VMA502

BASIC DIY KIT WITH ATMEGA2560 FOR ARDUINO®

USER MANUAL

USER MANUAL

1. Introduction

To all residents of the European Union

Important environmental information about this product

This symbol on the device or the package indicates that disposal of the device after its lifecycle could harm the environment. Do not dispose of the unit (or batteries) as unsorted municipal waste; it should be taken to a specialized company for recycling. This device should be returned to your distributor or to a local recycling service. Respect the local environmental rules.

If in doubt, contact your local waste disposal authorities.

Thank you for choosing Velleman®! Please read the manual thoroughly before bringing this device into service. If the device was damaged in transit, do not install or use it and contact your dealer.

2. Safety Instructions

- This device can be used by children aged from 8 years and above, and persons with reduced physical, sensory or mental capabilities or lack of experience and knowledge if they have been given supervision or instruction concerning the use of the device in a safe way and understand the hazards involved. Children shall not play with the device. Cleaning and user maintenance shall not be made by children without supervision.

- Indoor use only.
  Keep away from rain, moisture, splashing and dripping liquids.

3. General Guidelines

- Refer to the Velleman® Service and Quality Warranty on the last pages of this manual.
- Familiarise yourself with the functions of the device before actually using it.
- All modifications of the device are forbidden for safety reasons. Damage caused by user modifications to the device is not covered by the warranty.
- Only use the device for its intended purpose. Using the device in an unauthorised way will void the warranty.
- Damage caused by disregard of certain guidelines in this manual is not covered by the warranty and the dealer will not accept responsibility for any ensuing defects or problems.
- Nor Velleman nv nor its dealers can be held responsible for any damage (extraordinary, incidental or indirect) – of any nature (financial, physical…) arising from the possession, use or failure of this product.
- Due to constant product improvements, the actual product appearance might differ from the shown images.
- Product images are for illustrative purposes only.
- Do not switch the device on immediately after it has been exposed to changes in temperature. Protect the device against damage by leaving it switched off until it has reached room temperature.
- Keep this manual for future reference.

4. What is Arduino®

Arduino® is an open-source prototyping platform based in easy-to-use hardware and software. Arduino® boards are able to read inputs – light-on sensor, a finger on a button or a Twitter message – and turn it into an output – activating of a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so, you use the Arduino programming language (based on Wiring) and the Arduino® software IDE (based on Processing).

Surf to www.arduino.cc and www.arduino.org for more information.

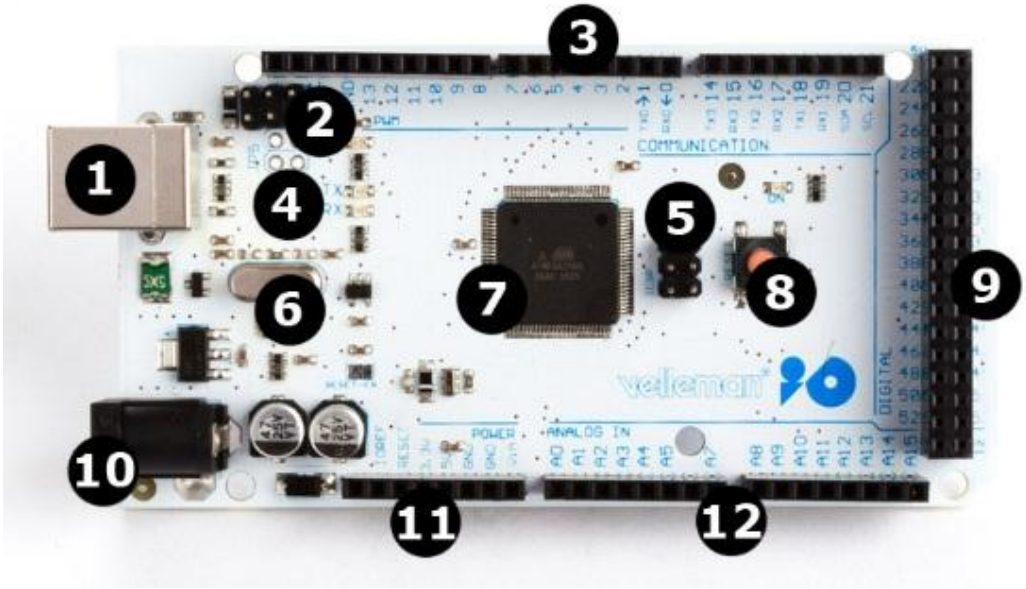
5. Contents

- 1 x ATmega2560 Mega development board (VMA101)
- 15 x LED (different colors)
- 8 x 220 Ω resistor (RA220E0)
- 5 x 1K resistor (RA1K0)
- 5 x 10K resistor (RA10K0)
- 1 x 830-hole breadboard
- 4 x 4-pin key switch
- 1 x active buzzer (VMA319)
- 1 x passive buzzer
- 1 x infrared sensor diode
- 1 x LM35 temperature sensor (LM35DZ)
- 2 x ball tilt switch (similar to MERS4 and MERS5)
- 3 x photosensitive resistor LDR (similar to LDR04)
- 1 x single-digit 7-segment LED display
- 30 x breadboard jumper wire
- 1 x USB cable

6. The ATmega2560 Mega

VMA101

The VMA101 (Arduino® compatible) Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analogue inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller. Connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino® Duemilanove or Diecimila.



1	USB interface
2	ICSP for 16U2
3	digital I/O
4	Atmel mega16U2
5	ICSP for mega2560
6	16 MHz clock

7	Atmel mega2560
8	reset button
9	digital I/O
10	7-12 VDC power input
11	power and ground pins
12	analogue input pins

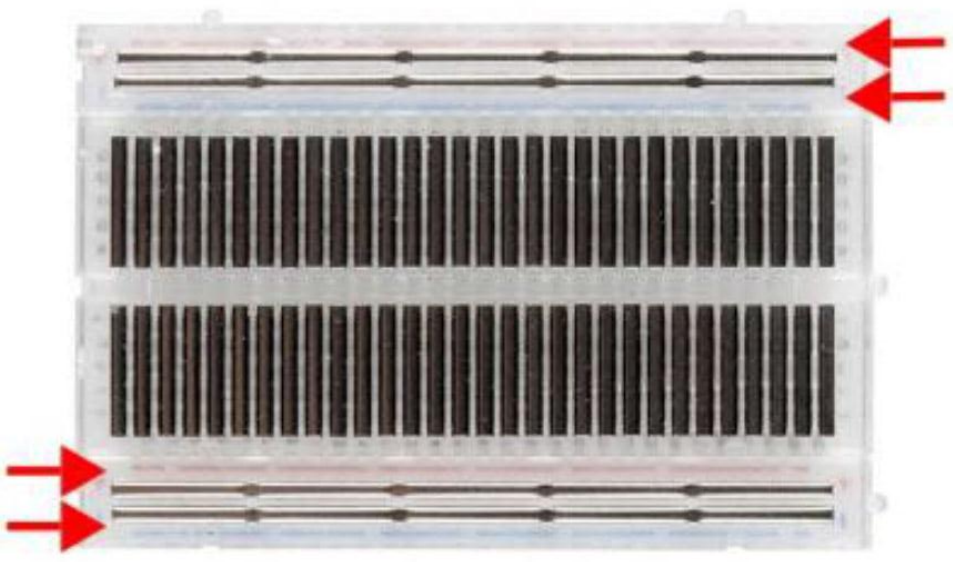
microcontroller ATmega2560
operating voltage ..... 5 VDC
input voltage (recommended) ..... 7-12 VDC
input voltage (limits) ..... 6-20 VDC
digital I/O pins ..... 54 (of which 15 provide PWM output)
analogue input pins 16
DC current per I/O pin ..... 40 mA
DC current for 3.3 V pin ..... 50 mA
flash memory ..... 256 kB of which 8 kB used by bootloader
SRAM ..... 8 kB
EEPROM ..... 4 kB
clock speed ..... 16 MHz
dimensions
length ..... 112 mm
width ..... 55 mm
weight ..... 62 g

7. Operation

7.1 The Breadboard

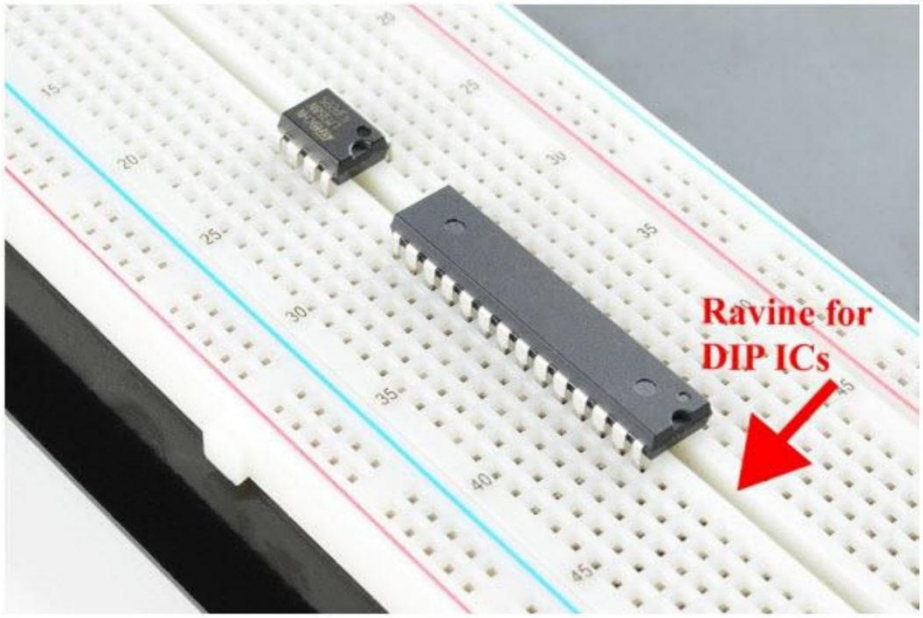
Breadboards are one of the most fundamental pieces when learning how to build circuits. In this tutorial, we will introduce you to what breadboards are and how they work.

Let us look at a larger, more typical breadboard. Aside from the horizontal rows, breadboards have what are called **power rails** that run vertically along the sides.



Power rails.

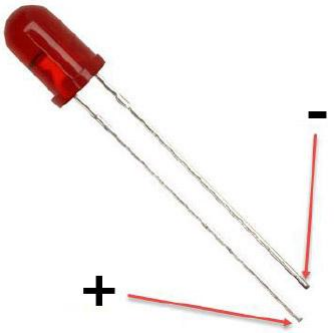
Chips have legs that come out of both sides and fit perfectly over the **ravine**. Since each leg on the IC is unique, we do not want both sides to be connected to each other. That is where the separation in the middle of the board comes in handy. Thus, we can connect components to each side of the IC without interfering with the functionality of the leg on the opposite side.



Ravine.

7.2 A Blinking LED

Let's start with a simple experiment. We are going to connect an LED to one of the digital pins rather than using LED13, which is soldered to the board.

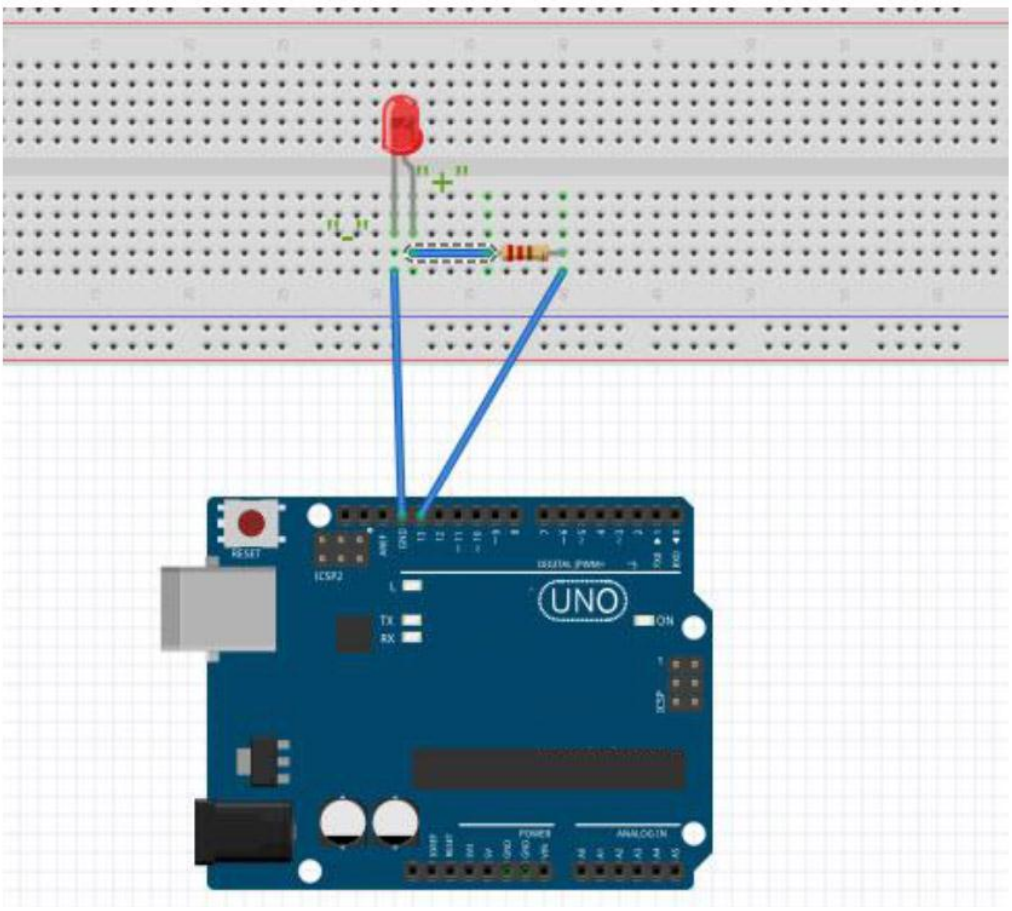


Required Hardware

- 1 x red M5 LED
- 1 x 220 Ω resistor
- 1 x breadboard
- jumper wires as needed

Follow the diagram below. We are using digital pin 10, and connecting the LED to a 220 Ω resistor to avoid high-current damaging the LED.

Connection



Programming Code

```
*******code begin*******
int ledPin = 10; // define digital pin 10.
void setup()
{
pinMode(ledPin, OUTPUT);// define pin with LED connected as output.
}
void loop()
{
digitalWrite(ledPin, HIGH); // set the LED on.
delay(1000); // wait for a second.
digitalWrite(ledPin, LOW); // set the LED off.
delay(1000); // wait for a second
}
*******code end*******
```

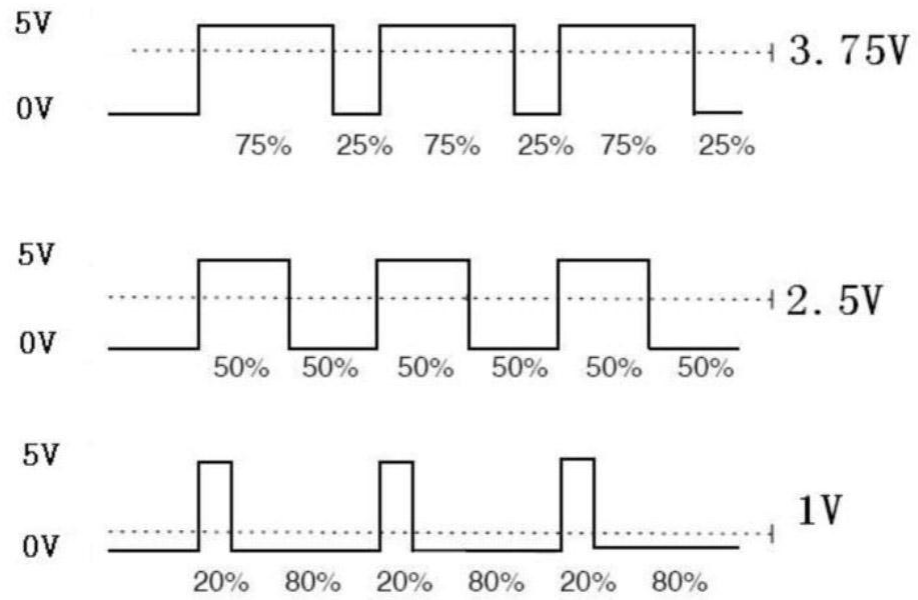
Result

After programming, you will see the LED connected to pin 10 blinking, with an interval of approximately one second. Congratulations, the experiment is now successfully completed!

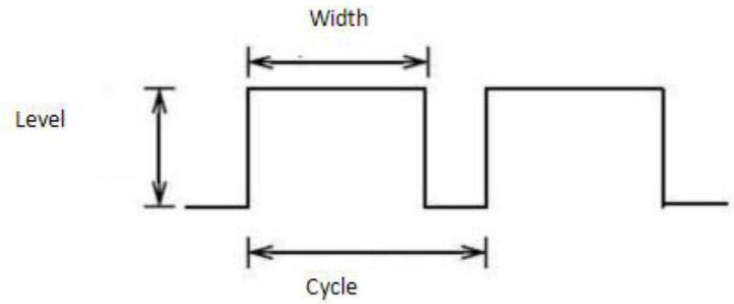
7.3 PWM Gradational LED

PWM (Pulse Width Modulation) is a technique used to encode analogue signal levels into digital ones. A computer cannot output analogue voltage but only digital voltage values. So, we will be using a high-resolution counter to encode a specific analogue signal level by modulating the duty cycle of PWM. The PWM signal is also digitalized because in any given moment, fully on DC power is either 5 V (on) of 0 V (off). The voltage or current is fed to the analogue load (the device using the power) by repeated pulse sequence being on or off. Being on, the current is fed to the load; being off, it is not. With the adequate bandwidth, any analogue value can be encoded using PWM. The output voltage value is calculated via the on and off time.

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



PWM has many applications: lamp brightness regulation, motor speed regulation, sound making, etc. The following are the basic parameters of PWM:

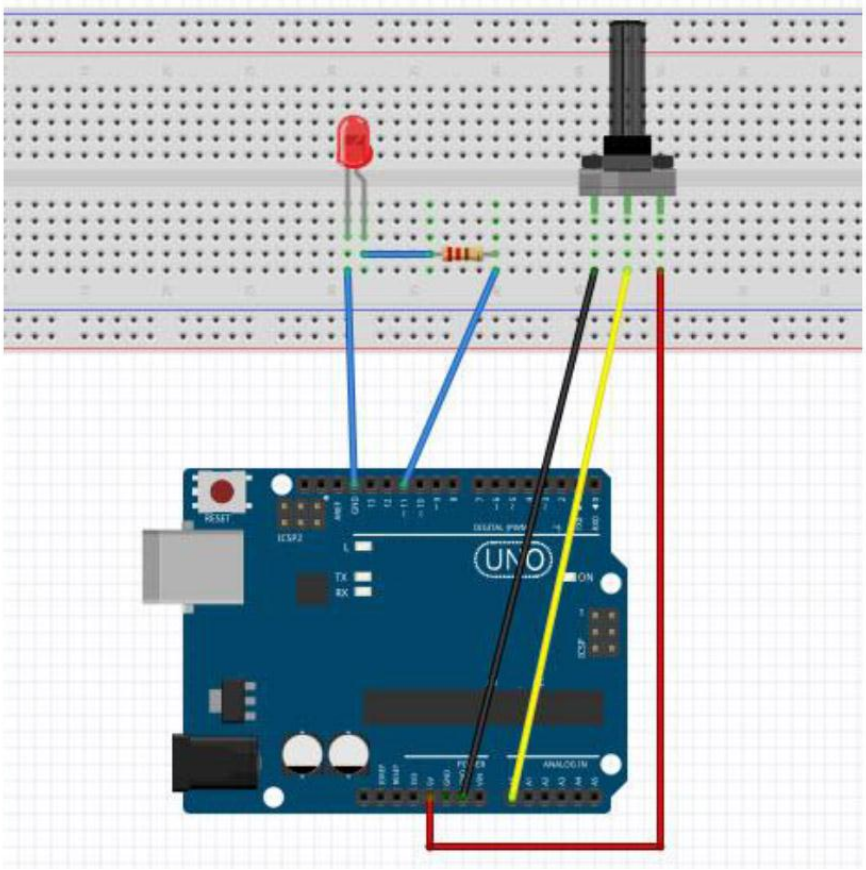


There are six PQM interfaces on Arduino®, namely digital pin, 3, 5, 6, 9, 10 and 11. In this experiment, we will be using a potentiometer to control the LED brightness.

Required Hardware

- 1 x variable resistor
- 1 x red M5 LED
- 1 x 220 Ω resistor
- 1 x breadboard
- jumper wires as needed

Connection



Programming Code

```
*******code begin*******
int potpin=0;// initialize analog pin 0
int ledpin=11;//initialize digital pin 11（PWM output）
int val=0;// Temporarily store variables' value from the sensor
void setup()
{
pinMode(ledpin,OUTPUT);// define digital pin 11 as “output”
Serial.begin(9600);// set baud rate at 9600
// attention: for analog ports, they are automatically set up as “input”
}
void loop()
{
val=analogRead(potpin);// read the analog value from the sensor and assign it to val
Serial.println(val);// display value of val
analogWrite(ledpin,val/4);// turn on LED and set up brightness（maximum output of
PWM is 255）
delay(10);// wait for 0.01 second
}
*******code end*******
```

In this code, we are using the analogWrite (PWM interface, analogue value) function. We will read the analogue value of the potentiometer and assign the value to PWM port, so there will be corresponding change to the brightness of the LED. One final part will be displaying the analogue value on the screen. You can consider this as the **analogue value reading** project adding the PWM analogue value assigning part.

Result

After programming, rotate the potentiometer knob to see changes of the displaying value. Also, note the obvious change of brightness on the breadboard.

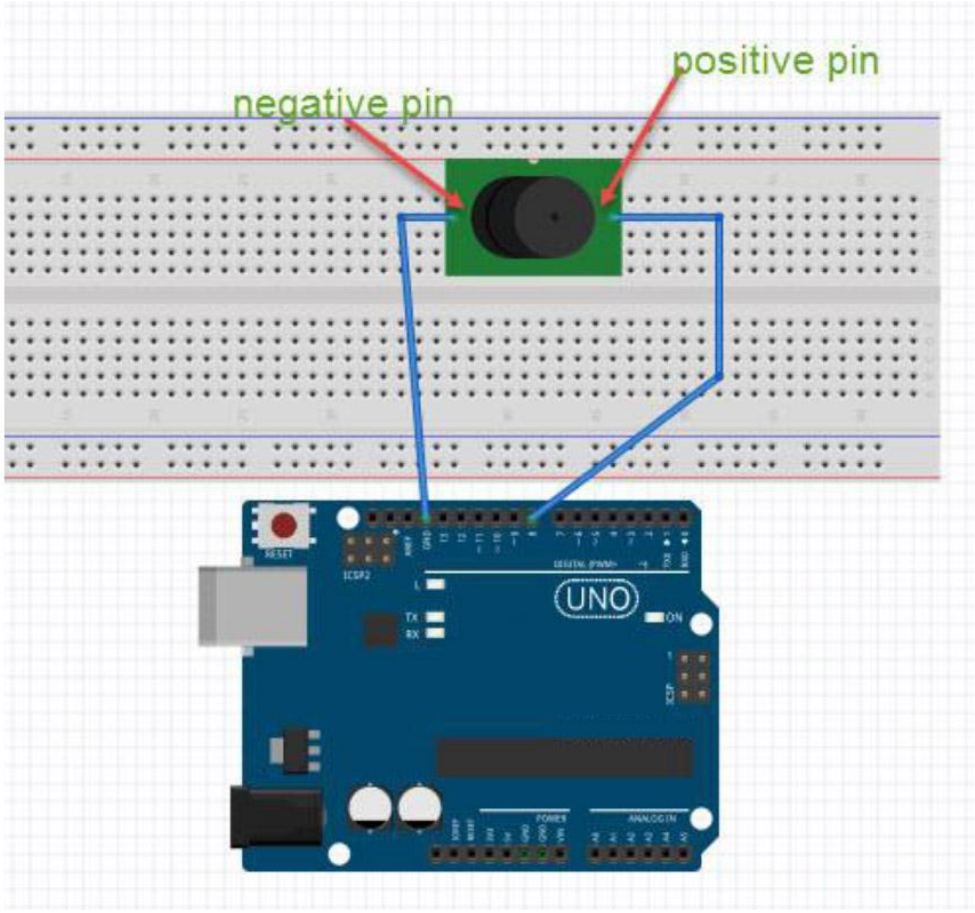
7.4 The Active Buzzer

An active buzzer is widely used on computers, printers, alarms, etc. as a sound-making element. It has an inner vibration source. Simply connect it with a 5 V power supply to make it buzz constantly.

Required Hardware

- 1 x buzzer
- 1 x key
- 1 x breadboard
- jumper wires as needed

Connection



Programming Code

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

Result

After programming, the buzzer should ring.

7.5 The Photosensitive Resistor

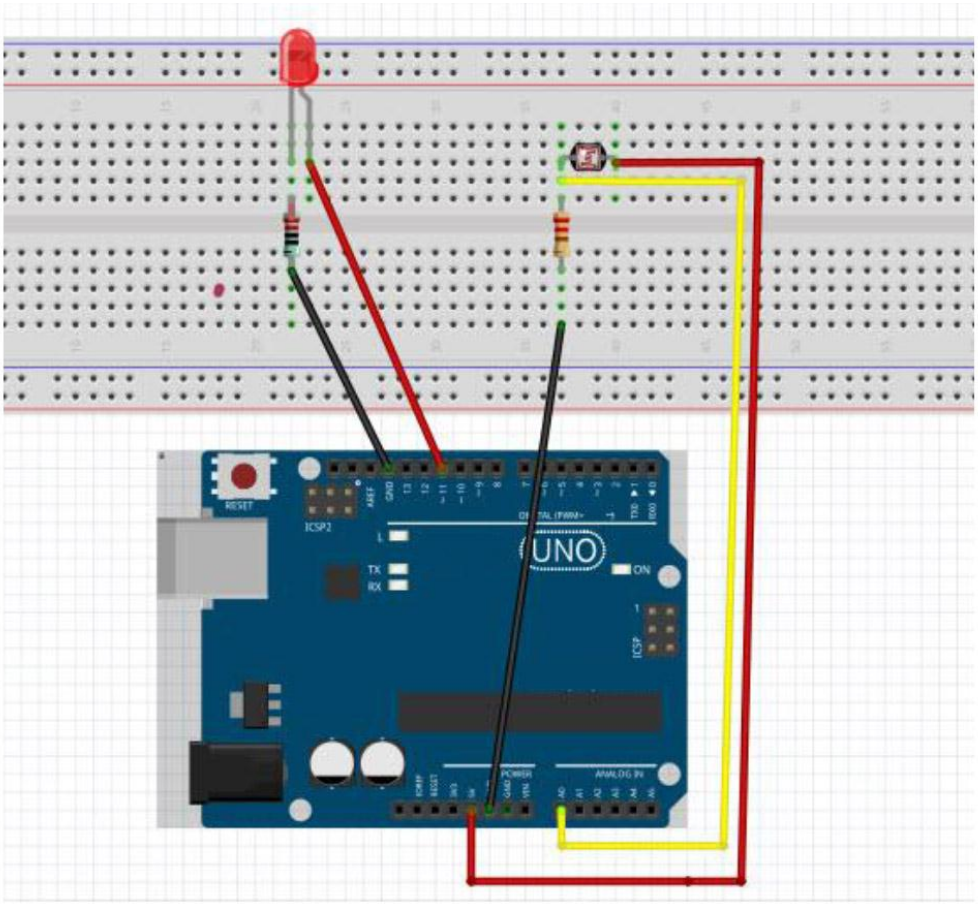
A photoresistor is a resistor whose resistance varies according to different light strengths. It is based on the photoelectric effect of a semiconductor. If the incident light is intense, the resistance reduces; if the incident light is weak, the resistant increases. A photovaristor is commonly applied in the measurement of light, light control and photovoltaic conversion.

Let’s start with a relative simple experiment. The photovaristor is an element that changes its resistance as light strength changes. Refer to the PWM experiment, replacing the potentiometer with a photovaristor. When there is a change in light strength, there will be a corresponding change on the LED.

Required Hardware

- 1 x photoresistor
- 1 x red M5 LED
- 1 x 10KΩ resistor
- 1 x 220 Ω resistor
- 1 x breadboard
- jumper wires as needed

Connection



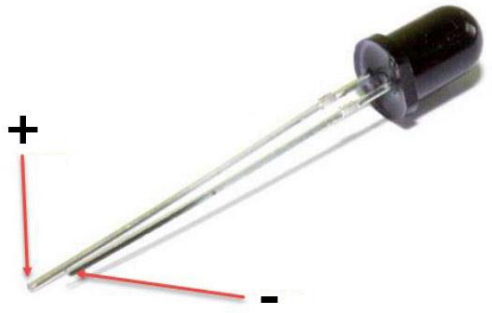
Programming Code

```
*******code begin*******

//////////////////////////////////////////////////////////
int potpin=0;// initialize analog pin 0, connected with photovaristor
int ledpin=11;// initialize digital pin 11, output regulating the brightness of LED
int val=0;// initialize variable va
void setup()
{
pinMode(ledpin,OUTPUT);// set digital pin 11 as “output”
Serial.begin(9600);// set baud rate at “9600”
}
void loop()
{
val=analogRead(potpin);// read the analog value of the sensor and assign it to val
Serial.println(val);// display the value of val
analogWrite(ledpin,val);// turn on the LED and set up brightness（maximum output
value 255）
delay(10);// wait for 0.01
}
//////////////////////////////////////////////////////////
*******code End*******
```

Result

After programming, change the light strength around the photovaristor and observe the LED changing!

7.6 The Flame Sensor



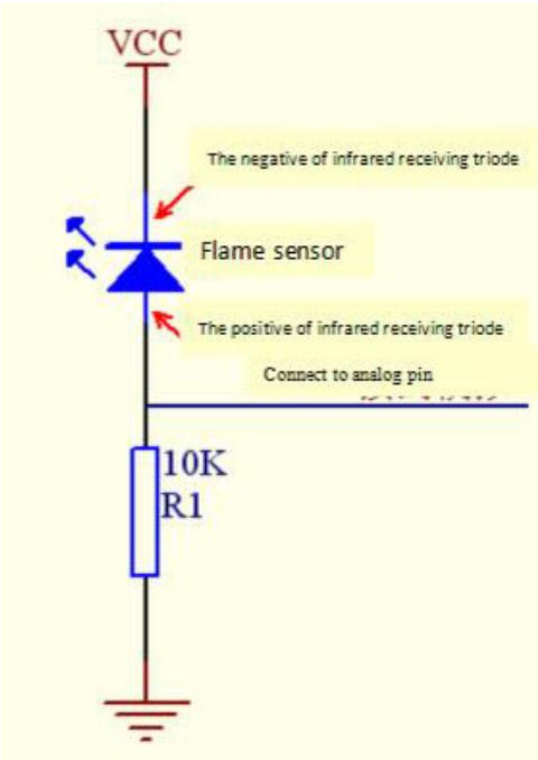
A flame sensor (IR receiving diode) is specifically used on robots to find the fire source. This sensor is highly sensitive to flames.

A flame sensor has a specifically designed IR tube to detect fire. The brightness of the flames will then be converted to a fluctuating level signal. The signals are the input into the central processor.

Required Hardware

- 1 x flame sensor
- 1 x buzzer
- 1 x 10KΩ resistor
- 1 x breadboard
- jumper wires as needed

Connection

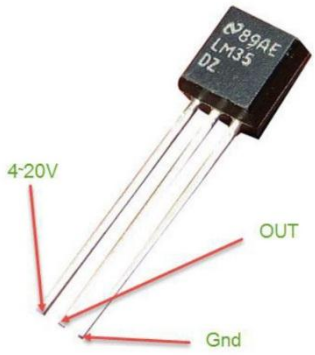


Connect the negative to the 5 V pin and the positive to the resistor. Connect the other end of the resistor to GND. Connect one end of a jumper wire to a clip, which is electrically connected to sensor positive, the other end to the analogue pin.

Programming Code

```
*******code End*******
int flame=0;// select analog pin 0 for the sensor
int Beep=9;// select digital pin 9 for the buzzer
int val=0;// initialize variable
void setup()
{
  pinMode(Beep,OUTPUT);// set LED pin as “output”
pinMode(flame,INPUT);// set buzzer pin as “input”
Serial.begin(9600);// set baud rate at “9600”
}
void loop()
{
  val=analogRead(flame);// read the analog value of the sensor
  Serial.println(val);// output and display the analog value
  if(val>=600)// when the analog value is larger than 600, the buzzer will buzz
  {
    digitalWrite(Beep,HIGH);
    }else
    {
       digitalWrite(Beep,LOW);
     }
    delay(500);
}
*******code End*******
```

7.7 The LM35 Temperature Sensor

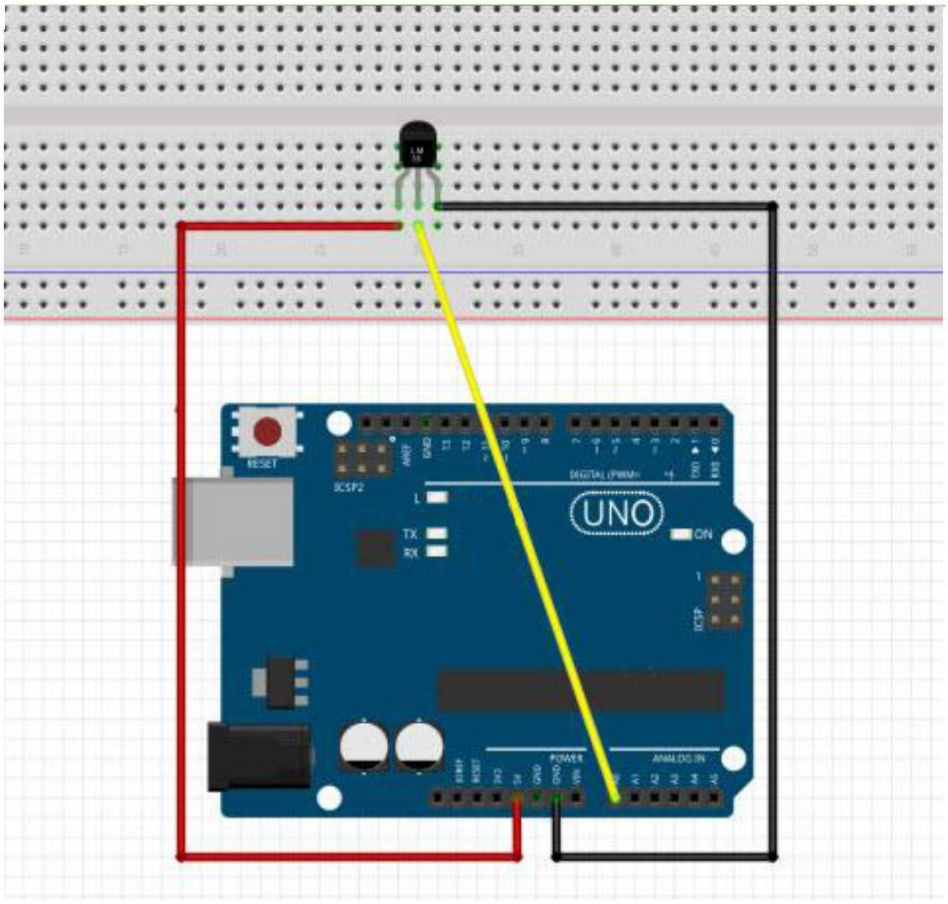


The LM35 is a common and easy-to-use temperature sensor. It does not require other hardware, you just need an analogue port to make it work. The difficulty lies in compiling the code to convert the analogue value it reads to Celsius temperature.

Required Hardware

- 1 x LM35 sensor
- 1 x breadboard
- jumper wires as needed

Connection



Programming Code

```
*******code begin*******
int potPin = 0; // initialize analog pin 0 for LM35 temperature sensor
void setup()
{
Serial.begin(9600);// set baud rate at”9600”
}
void loop()
{
int val;// define variable
int dat;// define variable
val=analogRead(0);// read the analog value of the sensor and assign it to val
dat=(125*val)>>8;// temperature calculation formula
Serial.print("Tep:");// output and display characters beginning with Tep
Serial.print(dat);// output and display value of dat
Serial.println("C");// display “C” characters
delay(500);// wait for 0.5 second

 }
 *******code End*******
```

Result

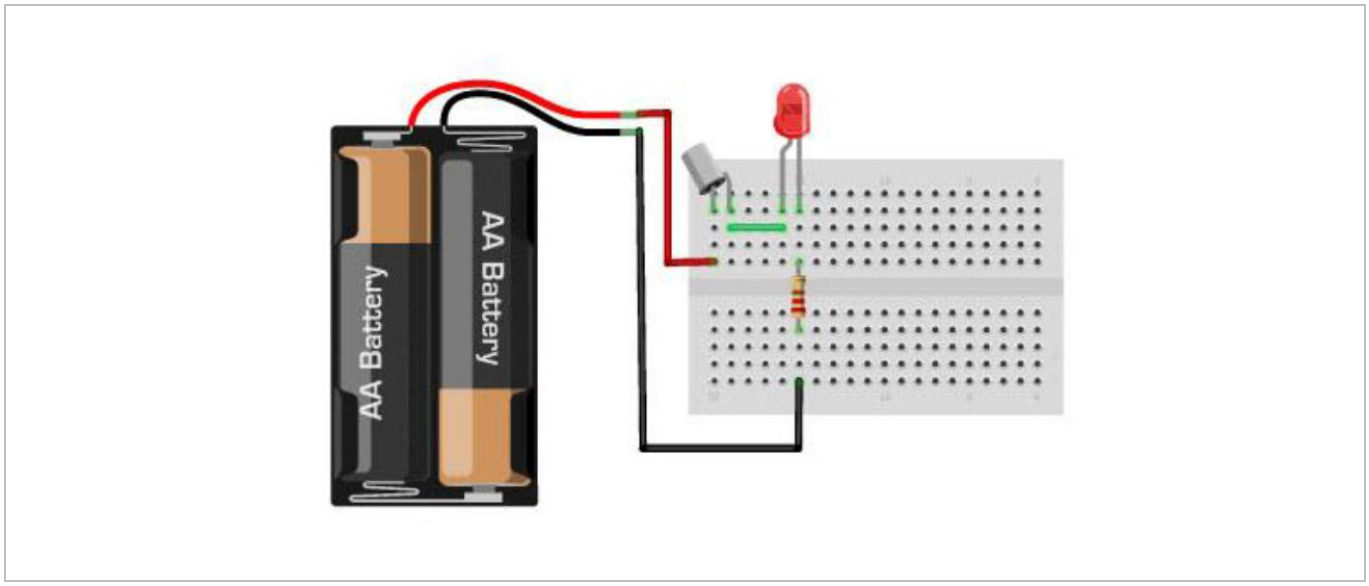
After programming, open the monitoring window to see the current temperature.

7.8 The Tilt Sensor Switch

A tilt sensor will detect orientation and inclination. They are small, low power and easy-to-use. If used properly, they will not wear out. Their simplicity makes them popular for toys, gadgets and other appliances. They are referred to as **mercury**, **tilt** or **rolling ball** switches.

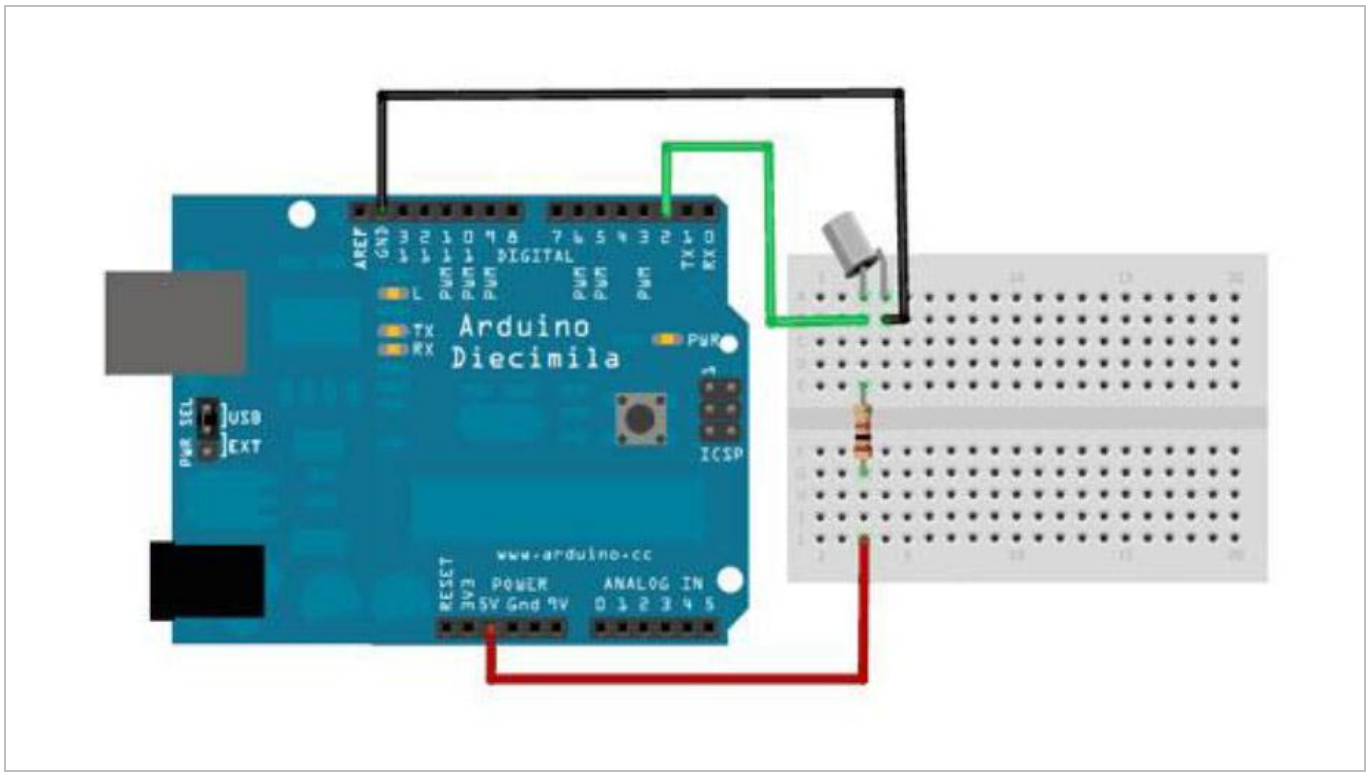
The Simple Tilt-Activated LED

This is the most basic connection of a tilt switch, but can be a handy while one is learning about them. Simply connect in series with an LED, resistor and battery.



Reading the Switch State with a Microcontroller

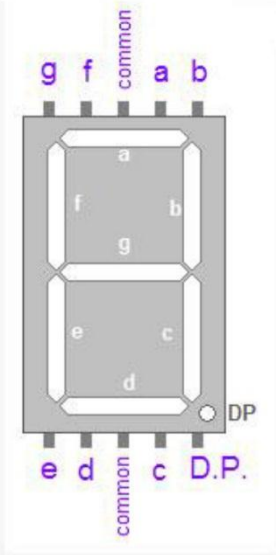
The layout below shows a 10K pull-up resistor. The code states the built-in pull-up resistor that you can turn on by setting an input pin to the high output. If you use the internal pull-up you can skip the external one.



Programming Code

```
*******code begin*******
int inPin = 2;               // the number of the input pin
int outPin = 13;             // the number of the output pin

int LEDstate = HIGH;         // the current state of the output pin

int reading;                 // the current reading from the input pin
int previous = LOW;          // the previous reading from the input pin

// the follow variables are long's because the time, measured in miliseconds,
// will quickly become a bigger number than can be stored in an int.
long time = 0;               // the last time the output pin was toggled
long debounce = 50;          // the debounce time, increase if the output flickers

void setup()
{
  pinMode(inPin, INPUT);
  digitalWrite(inPin, HIGH);     // turn on the built in pull-up resistor
  pinMode(outPin, OUTPUT);
}

void loop()
{
  int switchstate;

  reading = digitalRead(inPin);

 // If the switch changed, due to bounce or pressing...
 if (reading != previous) {
    // reset the debouncing timer
    time = millis();
 }

 if ((millis() - time) > debounce) {
     // whatever the switch is at, its been there for a long time
     // so lets settle on it!
     switchstate = reading;

     // Now invert the output on the pin13 LED
    if (switchstate == HIGH)
       LEDstate = LOW;
    else
       LEDstate = HIGH;
 }
 digitalWrite(outPin, LEDstate);

 // Save the last reading so we keep a running tally
 previous = reading;
}
*******code End*******
```

7.9 One-Digit Seven-Segment Display

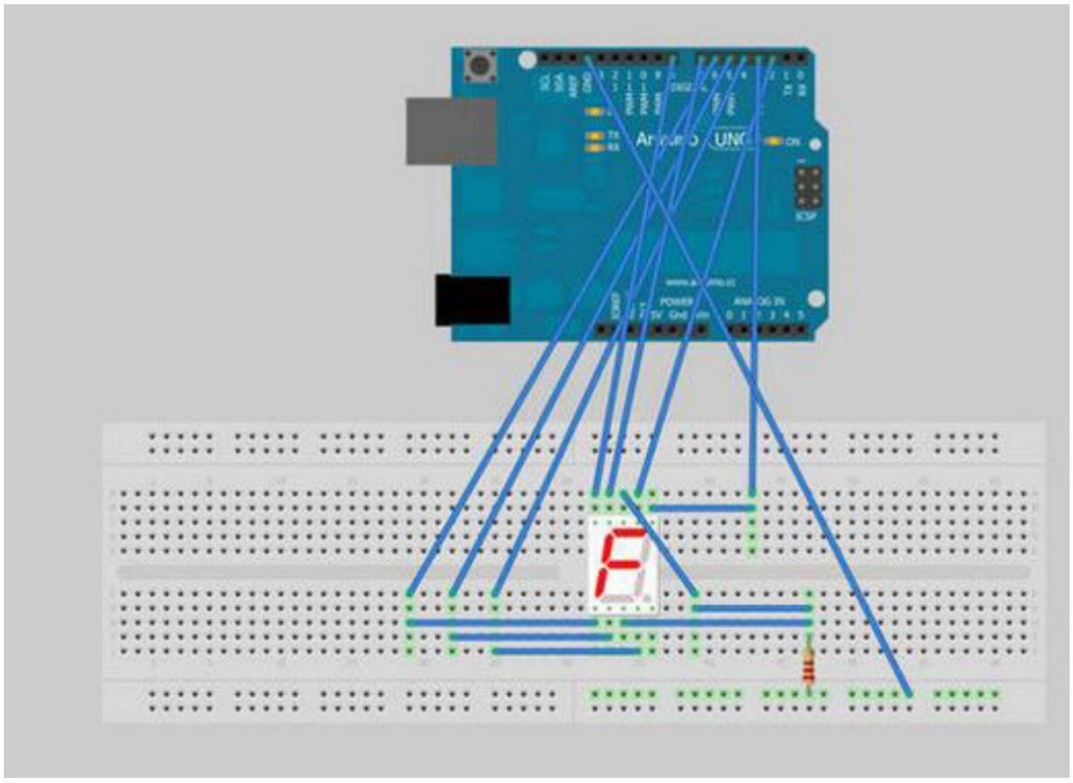


LED segment displays are common for displaying numerical information. They are widely applied on displays of ovens, washing machines, etc. the LED segment display is a semiconductor light-emitting device. Its basic unit is an LED (light-emitting diode). Segment displays can be divided into 7-segment and 8-segment displays.

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

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

Connection



Programming Code

```
*******code begin*******
// Define the LED digit patters, from 0 - 9
// Note that these patterns are for common cathode displays
// For common anode displays, change the 1's to 0's and 0's to 1's
// 1 = LED on, 0 = LED off, in this order:
//                                                  Arduino pin: 2,3,4,5,6,7,8
byte seven_seg_digits[10][7] = { { 1,1,1,1,1,1,0 },    // = 0

{ 0,1,1,0,0,0,0 },    // = 1

{ 1,1,0,1,1,0,1 },    // = 2

{ 1,1,1,1,0,0,1 },    // = 3

{ 0,1,1,0,0,1,1 },    // = 4

{ 1,0,1,1,0,1,1 },    // = 5

{ 1,0,1,1,1,1,1 },    // = 6

{ 1,1,1,0,0,0,0 },    // = 7

{ 1,1,1,1,1,1,1 },    // = 8

{ 1,1,1,0,0,1,1 }     // = 9
                                                                        };

void setup() {
   pinMode(2, OUTPUT);
   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
   pinMode(6, OUTPUT);
   pinMode(7, OUTPUT);
   pinMode(8, OUTPUT);
   pinMode(9, OUTPUT);
   writeDot(0);   // start with the "dot" off
}

void writeDot(byte dot) {
   digitalWrite(9, dot);
}
void sevenSegWrite(byte digit) {
   byte pin = 2;
   for (byte segCount = 0; segCount < 7; ++segCount) {
      digitalWrite(pin, seven_seg_digits[digit][segCount]);
      ++pin;
   }
}

void loop() {
   for (byte count = 10; count > 0; --count) {
     delay(1000);
     sevenSegWrite(count - 1);
   }
   delay(4000);
}
*******code End*******
```

Use this device with original accessories only. Velleman nv cannot be held responsible in the event of damage or injury resulting from (incorrect) use of this device. For more info concerning this product and the latest version of this manual, please visit our website www.velleman.eu. The information in this manual is subject to change without prior notice.

Velleman® Service and Quality Warranty

Since its foundation in 1972, Velleman® acquired extensive experience in the electronics world and currently distributes its products in over 85 countries.

All our products fulfil strict quality requirements and legal stipulations in the EU. In order to ensure the quality, our products regularly go through an extra quality check, both by an internal quality department and by specialized external organisations. If, all precautionary measures notwithstanding, problems should occur, please make appeal to our warranty (see guarantee conditions).

General Warranty Conditions Concerning Consumer Products (for EU):

• All consumer products are subject to a 24-month warranty on production flaws and defective material as from the original date of purchase.

• Velleman® can decide to replace an article with an equivalent article, or to refund the retail value totally or partially when the complaint is valid and a free repair or replacement of the article is impossible, or if the expenses are out of proportion.

You will be delivered a replacing article or a refund at the value of 100% of the purchase price in case of a flaw occurred in the first year after the date of purchase and delivery, or a replacing article at 50% of the purchase price or a refund at the value of 50% of the retail value in case of a flaw occurred in the second year after the date of purchase and delivery.

• Not covered by warranty:

- all direct or indirect damage caused after delivery to the article (e.g. by oxidation, shocks, falls, dust, dirt, humidity...), and by the article, as well as its contents (e.g. data loss), compensation for loss of profits;

- consumable goods, parts or accessories that are subject to an aging process during normal use, such as batteries (rechargeable, non-rechargeable, built-in or replaceable), lamps, rubber parts, drive belts... (unlimited list);

- flaws resulting from fire, water damage, lightning, accident, natural disaster, etc.…;

- flaws caused deliberately, negligently or resulting from improper handling, negligent maintenance, abusive use or use contrary to the manufacturer's instructions;

- damage caused by a commercial, professional or collective use of the article (the warranty validity will be reduced to six (6) months when the article is used professionally);

- damage resulting from an inappropriate packing and shipping of the article;

- all damage caused by modification, repair or alteration performed by a third party without written permission by Velleman®.

• Articles to be repaired must be delivered to your Velleman® dealer, solidly packed (preferably in the original packaging), and be completed with the original receipt of purchase and a clear flaw description.

• Hint: In order to save on cost and time, please reread the manual and check if the flaw is caused by obvious causes prior to presenting the article for repair. Note that returning a non-defective article can also involve handling costs.

• Repairs occurring after warranty expiration are subject to shipping costs.

• The above conditions are without prejudice to all commercial warranties.

The above enumeration is subject to modification according to the article (see article's manual).

Made in PRC
Imported by Velleman nv
Legen Heirweg 33, 9890 Gavere, Belgium
www.velleman.eu