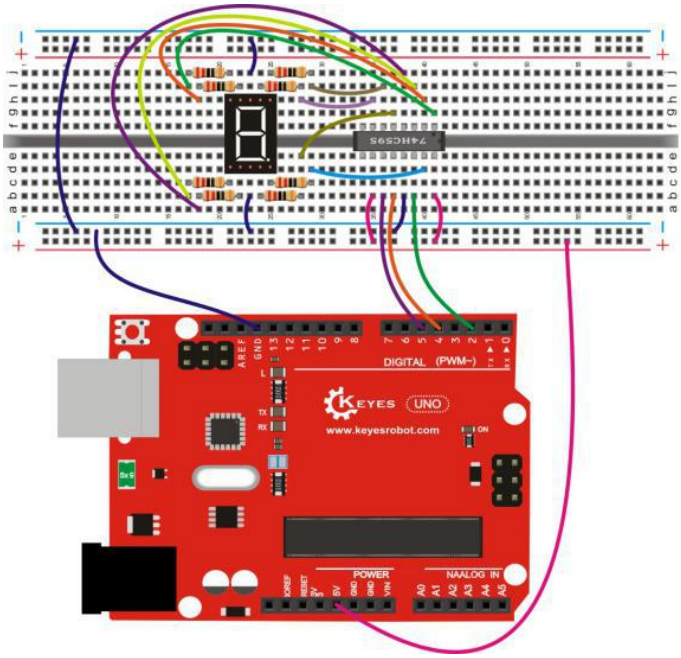
	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0	
	a	b	c	d	e	f	g	dp	
0	1	1	1	1	1	1	0	0	252
1	0	1	1	0	0	0	0	0	96
2	1	1	0	1	1	0	1	0	218
3	1	1	1	1	0	0	1	0	242
4	0	1	1	0	0	1	1	0	102
5	1	0	1	1	0	1	1	0	182
6	1	0	1	1	1	1	1	0	190
7	1	1	1	0	0	0	0	0	224
8	1	1	1	1	1	1	1	0	254
9	1	1	1	1	0	1	1	0	246

Part List

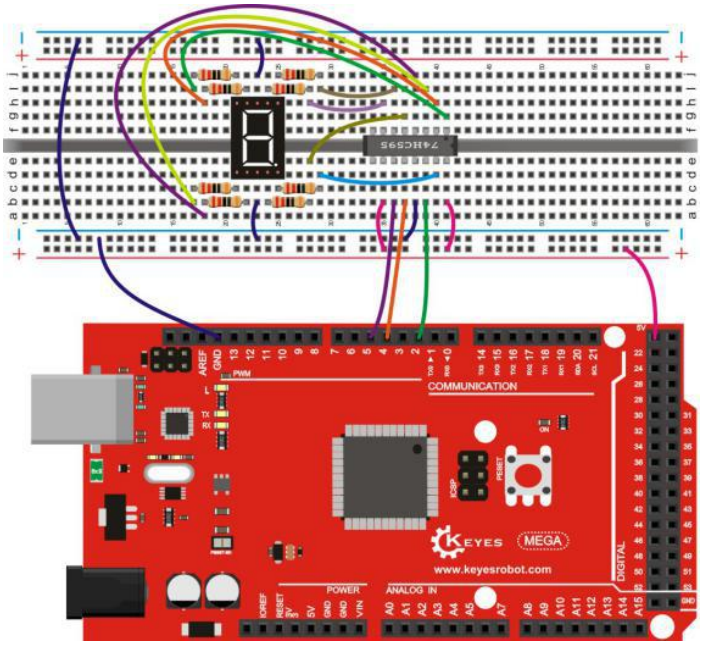
Development Board*1
USB Cable*1
74HC595*1
1-digit LED Display*1
220Ω resistor*8
Breadboard*1
Breadboard jumper wire*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
int latchPin = 4;
int clockPin = 5;
int dataPin = 2; //here define that three pins
void setup ()
{
  pinMode(latchPin,OUTPUT);
  pinMode(clockPin,OUTPUT);
  pinMode(dataPin,OUTPUT); //three pins are in output state
}
void loop()
{

  int a[10]={
        246,254,224,190,182,102,242,218,96,252};     //define the functional array
  for(int x=9; x>-1 ;x-- )                          //reciprocal function loop
  {
    digitalWrite(latchPin,LOW);
    shiftOut(dataPin,clockPin,MSBFIRST,a[x]);        //display the array a[x]
    digitalWrite(latchPin,HIGH);
    delay(1000);
  }
}
//////////////////////////////////////////////////////////
```

Test Result

Done uploading the code, you can see the LED display circularly show the number from 0 to 9.

Project 16: 8*8 Dot Matrix Display

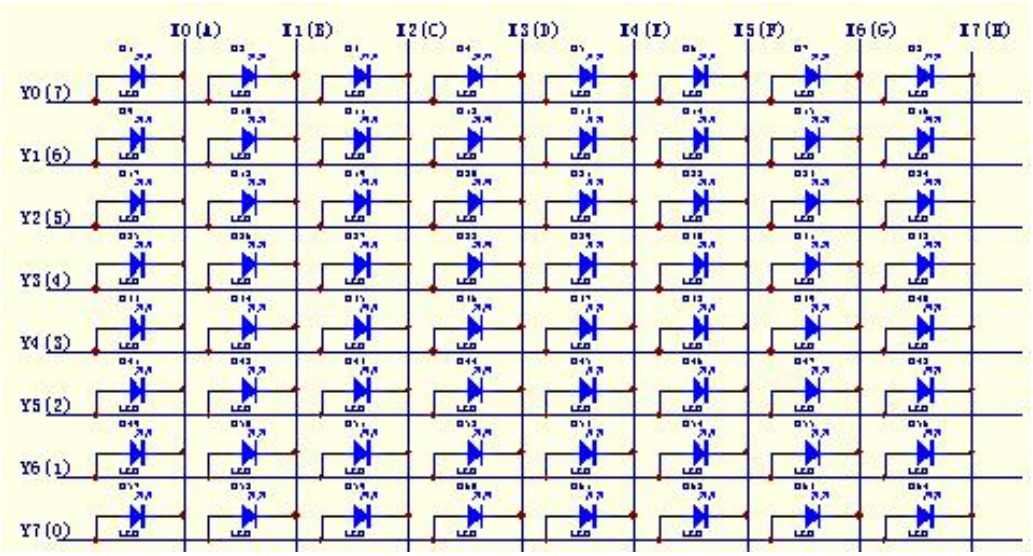
Description

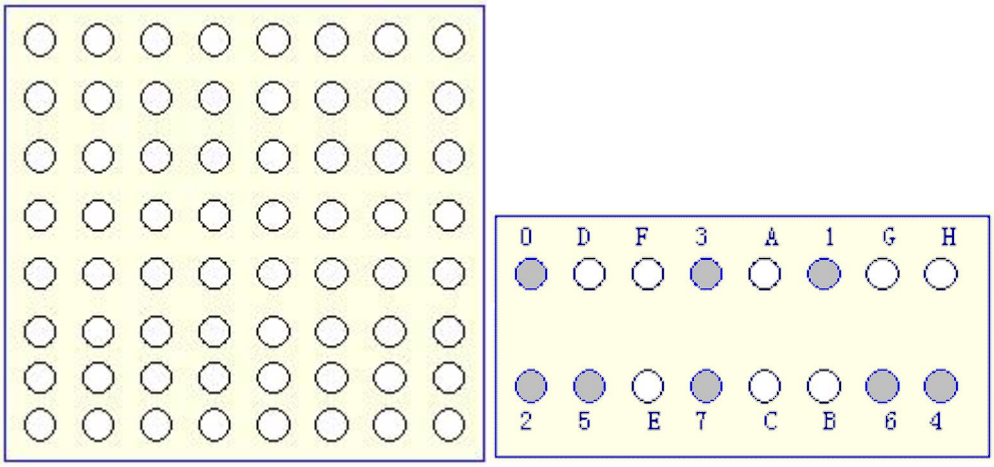
Actually dot matrix is very commonly seen in our life, such as LED advertising screen, elevator floor display and so on.

Now, we are going to learn a 8*8 dot matrix. It consists of 64 LEDs, located in every crossing of each row and column. When a row is at level 1, and a column at level 0 simultaneously, LED lights up that is between high and low level.

For instance, if you want to light up the first LED, connect pin 7 to high level, and pin A to low level; if you want to light up LEDs in first row, connect pin 7 to high level, and pin A, B,

C, D, E, F, G, H to low level; if you want to light up LEDs in first column, connect pin A to high level, and pin 0, 1, 2, 3, 4, 5, 6, 7 to low level.
Please refer to the schematic diagram shown as below figure.



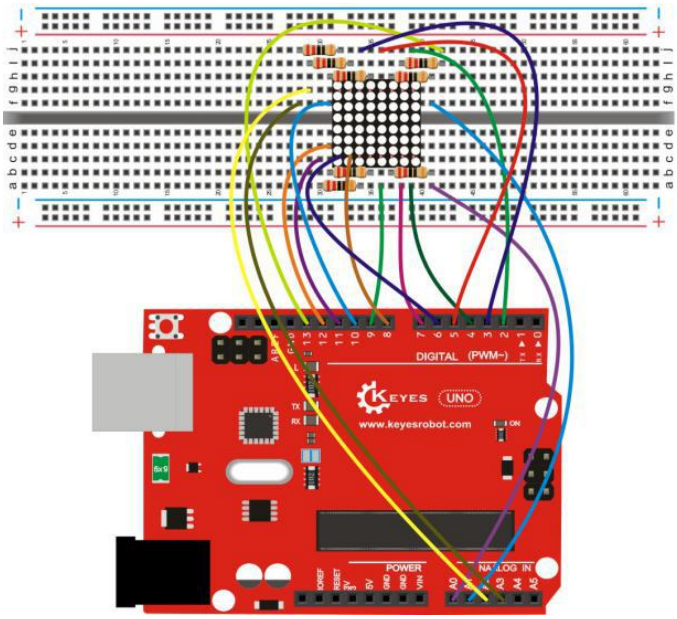


Part List

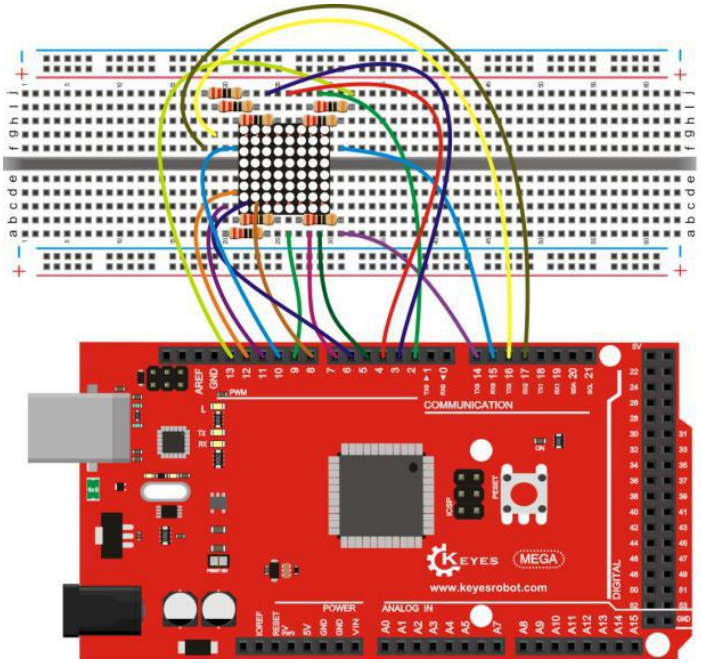
Development Board*1
USB Cable*1
8*8 Dot Matrix*1
220Ω resistor*8
Breadboard*1
Breadboard jumper wire*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
////////////////////////////////////////////////////////////////
//define an array to store “0”
unsigned char Text[]={0x00,0x1c,0x22,0x22,0x22,0x22,0x22,0x1c};
void Draw_point(unsigned char x,unsigned char y)//draw point function
{
    clear_();
    digitalWrite(x+2, HIGH);
    digitalWrite(y+10, LOW);
    delay(1);
```

```
}
void show_num(void)//display the function, finally call the draw point 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 the screen
{
   for(int i=2;i<10;i++)
   digitalWrite(i, LOW);
   for(int i=0;i<8;i++)
   digitalWrite(i+10, HIGH);
}
//////////////////////////////////////////////////////////////
```

Test Result

After wiring and uploading codes, the matrix will display the number “0”.

Project 17: 4-digit LED Display

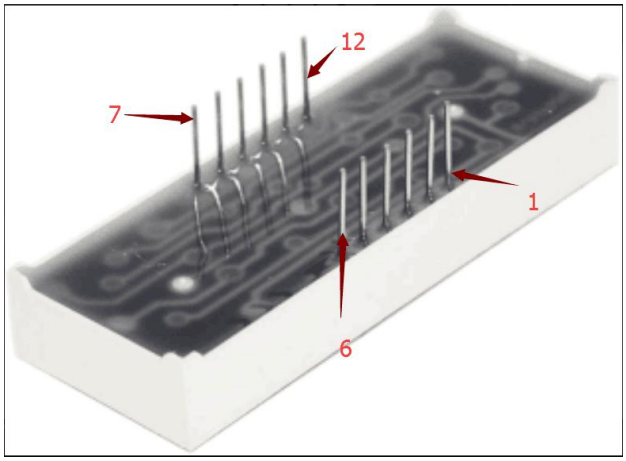
Description

We have done 1-digit LED display driven by development board before, so this time we are going to drive a common cathode 4-digit LED display. It is very essential to use the current limiting resistor when driving it.

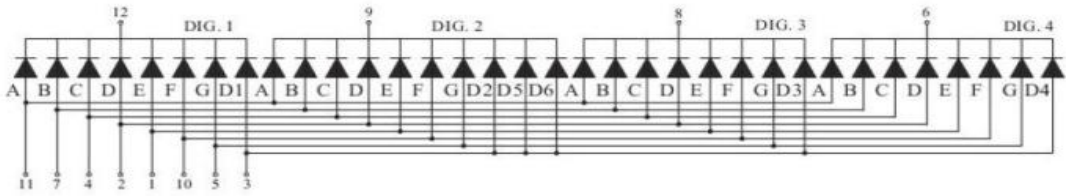
There are two connection methods for current limiting resistor. One is to connect total four current limiting resistors to D1-D4 anode end, using less resistors but the display brightness is uneven from the different numbers. Another method is to connect it to other eight pins, and this way, the display brightness is even but using more resistors.

This experiment will use eight 220Ω resistors.

Below figure you can see that this 4-digit LED display has total 12 pins (two rows of six), counting from 1 to 12 anticlockwise.



Schematic diagram of 4-digit LED Display shown below:



Part List

Development Board*1

USB Cable*1

4-digit LED Display*1

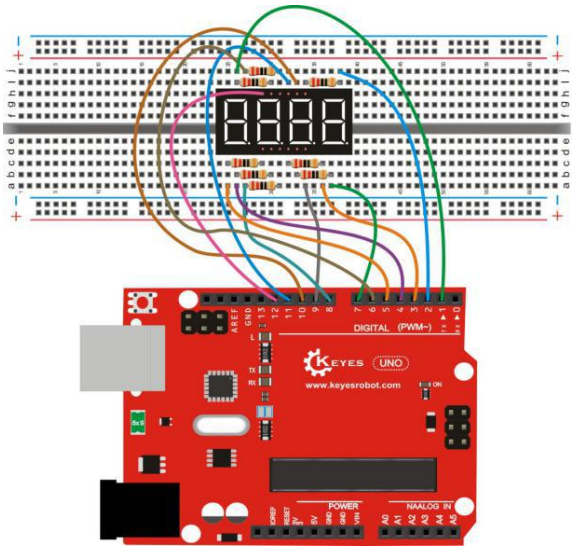
220Ω resistor*8

Breadboard*1

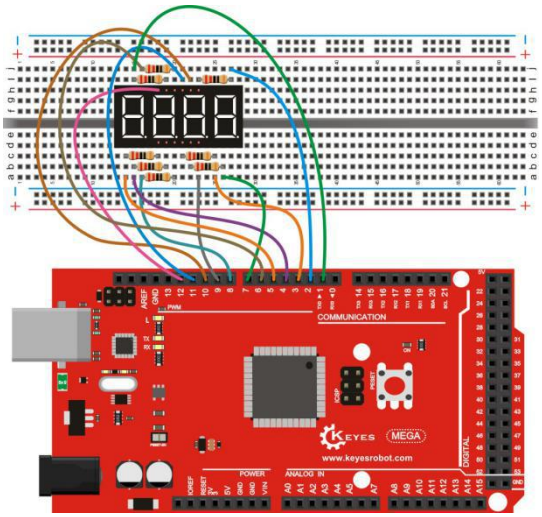
Jumper wire*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
////////////////////////////////////////////////////////////////
int a = 1;
int b = 2;
int c = 3;
int d = 4;
int e = 5;
int f = 6;
int g = 7;
int dp = 8;

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()
{
   int a=0;
   int b=0;
   int c=0;
   int d=0;
   unsigned long currentMillis = millis();

   while(d>=0)
   {
      while(millis()-currentMillis<1000)
        {
          Display(1,a);
          Display(2,b);
          Display(3,c);
          Display(4,d);
        }
      currentMillis = millis();
      d++;
```

```
   if (d>9)
   {
    c++;
    d=0;
   }
      if (c>9)
   {
    b++;
    c=0;
   }
      if (b>9)
   {
    a++;
    b=0;
   }
      if (a>9)
   {
    a=0;
    b=0;
    c=0;
    d=0;
   }
   }
}
//////////////////////////////////////////////////////////////
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
}
//////////////////////////////////////////////////////////////
```

Test Result

Done uploading the code, the LED display will show the number jumping from 0000 to 9999, circularly.

Project 18: 1602 LCD Display

Description

Since the IO port of development board is not enough to connect more sensors and modules. The original 1602 LCD screen needs 7 IO ports to drive it. While this 1602 I2C module integrates LCD pinboard and screen, and passes the I2C communication, so it only needs 2 IO ports can drive it.

The display capacity of 1602 LCD is 16x2 character. This project we will use the 1602 LCD to display the characters.

Part List

Development Board*1

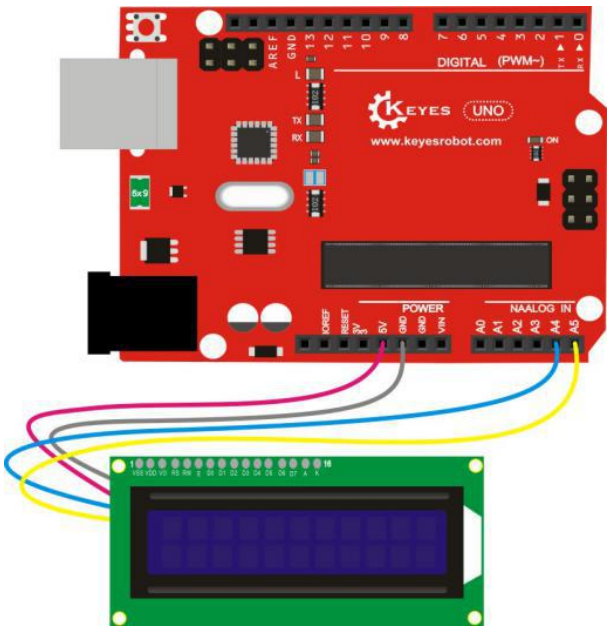
USB Cable*1

1602 LCD*1

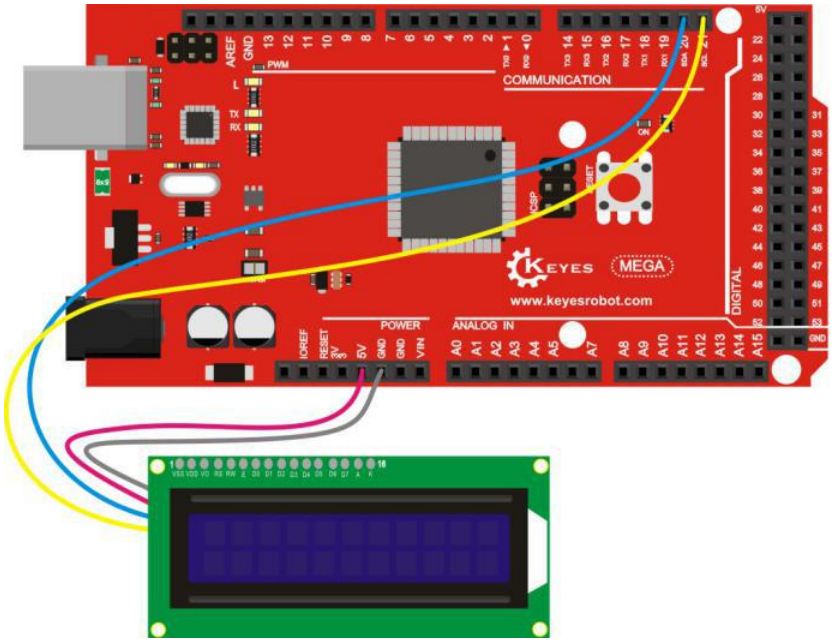
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);    // set the LCD address to 0x27 for a 16 chars and 2 line display
void setup()
{
  lcd.init();                        // initialize the lcd
  lcd.init();
  // Print a message to the LCD.
  lcd.backlight();
  lcd.setCursor(2,0);
  lcd.print("Hello, world!");
  lcd.setCursor(2,1);
  lcd.print("Hello, keyes!");
}
void loop()
{
}
//////////////////////////////////////////////////////////
```

Test Result

Done uploading the code to the board, rotate the potentiometer on the module to adjust its backlight, finally you can see the LCD display the character "Hello, world!" and "Hello, keyes!".

Project 19: Ultrasonic Rangefinder

Description

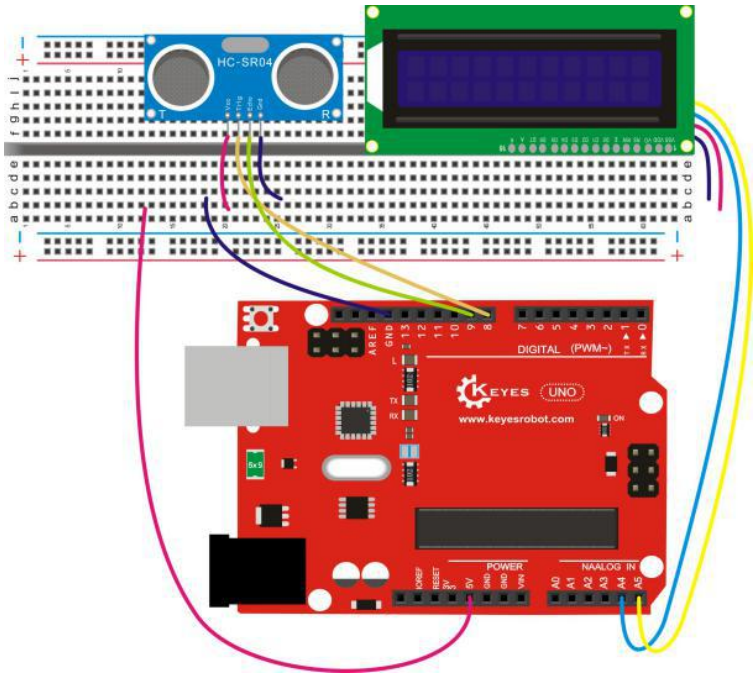
The ultrasonic sensor is mainly used for distance measurement. It has the features of high precision, super closeness in the blind area (2cm), and stable performance. In this experiment we mainly use an ultrasonic sensor and the 1602 I2C blue screen. We are going to detect the distance between the ultrasonic sensor and the obstacle, displaying it on the 1602 I2C blue screen.

Part List

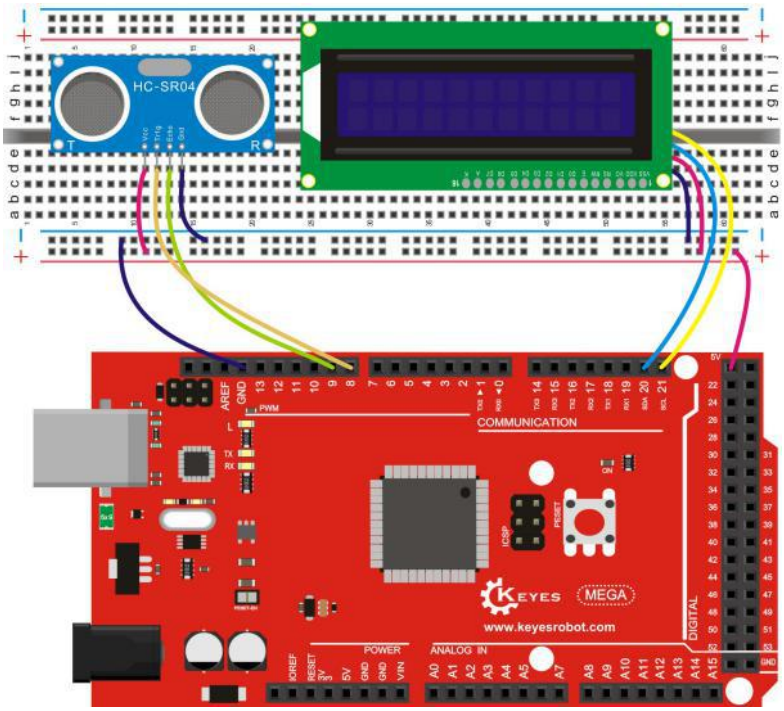
Development Board*1
USB Cable*1
1602 I2C Screen*1
Ultrasonic Sensor*1
Breadboard*1
Breadboard jumper wire*several
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
#define echoPin 9 // 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() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(LEDPin, OUTPUT); // Use LED indicator (if required)
    lcd.init();                                  // initialize the lcd
    // Print a message to the LCD.
    lcd.init();
    lcd.backlight();
    lcd.setCursor(0,0);
    lcd.print("The distance is:");
}

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" */
 lcd.setCursor(0,1);
 lcd.print("-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);
 if(distance<10)
{
 lcd.setCursor(0,1);
 lcd.print(distance);
 lcd.setCursor(1,1);
 lcd.print("    ");
}
if((distance >=10)&&(distance<100))
{
 lcd.setCursor(0,1);
 lcd.print(distance);
 lcd.setCursor(2,1);
 lcd.print("    ");
}
 if(distance>100)
{
 lcd.setCursor(0,1);
 lcd.print(distance);
}
 digitalWrite(LEDPin, LOW);
 }
```

```
  //Delay 50ms before next reading.
  delay(50);
}
////////////////////////////////////////////////////////
```

Test Result

Wire it up and upload well the code, you can rotate the potentiometer to adjust the backlight, finally the 1602 I2C screen will show the character "The distance is:"

If detecting the distance between the ultrasonic sensor and the obstacle, the data result will be displayed on1602 I2C screen . If no data, 1602 I2C screen will display the character “-1”.

Project 20: 1302 Clock Display

Description

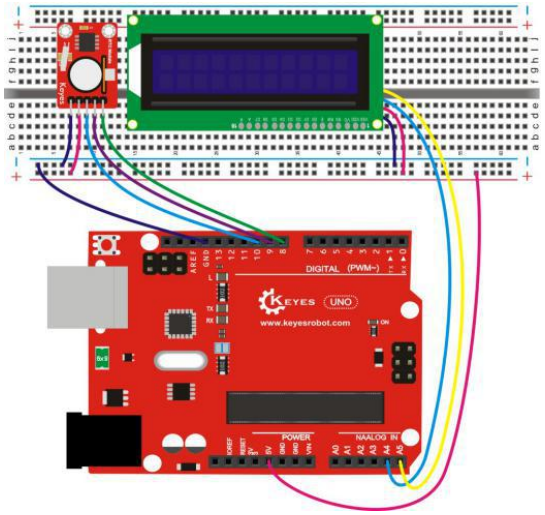
In this project, we also use 1602 I2C as the display. Let’s move on to make a real time clock by ourselves. In the code, you can set the year, month, day, week, hour, minute and second. Finally, you can see the time displayed on 1602 I2C screen when the clock starts to run.

Part List

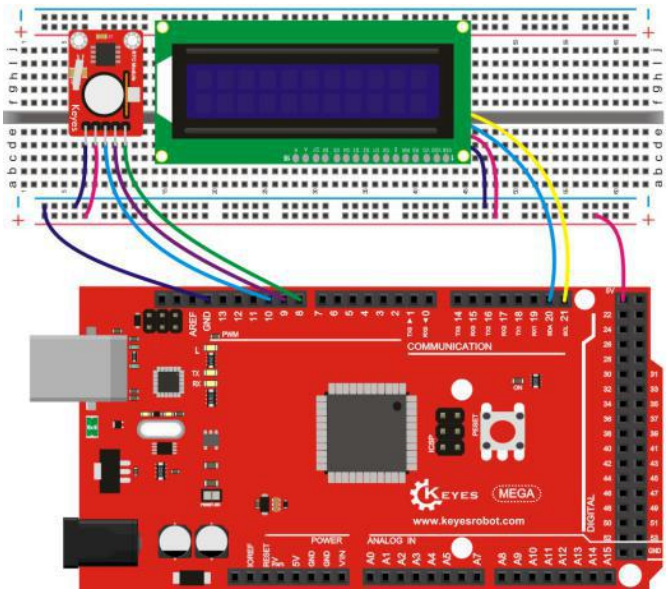
Development Board*1
USB Cable*1
1602 I2C Screen*1
1302 clock module*1
Breadboard*1
Jumper wire*several
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
#include <stdio.h>
#include <string.h>
#include <DS1302.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
/* Set the appropriate digital I/O pin connections */
uint8_t CE_PIN   = 10;    // RST
uint8_t IO_PIN   = 9;     // DAT
uint8_t SCLK_PIN = 8;     // CLK
/* Create buffers */
char buf[50];
char bf[50];
char bu[50];
char uf[50];
char day[10];
/* Create a DS1302 object */
DS1302 rtc(CE_PIN, IO_PIN, SCLK_PIN);
void print_time()
{
  /* Get the current time and date from the chip */
  Time t = rtc.time();

  /* Name the day of the week */
```

```
memset(day, 0, sizeof(day));    /* clear day buffer */
switch (t.day) {
   case 1:
      strcpy(day, "Sunday      ");
      break;
   case 2:
      strcpy(day, "Monday      ");
      break;
   case 3:
      strcpy(day, "Tuesday     ");
      break;
   case 4:
      strcpy(day, "Wednesday");
      break;
   case 5:
      strcpy(day, "Thursday ");
      break;
   case 6:
      strcpy(day, "Friday      ");
      break;
   case 7:
      strcpy(day, "Saturday ");
      break;
}

/* Format the time and date and insert into the temporary buffer */
snprintf(buf, sizeof(buf), "%s %04d-%02d-%02d %02d:%02d:%02d",
               day,
               t.yr, t.mon, t.date,
               t.hr, t.min, t.sec);
               Serial.println(buf);
snprintf(bf, sizeof(bf), "%s    %04d",
             day, t.yr);
lcd.setCursor(0,0);
lcd.print(bf);
snprintf(bu, sizeof(bu),"%02d:%02d:%02d",
               t.hr, t.min, t.sec);
/* Print the formatted string to serial so we can see the time */
lcd.setCursor(0,1);
lcd.print(bu);
snprintf(uf, sizeof(uf), "%02d-%02d",
           t.mon, t.date);
lcd.setCursor(11,1);
lcd.print(uf);
```

```
}
void setup()
{
  lcd.init();                    // initialize the lcd
  // Print a message to the LCD.
  lcd.init();
  lcd.backlight();
  Serial.begin(9600);

  /* Initialize a new chip by turning off write protection and clearing the
     clock halt flag. These methods needn't always be called. See the DS1302
     datasheet for details. */
  rtc.write_protect(false);
  rtc.halt(false);

  /* Make a new time object to set the date and time */
  /*      Tuesday, May 19, 2009 at 21:16:37.          */
  Time t(2017,10,24,10,11,22,3);

  /* Set the time and date on the chip */
  rtc.time(t);
}


/* Loop and print the time every second */
void loop()
{
  print_time();
  delay(1000);
}
////////////////////////////////////////////////////////////
```

Test Result

Wire it up well and upload the code, rotate the potentiometer to adjust well the screen backlight. Finally, you will see the 1602 I2C blue screen show the initial time you set, and then start running.

Project 21: PIR Motion Sensing

Description

The human body infrared pyroelectric sensor is an automatic control product based on infrared technology. It has the characteristics of high sensitivity, strong reliability, ultra-low power consumption, ultra-low voltage operation mode and so on.

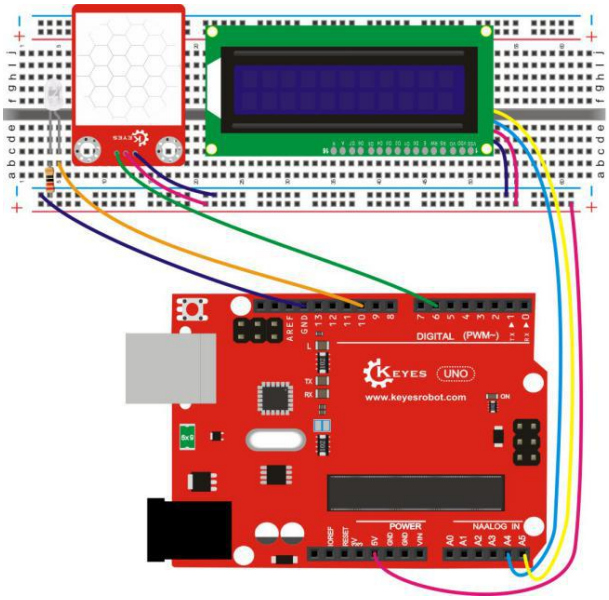
In this project, we are going to use the PIR motion sensor to detect whether there is someone moving nearby. When detecting the movement nearby or even no movement, 1602 I2C screen will display the corresponding character.

Part List

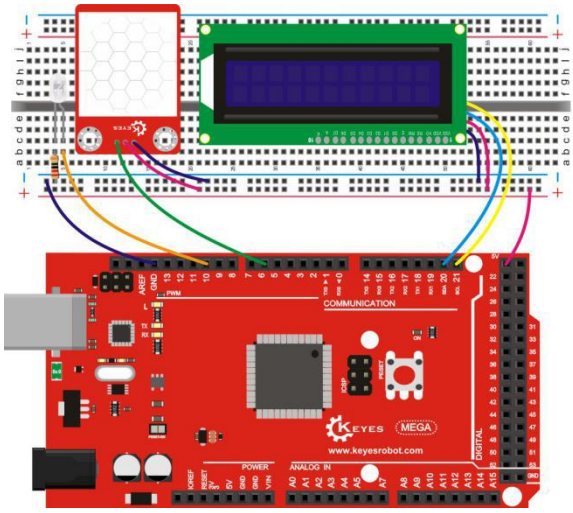
Development Board*1
USB Cable*1
1602 I2C Screen*1
PIR motion sensor*1
Breadboard*1
Jumper wire*several
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
// include the library code:
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
byte sensorPin = 6;//define the digital 6
byte indicator = 10;//define the digital 10
void setup()
{
   pinMode(sensorPin,INPUT);//set the digital 6 as input
   pinMode(indicator,OUTPUT);//set the digital 10 as output
      lcd.init();                                  // initialize the lcd
   // Print a message to the LCD.
   lcd.init();
   lcd.backlight();
}
void loop()
{
   byte state = digitalRead(sensorPin);//read the value of digital 6 and assign it to state
   digitalWrite(indicator,state);//control the state of digital10
   if(state == 1)//if digital 6 is high level, monitor will print out corresponding character and
   line wrap
  {
   lcd.setCursor(0,0);
   lcd.print("Somebody is");
   lcd.setCursor(0,1);
   lcd.print("in this area!");
   }
   else if(state == 0) //if digital 6 is low level, monitor will print out corresponding character
```

```
and line wrap
   {
   lcd.setCursor(0,0);
   lcd.print("No one!          ");
   lcd.setCursor(0,1);
   lcd.print("No one!          ");
   }
   delay(500);//delay 0.5S
}
//////////////////////////////////////////////////////////
```

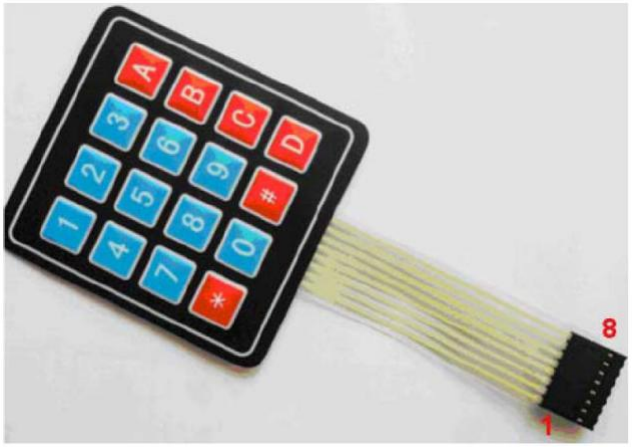
Test Result

Wire it up well and upload the code, rotate the potentiometer to adjust well the screen backlight. When detecting someone moving nearby, 1602 LCD will display the character "Somebody is " on the first line, and "in this area!" is displayed on the second line, you can see the LED is on. If no one moves nearby, 1602 LCD will display the character "No one!" in both two lines, LED off.

Project 22: 4*4 Keypad Display

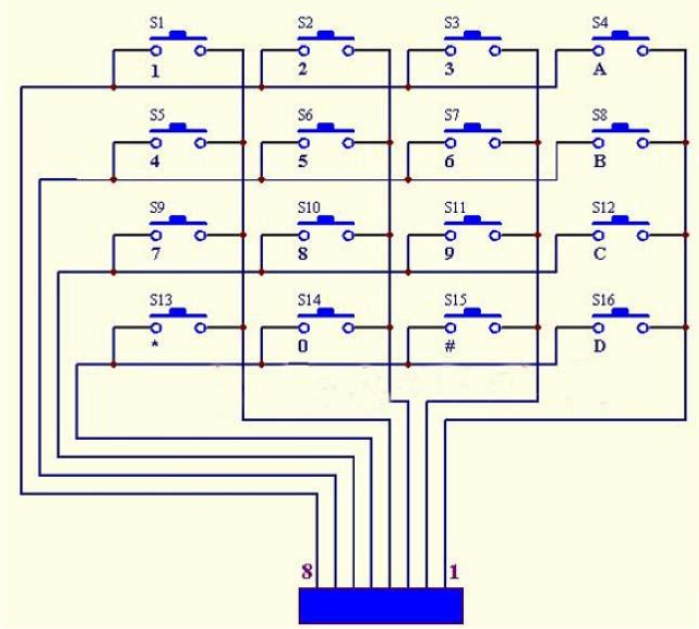
Description

In order to save the IO port of microcontroller, we have made a matrix keypad integrated with several buttons. In this project, press down the buttons on the keypad, finally show you the corresponding button value on the serial monitor.



You can see the button pin of 4*4 thin-film keypad in the picture.

Schematic Diagram Shown Below:



Part List

Development Board*1

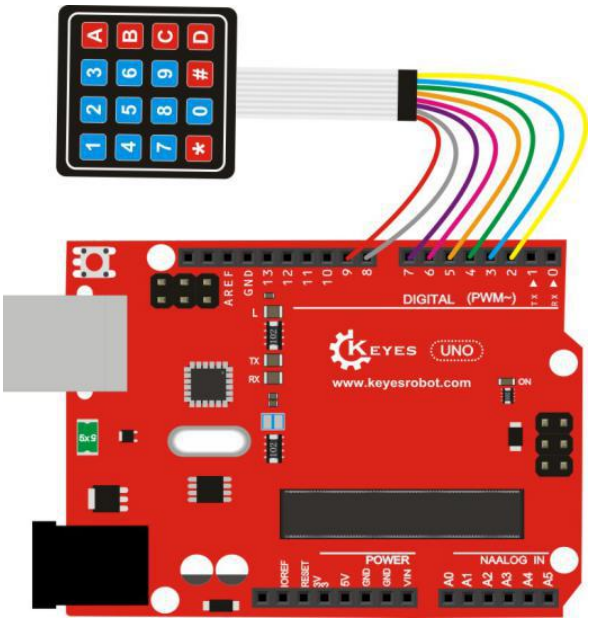
USB Cable*1

4*4 thin-film keypad*1

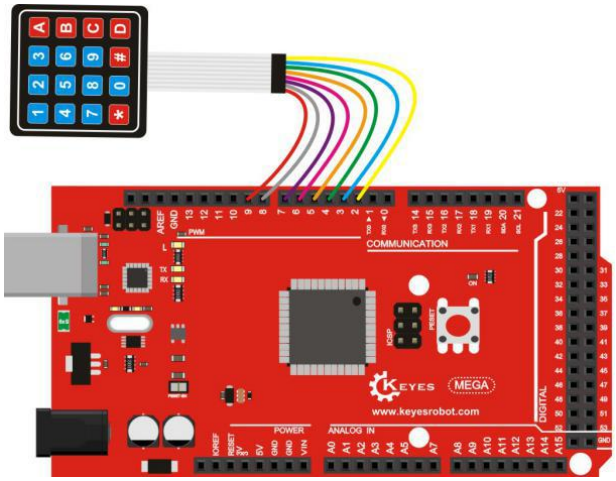
Breadboard jumper wire*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3

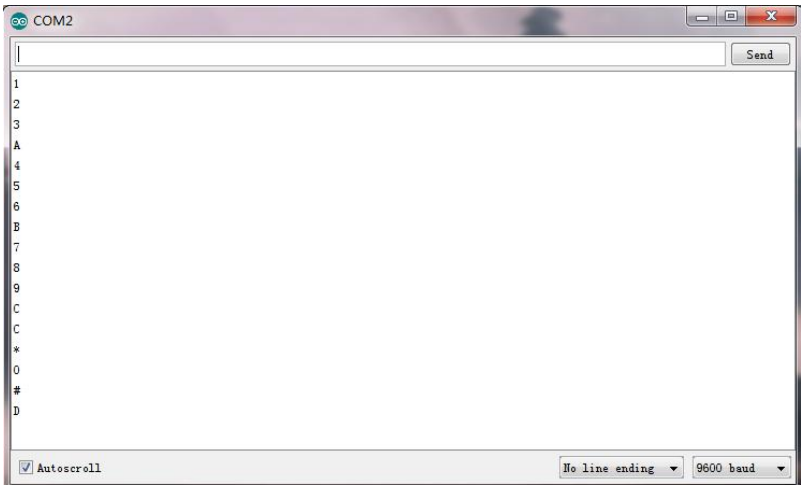


Sample Code

```
//////////////////////////////////////////////////////////
#include <Keypad.h>
const byte ROWS = 4; //define 4 rows
const byte COLS = 4; //define 4 cols
char keys[ROWS][COLS] = {
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};
//connect the 4*4 keypad row terminal, to control the digital IO ports of control board.
  byte rowPins[ROWS] = {9,8,7,6};
//connect the 4*4 keypad col terminal, to control the digital IO ports of control board.
byte colPins[COLS] = {5,4,3,2};
//call Keypad Class library function
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
void setup(){
Serial.begin(9600);
}
void loop(){
char key = keypad.getKey();
if (key != NO_KEY){
Serial.println(key);
}
}
//////////////////////////////////////////////////////////
```

Test Result

Done uploading the code to the board, open the serial monitor, then press down a button on the keypad, you can see the value of that button on the serial monitor. Shown below.



Project 23: Driving Stepper Motor

Description

The stepper motor is an actuator that converts electrical pulses into angular displacements. Simple to say, when the stepper driver receives a pulse signal, it drives the stepper motor to rotate a fixed angle (stepping angle) in the setting direction.

You can control the number of pulses to control the amount of angular displacement, so as to achieve the purpose of accurate positioning. At the same time, you can also control the speed and acceleration of the motor rotation by controlling the pulse frequency, so as to achieve the speed control.

The figure as shown below is the stepper motor we used in this project:

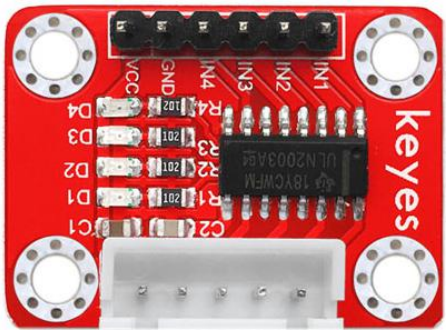
Deceleration Stepper Motor

- Diameter: 28mm
- Voltage: 5V
- Stepping angle: 5.625 x 1/64
- Reduction ratio: 1/64
- 5-wire 4-phase, can be driven with ordinary ULN2003 chip; can also used as a 2-phase.

The stepper motor's no-load power consumption is below 50mA, with 64 times reducer, and its output torque is relatively large, can drive heavy load, very suitable for the development board.

Note: The rotation speed of this stepper motor with 64 times reducer is slower, you can stick a small paper on its output shaft for convenient observation.

Stepper Motor (five-wire four-phase) Drive Board (ULN2003):



Part List

Development Board*1

USB Cable*1

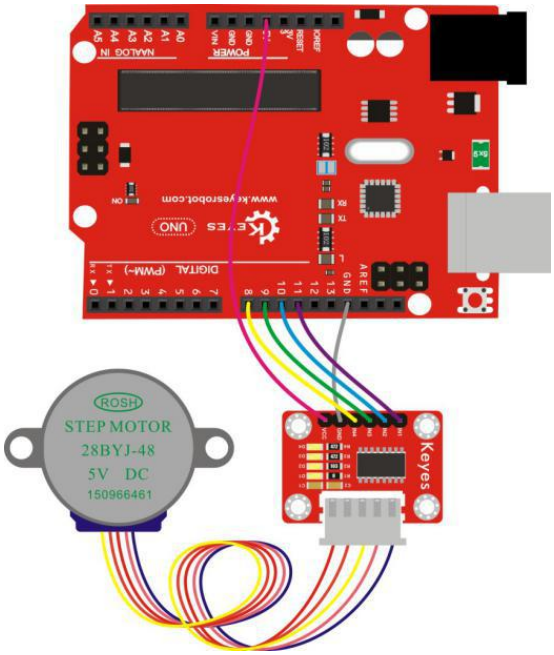
Stepper motor*1

ULN2003*1

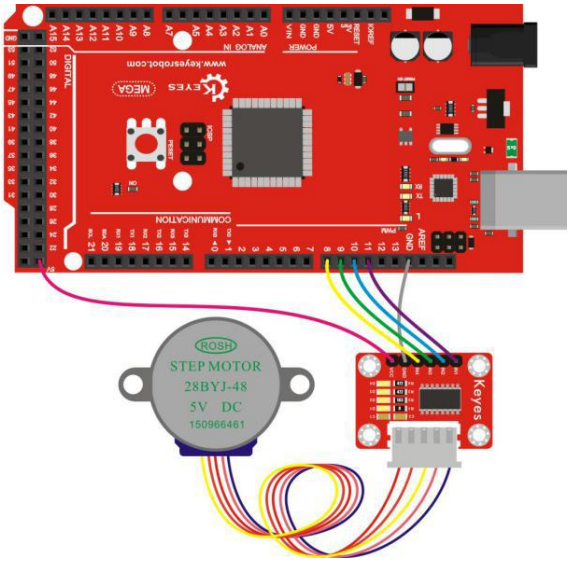
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
#include <Stepper.h>
//set the step number of stepper motor for rotating a circle.
#define STEPS 100
//set the step number and pin of stepper motor
Stepper stepper(STEPS, 11, 10, 9, 8);
//define the variable for storing the history steps
int previous = 0;
void setup()
{
 //set the speed of motor as 90steps per minute
  stepper.setSpeed(90);
}
void loop()
{
 //get the sensor reading number
 int val = analogRead(0);
 //The number of moves is the current reading minus previous readings
 stepper.step(val - previous);
 //save the previous steps
 previous = val;
}
//////////////////////////////////////////////////////////
```

Test Result

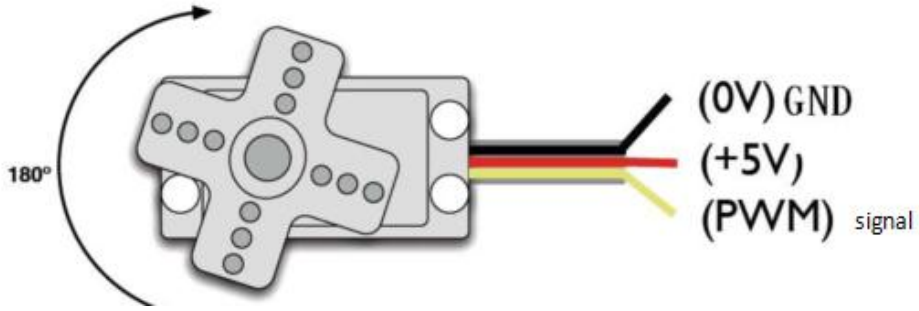
Done uploading the code, you can see the stepper motor rotate in a very slow speed.

Project 24: Controlling Servo Motor

Description

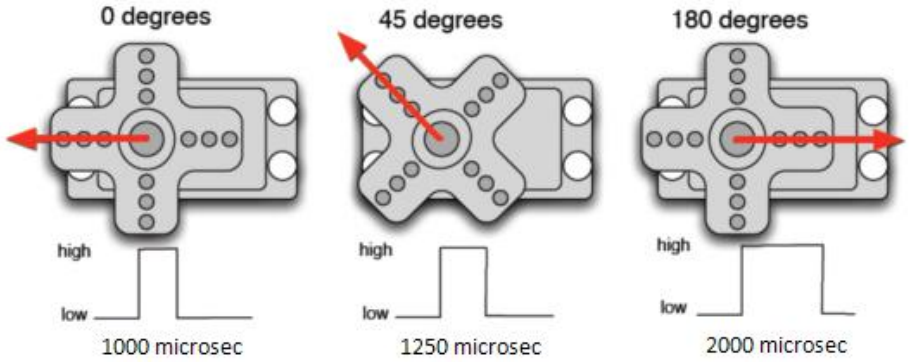
Servo motor is a kind of location servo driver, mainly consisting of shell, circuit board, no-core motor, gear and location detector.

Servo motor has many different sizes, but all of them have three external connection wires, distinguished by brown, red and orange (different brand may have different color). Brown one is for GND, red one for power supply positive-end, 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, but in fact, it's between 0.5ms-2.5ms. The width corresponds the rotation angle of servo motor from 0°to 180°.

Note that for different brands of motor, the same signal may have different rotation angles.



Part List

Development Board*1

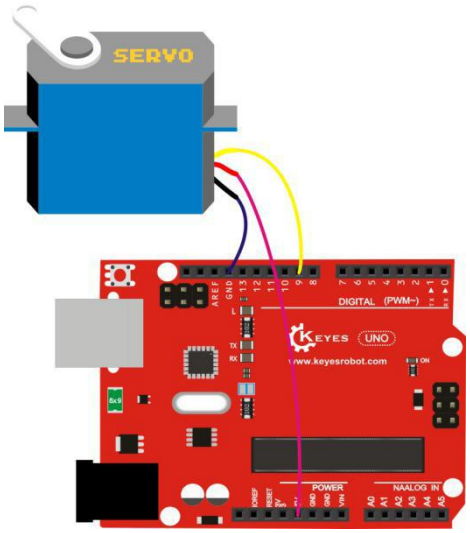
USB Cable*1

Servo motor*1

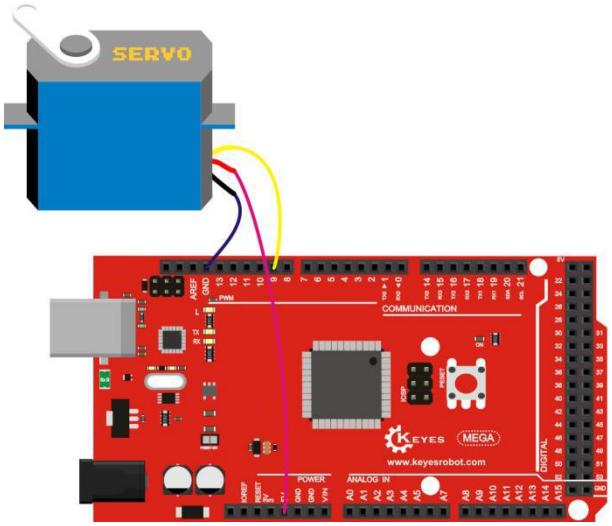
Breadboard jumper wire*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

Program A:

```
//////////////////////////////////////////////////////////////
int servopin=9;//digital 9 is connected to the signal line of servo motor
int myangle;//define the angle variable
int pulsewidth;//define the pulsewidth variable
int val;
void servopulse(int servopin,int myangle)//define a pulse function
{
pulsewidth=(myangle*11)+500;//convert the angle into the plusewidth value of 500-2480
digitalWrite(servopin,HIGH);//set servo pin as high level
delayMicroseconds(pulsewidth);//delay the microseconds of pulsewidth
digitalWrite(servopin,LOW);//set servo pin as low level
delay(20-pulsewidth/1000);
```

```
}
void setup()
{
pinMode(servopin,OUTPUT);//set the servo as output
Serial.begin(9600);//connecting to serial port, baud rate as 9600
Serial.println("servo=o_seral_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 the value of serial port
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 enough time to rotate to commanded angle
{
servopulse(servopin,val);//call the pulse function

}
}
}
////////////////////////////////////////////////////////////
```

Program B:

```
////////////////////////////////////////////////////////////
#include <Servo.h>
Servo myservo;///define servo motor’s variable name
void setup()
{
myservo.attach(9);//select servo pin(9 or 10, only to control two pins)
}
void loop()
{
myservo.write(90);//set the rotating angle of servo
}
////////////////////////////////////////////////////////////
```

Test Result

For Program A:

Enter the number on serial monitor, and click “Send”. The position where the servo rotates to

the corresponding number of angles, and the angle will be displayed on the screen.

For Program B:

The servo motor turns to the position of 90 degrees by itself.

Project 25: RFID Card Reader

Description

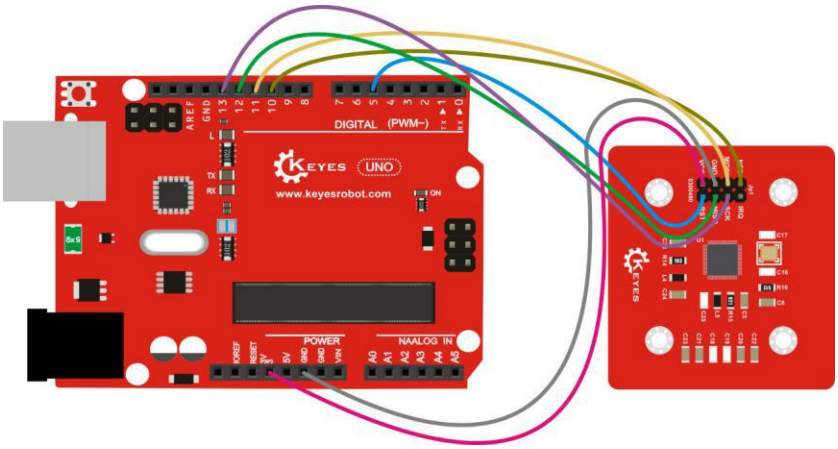
Radio frequency identification or technology is referred to as RFID. This experiment will use the RFID module to read the contents of the IC card and key chain. For RFID module, must use +3.3V power supply, otherwise it will burn out the module.

Part List

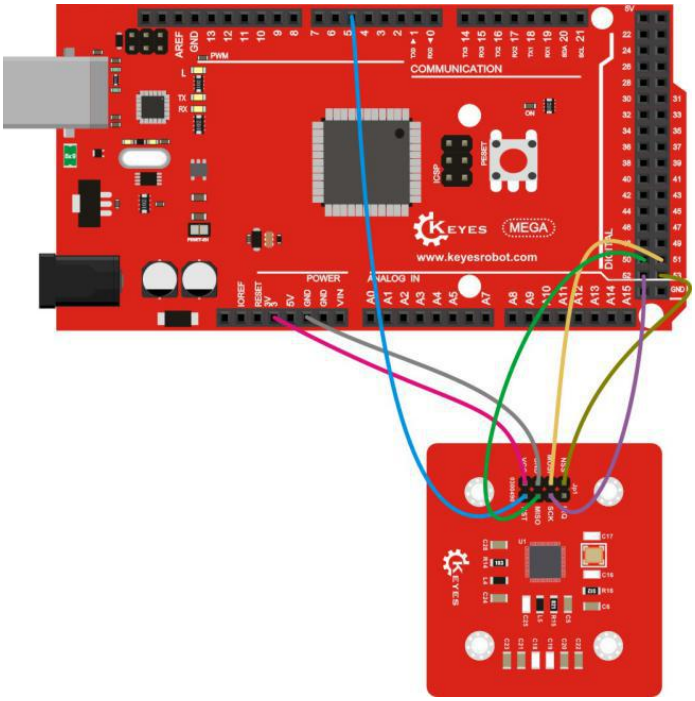
Development Board*1
USB Cable*1
RFID－RC522 module*1
IC card *1
Key chain*1
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
#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 chipSelectPin = 53;//if the controller is MEGA 2560
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_SElECTTAG             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     DivlEnReg                 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};

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,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 into sleeping 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 original value 0x6363  ???

    AntennaOn();        // open antenna
}
uchar MFRC522_Request(uchar reqMode, uchar *TagType)
{
    uchar status;

    uint backBits;              // bits of data received
    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);     // permission for interrupt request
    ClearBitMask(CommIrqReg, 0x80);                // clear all bits of the interrupt request
    SetBitMask(FIFOLevelReg, 0x80);                //FlushBuffer=1, FIFO initialize

    Write_MFRC522(CommandReg, PCD_IDLE);   //NO action; clear 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 transmission
    i = 2000; // adjust i according to clock frequency, max wait time for M1 card operation
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 the data received in FIFO
                    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 completion of CRC calculation
    i = 0xFF;
    do
    {
        n = Read_MFRC522(DivIrqReg);
        i--;
    }
    while ((i!=0) && !(n&0x04));                //CRCIrq = 1
```

```
    // read result from CRC calculation
    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);                    //MFCrypto1On=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 instructions + 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;
}
```

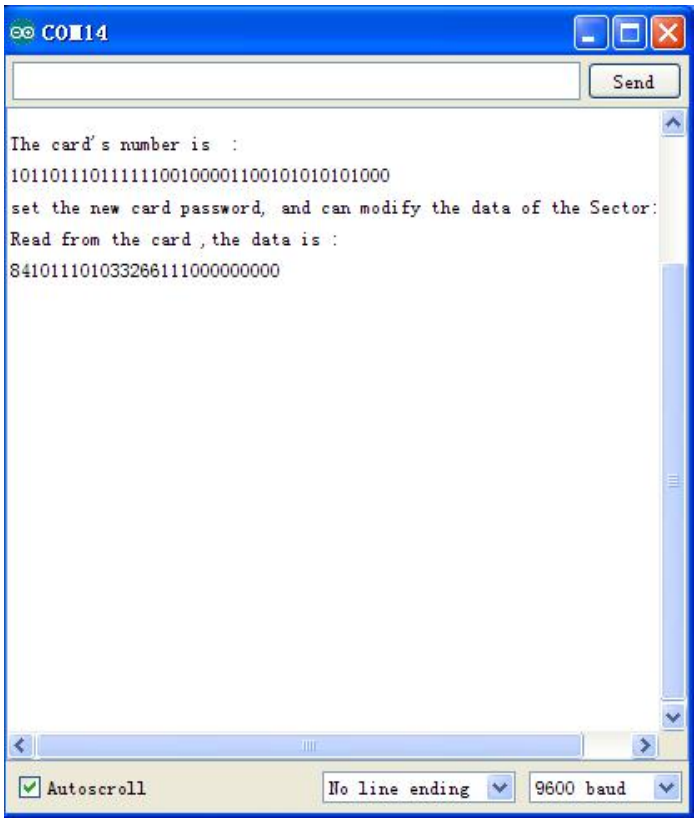
```
uchar MFRC522_Write(uchar blockAddr, uchar *writeData)
{
    uchar status;
    uint recvBits;
    uchar i;
    uchar buff[18];


    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 connect to MEGA 2560 development board, in the code **const int chipSelectPin = 10;//if the controller is UNO,328,168** should be changed as **const int chipSelectPin = 53;//if the controller is MEGA 2560**

Test Result

Done uploading the code to the board, put the IC card and key chain close to the module, you can read the data showed on the serial monitor. Shown below.



Project 26: Sound-Controlled Light

Description

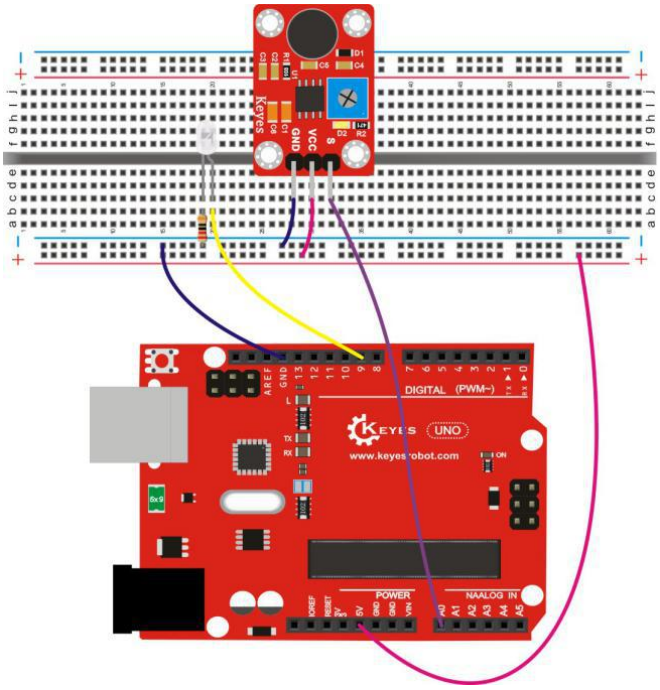
The sound sensor is specifically designed to detect the sounds. The S terminal of sensor is for analog output, which is for the microphone's voltage signal output in real time. The potentiometer on the sensor can be used to adjust the signal gain. In the experiment, we use the sensor to detect the sound volume, so as to control an LED on and off.

Part List

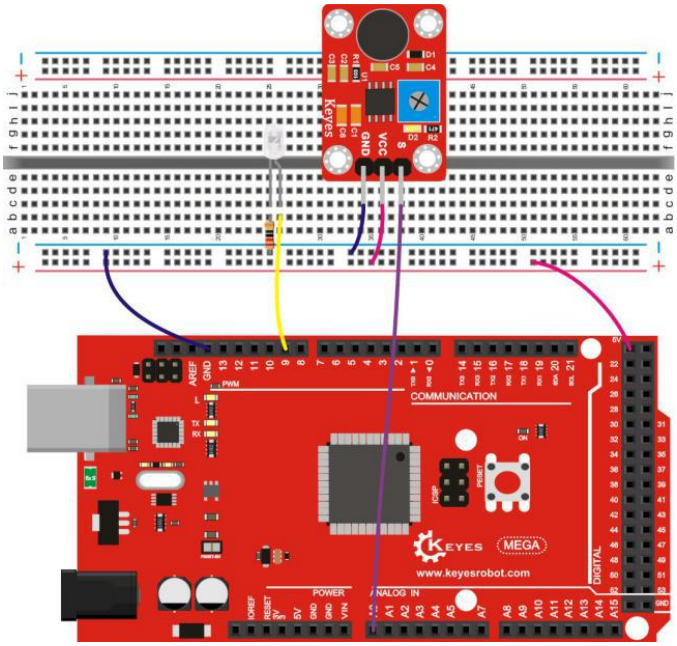
Development Board*1
USB Cable*1
Sound sensor*1
LED*1
220Ω resistor*1
Breadboard*1
Jumper wire*several
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
int MIC=0;//set the sound sensor as analog 0 interface
int LED=9;//define LED interface as digital 9
int val=0;//define the digital variable
  void setup()
{
   pinMode(LED,OUTPUT);//define LED as output
  pinMode(MIC,INPUT);//define sound sensor as input
  Serial.begin(9600);//set the baud rate as 9600
  }
void loop()
{
   val=analogRead(MIC);//read the analog value of sound sensor
   Serial.println(val);//output the analog value and print out
   if(val>=300)//analog value is greater than300, LED is on.
   {
    digitalWrite(LED,HIGH);
    }else
    {
       digitalWrite(LED,LOW);
     }
    delay(500);
}
//////////////////////////////////////////////////////////
```

Test Result

Powered up and upload well the code, it can detect the sound volume and output the analog value. The greater the sound, the bigger the value. When the sound volume reaches to a certain value, LED on, otherwise LED is off.

Project 27: Relay-Controlled LED

Description

The relay module is used for controlling the high power by low power to protect the circuits. The 5V single relay module used in this experiment is active at high level. It has a control indicator light indicated for the connected and disconnected state.

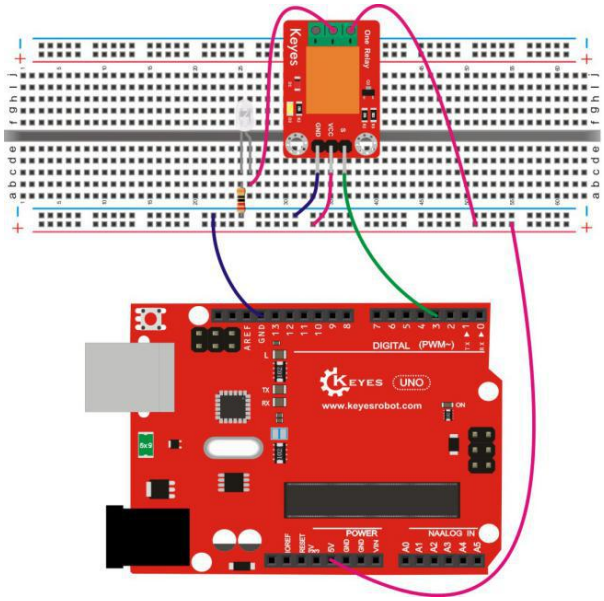
In the experiment we are going to control the on and off of an LED through controlling the relay.

Part List

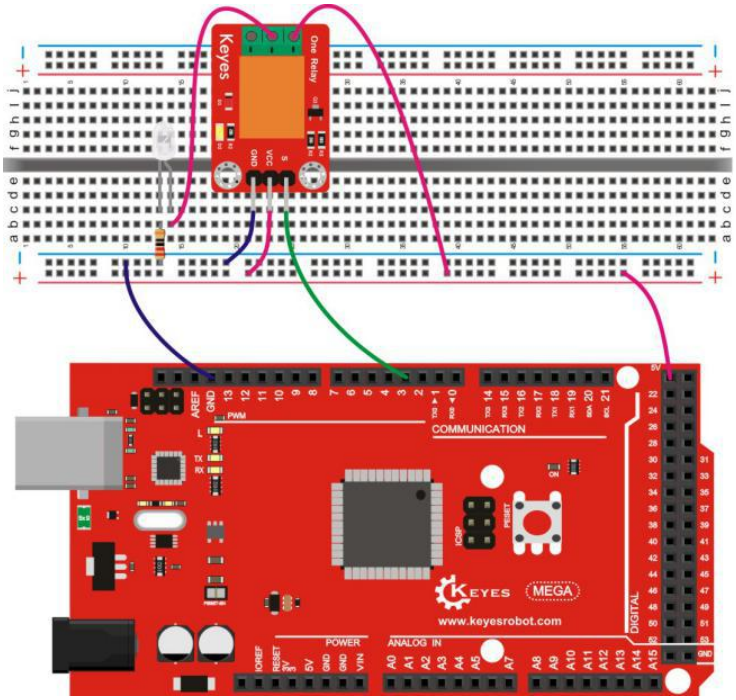
Development Board*1
USB Cable*1
Relay module*1
LED*1
220Ω resistor*1
Breadboard*1
Jumper wire*several
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
int Relay = 3; //define the digital 3
void setup()
{
pinMode(Relay, OUTPUT); //set the Relay as output
}
void loop()
{
digitalWrite(Relay, HIGH); //turn on the relay
delay(2000); //delay 2S
digitalWrite(Relay, LOW); //turn off the relay
delay(2000); //delay 2S
}
//////////////////////////////////////////////////////////
```

Test Result

Wire it up and upload the code well, the relay on for 2S (ON and COM end are connected), and LED on; the off for 2S (NC and COM end are connected), and LED off, circularly. The D2 light on the relay is on when connected.

Project 28: Displaying Temperature and Humidity

Description

In this experiment we mainly use the DHT11 temperature and humidity sensor and the 1602 I2C blue screen.

DHT11 temperature and humidity sensor is a temperature and humidity composite sensor with calibrated digital signal output. It applies special digital module acquisition technology and temperature and humidity sensing technology to ensure the high reliability and excellent long-term stability.

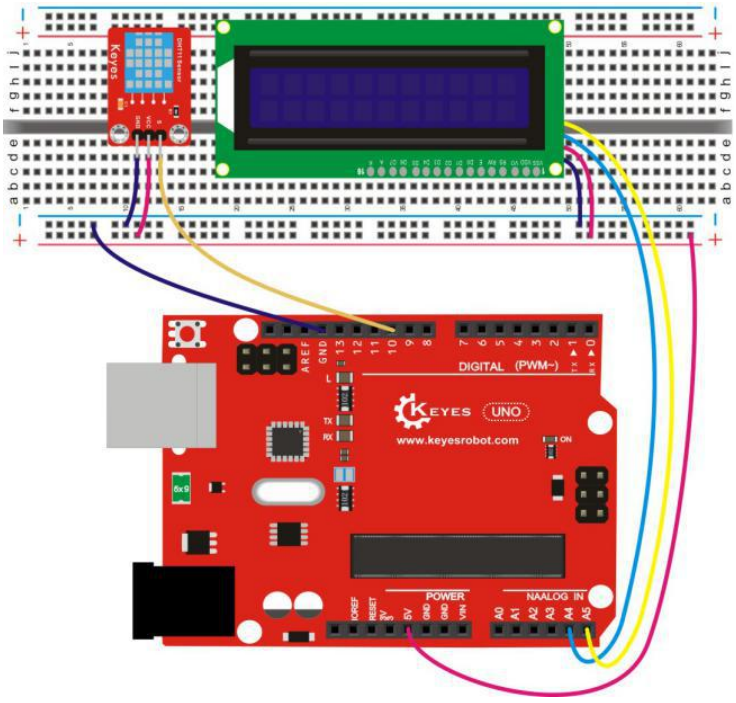
In the experiment, we make use of DHT11 temperature and humidity sensor to test the temperature and humidity in the current environment, displaying the test results on the 1602 I2C blue screen.

Part List

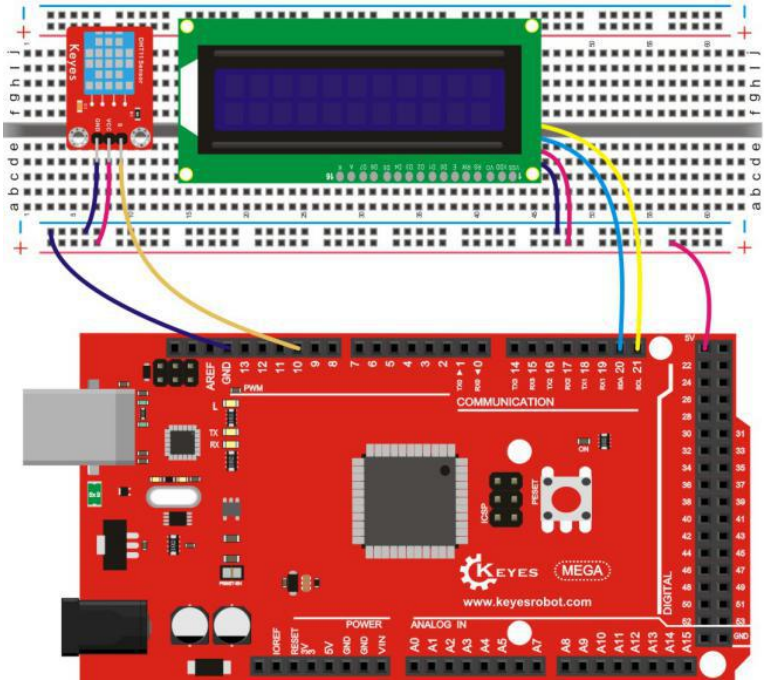
Development Board*1
USB Cable*1
1602 I2C Screen*1
DHT11 sensor*1
Breadboard*1
Jumper wire*several
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
#include <dht11.h>
// include the library code:
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
dht11 DHT;
#define DHT11_PIN 10
void setup(){
   lcd.init();                                  // initialize the lcd
   // Print a message to the LCD.
   lcd.init();
   lcd.backlight();
   lcd.setCursor(0,0);
   lcd.print("Humidity (%):");
   lcd.setCursor(0,1);
   lcd.print("Temp (C):");
}
void loop(){
   int chk;
   chk = DHT.read(DHT11_PIN);        // READ DATA
   switch (chk){
      case DHTLIB_OK:
                        break;
```

```
case DHTLIB_ERROR_CHECKSUM:
                    break;
      case DHTLIB_ERROR_TIMEOUT:
                    break;
      default:
                    break;
   }
  // DISPLAT DATA
   lcd.setCursor(13,0);
   lcd.print(DHT.humidity);
   lcd.setCursor(9,1);
   lcd.print(DHT.temperature);
   delay(1000);
}
//////////////////////////////////////////////////////////
```

Test Result

Wire it up and upload the code well, rotate the potentiometer knob to adjust the backlight, finally you can see the current temperature and humidity value displayed on 1602 I2C blue screen.

Project 29: Gas Detection

Description

This project we are going to use the MQ-2 gas sensor to detect the combustible gases in the air, displaying the results on a 1602 I2C blue screen.

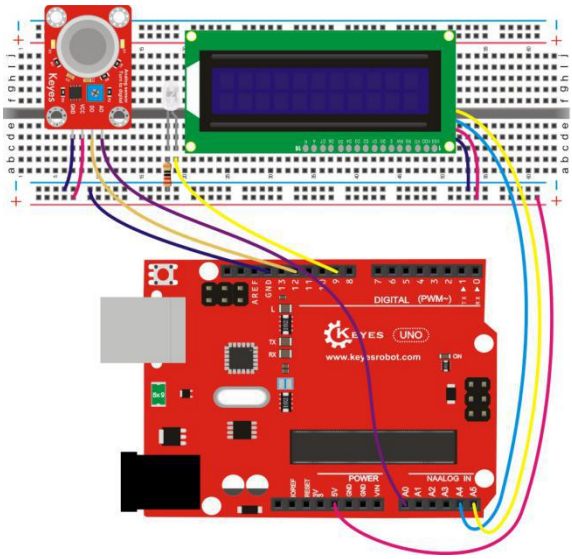
MQ-2 gas sensor is mainly used for detecting liquefied gas, propane and hydrogen, etc. It has two output ports: analog output and digital output. Its analog output voltage increases with the increase of gas concentration in the detected environment, and has the features of fast response recovery, sensitivity adjustable, signal output indication and so on.

Part List

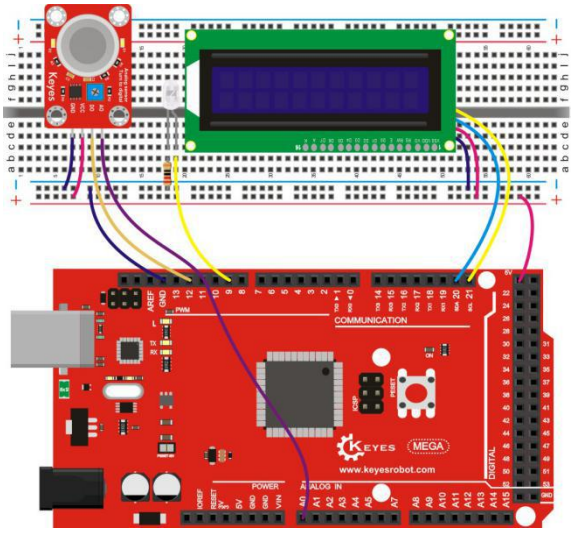
Development Board*1
USB Cable*1
LED*1
220Ω resistor*1
1602 I2C Screen*1
MQ-2 gas sensor*1
Breadboard*1
Jumper wire*several
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////////
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
int gas_din=12;
int gas_ain=A0;
int led=9;
int ad_value;
void setup()
```

```
{
  pinMode(led,OUTPUT);
  pinMode(gas_din,INPUT);
  pinMode(gas_ain,INPUT);
  lcd.init();                                  // initialize the lcd
  lcd.init();
  // Print a message to the LCD.
  lcd.backlight();
}
void loop()
{
  ad_value=analogRead(gas_ain);
  if(digitalRead(gas_din)==LOW)
  {
    digitalWrite(led,HIGH);
    lcd.setCursor(0,0);
    lcd.print("Gas leakage!        ");
    lcd.setCursor(0,1);
    lcd.print("Value:");
     if(ad_value<10)
  {
  lcd.setCursor(6,1);
  lcd.print(ad_value);
  lcd.setCursor(7,1);
  lcd.print("                 ");
  }
  if((ad_value>=10)&&(ad_value<100))
  {
  lcd.setCursor(6,1);
  lcd.print(ad_value);
  lcd.setCursor(8,1);
  lcd.print("               ");
  }
  if( ad_value>=100)
  {
  lcd.setCursor(6,1);
  lcd.print(ad_value);
  lcd.setCursor(9,1);
  lcd.print("             ");
  }
  }
  else
  {
   digitalWrite(led,LOW);
```

```
      lcd.setCursor(0,0);
      lcd.print("Gas not leak!      ");
      lcd.setCursor(0,1);
      lcd.print("Gas not leak!     ");
    }
   delay(500);
}
//////////////////////////////////////////////////////////
```

Test Result

Done uploading the code to the board, rotate the potentiometer on the gas sensor to adjust its sensitivity. The sensitivity becomes the best when the LED on the sensor is adjusted to the critical point between off and on state. You can rotate the potentiometer on the 1602 I2C screen to adjust the backlight.

When no flammable gas is detected, both the first line and second line of the 1602 I2C screen will display the character "Gas not leak!", and the plug-in LED does not light.

If detecting the combustible gas, the first line of 1602 I2C blue screen will display the character "Gas leakage!", and the character "Value:" and the output analog value will be displayed on the second line.

Project 30: Controlling RGB Module

Description

The RGB color model is an additive color model in which red, green and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red (R), green (G), and blue (B).

This color standard includes almost all the colors that human vision can perceive.

In this project, we will use an RGB module. By adjusting the joystick module and potentiometer, you can get the color change of the RGB light, and the data is displayed on the 1602 I2C blue screen. This way, you can get the overlay method of all colors.

For example, in the experiment we have adjusted R to 255, G to 255, B to 255, so RGB module will show the white light, in this way we can know that white light can be obtained by superimposing RGB 1:1:1.

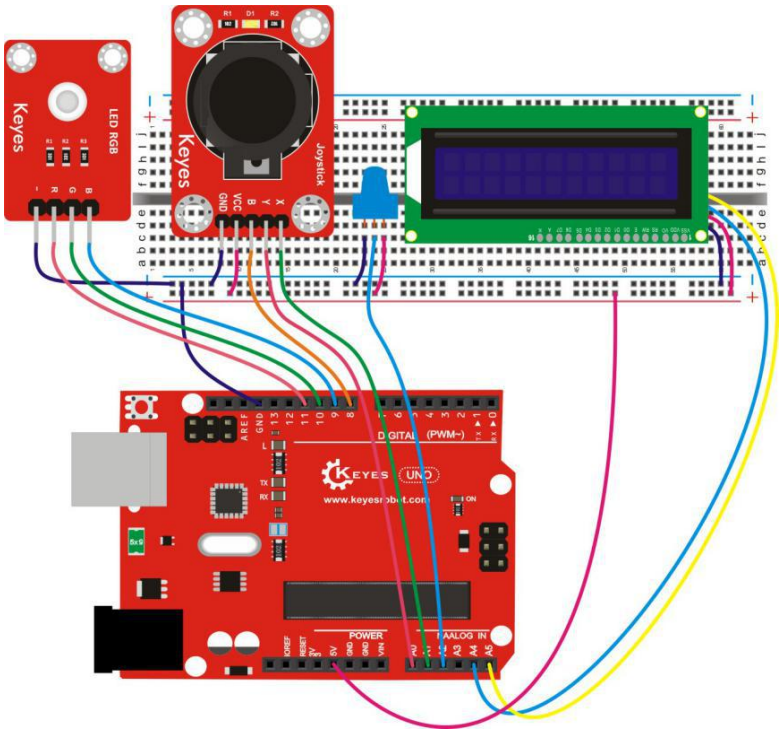
Part List

Development Board*1
USB Cable*1
Joystick module*1
Potentiometer*1
1602 I2C Screen*1

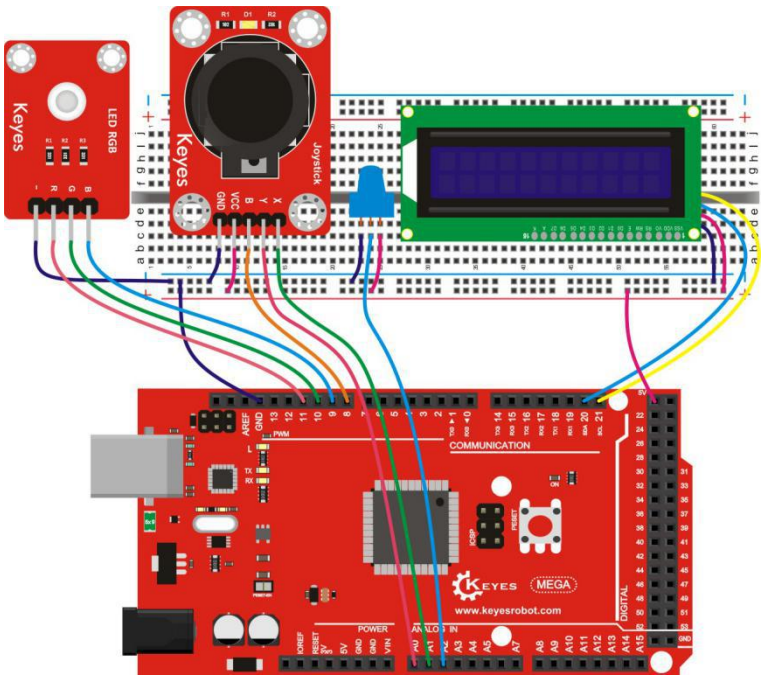
Keyes RGB module*1
Breadboard*1
Jumper wire*several
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
int redpin = 11; //select the pin for the red LED
int greenpin =10;// select the pin for the green LED
int bluepin =9; // select the pin for the blue LED
int Z =8;
int val;
int value1;
int value2;
int value3;

void setup() {
   pinMode(redpin, OUTPUT);
   pinMode(bluepin, OUTPUT);
   pinMode(greenpin, OUTPUT);
   pinMode(Z, INPUT);
   lcd.init();    // initialize the lcd
   // Print a message to the LCD.
   lcd.init();
   lcd.backlight();
   lcd.setCursor(0,0);
   lcd.print("R;");
   lcd.setCursor(8,0);
   lcd.print("G;");
   lcd.setCursor(0,1);
   lcd.print("B;");
}

void loop()
{
val=digitalRead(Z);
if(val==HIGH)
{
   analogWrite(redpin, 255);
   lcd.setCursor(2,0);
   lcd.print("255");
   analogWrite(greenpin, 255);
   lcd.setCursor(10,0);
   lcd.print("255");
   analogWrite(bluepin, 255);
```

```
   lcd.setCursor(2,1);
   lcd.print("255");
}
else
{
value1=map( analogRead(0),0,1023,0,255);
value2=map( analogRead(1),0,1023,0,255);
value3=map( analogRead(2),0,1023,0,255);
analogWrite(redpin, value1);
if(value1<10)
{
   lcd.setCursor(2,0);
   lcd.print(value1);
   lcd.setCursor(3,0);
   lcd.print("      ");
}
if((value1>=10)&&(value1<100))
{
   lcd.setCursor(2,0);
   lcd.print(value1);
   lcd.setCursor(4,0);
   lcd.print("    ");
}
if(value1>=100)
{
   lcd.setCursor(2,0);
   lcd.print(value1);
}
delay(100);
analogWrite(greenpin, value2);
if(value2<10)
{
   lcd.setCursor(10,0);
   lcd.print(value2);
   lcd.setCursor(11,0);
   lcd.print("      ");
}
if((value2>=10)&&(value2<100))
{
   lcd.setCursor(10,0);
   lcd.print(value2);
   lcd.setCursor(12,0);
   lcd.print("    ");
}
```

```
if(value2>=100)
{
  lcd.setCursor(10,0);
  lcd.print(value2);
}
delay(100);
analogWrite(bluepin, value3);
if(value3<10)
{
  lcd.setCursor(2,1);
  lcd.print(value3);
  lcd.setCursor(3,1);
  lcd.print("  ");
}
if((value3>=10)&&(value3<100))
{
  lcd.setCursor(2,1);
  lcd.print(value3);
  lcd.setCursor(4,1);
  lcd.print("  ");
}
if(value3>=100)
{
  lcd.setCursor(2,1);
  lcd.print(value3);
}
delay(100);
}
}
//////////////////////////////////////////////////////////////
```

Test Result

Upload the code to the board, press the Z direction of joystick module, RGB module will display the white light, and 1602 I2C screen will display the value 255. Release the Z direction of joystick module, then adjust the joystick module and potentiometer, you can see the RGB module display different colors and the corresponding data is shown on the 1602 I2C screen.

Project 31: TMD27713 Distance Sensor

Description

Keyes TMD27713 distance sensor is a three-in-one sensor that integrates ambient light, proximity sensor and infrared LED.

It has two main functions as follows:

For one thing, it can be used to detect the current ambient light brightness (ALS). And can automatically adjust the backlight brightness to suit the ambient brightness using software regulation. So this way can not only save power but also protect your eyesight.

For another is called the proximity sensing function (PROX).

This experiment is simply to test the basic function of this sensor.

Part List

Development Board*1

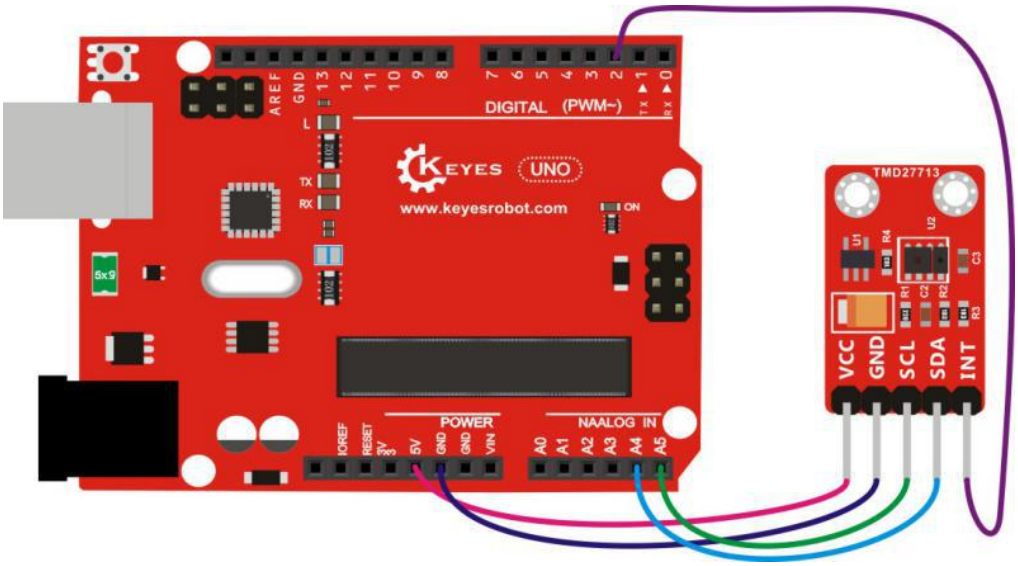
USB Cable*1

keyes TMD27713 distance sensor*1

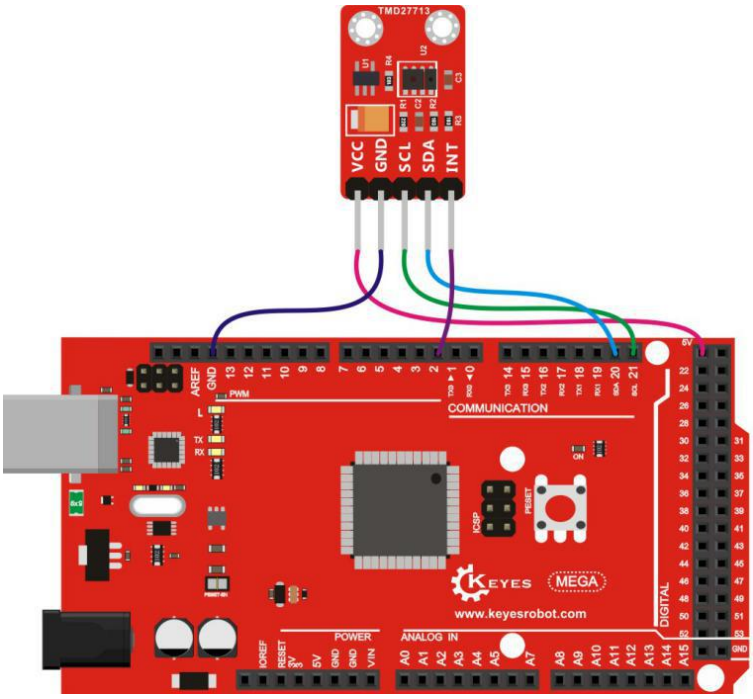
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
#define DUMP_REGS

#include <Wire.h>
#include <APDS9930.h>

// Pins
#define APDS9930_INT    2   // Needs to be an interrupt pin
#define LED_PIN         13 // LED for showing interrupt

// Constants
#define PROX_INT_HIGH   600 // Proximity level for interrupt
#define PROX_INT_LOW    0   // No far interrupt

// Global variables
APDS9930 apds = APDS9930();
float ambient_light = 0; // can also be an unsigned long
uint16_t ch0 = 0;
uint16_t ch1 = 1;
uint16_t proximity_data = 0;
volatile bool isr_flag = false;

void setup() {
```

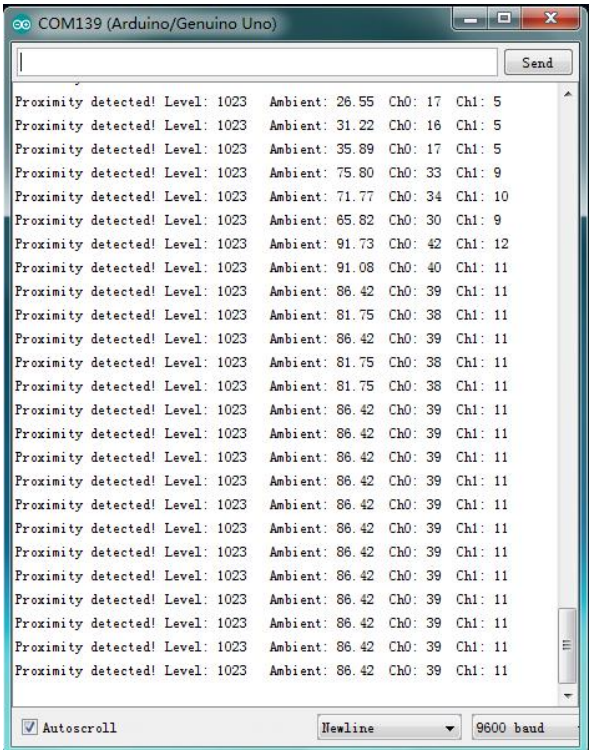
```
// Set LED as output
pinMode(LED_PIN, OUTPUT);
pinMode(APDS9930_INT, INPUT);

// Initialize Serial port
Serial.begin(9600);
Serial.println();
Serial.println(F("------------------------------"));
Serial.println(F("APDS-9930 - ProximityInterrupt"));
Serial.println(F("------------------------------"));

// Initialize interrupt service routine
attachInterrupt(digitalPinToInterrupt(APDS9930_INT), interruptRoutine, FALLING);

// Initialize APDS-9930 (configure I2C and initial values)
if (apds.init()) {
  Serial.println(F("APDS-9930 initialization complete"));
}
else {
  Serial.println(F("Something went wrong during APDS-9930 init!"));
}

// Adjust the Proximity sensor gain
if (!apds.setProximityGain(PGAIN_2X)) {
  Serial.println(F("Something went wrong trying to set PGAIN"));
}

// Set proximity interrupt thresholds
if (!apds.setProximityIntLowThreshold(PROX_INT_LOW)) {
  Serial.println(F("Error writing low threshold"));
}
if (!apds.setProximityIntHighThreshold(PROX_INT_HIGH)) {
  Serial.println(F("Error writing high threshold"));
}

// Start running the APDS-9930 proximity sensor (interrupts)
if (apds.enableProximitySensor(true)) {
  Serial.println(F("Proximity sensor is now running"));
}
else {
  Serial.println(F("Something went wrong during sensor init!"));
}

// Start running the APDS-9930 light sensor (no interrupts)
```

```
  if (apds.enableLightSensor(false)) {
    Serial.println(F("Light sensor is now running"));
  }
  else {
    Serial.println(F("Something went wrong during light sensor init!"));
  }

#ifdef DUMP_REGS
  /* Register dump */
  uint8_t reg;
  uint8_t val;

  for (reg = 0x00; reg <= 0x19; reg++) {
    if ((reg != 0x10) && \
      (reg != 0x11))
    {
      apds.wireReadDataByte(reg, val);
      Serial.print(reg, HEX);
      Serial.print(": 0x");
      Serial.println(val, HEX);
    }
  }
  apds.wireReadDataByte(0x1E, val);
  Serial.print(0x1E, HEX);
  Serial.print(": 0x");
  Serial.println(val, HEX);
#endif

}

void loop() {

  // If interrupt occurs, print out the proximity level
  if (isr_flag) {

    // Read proximity level and print it out
    if (!apds.readProximity(proximity_data)) {
      Serial.println("Error reading proximity value");
    }
    else {
      Serial.print("Proximity detected! Level: ");
      Serial.print(proximity_data);
      Serial.print("      ");
    }
```

```
      apds.readAmbientLightLux(ambient_light);
      // Read the light levels (ambient, red, green, blue)
      if (!apds.readAmbientLightLux(ambient_light) ||
         !apds.readCh0Light(ch0) ||
         !apds.readCh1Light(ch1)) {
         Serial.println(F("Error reading light values"));
      }
      else {
         Serial.print(F("Ambient: "));
         Serial.print(ambient_light);
         Serial.print(F("    Ch0: "));
         Serial.print(ch0);
         Serial.print(F("    Ch1: "));
         Serial.println(ch1);
      }

      // Turn on LED for a half a second
      digitalWrite(LED_PIN, HIGH);
      delay(300);
      digitalWrite(LED_PIN, LOW);

      // Reset flag and clear APDS-9930 interrupt (IMPORTANT!)
      isr_flag = false;
      if (!apds.clearProximityInt()) {
         Serial.println("Error clearing interrupt");
      }

   }
}

void interruptRoutine() {
   isr_flag = true;
}
////////////////////////////////////////////////////////////
```

Test Result

When testing, you need to use the software Arduino-1.8.2, powered up and upload well the code, open the serial monitor, you can see the data as the following figure showed.



Project 32: Acceleration Sensor

Description

The MMA8452Q sensor is a smart low power, triaxial and capacitive micromechanical accelerometer with 12-bit resolution. This accelerometer has rich embedded features with flexible user-programmable options that can configure up to two interrupt pins. The embedded interrupt function can save overall power consumption and relieve the host processor of the burden of constantly polling the data.

The MMA8452Q has a user-selectable range of ±2g/±4g/±8g and can output high-pass filtered data and unfiltered data in real time. The device can be configured to generate inertial wake-up interrupt signals using any combination of configurable embedded functions. This allows the MMA8452Q to remain in a low-power mode while still monitoring events.

This experiment will use the keyes MMA8452Q triaxial digital acceleration sensor to test the triaxial acceleration of the object.

Part List

Development Board*1

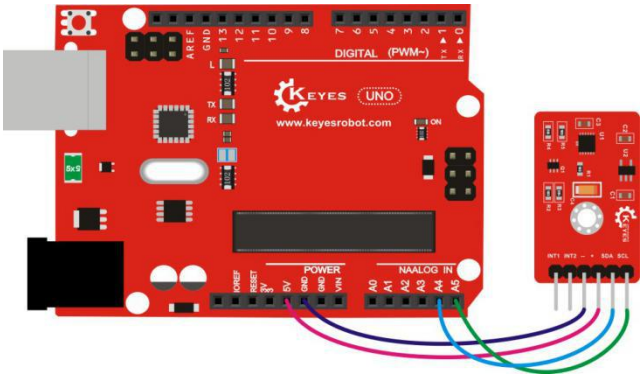
USB Cable*1

keyes MMA8452Q acceleration sensor*1

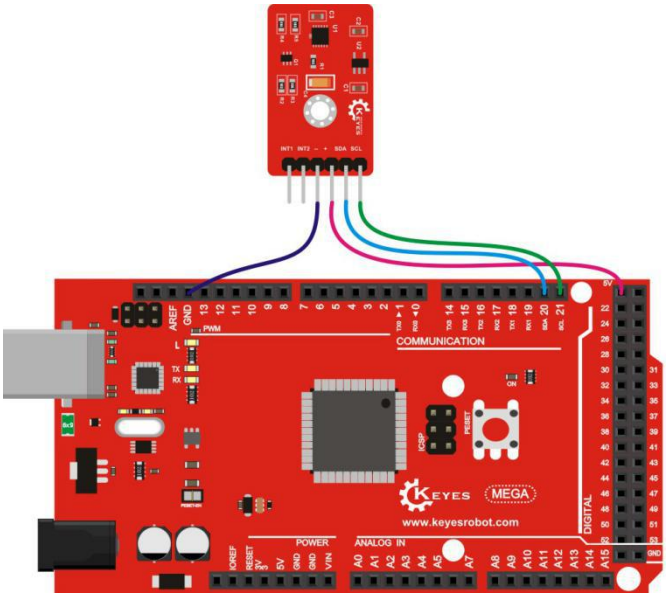
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////////////
#include <Wire.h> // Must include Wire library for I2C
#include <SparkFun_MMA8452Q.h> // Includes the SFE_MMA8452Q library

// Begin using the library by creating an instance of the MMA8452Q
//    class. We'll call it "accel". That's what we'll reference from
//    here on out.
MMA8452Q accel;

// The setup function simply starts serial and initializes the
//    accelerometer.
void setup()
```

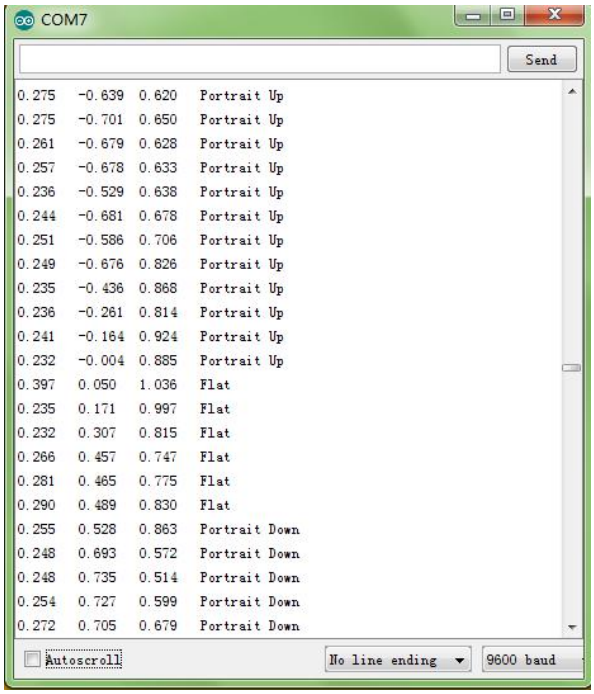
```
{
  Serial.begin(9600);
  Serial.println("MMA8452Q Test Code!");

  // Choose your adventure! There are a few options when it comes
  // to initializing the MMA8452Q:
  //  1. Default init. This will set the accelerometer up
  //     with a full-scale range of +/-2g, and an output data rate
  //     of 800 Hz (fastest).
  accel.init();
  //  2. Initialize with FULL-SCALE setting. You can set the scale
  //     using either SCALE_2G, SCALE_4G, or SCALE_8G as the value.
  //     That'll set the scale to +/-2g, 4g, or 8g respectively.
  //accel.init(SCALE_4G); // Uncomment this out if you'd like
  //  3. Initialize with FULL-SCALE and DATA RATE setting. If you
  //     want control over how fast your accelerometer produces
  //     data use one of the following options in the second param:
  //     ODR_800, ODR_400, ODR_200, ODR_100, ODR_50, ODR_12,
  //     ODR_6, or ODR_1.
  //     Sets to 800, 400, 200, 100, 50, 12.5, 6.25, or 1.56 Hz.
  //accel.init(SCALE_8G, ODR_6);
}

// The loop function will simply check for new data from the
//  accelerometer and print it out if it's available.
void loop()
{
  // Use the accel.available() function to wait for new data
  //  from the accelerometer.
  if (accel.available())
  {
    // First, use accel.read() to read the new variables:
    accel.read();

    // accel.read() will update two sets of variables.
    // * int's x, y, and z will store the signed 12-bit values
    //     read out of the accelerometer.
    // * floats cx, cy, and cz will store the calculated
    //     acceleration from those 12-bit values. These variables
    //     are in units of g's.
    // Check the two function declarations below for an example
    // of how to use these variables.
    printCalculatedAccels();
    //printAccels(); // Uncomment to print digital readings
```

```
      // The library also supports the portrait/landscape detection
      //  of the MMA8452Q. Check out this function declaration for
      //  an example of how to use that.
      printOrientation();

      Serial.println(); // Print new line every time.
    }
}

// The function demonstrates how to use the accel.x, accel.y and
//  accel.z variables.
// Before using these variables you must call the accel.read()
//  function!
void printAccels()
{
  Serial.print(accel.x, 3);
  Serial.print("\t");
  Serial.print(accel.y, 3);
  Serial.print("\t");
  Serial.print(accel.z, 3);
  Serial.print("\t");
}

// This function demonstrates how to use the accel.cx, accel.cy,
//  and accel.cz variables.
// Before using these variables you must call the accel.read()
//  function!
void printCalculatedAccels()
{
  Serial.print(accel.cx, 3);
  Serial.print("\t");
  Serial.print(accel.cy, 3);
  Serial.print("\t");
  Serial.print(accel.cz, 3);
  Serial.print("\t");
}

// This function demonstrates how to use the accel.readPL()
// function, which reads the portrait/landscape status of the
// sensor.
void printOrientation()
{
  // accel.readPL() will return a byte containing information
```

```
    // about the orientation of the sensor. It will be either
    // PORTRAIT_U, PORTRAIT_D, LANDSCAPE_R, LANDSCAPE_L, or
    // LOCKOUT.
    byte pl = accel.readPL();
    switch (pl)
    {
    case PORTRAIT_U:
       Serial.print("Portrait Up");
       break;
    case PORTRAIT_D:
       Serial.print("Portrait Down");
       break;
    case LANDSCAPE_R:
       Serial.print("Landscape Right");
       break;
    case LANDSCAPE_L:
       Serial.print("Landscape Left");
       break;
    case LOCKOUT:
       Serial.print("Flat");
       break;
    }
}
////////////////////////////////////////////////////////////
```

Test Result

Done uploading the code, open the serial monitor, tilt the sensor, you can see the value change shown below.



Project 33: Detecting Ultraviolet Light

Description

Keyes GUVA-S12SD 3528 solar UV sensor is used to detect the ultraviolet light.

It contains GUVA-S12SD, which can be widely used in UV detection of smart wearable devices, such as watches with UV index detection function, mobile phones with UV index detection, outdoor detecting UV index equipment, etc. It can also be used for UV disinfection, used to monitor UV intensity or used as UV flame detectors and so on.

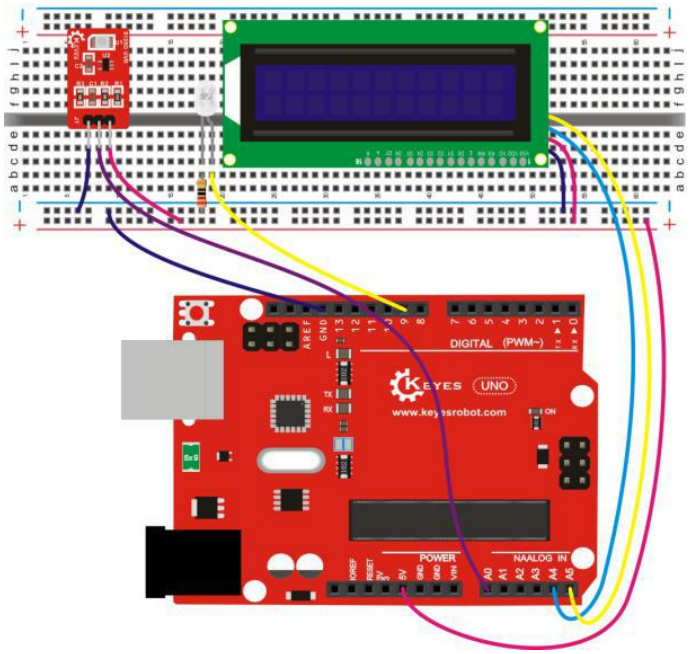
In this experiment, the ultraviolet sensor is used to detect the UV in the current environment and the result will be displayed on the 1602 LCD.

Part List

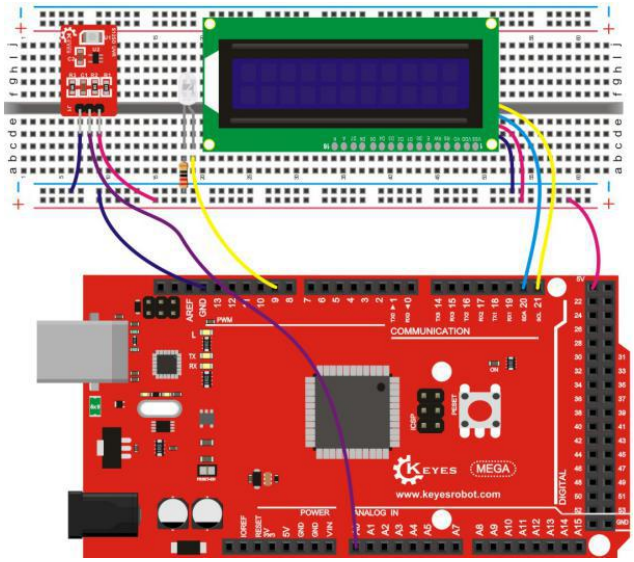
Development Board*1
USB Cable*1
LED*1
220Ω resistor*1
1602 I2C screen *1
GUVA-S12SD 3528 ultraviolet sensor*1
Breadboard*1
Jumper wire *several
Dupont line*several

Wiring Diagram

Connect to Keyes UNO R3



Connect to Keyes 2560 R3



Sample Code

```
//////////////////////////////////////////////////////////
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
int led=9;
void setup()
{
   pinMode(led,OUTPUT);
   lcd.init();                              // initialize the lcd
   lcd.init();
   // Print a message to the LCD.
   lcd.backlight();
   lcd.setCursor(0,0);
   lcd.print("Ultra-Violet ");
   lcd.setCursor(0,1);
   lcd.print("Detection:");
}
void loop()
{
  int sensorValue = analogRead(A0);
   if(sensorValue<10)
   {
   lcd.setCursor(10,1);
   lcd.print(sensorValue);
   lcd.setCursor(11,1);
   lcd.print("          ");
   digitalWrite(led,LOW);
```

```
    }
    if((sensorValue>=10)&&(sensorValue<100))
    {
    lcd.setCursor(10,1);
    lcd.print(sensorValue);
    lcd.setCursor(12,1);
    lcd.print("       ");
    digitalWrite(led,HIGH);
    }
    if( sensorValue>=100)
    {
    lcd.setCursor(10,1);
    lcd.print(sensorValue);
    lcd.setCursor(13,1);
    lcd.print("      ");
    digitalWrite(led,HIGH);
    }
    delay(500);
}
//////////////////////////////////////////////////////////
```

Test Result

Wire it up and upload well the code, the data representing the contents of ultraviolet will be displayed on the 1602 LCD. When the value is smaller than 10, LED off. If greater than or equal to 10, LED on.

6. Related Data Link

https://pan.baidu.com/s/1nu6V4pF