
SunFounder vincent_kit_for_arduino

www.sunfounder.com

Oct 14, 2022

CONTENTS

1	Components List and Introduction	3
1.1	SunFounder Mega Board	3
1.2	Prototype Shield	5
1.3	Breadboard	6
1.4	Resistor	7
1.5	Transistor	9
1.6	Capacitor	12
1.7	Diode	14
1.8	Jumper Wires	15
1.9	74HC595	15
1.10	L293D	17
1.11	LED	18
1.12	RGB LED	19
1.13	LED Bar Graph	21
1.14	7-segment Display	22
1.15	4-Digit 7-Segment Display	25
1.16	LED Matrix Module	26
1.17	I2C LCD1602	28
1.18	Buzzer	30
1.19	DC Motor	31
1.20	Servo	33
1.21	Stepper Motor	34
1.22	Power Supply Module	38
1.23	Relay Module	39
1.24	Button	42
1.25	Slide Switch	43
1.26	Potentiometer	44
1.27	Joystick Module	46
1.28	Rotary Encoder Module	48
1.29	Keypad	50
1.30	IR Receiver Module	50
1.31	MPR121	52
1.32	Photoresistor	53
1.33	Thermistor	54
1.34	Tilt Switch	56
1.35	Water Level Sensor Module	57
1.36	Sound Sensor Module	58
1.37	Touch Switch Module	59
1.38	Obstacle Avoidance Module	60
1.39	PIR Motion Sensor Module	61

1.40	Ultrasonic Module	63
1.41	Humiture Sensor Module	64
1.42	MPU6050 Module	66
1.43	MFRC522 Module	68
2	Play with Arduino	69
2.1	1.1 Get Started with Arduino IDE	69
2.2	1.2 Digital Write	81
2.3	1.3 Analog Write	85
2.4	1.4 Digital Read	90
2.5	1.5 Analog Read	95
2.6	1.6 Digital Input Control Output	98
2.7	1.7 Analog Input Control Output	102
2.8	1.8 Serial Read	106
2.9	1.9 Digital Input Pull-Up	112
2.10	1.10 State Change Detection	115
2.11	1.11 Interval	120
2.12	2.2 LED	125
2.13	2.3 RGB LED	128
2.14	2.4 LED Bar Graph	132
2.15	2.5 7-Segment Display	137
2.16	2.6 74HC595	143
2.17	2.7 4-Digital 7-Segment Display	147
2.18	2.8 LED Matrix Module	153
2.19	2.9 I2C LCD1602 Module	157
2.20	2.10 Active Buzzer	161
2.21	2.11 Passive Buzzer	165
2.22	2.12 Servo	172
2.23	2.13 Motor	176
2.24	2.14 Stepper Motor	180
2.25	2.15 Button	184
2.26	2.16 Slide Switch	188
2.27	2.17 Tilt Switch	192
2.28	2.18 Touch Switch Module	196
2.29	2.19 Keypad	199
2.30	2.20 IR Receiver Module	204
2.31	2.21 Relay Module	208
2.32	2.22 Potentiometer	210
2.33	2.23 Joystick Module	214
2.34	2.24 MPR121 Module	217
2.35	2.25 Rotary Encoder Module	222
2.36	2.26 Photoresistor	226
2.37	2.27 Thermistor	230
2.38	2.28 Sound Sensor Module	234
2.39	2.29 Water Sensor Module	237
2.40	2.30 IR Obstacle Avoidance Sensor	240
2.41	2.31 PIR Module	243
2.42	2.32 DHT11 Module	246
2.43	2.33 Ultrasonic Module	250
2.44	2.34 MPU6050 Module	254
2.45	2.35 RFID-RC522 Module	258
2.46	3.1 Reversing Aid	262
2.47	3.2 Pedestrian Crossing Button	267
2.48	3.3 Overheat Monitor	273

2.49	3.4	Guess Number	278
2.50	3.5	Access Control System	282
3		Play with Scratch	289
3.1	1.1	Install PictoBlox	290
3.2	1.2	Interface Introduction	291
3.3	1.3	Quick Guide on PictoBlox	292
3.4	2.1	Table Lamp	308
3.5	2.2	Breathing LED	314
3.6	2.3	Colorful Balls	320
3.7	2.4	LCD1602	329
3.8	2.5	Moving Mouse	337
3.9	2.6	Doorbell	343
3.10	2.7	Tumbler	349
3.11	2.8	Low Temperature Alarm	355
3.12	2.9	Light Alarm Clock	360
3.13	2.10	Read Temperature and Humidity	367
3.14	2.11	Pendulum	372
3.15	2.12	Rotating Fan	380
3.16	2.13	Blow Ball	389
3.17	2.14	GAME - Shooting	401
3.18	2.15	GAME - Inflating the Balloon	414
3.19	2.16	GAME - Star-Crossed	422
3.20	2.17	GAME - Eat Apple	430
3.21	2.18	GAME - Flappy Parrot	441
3.22	2.19	GAME - Breakout Clone	450
3.23	2.20	GAME - Fishing	461
3.24	2.21	Catching Starfish	470
3.25	2.22	GAME - Kill Dragon	480
4		Thank You	503
5		German Online-Tutorials	505
6		Copyright Notice	507

This is a kit suitable for beginners and knowledgeable Arduino learners. The kit uses Mega2560 as the main control board, containing 64 commonly used input and output components, modules and so many basic components (such as resistors, capacitors, transistors, etc.), and it can provide powerful help for you to finish your programming projects.

Download the relevant code from the link below.

- [SunFounder Vincent Kit for Arduino](#)
- Or check out the code at [Vincent Kit for Arduino - GitHub](#)

If you want to learn another projects which we don't have, please feel free to send Email and we will update to our online tutorials as soon as possible, any suggestions are welcomed.

Here is the Email: cs@sunfounder.com.

COMPONENTS LIST AND INTRODUCTION

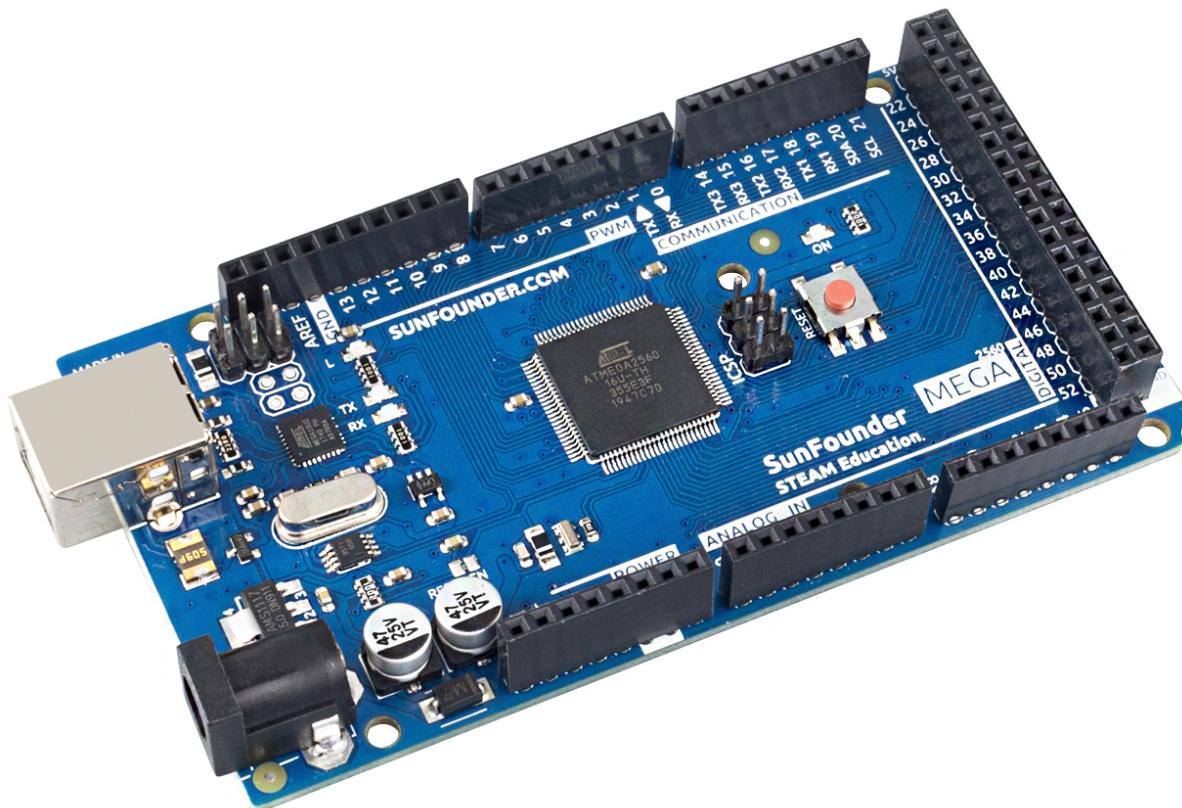
After opening the package, please check whether the quantity of components is compliance with product description and whether all components are in good condition.

- Vincent Kit Components List

Below is the introduction to each component, which contains the operating principle of the component and the corresponding projects.

Control Board

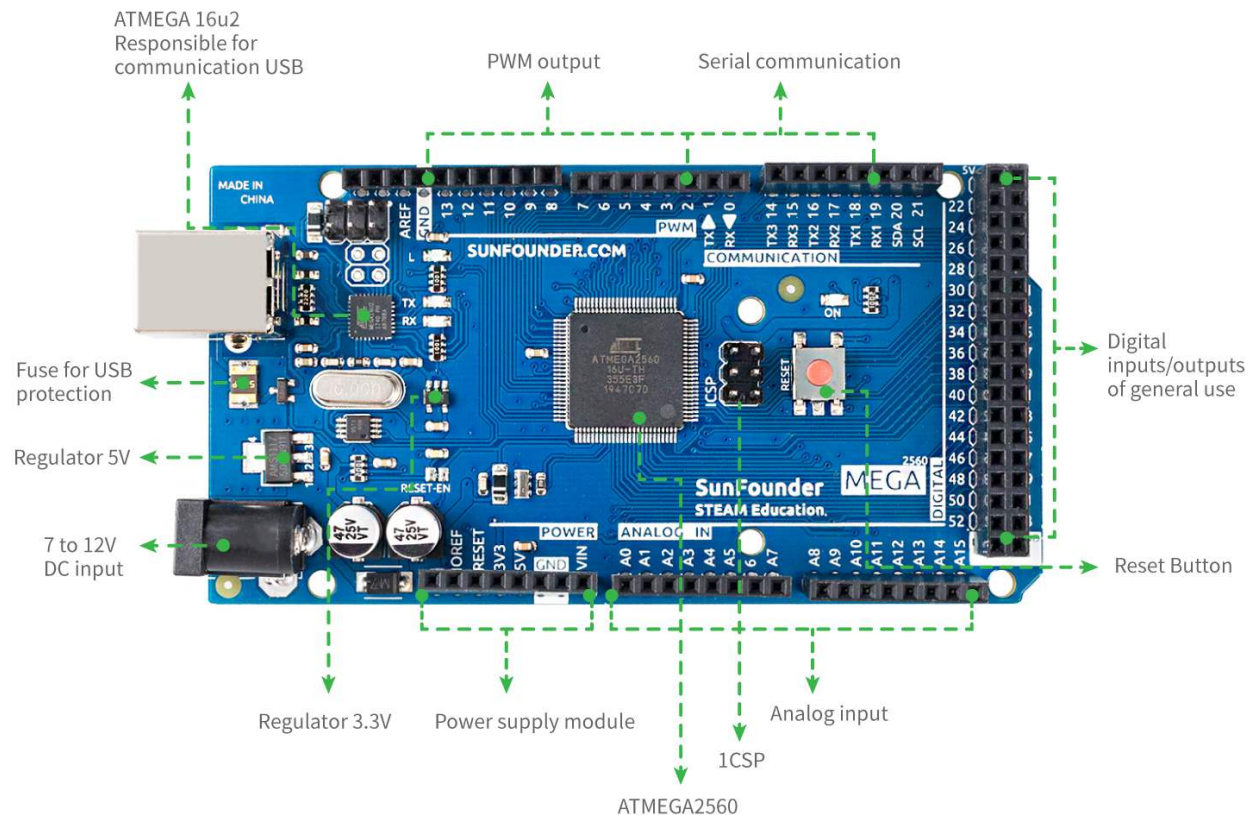
1.1 SunFounder Mega Board



Note: The SunFounder Mega board is a mainboard with almost the same functions as the [Arduino Mega 2560 Rev3](#), and the two boards can be used interchangeably.

The SunFounder Mega Board is a microcontroller board based on the ATmega2560 ([datasheet](#)). It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The SunFounder Mega Board board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

Technical Parameters



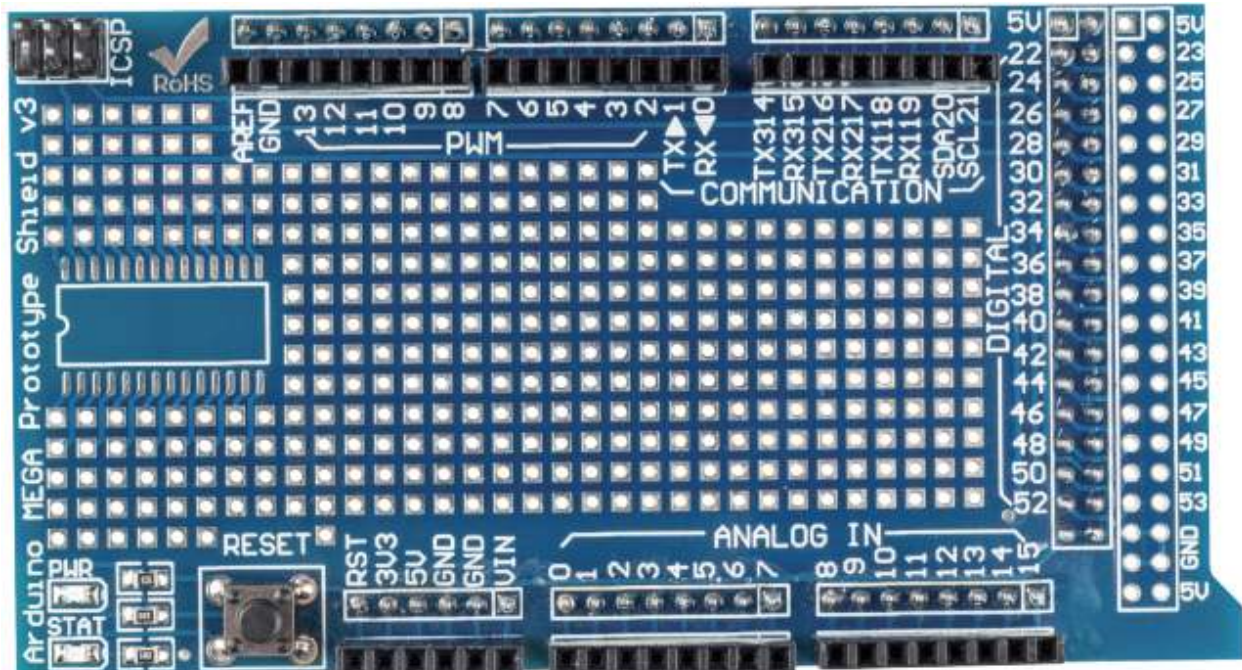
- MICROCONTROLLER: ATmega2560
- OPERATING VOLTAGE: 5V
- INPUT VOLTAGE (RECOMMENDED): 7-12V
- INPUT VOLTAGE (LIMIT): 6-20V
- DIGITAL I/O PINS: 54 (0-53, of which 15 provide PWM output(2-13, 44-46))
- ANALOG INPUT PINS: 16 (A0-A15)
- DC CURRENT PER I/O PIN: 20 mA
- DC CURRENT FOR 3.3V PIN: 50 mA
- FLASH MEMORY: 256 KB of which 8 KB used by bootloader
- SRAM: 8 KB

- EEPROM: 4 KB
- CLOCK SPEED: 16 MHz
- LED_BUILTIN: 13
- LENGTH: 101.52 mm
- WIDTH: 53.3 mm
- WEIGHT: 37 g
- I2C Port: A4(SDA), A5(SCL); 20(SDA), 21(SCL)

What's More

- Arduino IDE
- Arduino Programming Language Reference
- *Install and Introduce Arduino IDE*
- ATmega2560 Datasheet

1.2 Prototype Shield



The Prototype Shield makes it easy for you to design custom circuits. You can solder parts to the prototyping area to create your project, or use it with a Tiny breadboard to quickly test circuit ideas without having to solder. It's got extra connections for all of the Arduino I/O pins, and it's got space to mount through-hole and surface mount integrated circuits. It's a convenient way to make your custom circuit and Arduino into a single module.

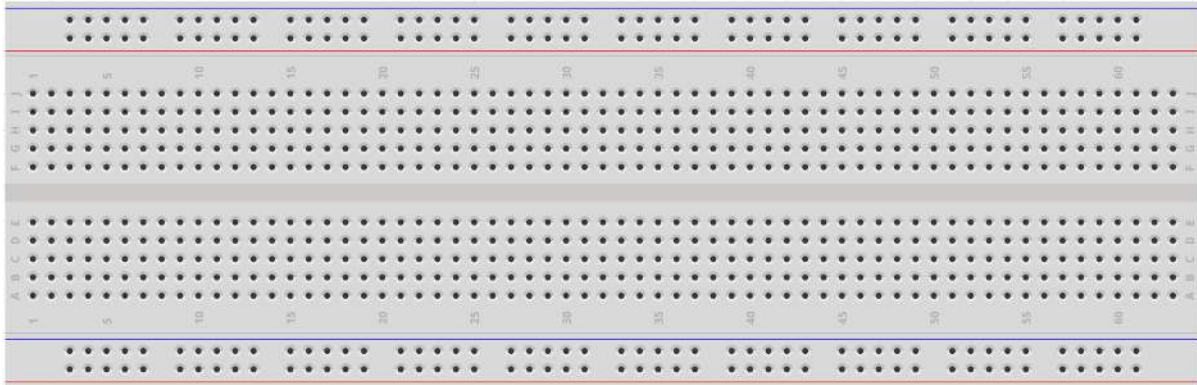
Specifications

- It can be overlaid on the Sunfounder Mega2560 board directly. A Tiny breadboard is provided, which you can use to do some simple experiments.
- Provide a footprint for SOP28.

- It's got extra connections for all of the Arduino I/O pins
- We offer you pin22-pin53 bonding pad and you can use it to weld the component directly.

Basic

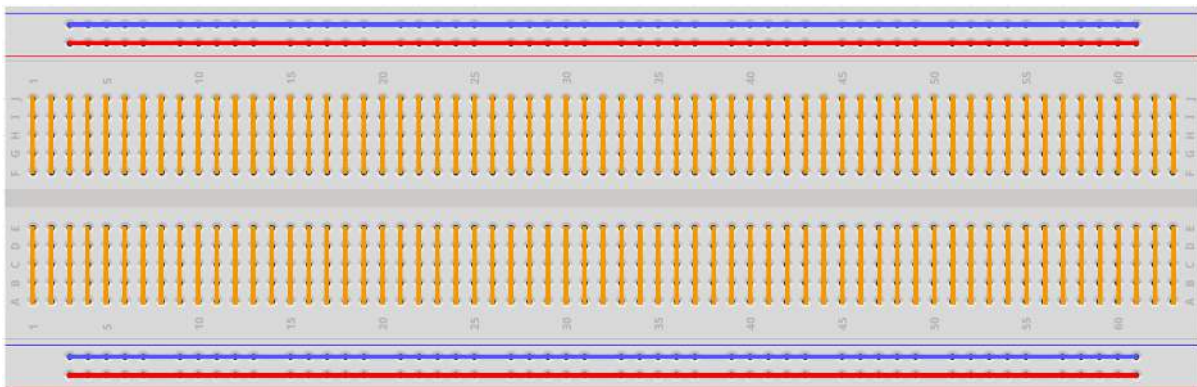
1.3 Breadboard



A breadboard is a construction base for prototyping of electronics. Originally the word referred to a literal bread board, a polished piece of wood used for slicing bread.[1] In the 1970s the solderless breadboard (a.k.a. plugboard, a terminal array board) became available and nowadays the term “breadboard” is commonly used to refer to these.

It is used to build and test circuits quickly before finishing any circuit design. And it has many holes into which components mentioned above can be inserted like ICs and resistors as well as jumper wires. The breadboard allows you to plug in and remove components easily.

The picture shows the internal structure of a breadboard. Although these holes on the breadboard appear to be independent of each other, they are actually connected to each other through metal strips internally.



If you want to know more about breadboard, refer to: [How to Use a Breadboard - Science Buddies](#)

1.4 Resistor



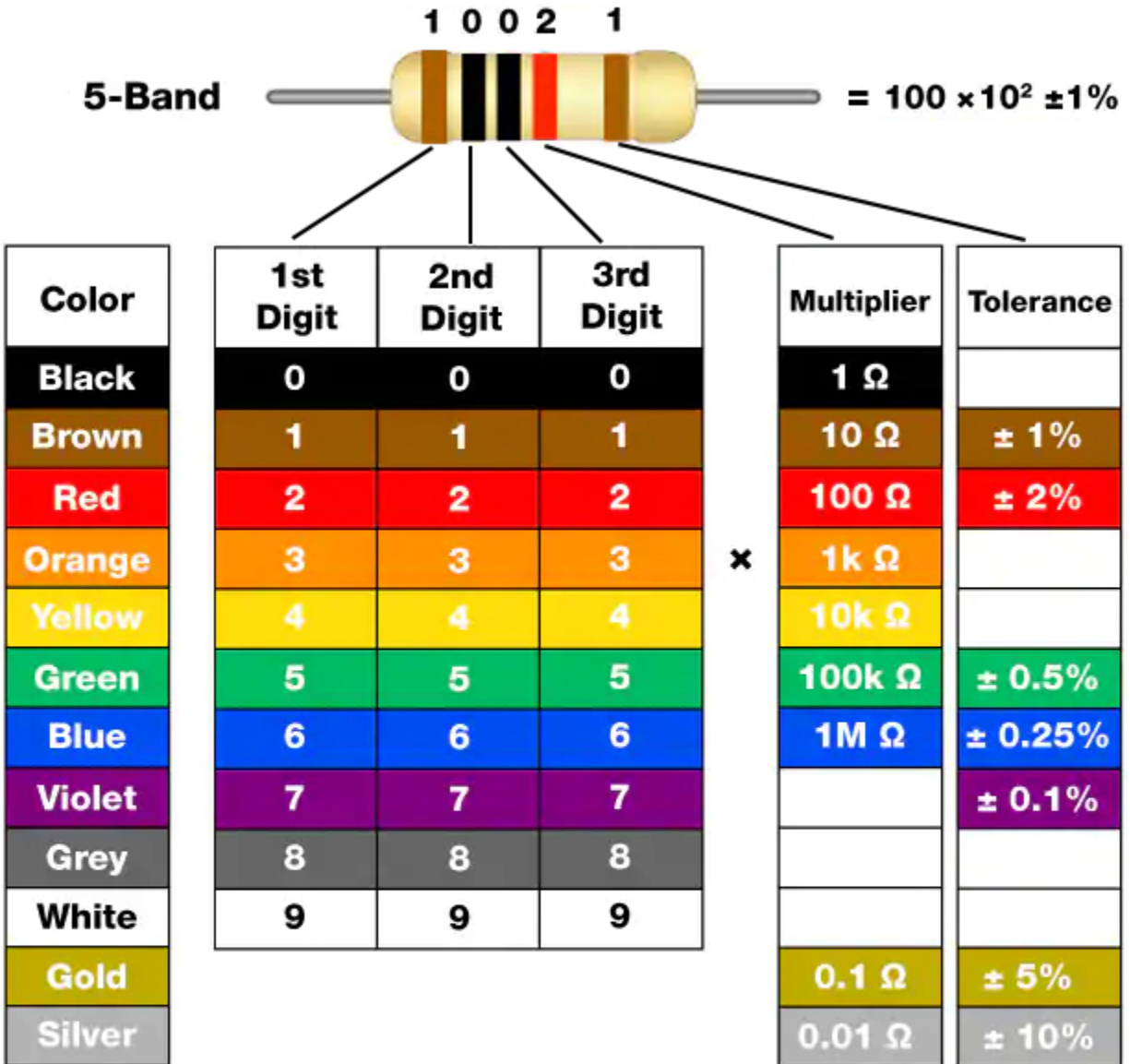
Resistor is an electronic element that can limit the branch current. A fixed resistor is a kind of resistor whose resistance cannot be changed, while that of a potentiometer or a variable resistor can be adjusted.

Two generally used circuit symbols for resistor. Normally, the resistance is marked on it. So if you see these symbols in a circuit, it stands for a resistor.



Ω is the unit of resistance and the larger units include K, M, etc. Their relationship can be shown as follows: 1 M=1000 K, 1 K = 1000 Ω . Normally, the value of resistance is marked on it.

When using a resistor, we need to know its resistance first. Here are two methods: you can observe the bands on the resistor, or use a multimeter to measure the resistance. You are recommended to use the first method as it is more convenient and faster.



As shown in the card, each color stands for a number.

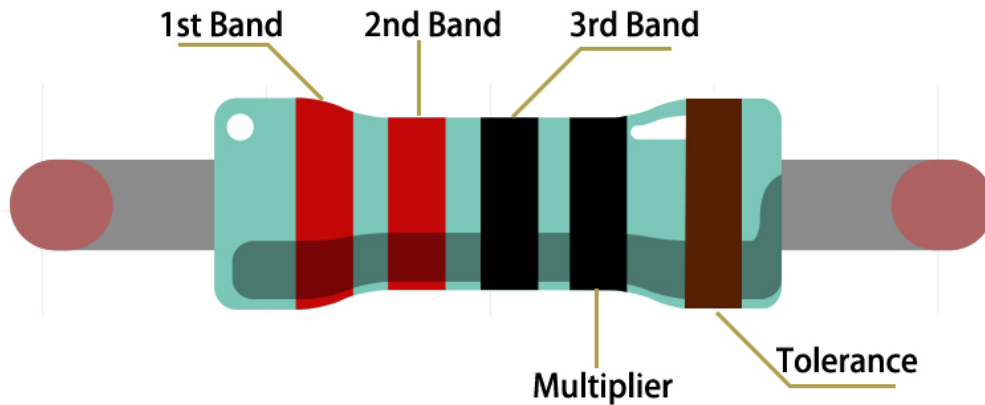
Black	Brown	Red	Orange	Yellow	Green	Blue	Violet	Grey	White	Gold	Silver
0	1	2	3	4	5	6	7	8	9	0.1	0.01

The 4- and 5-band resistors are frequently used, on which there are 4 and 5 chromatic bands.

Normally, when you get a resistor, you may find it hard to decide which end to start for reading the color. The tip is that the gap between the 4th and 5th band will be comparatively larger.

Therefore, you can observe the gap between the two chromatic bands at one end of the resistor; if it's larger than any other band gaps, then you can read from the opposite side.

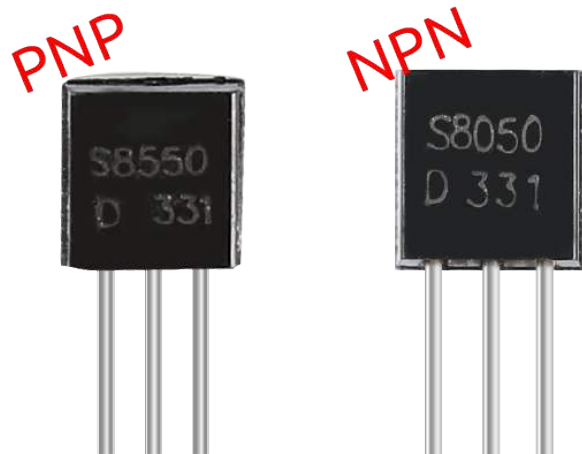
Let's see how to read the resistance value of a 5-band resistor as shown below.



So for this resistor, the resistance should be read from left to right. The value should be in this format: 1st Band 2nd Band 3rd Band $\times 10^{\text{Multiplier}}$ and the permissible error is $\pm \text{Tolerance}\%$. So the resistance value of this resistor is $2(\text{red}) 2(\text{red}) 0(\text{black}) \times 10^0(\text{black}) = 220$, and the permissible error is $\pm 1\%$ (brown).

You can learn more about resistor from Wiki: [Resistor - Wikipedia](#).

1.5 Transistor



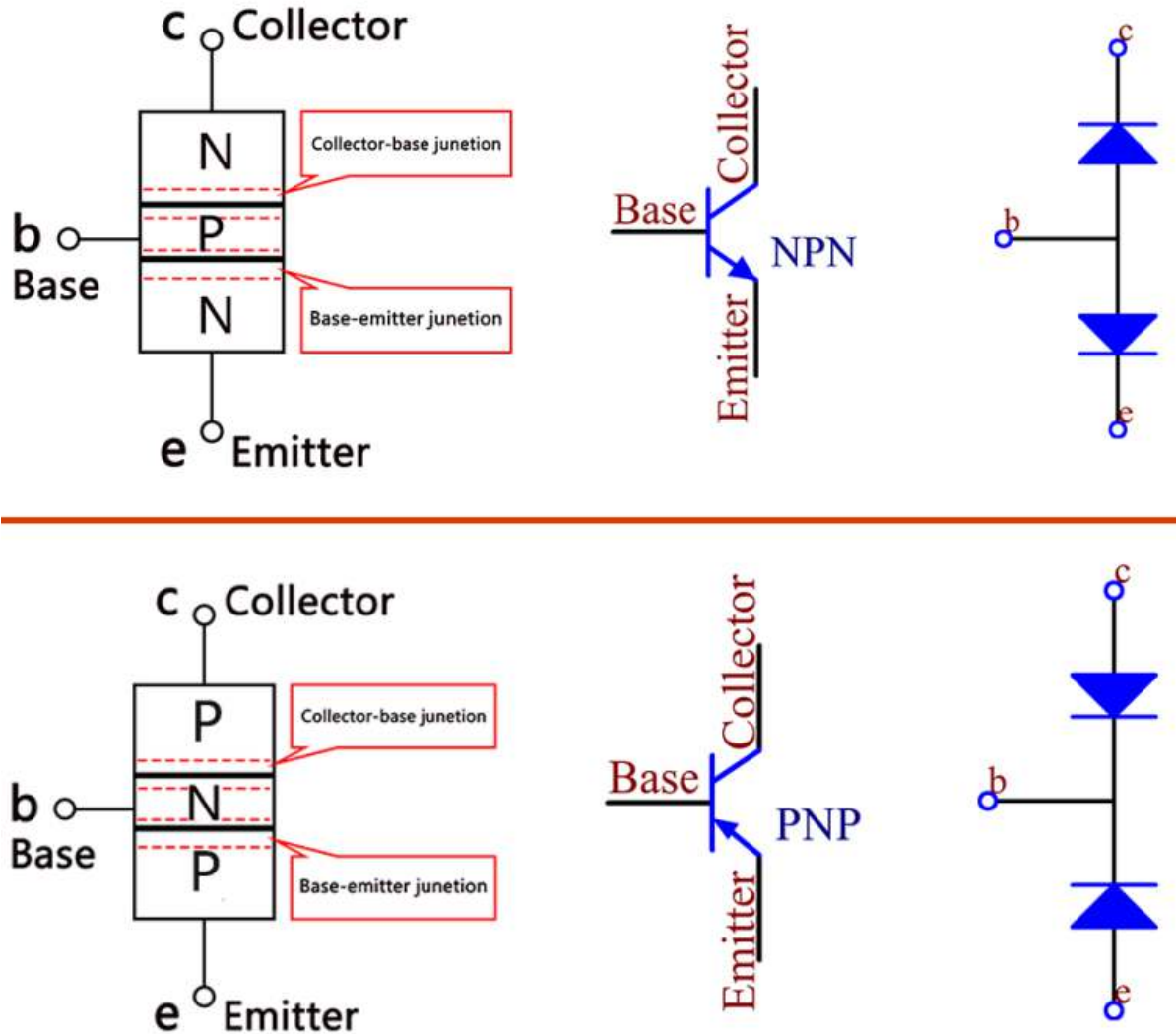
Transistor is a semiconductor device that controls current by current. It functions by amplifying weak signal to larger amplitude signal and is also used for non-contact switch.

A transistor is a three-layer structure composed of P-type and N-type semiconductors. They form the three regions internally. The thinner in the middle is the base region; the other two are both N-type or P-type ones – the smaller region with intense majority carriers is the emitter region, when the other one is the collector region. This composition enables the transistor to be an amplifier. From these three regions, three poles are generated respectively, which are base (b), emitter (e), and collector (c). They form two P-N junctions, namely, the emitter junction and collection junction. The direction of the arrow in the transistor circuit symbol indicates that of the emitter junction.

- [P-N junction - Wikipedia](#)

Based on the semiconductor type, transistors can be divided into two groups, the NPN and PNP ones. From the abbreviation, we can tell that the former is made of two N-type semiconductors and one P-type and that the latter is the opposite. See the figure below.

Note: s8550 is PNP transistor and the s8050 is the NPN one, They look very similar, and we need to check carefully to see their labels.



When a High level signal goes through an NPN transistor, it is energized. But a PNP one needs a Low level signal to manage it. Both types of transistor are frequently used for contactless switches, just like in this experiment.

Put the label side facing us and the pins facing down. The pins from left to right are emitter(e), base(b), and collector(c).



- [S8050 Transistor Datasheet](#)
- [S8550 Transistor Datasheet](#)

1.6 Capacitor





Capacitor, refers to the amount of charge storage under a given potential difference, denoted as C , and the international unit is farad (F). Generally speaking, electric charges move under force in an electric field. When there is a medium between conductors, the movement of electric charges is hindered and the electric charges accumulate on the conductors, resulting in accumulation of electric charges.

The amount of stored electric charges is called capacitance. Because capacitors are one of the most widely used electronic components in electronic equipment, they are widely used in direct current isolation, coupling, bypass, filtering, tuning loops, energy conversion, and control circuits. Capacitors are divided into electrolytic capacitors, solid capacitors, etc.

According to material characteristics, capacitors can be divided into: aluminum electrolytic capacitors, film capacitors, tantalum capacitors, ceramic capacitors, super capacitors, etc.

In this kit, ceramic capacitors and electrolytic capacitors are used.

- [Ceramic Capacitor - Wikipedia](#)
- [Electrolytic Capacitor - Wikipedia](#)

There are 103 or 104 label on the ceramic capacitors, which represent the capacitance value, 103= $10 \times 10^3 \text{pF}$, 104= $10 \times 10^4 \text{pF}$

Unit Conversion

$$1\text{F}=10^3\text{mF}=10^6\mu\text{F}=10^9\text{nF}=10^{12}\text{pF}$$

Example

- [2.15 Button](#) (Arduino Project)
- [2.16 Slide Switch](#) (Arduino Project)
- [2.6 Doorbell](#) (Scratch Project)
- [2.17 GAME - Eat Apple](#) (Scratch Project)
- [2.20 GAME - Fishing](#) (Scratch Project)

1.7 Diode

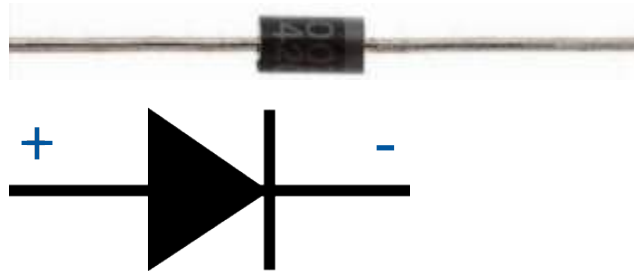
A diode is an electronic component with two electrodes. It allows current to flow in only one direction, which is often called the “Rectifying” function. Thus, a diode can be thought of as an electronic version of a check valve.

Because of its unidirectional conductivity, the diode is used in almost all electronic circuits of some complexity. It is one of the first semiconductor devices and has a wide range of applications.

According to its use classification, it can be divided into detector diodes, rectifier diodes, limiter diodes, voltage regulator diodes, etc.

Rectifier diodes and voltage regulator diodes are included in this kit.

Rectifier Diode



A rectifier diode is a semiconductor diode, used to rectify AC (alternating current) to DC (direct current) using the rectifier bridge application. The alternative of rectifier diode through the Schottky barrier is mainly valued within digital electronics. This diode is capable to conduct the values of current which changes from mA to a few kA & voltages up to a few kV.

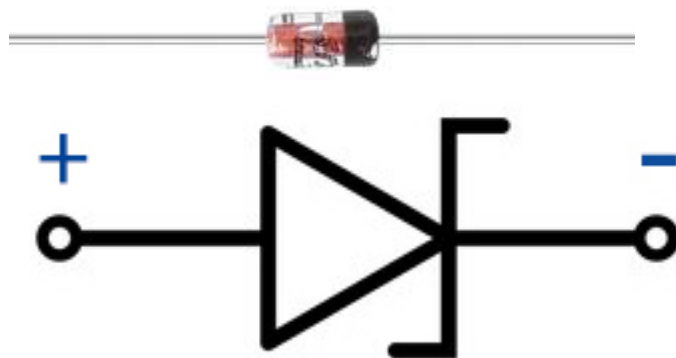
The designing of rectifier diodes can be done with Silicon material and they are capable of conducting high electric current values. These diodes are not famous but still used Ge or gallium arsenide-based semiconductor diodes. Ge diodes have less allowable reversed voltage as well as a lesser allowable junction temperature. The Ge diode has a benefit as compared to Si diode that is low threshold voltage value while operating in a forward-bias.

- [1N400x general-purpose diode - Wikipedia](#)

Zener Diode

A Zener diode is a special type of diode designed to reliably allow current to flow “backwards” when a certain set reverse voltage, known as the Zener voltage, is reached.

This diode is a semiconductor device that has a very high resistance up to the critical reverse breakdown voltage. At this critical breakdown point, the reverse resistance is reduced to a very small value, and the current increases while the voltage remains constant in this low resistance region.



- [Zener diode - Wikipedia](#)

1.8 Jumper Wires

Wires that connect two terminals are called jumper wires. There are various kinds of jumper wires. Here we focus on those used in breadboard. Among others, they are used to transfer electrical signals from anywhere on the breadboard to the input/output pins of a microcontroller.

Jumper wires are fitted by inserting their “end connectors” into the slots provided in the breadboard, beneath whose surface there are a few sets of parallel plates that connect the slots in groups of rows or columns depending on the area. The “end connectors” are inserted into the breadboard, without soldering, in the particular slots that need to be connected in the specific prototype.

There are three types of jumper wire: Female-to-Female, Male-to-Male, and Male-to-Female. The reason we call it Male-to-Female is because it has the outstanding tip in one end as well as a sunk female end. Male-to-Male means both side are male and Female-to-Female means both ends are female.



More than one type of them may be used in a project. **The color of the jump wires is different but it doesn't mean their function is different accordingly; it's just designed so to better identify the connection between each circuit.**

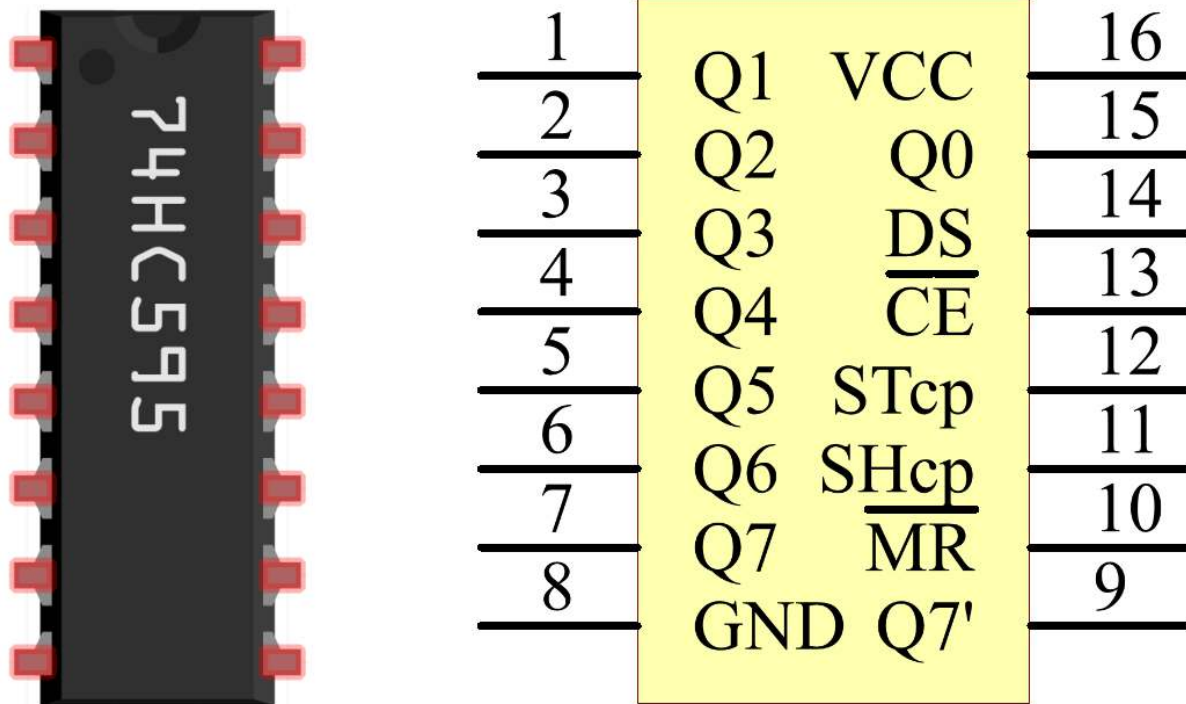
Chip

1.9 74HC595



The 74HC595 consists of an 8bit shift register and a storage register with threestate parallel outputs. It converts serial input into parallel output so you can save IO ports of an MCU. When MR (pin10) is high level and OE (pin13) is low level, data is input in the rising edge of SHcp and goes to the memory register through the rising edge of SHcp. If the two clocks are connected together, the shift register is always one pulse earlier than the memory register. There is a serial shift input pin (Ds), a serial output pin (Q) and an asynchronous reset button (low level) in the memory register. The memory register outputs a Bus with a parallel 8-bit and in three states. When OE is enabled (low level), the data in memory register is output to the bus.

- [74HC595 Datasheet](#)



Pins of 74HC595 and their functions:

- **Q0-Q7**: 8-bit parallel data output pins, able to control 8 LEDs or 8 pins of 7-segment display directly.
- **Q7'**: Series output pin, connected to DS of another 74HC595 to connect multiple 74HC595s in series
- **MR**: Reset pin, active at low level;
- **SHcp**: Time sequence input of shift register. On the rising edge, the data in shift register moves successively one bit, i.e. data in Q1 moves to Q2, and so forth. While on the falling edge, the data in shift register remain unchanged.
- **STcp**: Time sequence input of storage register. On the rising edge, data in the shift register moves into memory register.
- **CE**: Output enable pin, active at low level.
- **DS**: Serial data input pin
- **VCC**: Positive supply voltage.
- **GND**: Ground.

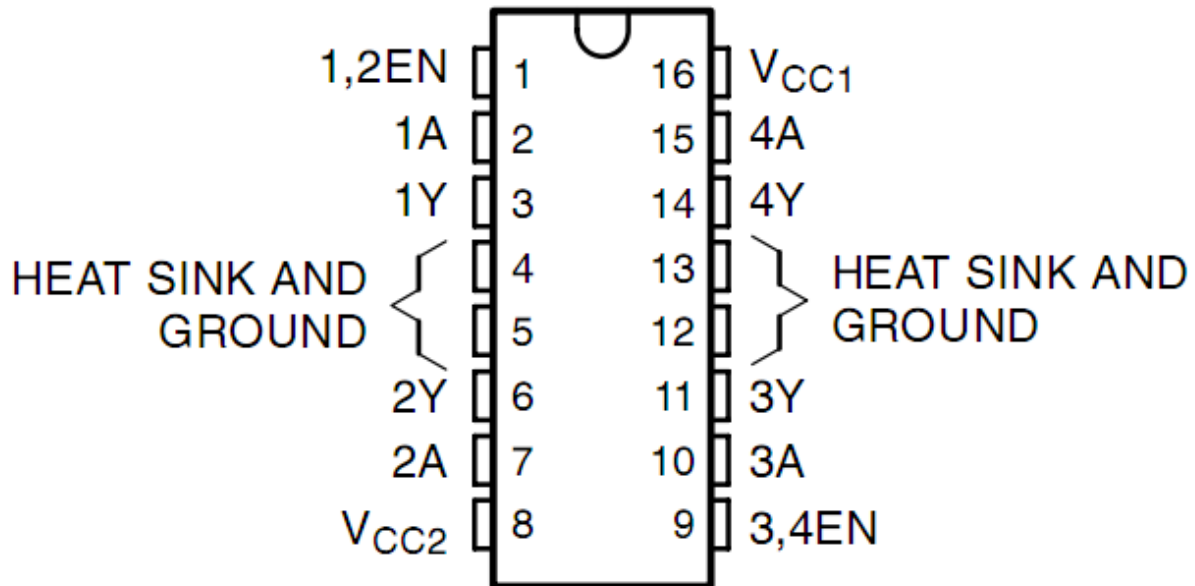
Example

- [2.5 7-Segment Display](#) (Arduino Project)
- [2.7 4-Digital 7-Segment Display](#) (Arduino Project)
- [3.2 Pedestrian Crossing Button](#) (Arduino Project)

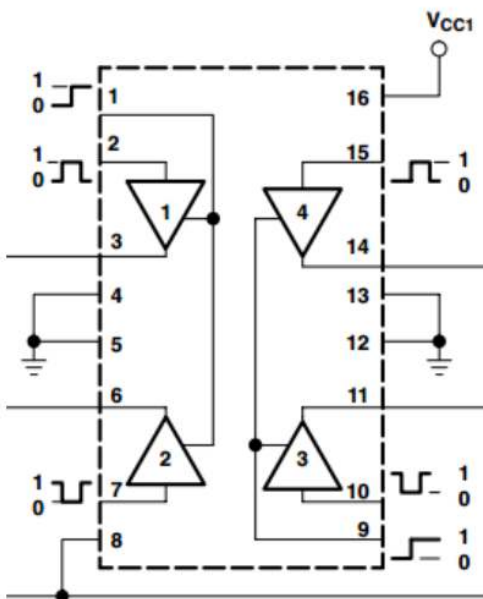
1.10 L293D

L293D is a 4-channel motor driver integrated by chip with high voltage and high current. It's designed to connect to standard DTL, TTL logic level, and drive inductive loads (such as relay coils, DC, Stepper Motors) and power switching transistors etc. DC Motors are devices that turn DC electrical energy into mechanical energy. They are widely used in electrical drive for their superior speed regulation performance.

See the figure of pins below. L293D has two pins (V_{cc1} and V_{cc2}) for power supply. V_{cc2} is used to supply power for the motor, while V_{cc1} to supply for the chip. Since a small-sized DC motor is used here, connect both pins to +5V.



The following is the internal structure of L293D. Pin EN is an enable pin and only works with high level; A stands for input and Y for output. You can see the relationship among them at the right bottom. When pin EN is High level, if A is High, Y outputs high level; if A is Low, Y outputs Low level. When pin EN is Low level, the L293D does not work.



INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant, Z = high impedance (off)

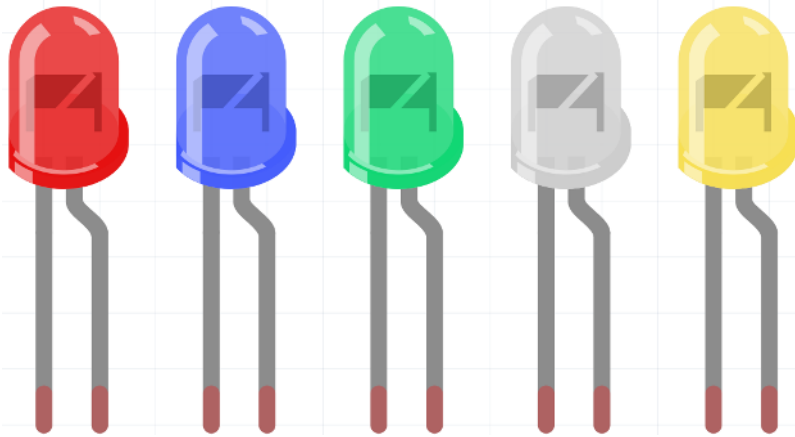
- [L293D Datasheet](#)

Example

- [2.13 Motor](#) (Arduino Project)
- [2.12 Rotating Fan](#) (Scratch Project)

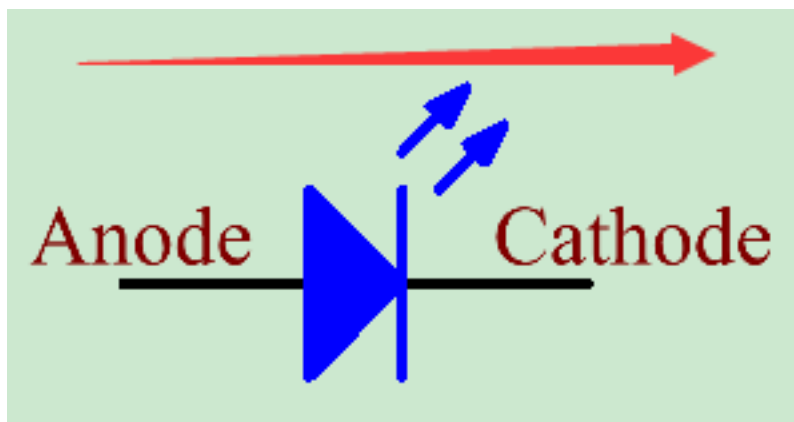
Display

1.11 LED



Semiconductor light-emitting diode is a type of component which can turn electric energy into light energy via PN junctions. By wavelength, it can be categorized into laser diode, infrared light-emitting diode and visible light-emitting diode which is usually known as light-emitting diode (LED).

Diode has unidirectional conductivity, so the current flow will be as the arrow indicates in figure circuit symbol. You can only provide the anode with a positive power and the cathode with a negative. Thus the LED will light up.



An LED has two pins. The longer one is the anode, and shorter one, the cathode. Pay attention not to connect them inversely. There is fixed forward voltage drop in the LED, so it cannot be connected with the circuit directly because the supply voltage can outweigh this drop and cause the LED to be burnt. The forward voltage of the red, yellow, and green LED is 1.8 V and that of the white one is 2.6 V. Most LEDs can withstand a maximum current of 20 mA, so we need to connect a current limiting resistor in series.

The formula of the resistance value is as follows:

$$R = (V_{\text{supply}} - V_D) / I$$

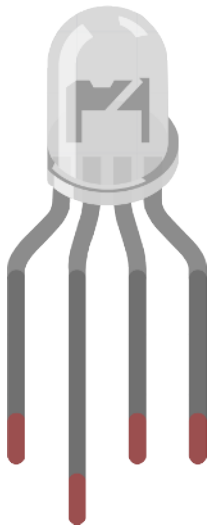
R stands for the resistance value of the current limiting resistor, **V_{supply}** for voltage supply, **V_D** for voltage drop and **I** for the working current of the LED.

Here is the detailed introduction for the LED: [LED - Wikipedia](#).

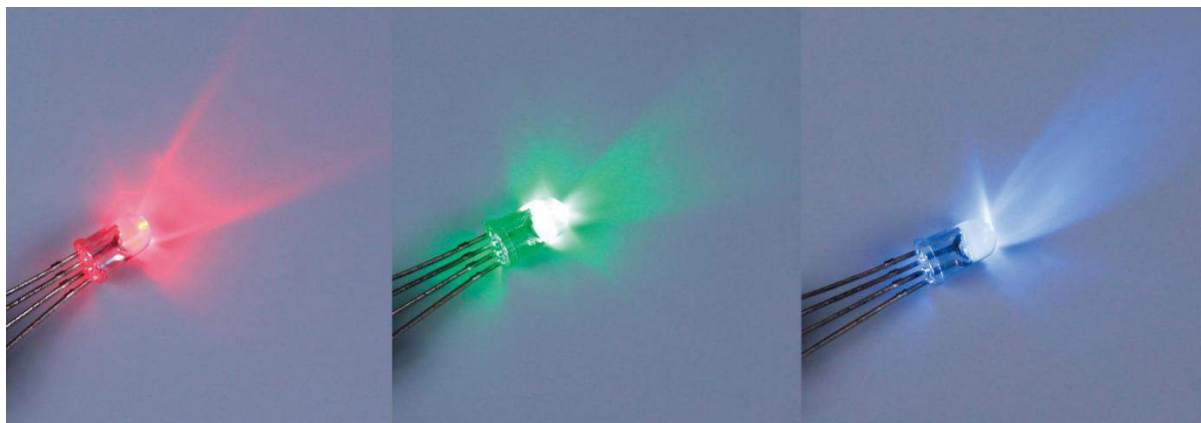
Example

- [2.2 LED](#) (Arduino Project)
- [2.2 Breathing LED](#) (Scratch Project)
- [2.1 Table Lamp](#) (Scratch Project)

1.12 RGB LED

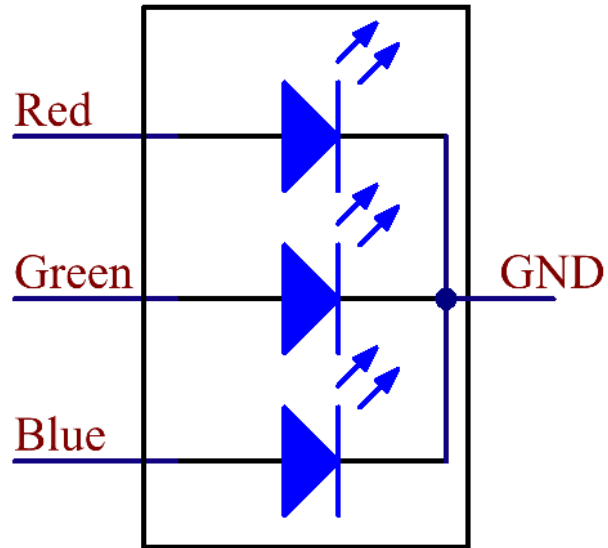


RGB LEDs emit light in various colors. An RGB LED packages three LEDs of red, green, and blue into a transparent or semitransparent plastic shell. It can display various colors by changing the input voltage of the three pins and superimpose them, which, according to statistics, can create 16,777,216 different colors.

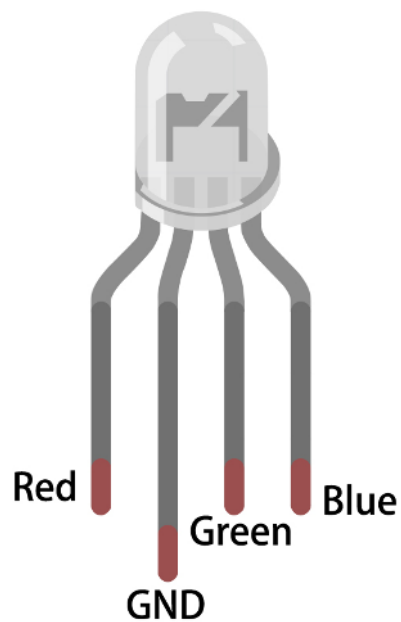


RGB LEDs can be categorized into common anode and common cathode ones. In this kit, the latter is used. The **common cathode**, or CC, means to connect the cathodes of the three LEDs. After you connect it with GND and plug in the three pins, the LED will flash the corresponding color.

Its circuit symbol is shown as figure.



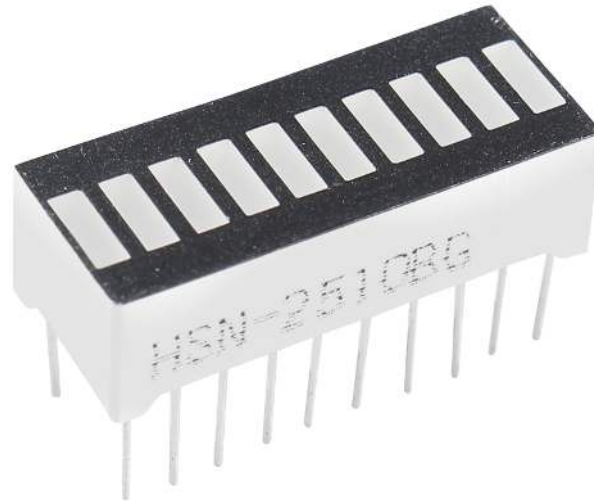
An RGB LED has 4 pins: the longest one is GND; the others are Red, Green and Blue. Touch its plastic shell and you will find a cut. The pin closest to the cut is the first pin, marked as Red, then GND, Green and Blue in turn.



Example

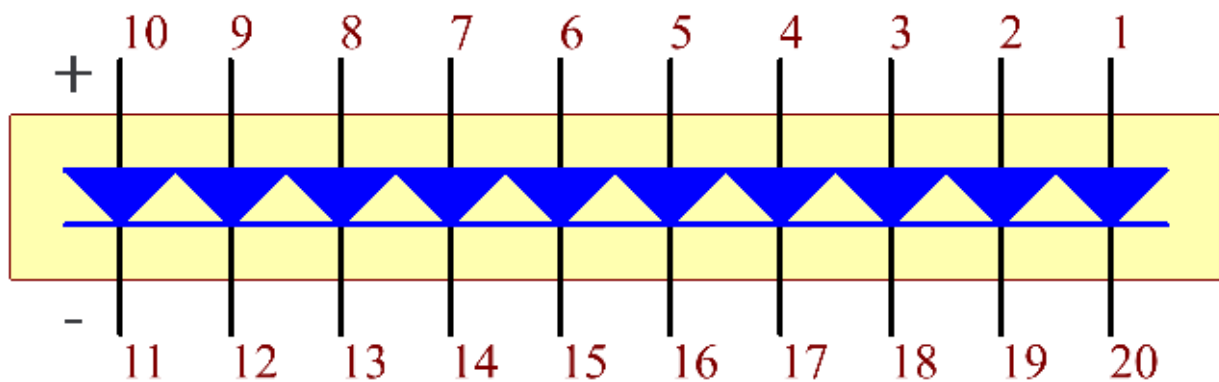
- [2.3 RGB LED](#) (Arduino Project)
- [3.3 Overheat Monitor](#) (Arduino Project)

1.13 LED Bar Graph



LED Bar Graph is an LED array, which is used to connect with electronic circuit or microcontroller. It's easy to connect LED bar graph with the circuit like as connecting 10 individual LEDs with 10 output pins. Generally we can use the LED bar graph as a Battery level Indicator, Audio equipments, and Industrial Control panels. There are many other applications of LED bar graphs.

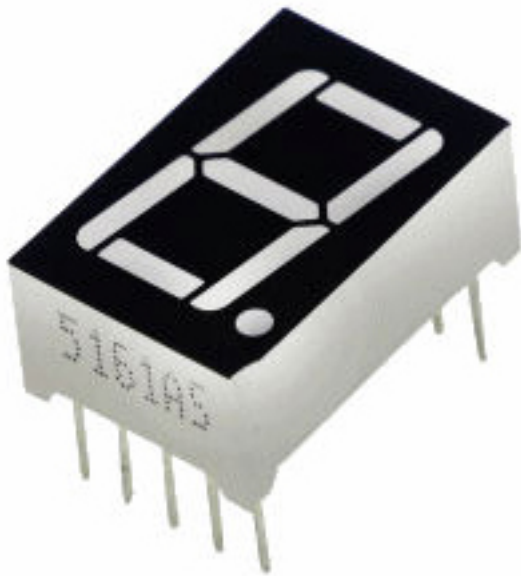
The following is the internal **Schematic** of LED Bar Graph. Generally speaking, the side with the label is the anode and the other side is the cathode.



Example

- [2.4 LED Bar Graph](#) (Arduino Project)
- [2.14 GAME - Shooting](#) (Scratch Project)

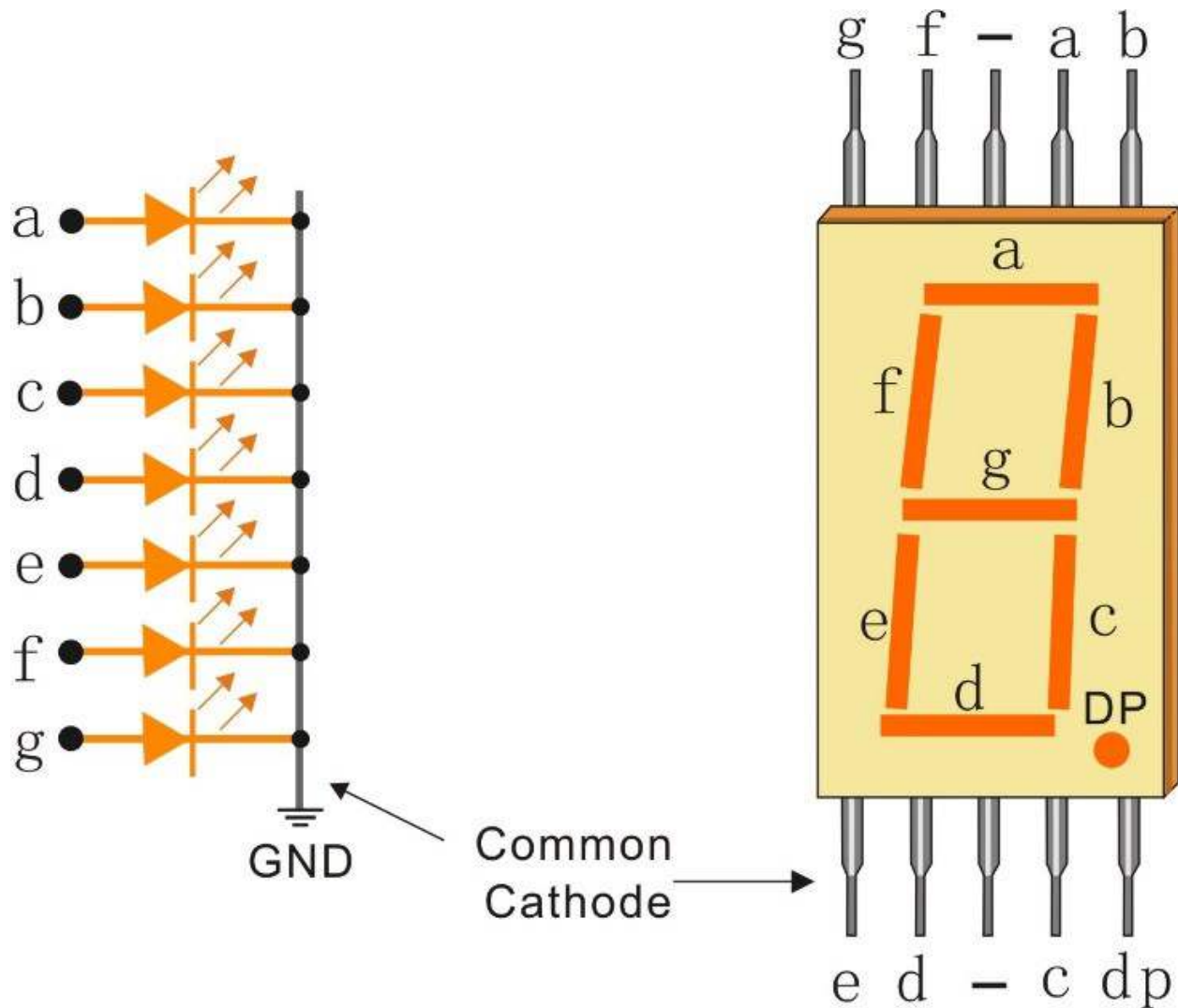
1.14 7-segment Display



A 7-segment display is an 8-shaped component which packages 7 LEDs. Each LED is called a segment - when energized, one segment forms part of a numeral to be displayed.

There are two types of pin connection: Common Cathode (CC) and Common Anode (CA). As the name suggests, a CC display has all the cathodes of the 7 LEDs connected when a CA display has all the anodes of the 7 segments connected.

In this kit, we use the Common Cathode 7-segment display, here is the electronic symbol.



Each of the LEDs in the display is given a positional segment with one of its connection pins led out from the rectangular plastic package. These LED pins are labeled from “a” through to “g” representing each individual LED. The other LED pins are connected together forming a common pin. So by forward biasing the appropriate pins of the LED segments in a particular order, some segments will brighten and others stay dim, thus showing the corresponding character on the display.

Display Codes

To help you get to know how 7-segment displays(Common Cathode) display Numbers, we have drawn the following table. Numbers are the number 0-F displayed on the 7-segment display; (DP) GFEDCBA refers to the corresponding LED set to 0 or 1, For example, 00111111 means that DP and G are set to 0, while others are set to 1. Therefore, the number 0 is displayed on the 7-segment display, while HEX Code corresponds to hexadecimal number.

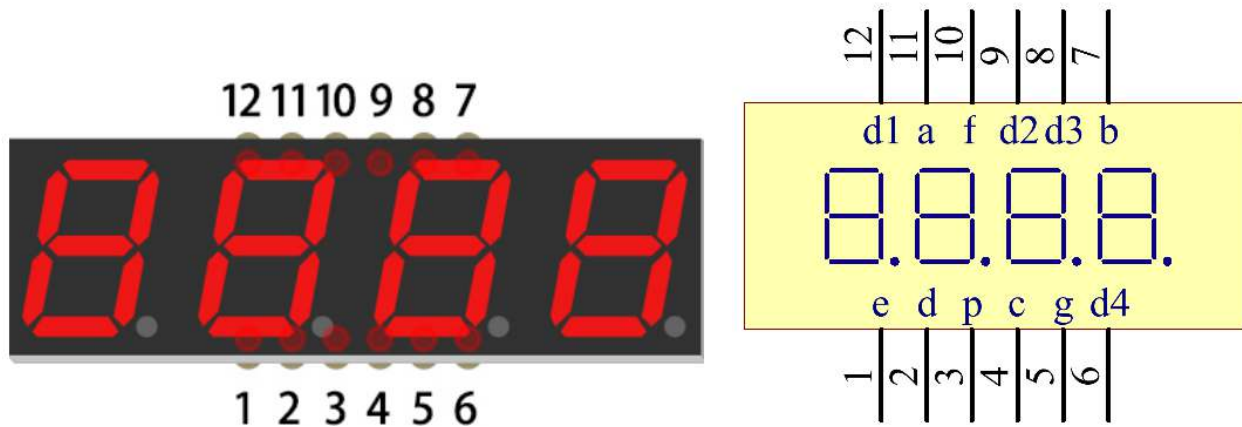
Numbers	Common Cathode		Numbers	Common Cathode	
	(DP)GFEDCBA	Hex Code		(DP)GFEDCBA A	Hex Code
0	00111111	0x3f	A	01110111	0x77
1	00001110	0x06	B	01111100	0x7c
2	01011011	0x5b	C	00111001	0x39
3	01001111	0x4f	D	01011110	0x5e
4	01100110	0x66	E	01111001	0x79
5	01101101	0x6d	F	01110001	0x71
6	01111101	0x7d			
7	00001111	0x07			
8	01111111	0x7f			
9	01101111	0x6f			

Example

- *2.5 7-Segment Display* (Arduino Project)
- *3.2 Pedestrian Crossing Button* (Arduino Project)

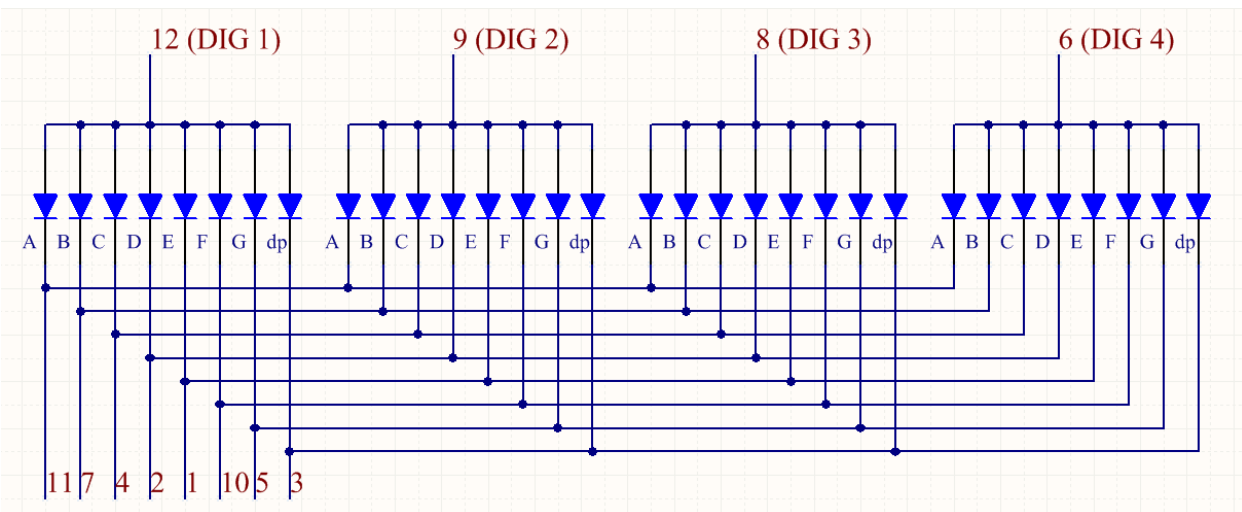
1.15 4-Digit 7-Segment Display

4-Digit 7-segment display consists of four 7-segment displays working together.



The 4-digit 7-segment display works independently. It uses the principle of human visual persistence to quickly display the characters of each 7-segment in a loop to form continuous strings.

For example, when “1234” is displayed on the display, “1” is displayed on the first 7-segment, and “234” is not displayed. After a period of time, the second 7-segment shows “2”, the 1st 3th 4th of 7-segment does not show, and so on, the four digital display show in turn. This process is very short (typically 5ms), and because of the optical afterglow effect and the principle of visual residue, we can see four characters at the same time.



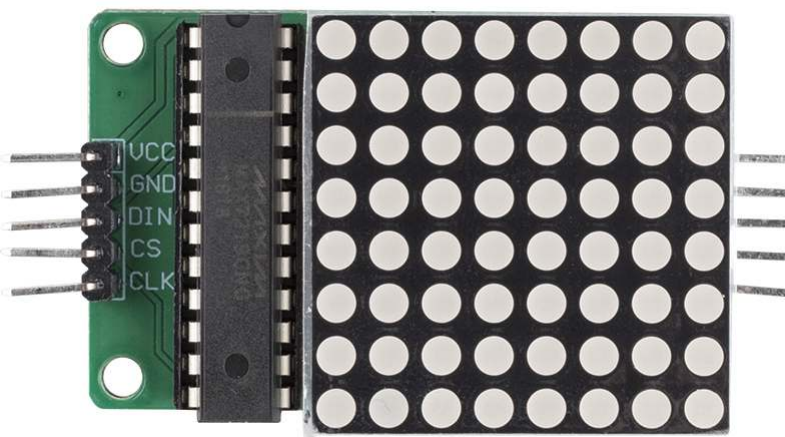
Display Codes

To help you get to know how 7-segment displays(Common Anode) display Numbers, we have drawn the following table. Numbers are the number 0-F displayed on the 7-segment display; (DP) GFEDCBA refers to the corresponding LED set to 0 or 1, For example, 11000000 means that DP and G are set to 1, while others are set to 0. Therefore, the number 0 is displayed on the 7-segment display, while HEX Code corresponds to hexadecimal number.

Numbers	Common Anode		Numbers	Common Anode	
	(DP)GFEDCBA	Hex Code		(DP)GFEDCBA A	Hex Code
0	11000000	0xc0	A	10001000	0x88
1	11111001	0xf9	B	10000011	0x83
2	10100100	0xa4	C	11000110	0xc6
3	10110000	0xb0	D	10100001	0xa1
4	10011001	0x99	E	10000110	0x86
5	10010010	0x92	F	10001110	0x8e
6	10000010	0x82	.	01111111	0x7f
7	11111000	0xf8			
8	10000000	0x80			
9	10010000	0x90			

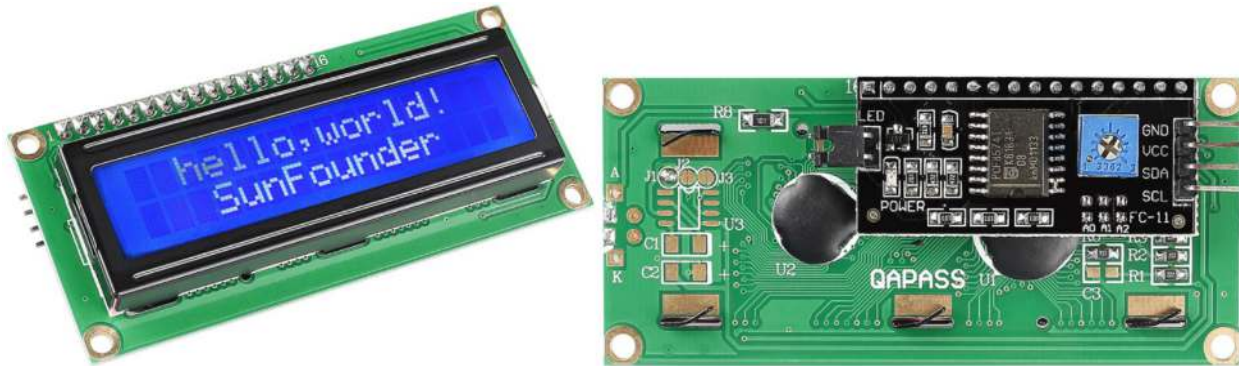
Example

- *2.7 4-Digital 7-Segment Display* (Arduino Project)

1.16 LED Matrix Module

This is a common cathode 8x8 dot matrix module driven by MAX7219, the module operating voltage is 5V, the size is 50mmx32mmx15mm, the left side is input port, the right side is output port, support multiple modules cascade.

1.17 I2C LCD1602



- **GND:** Ground
- **VCC:** Voltage supply, 5V.
- **SDA:** Serial data line. Connect to VCC through a pullup resistor.
- **SCL:** Serial clock line. Connect to VCC through a pullup resistor.

As we all know, though LCD and some other displays greatly enrich the man-machine interaction, they share a common weakness. When they are connected to a controller, multiple IOs will be occupied of the controller which has no so many outer ports. Also it restricts other functions of the controller.

Therefore, LCD1602 with an I2C module is developed to solve the problem. The I2C module has a built-in PCF8574 I2C chip that converts I2C serial data to parallel data for the LCD display.

- [PCF8574 Datasheet](#)

I2C Address

The default address is basically 0x27, in a few cases it may be 0x3F.

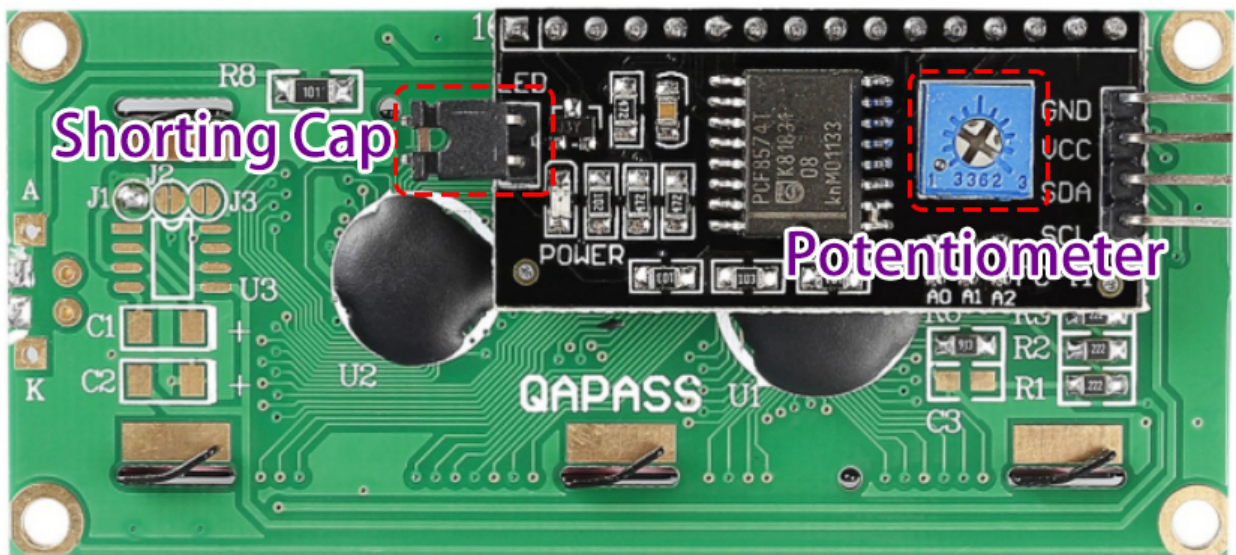
Taking the default address of 0x27 as an example, the device address can be modified by shorting the A0/A1/A2 pads; in the default state, A0/A1/A2 is 1, and if the pad is shorted, A0/A1/A2 is 0.

Slave Address

0	0	1	0	0	A2	A1	A0	
0	0	1	0	0	1	1	1	0x27
0	0	1	0	0	1	1	0	0x26
0	0	1	0	0	1	0	1	0x25
0	0	1	0	0	0	1	1	0x23
.....								
0	0	1	0	0	0	0	0	0x20

Backlight/Contrast

Backlight can be enabled by jumper cap, unplug the jumper cap to disable the backlight. The blue potentiometer on the back



- **Shorting Cap:** Backlight can be enabled by this cap unplug this cap to disable the backlight.
- **Potentiometer:** It is used to adjust the contrast (the clarity of the displayed text), which is increased in the clockwise direction and decreased in the counterclockwise direction.

Example

- *2.9 I2C LCD1602 Module* (Arduino Project)
- *3.5 Access Control System* (Arduino Project)
- *3.4 Guess Number* (Arduino Project)
- *3.3 Overheat Monitor* (Arduino Project)
- *2.4 LCD1602* (Scratch Project)

Sound

1.18 Buzzer



As a type of electronic buzzer with an integrated structure, buzzers, which are supplied by DC power, are widely used in computers, printers, photocopiers, alarms, electronic toys, automotive electronic devices, telephones, timers and other electronic products or voice devices.

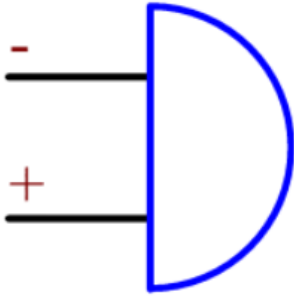
Buzzers can be categorized as active and passive ones (see the following picture). Turn the buzzer so that its pins are facing up, and the buzzer with a green circuit board is a passive buzzer, while the one enclosed with a black tape is an active one.

The difference between an active buzzer and a passive buzzer:

An active buzzer has a built-in oscillating source, so it will make sounds when electrified. But a passive buzzer does not have such source, so it will not beep if DC signals are used; instead, you need to use square waves whose frequency is between 2K and 5K to drive it. The active buzzer is often more expensive than the passive one because of multiple built-in oscillating circuits.

The following is the electrical symbol of a buzzer. It has two pins with positive and negative poles. With a + in the surface represents the anode and the other is the cathode.

Buzzer



You can check the pins of the buzzer, the longer one is the anode and the shorter one is the cathode. Please don't mix them up when connecting, otherwise the buzzer will not make sound.

[Buzzer - Wikipedia](#)

Example

- [2.10 Active Buzzer](#) (Arduino Project)
- [2.11 Passive Buzzer](#) (Arduino Project)

Driver

1.19 DC Motor



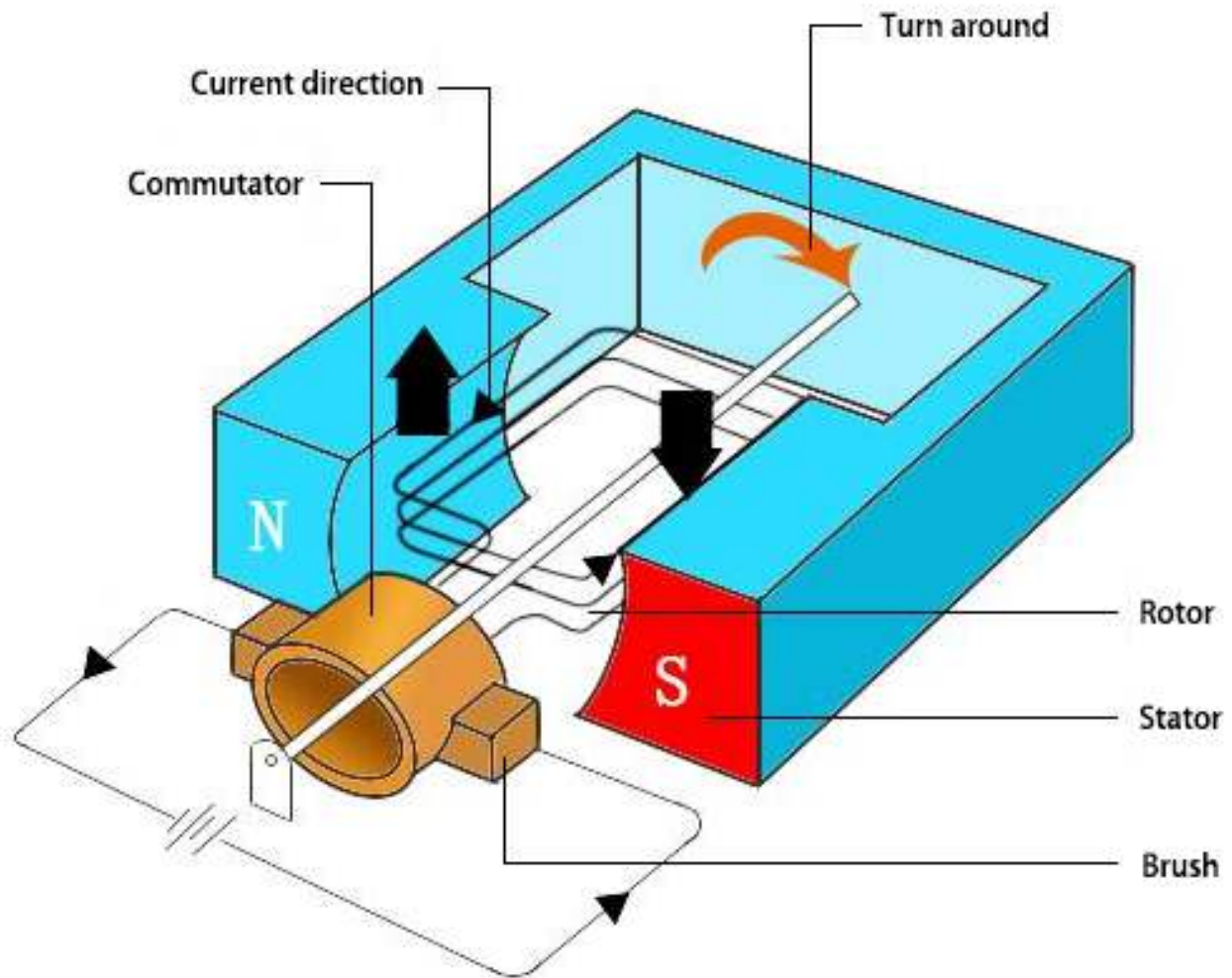
This is a 3V DC motor. When you give a high level and a low level to each of the 2 terminals, it will rotate.

- **Size:** 25*20*15MM
- **Operation Voltage:** 1-6V
- **Free-run Current (3V):** 70m
- **A Free-run Speed (3V):** 13000RPM
- **Stall Current (3V):** 800mA
- **Shaft Diameter:** 2mm

Direct current (DC) motor is a continuous actuator that converts electrical energy into mechanical energy. DC motors make rotary pumps, fans, compressors, impellers, and other devices work by producing continuous angular rotation.

A DC motor consists of two parts, the fixed part of the motor called the **stator** and the internal part of the motor called the **rotor** (or **armature** of a DC motor) that rotates to produce motion. The key to generating motion is to position

the armature within the magnetic field of the permanent magnet (whose field extends from the north pole to the south pole). The interaction of the magnetic field and the moving charged particles (the current-carrying wire generates the magnetic field) produces the torque that rotates the armature.



Current flows from the positive terminal of the battery through the circuit, through the copper brushes to the commutator, and then to the armature. But because of the two gaps in the commutator, this flow reverses halfway through each complete rotation. This continuous reversal essentially converts the DC power from the battery to AC, allowing the armature to experience torque in the right direction at the right time to maintain rotation.

- [DC Motor - MagLab](#)

Example

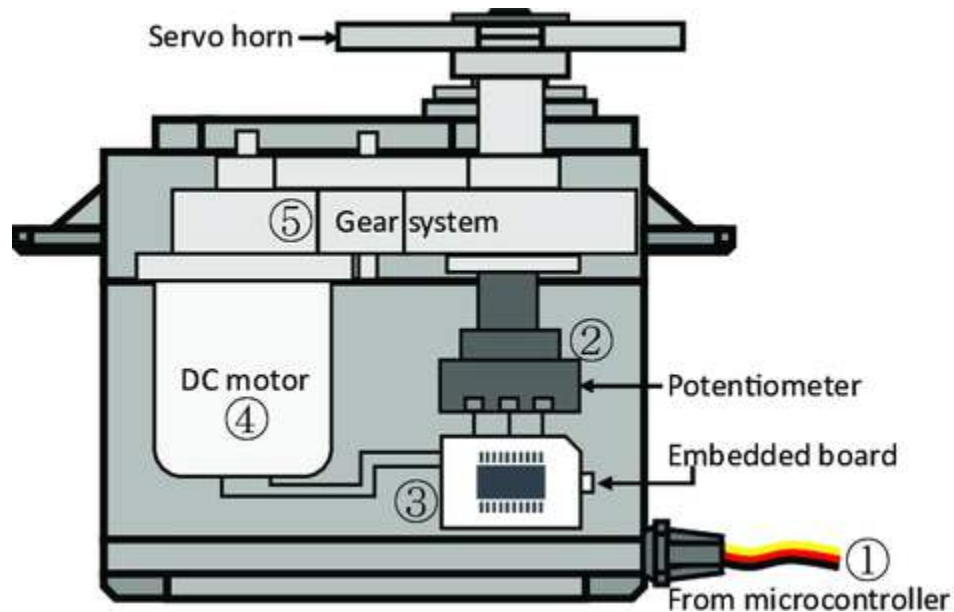
- [2.13 Motor](#) (Arduino Project)
- [2.12 Rotating Fan](#) (Scratch Project)

1.20 Servo

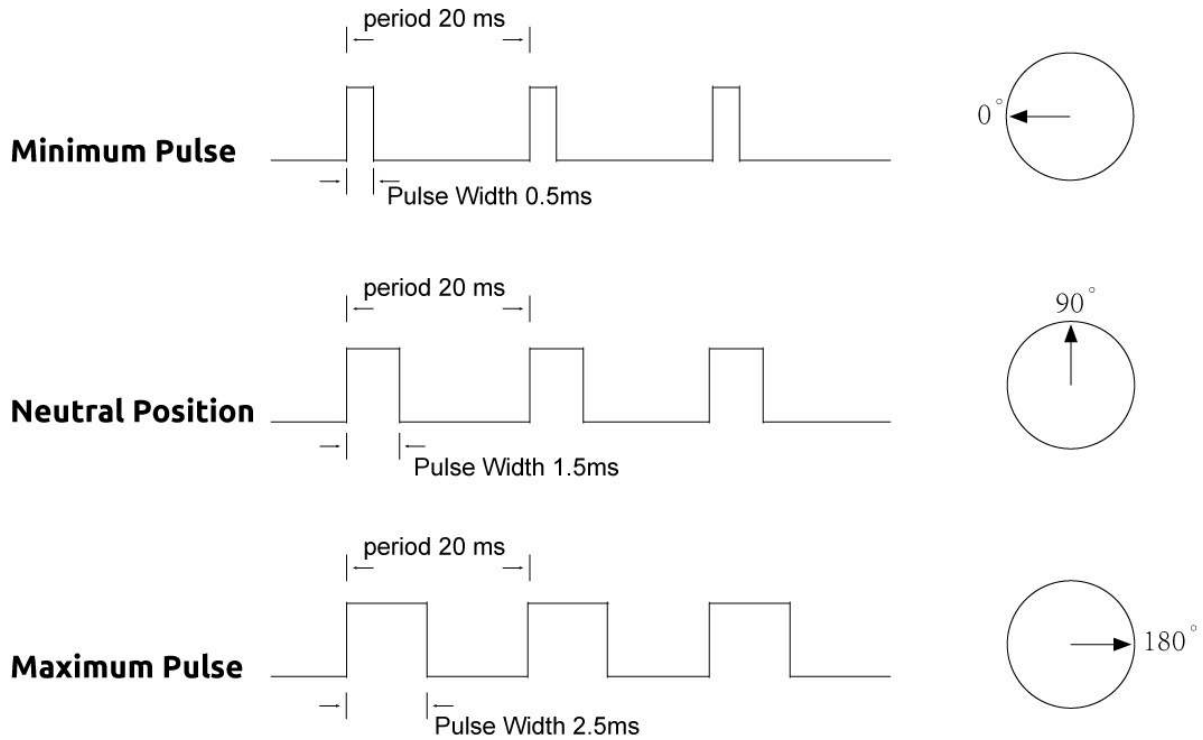


A servo is generally composed of the following parts: case, shaft, gear system, potentiometer, DC motor, and embedded board.

It works like this: The microcontroller sends out PWM signals to the servo, and then the embedded board in the servo receives the signals through the signal pin and controls the motor inside to turn. As a result, the motor drives the gear system and then motivates the shaft after deceleration. The shaft and potentiometer of the servo are connected together. When the shaft rotates, it drives the potentiometer, so the potentiometer outputs a voltage signal to the embedded board. Then the board determines the direction and speed of rotation based on the current position, so it can stop exactly at the right position as defined and hold there.



The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse width Modulation. The servo expects to see a pulse every 20 ms. The length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90 degree position (neutral position). When a pulse is sent to a servo that is less than 1.5 ms, the servo rotates to a position and holds its output shaft some number of degrees counterclockwise from the neutral point. When the pulse is wider than 1.5 ms the opposite occurs. The minimal width and the maximum width of pulse that will command the servo to turn to a valid position are functions of each servo. Generally the minimum pulse will be about 0.5 ms wide and the maximum pulse will be 2.5 ms wide.



Example

- [2.12 Servo](#) (Arduino Project)
- [2.11 Pendulum](#) (Scratch Project)

1.21 Stepper Motor



Stepper motors, due to their unique design, can be controlled to a high degree of accuracy without any feedback mechanisms. The shaft of a stepper, mounted with a series of magnets, is controlled by a series of electromagnetic

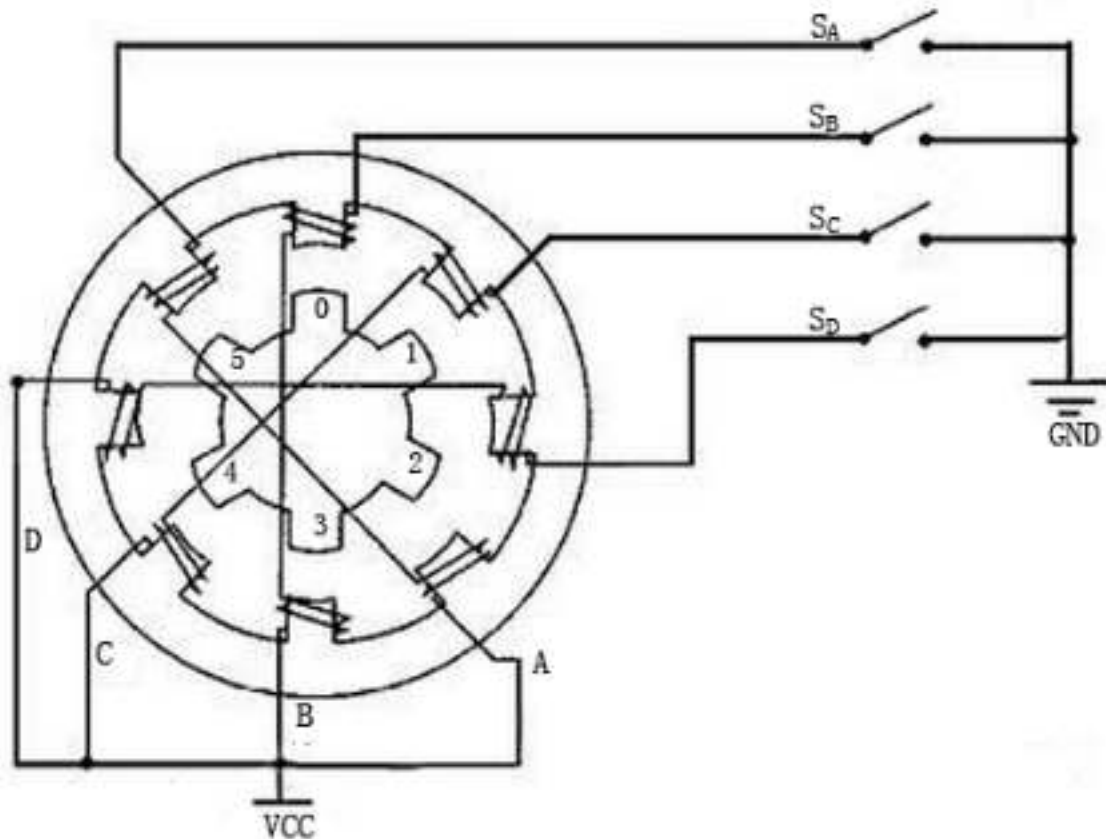
coils that are charged positively and negatively in a specific sequence, precisely moving it forward or backward in small “steps”.

Principle

There are two types of steppers, unipolars and bipolars, and it is very important to know which type you are working with. In this experiment, we will use a unipolar stepper.

The stepper motor is a four-phase one, which uses a unipolarity DC power supply. As long as you electrify all phase windings of the motor by an appropriate timing sequence, you can make it rotate step by step. The **Schematic** of a four-phase reactive stepper motor:

Here’s how a 4-phase stepper motor works:



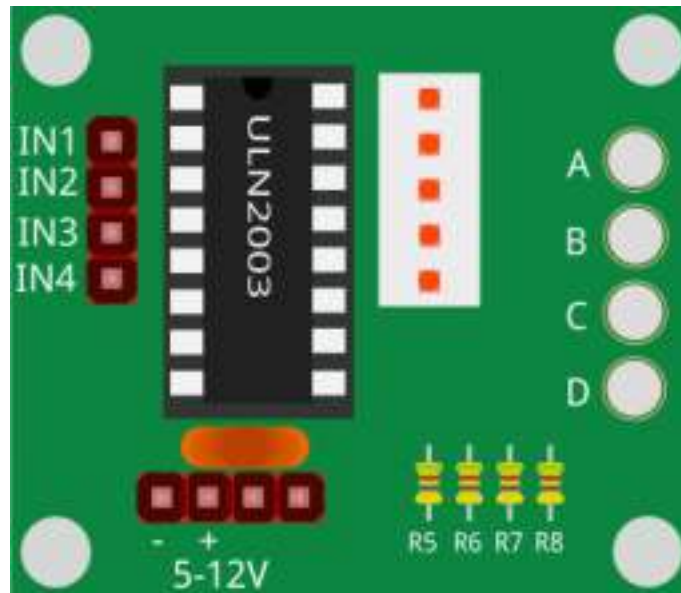
In the figure, in the middle of the motor is a rotor – a gear-shaped permanent magnet. Around the rotor, 0 to 5 are teeth. Then more outside, there are 8 magnetic poles, with each two opposite ones connected by coil winding. So they form four pairs from A to D, which is called a phase. It has four lead wires to be connected with switches SA, SB, SC, and SD. Therefore, the four phases are in parallel in the circuit, and the two magnetic poles in one phase are in series.

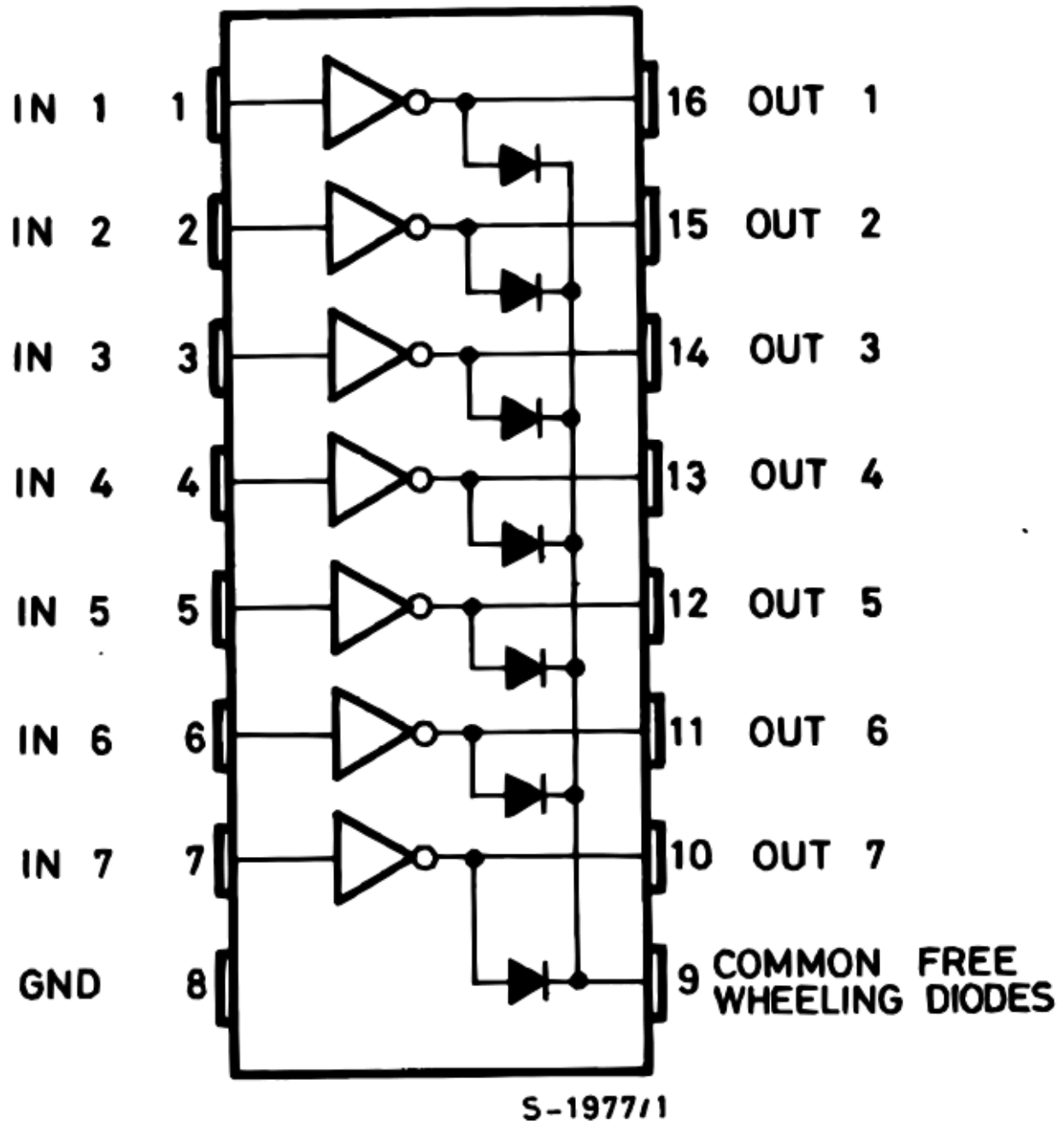
At the beginning, switch SB is power on, switch SA, SC, and SD is power off, and B-phase magnetic poles align with tooth 0 and 3 of the rotor. At the same time, tooth 1 and 4 generate staggered teeth with C- and D-phase poles. Tooth 2 and 5 generate staggered teeth with D- and A-phase poles. When switch SC is power on, switch SB, SA, and SD is power off, the rotor rotates under magnetic field of C-phase winding and that between tooth 1 and 4. Then tooth 1 and 4 align with the magnetic poles of C-phase winding. While tooth 0 and 3 generate staggered teeth with A- and B-phase poles, and tooth 2 and 5 generate staggered teeth with the magnetic poles of A- and D-phase poles. The similar situation goes on and on. Energize the A, B, C and D phases in turn, and the rotor will rotate in the order of A, B, C and D.

The four-phase stepper motor has three operating modes: single four-step, double four-step, and eight-step. The step angle for the single four-step and double four-step are the same, but the driving torque for the single four-step is

smaller. The step angle of the eight-step is half that of the single four-step and double four-step. Thus, the eight-step operating mode can keep high driving torque and improve control accuracy. In this experiment, we let the stepper motor work in the eight-step mode.

ULN2003 Module





To apply the motor in the circuit, a driver board needs to be used. Stepper Motor Driver-ULN2003 is a 7-channel inverter circuit. That is, when the input end is at high level, the output end of ULN2003 is at low level, and vice versa. If we supply high level to IN1, and low level to IN2, IN3 and IN4, then the output end OUT1 is at low level, and all the other output ends are at high level. So D1 lights up, switch SA is power on, and the stepper motor rotates one step. The similar case repeats on and on. Therefore, just give the stepper motor a specific timing sequence, it will rotate step by step. The ULN2003 here is used to provide particular timing sequences for the stepper motor.

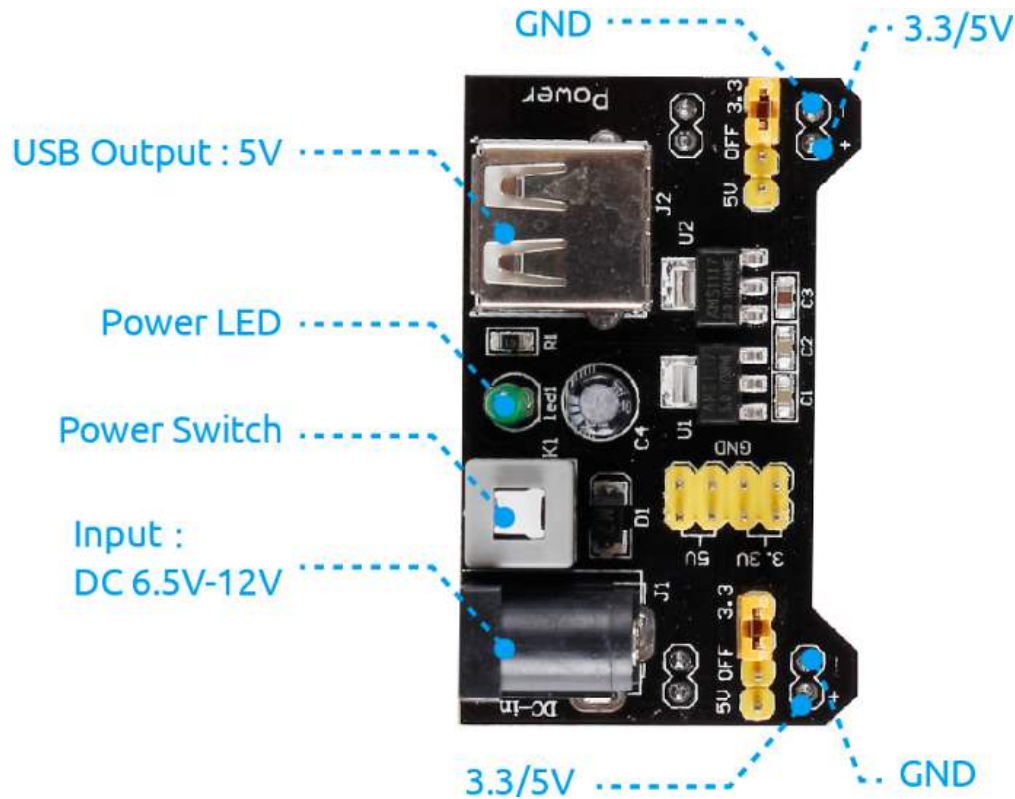
Example

- [2.14 Stepper Motor](#) (Arduino Project)
- [3.5 Access Control System](#) (Arduino Project)

1.22 Power Supply Module

When we need a large current to drive a component, which will severely interfere with the normal work of Raspberry Pi. Therefore, we separately supply power for the component by this module to make it run safely and steadily.

You can just plug it in the breadboard to supply power. It provides a voltage of 3.3V and 5V, and you can connect either via a jumper cap included.



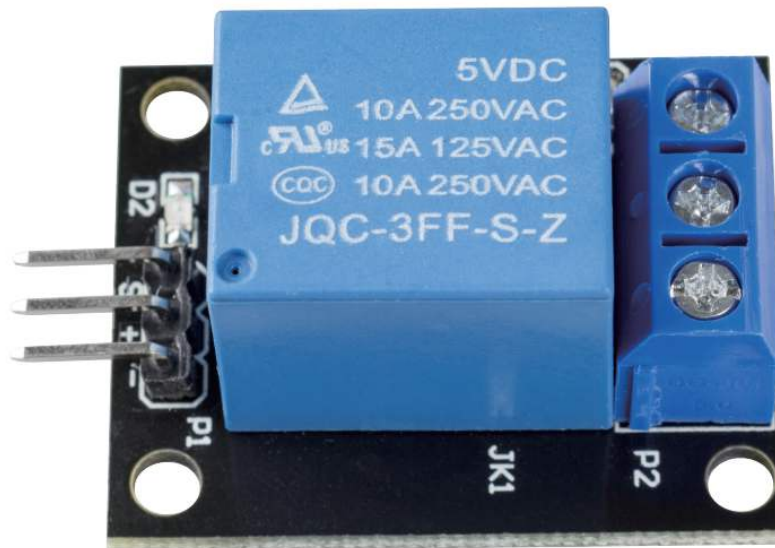
Features and specifications

- Input voltage: 6.5 - 12V
- Two Independent Channel
- Output voltage: 5V, 3.3V (adjustable via jumpers. 0V, 3.3V, and 5V configuration)
- Output current: Maximum output current 700mA
- Onboard berg male header for GND, 5V, 3.3V output
- ON-OFF Switch available.
- USB (Type-A) input available.
- DC Barrel Jack input available.
- Onboard power LED
- Dimension: 53mm x 33mm (L x W)

Example

- [2.13 Motor](#) (Arduino Project)
- [2.14 Stepper Motor](#) (Arduino Project)

1.23 Relay Module

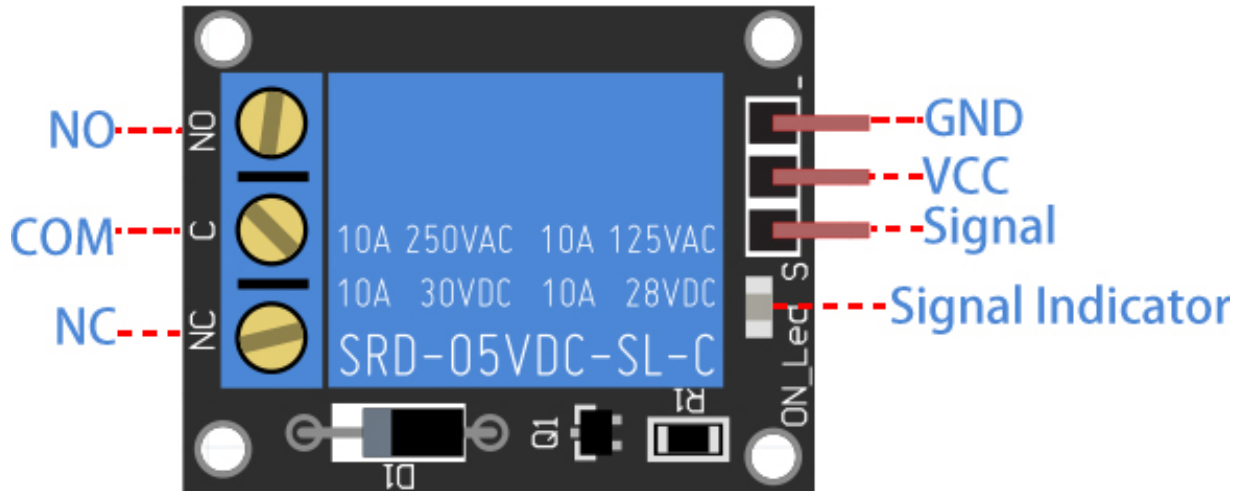


The relay module is a module consisting of a relay and a simple circuit that can be used to control large voltage devices, such as household appliances, by outputting a low voltage such as 3.3V from the control board.

1.23.1 FEATURES

- Output: 250VAC-10A, 125VAC-10A, 30VDC-10A, 28VDC-10A.
- Operating voltage 5V, suction current about 70mA.
- With signal indicator
- Input high and the relay closes, input low and the relay opens.
- Using 8050 transistor to drive the relay action
- With mounting screw holes
- PCB size: 1.8cm*4.0cm*1.9cm, weight 15g

1.23.2 PINS OUT



INPUT

It has a 1×3 (2.54mm pitch) pin header for connecting power (5V and 0V), and for controlling the relay. The pins are marked on the PCB:

- -: GND
- +: VCC
- S: Signal pin, used to control this relay. Input high and the relay closes, input low and the relay opens.

OUTPUT

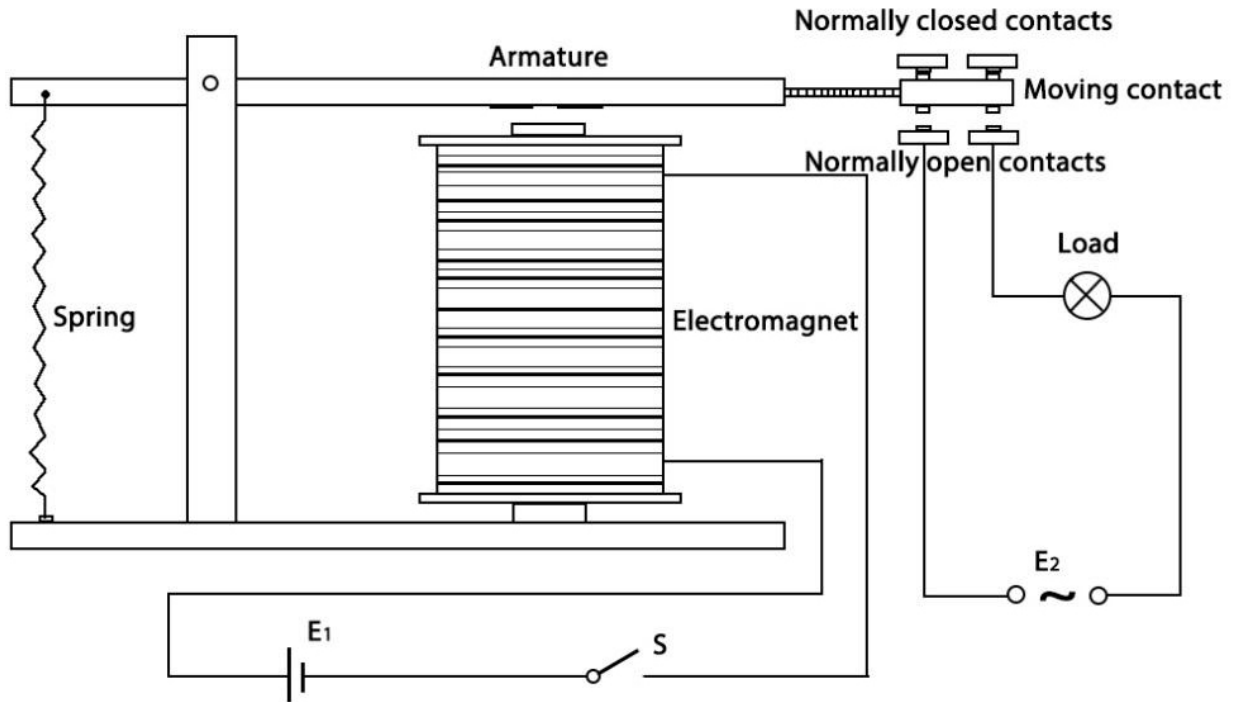
The 1 channel relay module could be considered like a series switches: 1 normally Open (NO), 1 normally closed (NC) and 1 common Pins (COM).

- **COM** - Common pin
- **NC** - Normally Closed, in which case NC is connected with COM when pin **S** is set low and disconnected when pin **S** is high.
- **NO** - Normally Open, in which case NO is disconnected with COM when pin **S** is set low and connected when pin **S** is high.

1.23.3 HOW RELAY WORKS?

As we may know, relay is a device which is used to provide connection between two or more points or devices in response to the input signal applied. In other words, relays provide isolation between the controller and the device as devices may work on AC as well as on DC. However, they receive signals from a microcontroller which works on DC hence requiring a relay to bridge the gap. Relay is extremely useful when you need to control a large amount of current or voltage with small electrical signal.

There are 5 parts in every relay:



Electromagnet - It consists of an iron core wound by coil of wires. When electricity is passed through, it becomes magnetic. Therefore, it is called electromagnet.

Armature - The movable magnetic strip is known as armature. When current flows through them, the coil is energized thus producing a magnetic field which is used to make or break the normally open (N/O) or normally close (N/C) points. And the armature can be moved with direct current (DC) as well as alternating current (AC).

Spring - When no currents flow through the coil on the electromagnet, the spring pulls the armature away so the circuit cannot be completed.

Set of electrical **contacts** - There are two contact points:

- Normally open - connected when the relay is activated, and disconnected when it is inactive.
- Normally close - not connected when the relay is activated, and connected when it is inactive.

Molded frame - Relays are covered with plastic for protection.

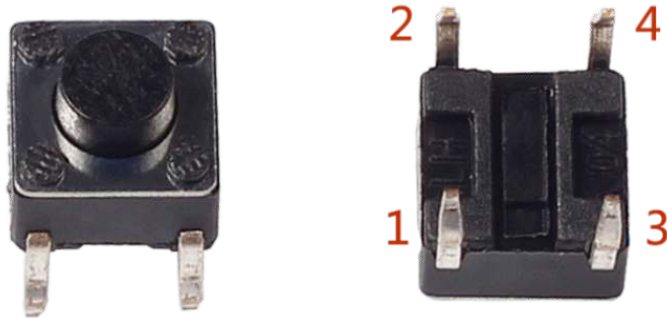
The working principle of relay is simple. When power is supplied to the relay, currents start flowing through the control coil; as a result, the electromagnet starts energizing. Then the armature is attracted to the coil, pulling down the moving contact together thus connecting with the normally open contacts. So the circuit with the load is energized. Then breaking the circuit would a similar case, as the moving contact will be pulled up to the normally closed contacts under the force of the spring. In this way, the switching on and off of the relay can control the state of a load circuit.

Example

- [2.21 Relay Module](#) (Arduino Project)
- [3.3 Overheat Monitor](#) (Arduino Project)

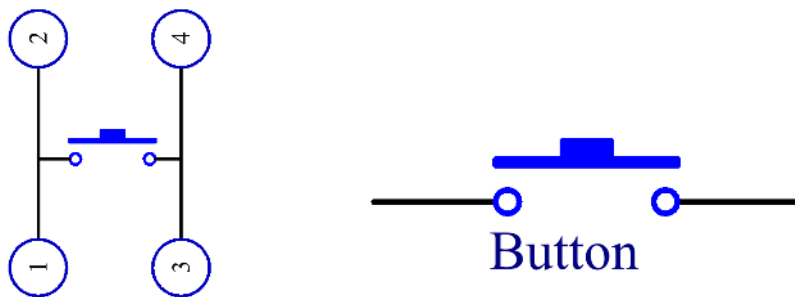
Controller

1.24 Button

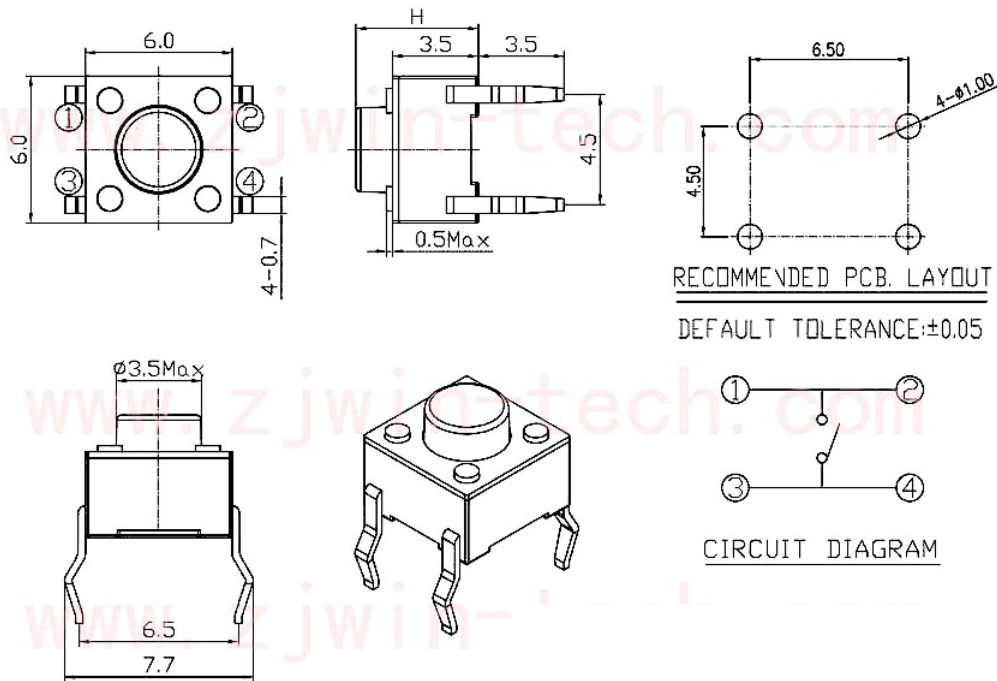


Buttons are a common component used to control electronic devices. They are usually used as switches to connect or break circuits. Although buttons come in a variety of sizes and shapes, the one used here is a 6mm mini-button as shown in the following pictures. Pin 1 is connected to pin 2 and pin 3 to pin 4. So you just need to connect either of pin 1 and pin 2 to pin 3 or pin 4.

The following is the internal structure of a button. The symbol on the right below is usually used to represent a button in circuits.



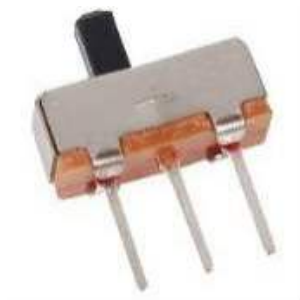
Since the pin 1 is connected to pin 2, and pin 3 to pin 4, when the button is pressed, the 4 pins are connected, thus closing the circuit.



Example

- 2.15 *Button* (Arduino Project)
- 3.3 *Overheat Monitor* (Arduino Project)
- 2.6 *Doorbell* (Scratch Project)
- 2.17 *GAME - Eat Apple* (Scratch Project)
- 2.20 *GAME - Fishing* (Scratch Project)

1.25 Slide Switch

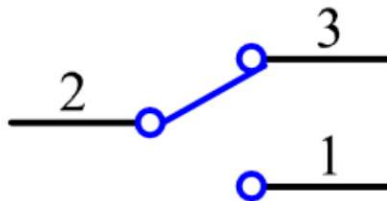


A slide switch, just as its name implies, is to slide the switch bar to connect or break the circuit, and further switch circuits. The common-used types are SPDT, SPTT, DPDT, DPTT etc. The slide switch is commonly used in low-voltage circuit. It has the features of flexibility and stability, and applies in electric instruments and electric toys widely. How it works: Set the middle pin as the fixed one. When you pull the slide to the left, the two pins on the

left are connected; when you pull it to the right, the two pins on the right are connected. Thus, it works as a switch connecting or disconnecting circuits. See the figure below:



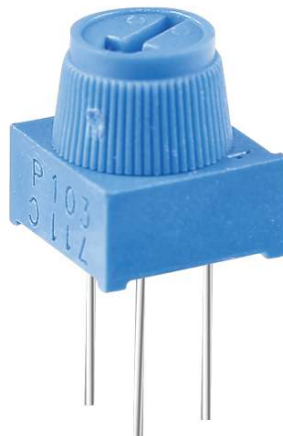
The circuit symbol of the slide switch is shown as below. The pin2 in the figure refers to the middle pin.



Example

- [2.16 Slide Switch](#) (Arduino Project)

1.26 Potentiometer

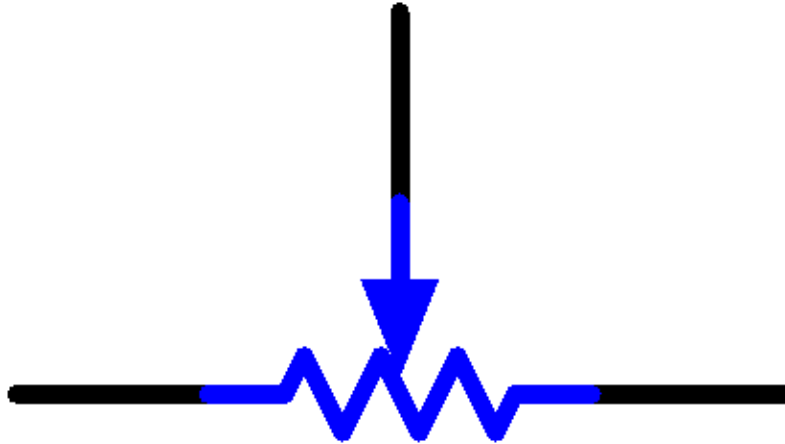


Potentiometer is also a resistance component with 3 terminals and its resistance value can be adjusted according to some regular variation.

Potentiometers come in various shapes, sizes, and values, but they all have the following things in common:

- They have three terminals (or connection points).
- They have a knob, screw, or slider that can be moved to vary the resistance between the middle terminal and either one of the outer terminals.
- The resistance between the middle terminal and either one of the outer terminals varies from 0 to the maximum resistance of the pot as the knob, screw, or slider is moved.

Here is the circuit symbol of potentiometer.



The functions of the potentiometer in the circuit are as follows:

1. Serving as a voltage divider

Potentiometer is a continuously adjustable resistor. When you adjust the shaft or sliding handle of the potentiometer, the movable contact will slide on the resistor. At this point, a voltage can be output depending on the voltage applied onto the potentiometer and the angle the movable arm has rotated to or the travel it has made.

2. Serving as a rheostat

When the potentiometer is used as a rheostat, connect the middle pin and one of the other 2 pins in the circuit. Thus you can get a smoothly and continuously changed resistance value within the travel of the moving contact.

3. Serving as a current controller

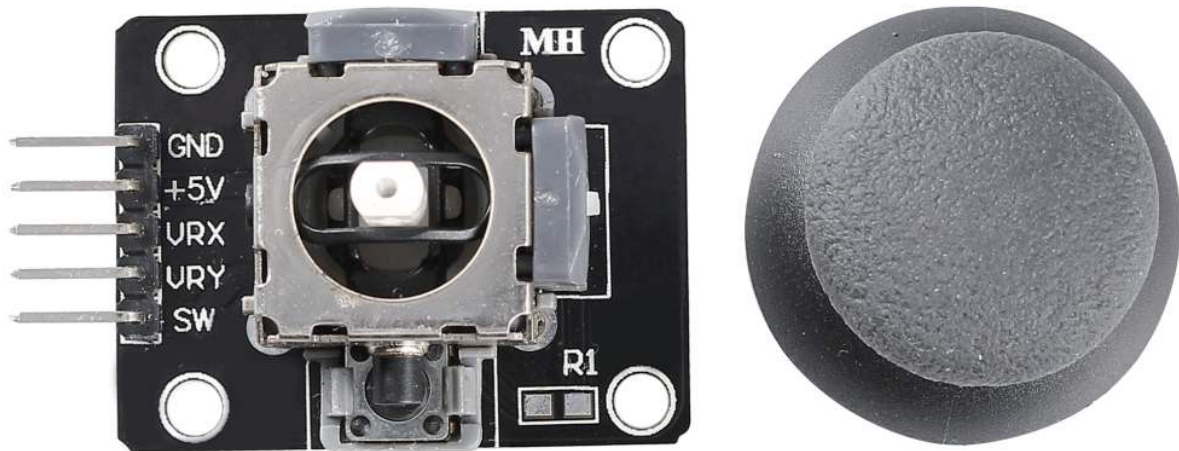
When the potentiometer acts as a current controller, the sliding contact terminal must be connected as one of the output terminals.

If you want to know more about potentiometer, refer to: [Potentiometer - Wikipedia](#)

Example

- [2.22 Potentiometer](#) (Arduino Project)
- [2.5 Moving Mouse](#) (Scratch Project)
- [2.19 GAME - Breakout Clone](#) (Scratch Project)

1.27 Joystick Module

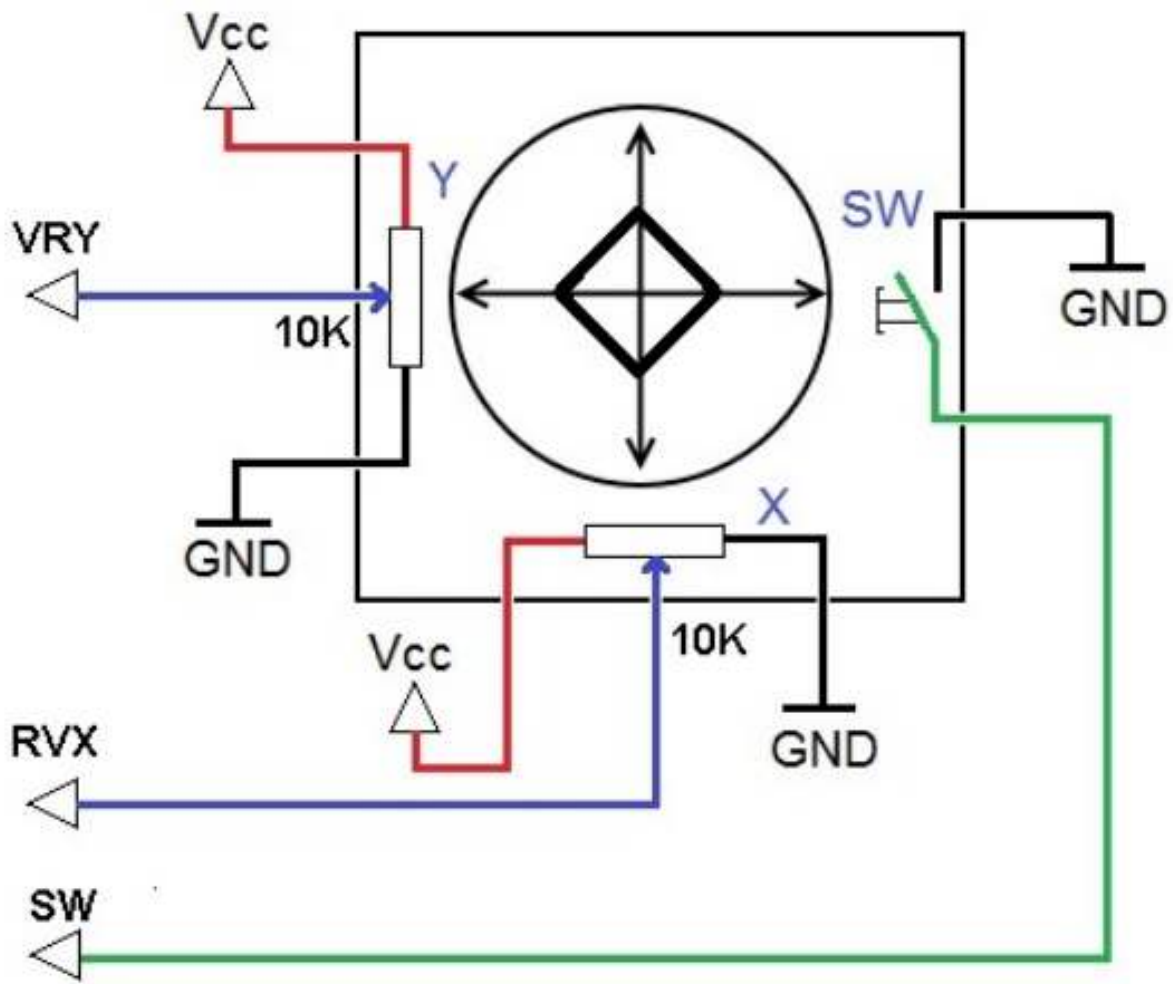


The basic idea of a joystick is to translate the movement of a stick into electronic information that a computer can process.

In order to communicate a full range of motion to the computer, a joystick needs to measure the stick's position on two axes – the X-axis (left to right) and the Y-axis (up and down). Just as in basic geometry, the X-Y coordinates pinpoint the stick's position exactly.

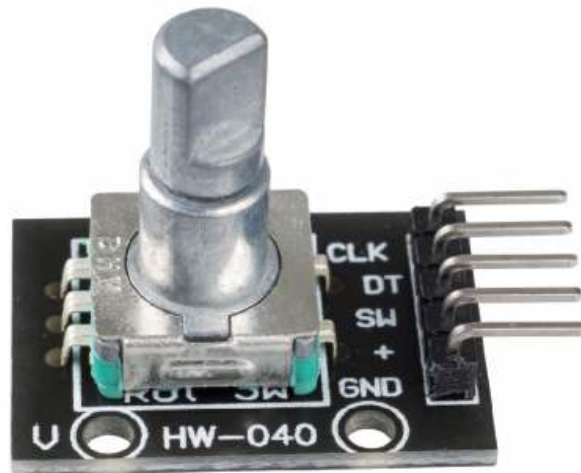
To determine the location of the stick, the joystick control system simply monitors the position of each shaft. The conventional analog joystick design does this with two potentiometers, or variable resistors.

The joystick also has a digital input that is actuated when the joystick is pressed down.

**Example**

- 2.23 *Joystick Module* (Arduino Project)
- 2.16 *GAME - Star-Crossed* (Scratch Project)
- 2.22 *GAME - Kill Dragon* (Scratch Project)

1.28 Rotary Encoder Module

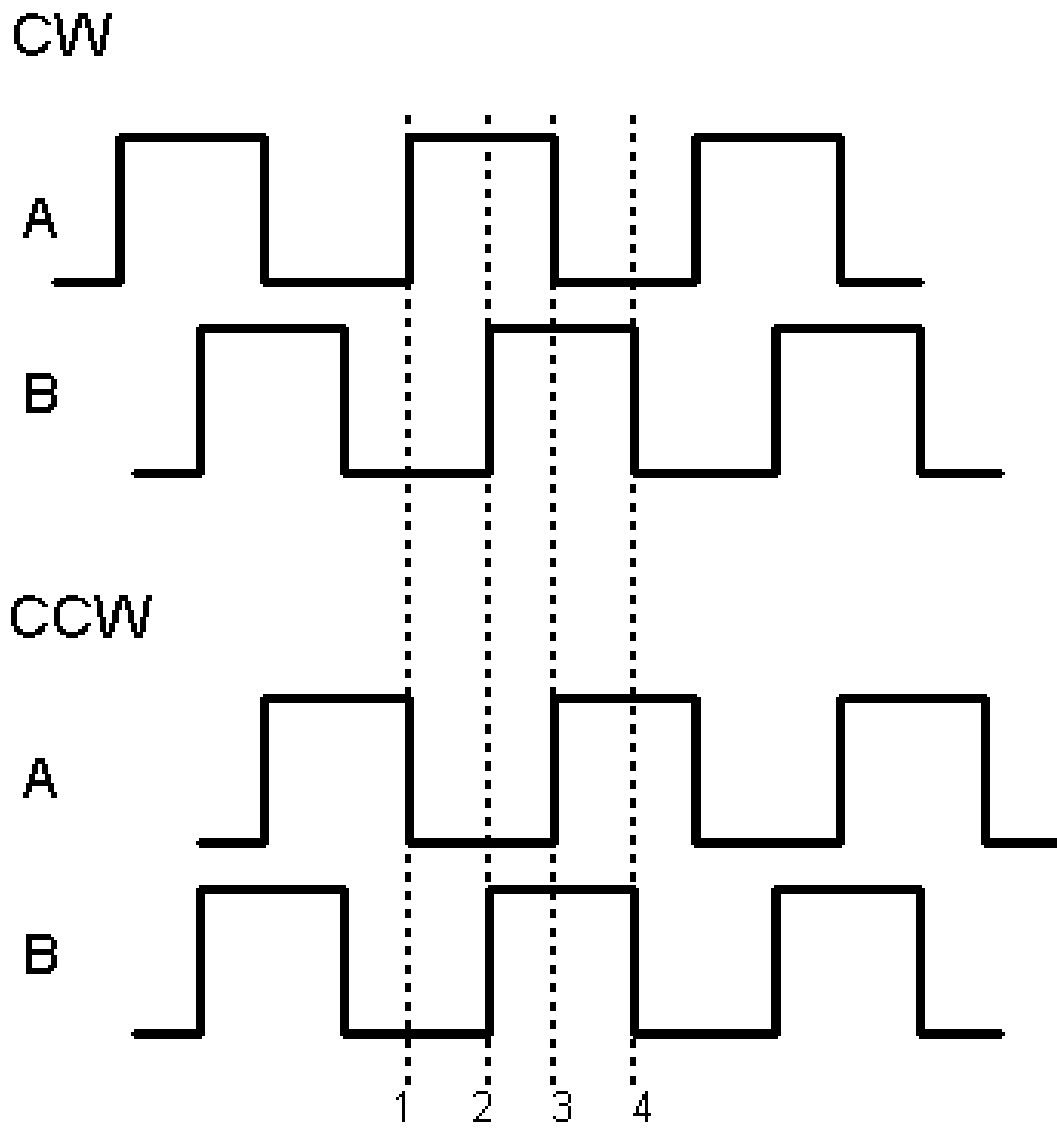


The rotary encoder module counts the number of pulses output in the forward and reverse directions during rotation. Unlike a potentiometer, this rotation count is unlimited and the number of pulses per cycle is 20. Press the key (SW) on the rotary encoder to start counting from zero.

There are mainly two types of rotary encoders: absolute and incremental (relative) encoders. An incremental one is used in this kit.

Incremental encoders give two-phase square waves, their phase difference is 90, usually called A channel and B channel.

As shown on the right, when channel A changes from high level to low level, if channel B is high level, it indicates the rotary encoder spins clockwise (CW); if at that moment channel B is low level, it means spins counterclockwise (CCW). So if we read the value of channel B when channel A is low level, we can know in which direction the rotary encoder rotates.

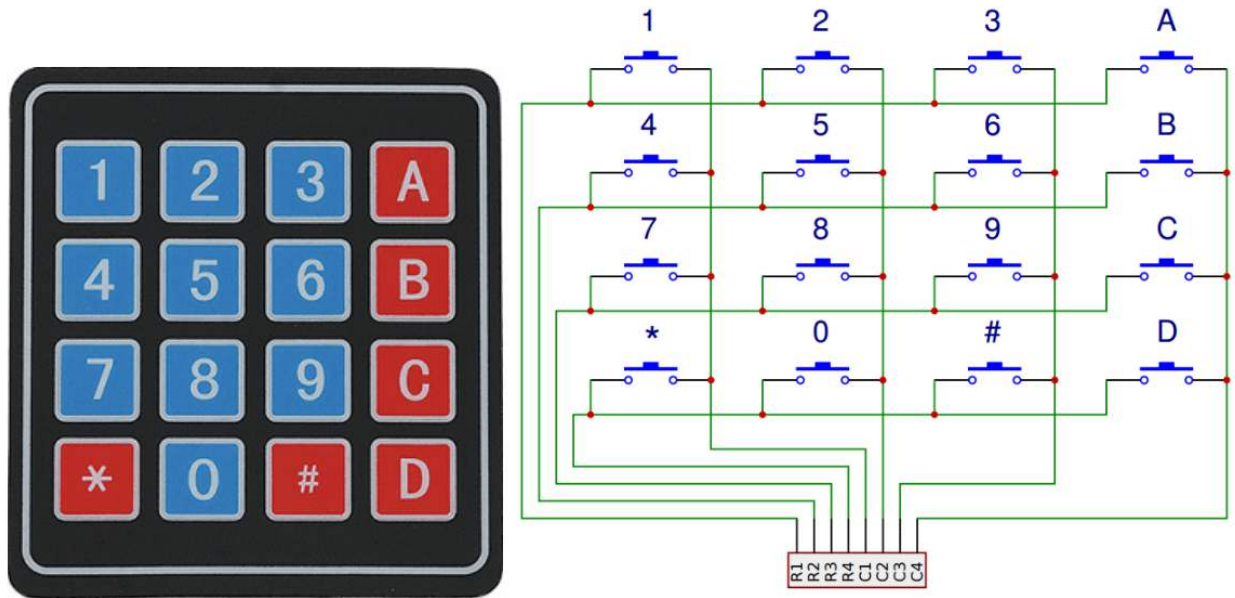


Example

- [2.25 Rotary Encoder Module \(Arduino Project\)](#)

1.29 Keypad

A keypad is a rectangular array of 12 or 16 OFF-(ON) buttons. Their contacts are accessed via a header suitable for connection with a ribbon cable or insertion into a printed circuit board. In some keypads, each button connects with a separate contact in the header, while all the buttons share a common ground.



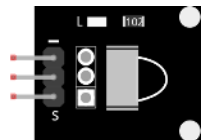
More often, the buttons are matrix encoded, meaning that each of them bridges a unique pair of conductors in a matrix. This configuration is suitable for polling by a microcontroller, which can be programmed to send an output pulse to each of the four horizontal wires in turn. During each pulse, it checks the remaining four vertical wires in sequence, to determine which one, if any, is carrying a signal. Pullup or pulldown resistors should be added to the input wires to prevent the inputs of the microcontroller from behaving unpredictably when no signal is present.

Example

- [2.19 Keypad](#) (Arduino Project)
- [3.5 Access Control System](#) (Arduino Project)

1.30 IR Receiver Module

IR Receiver Module



- S: Signal output
- +VCC
- -: GND

An infrared-receiver is a component which receives infrared signals and can independently receive infrared rays and output signals compatible with TTL level. It is similar with a normal plastic-packaged transistor in size and is suitable

for all kinds of infrared remote control and infrared transmission.

Infrared, or IR, communication is a popular, low-cost, easy-to-use wireless communication technology. Infrared light has a slightly longer wavelength than visible light, so it is imperceptible to the human eye - ideal for wireless communication. A common modulation scheme for infrared communication is 38KHz modulation.

- Adopted HX1838 IR Receiver Sensor, high sensitivity
- Can be used for remote control
- Power Supply: 5V
- Interface: Digital
- Modulate Frequency: 38Khz
- Pin Definitions: (1) Output (2) Vcc (3) GND
- Size: 23.5mm x 21.5mm

Remote Control



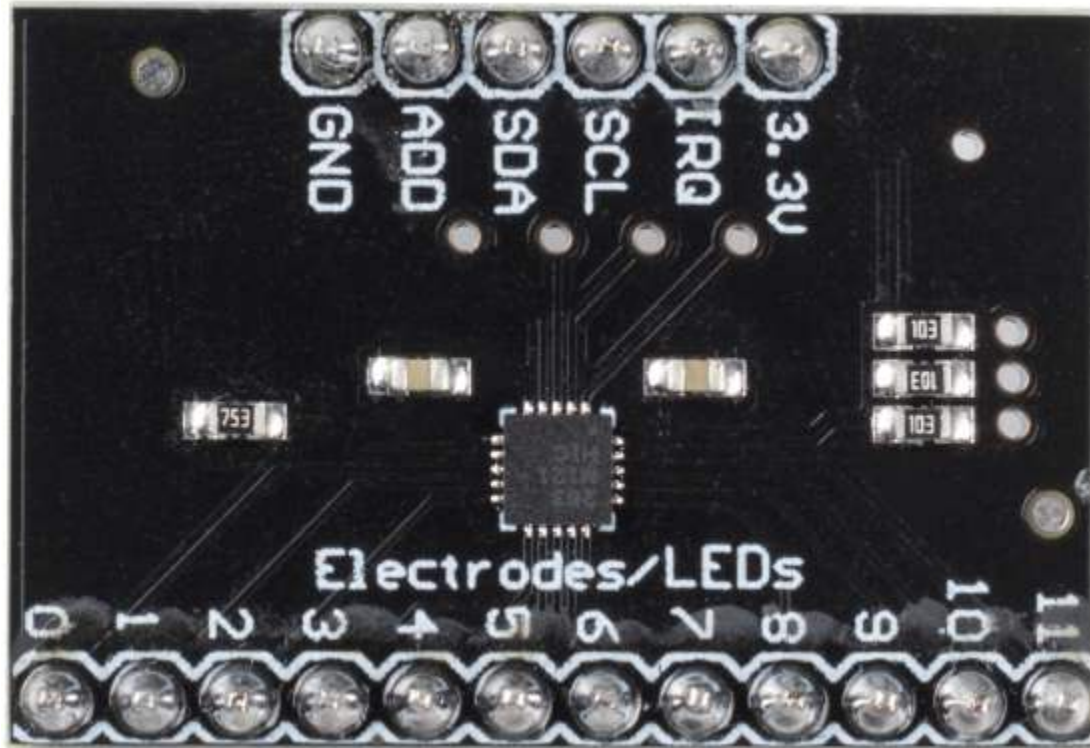
This is a Mini thin infrared wireless remote control with 21 function buttons and a transmitting distance of up to 8 meters, which is suitable for operating a wide range of devices in a kid's room.

- Size: 85x39x6mm
- Remote control range: 8-10m
- Battery: 3V button type lithium manganese battery
- Infrared carrier frequency: 38KHz
- Surface paste material: 0.125mm PET
- Effective life: more than 20,000 times

Example

- [2.20 IR Receiver Module](#) (Arduino Project)
- [3.4 Guess Number](#) (Arduino Project)

1.31 MPR121



- **3.3V**: Power supply
- **IRQ**: Open Collector Interrupt Output Pin, active low
- **SCL**: I2C Clock
- **SDA**: I2C Data
- **ADD**: I2C Address Select Input Pin. Connect the ADDR pin to the VSS, VDD, SDA or SCL line, the resulting I2C addresses are 0x5A, 0x5B, 0x5C and 0x5D respectively
- **GND**: Ground
- **0~11**: Electrode 0~11, electrode is a touch sensor. Typically, electrodes can just be some piece of metal, or a wire. But some times depending on the length of our wire, or the material the electrode is on, it can make triggering the sensor difficult. For this reason, the MPR121 allows you to configure what is needed to trigger and untrigger an electrode.

MPR121 OVERVIEW

The MPR121 is the second generation capacitive touch sensor controller after the initial release of the MPR03x series devices. The MPR121 features increased internal intelligence, some of the major additions include an increased electrode count, a hardware configurable I2C address, an expanded filtering system with debounce, and completely independent electrodes with auto-configuration built in. The device also features a 13th simulated sensing channel dedicated for near proximity detection using the multiplexed sensing inputs.

- [MPR121 Datasheet](#)

Features

- **Low power operation**

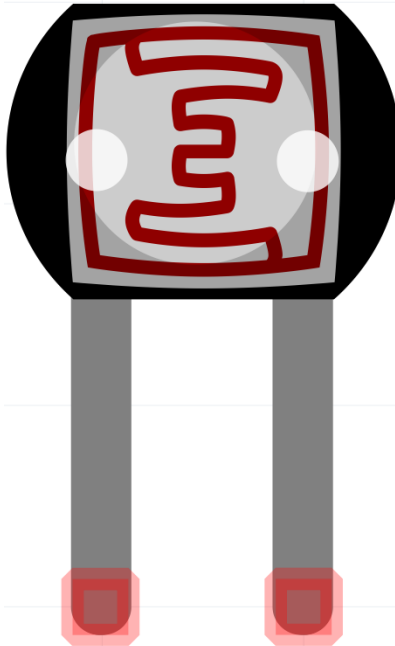
- 1.71 V to 3.6 V supply operation
- 29 A supply current at 16 ms sampling interval period
- 3 A Stop mode current
- **12 capacitance sensing inputs**
 - 8 inputs are multifunctional for LED driver and GPIO
- **Complete touch detection**
 - Auto-configuration for each sensing input
 - Auto-calibration for each sensing input
 - Touch/release threshold and debounce for touch detection
- I2C interface, with Interrupt output
- 3 mm x 3 mm x 0.65 mm 20 lead QFN package
- -40°C to +85°C operating temperature range

Example

- [2.24 MPR121 Module](#) (Arduino Project)

Sensor

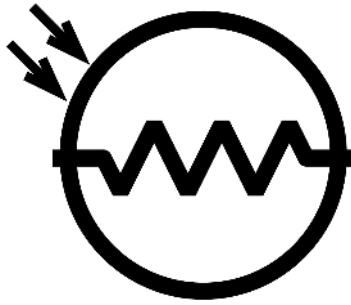
1.32 Photoresistor



A photoresistor or photocell is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity; in other words, it exhibits photo conductivity.

A photoresistor can be applied in light-sensitive detector circuits and light-activated and dark-activated switching circuits acting as a resistance semiconductor. In the dark, a photoresistor can have a resistance as high as several megaohms (M), while in the light, a photoresistor can have a resistance as low as a few hundred ohms.

Here is the electronic symbol of photoresistor.

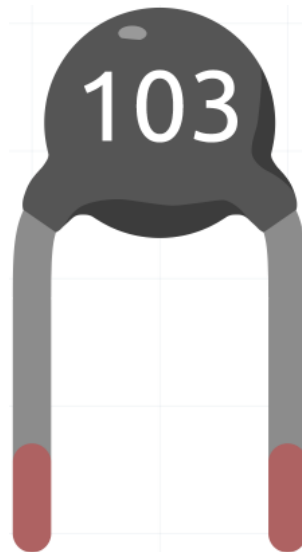


- [Photoresistor - Wikipedia](#)

Example

- [2.26 Photoresistor](#) (Arduino Project)
- [2.9 Light Alarm Clock](#) (Scratch Project)

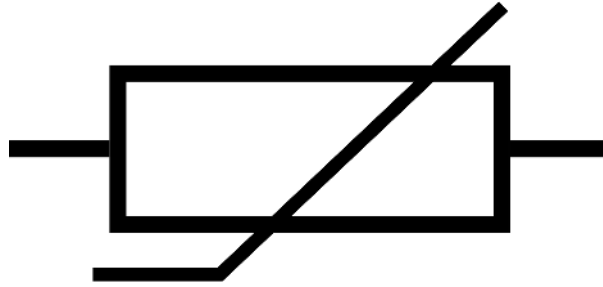
1.33 Thermistor



A thermistor is a type of resistor whose resistance is strongly dependent on temperature, more so than in standard resistors. The word is a combination of thermal and resistor. Thermistors are widely used as inrush current limiters, temperature sensors (negative temperature coefficient or NTC type typically), self-resetting overcurrent protectors, and self-regulating heating elements (positive temperature coefficient or PTC type typically).

- [Thermistor - Wikipedia](#)

Here is the electronic symbol of thermistor.



Thermistors are of two opposite fundamental types:

- With NTC thermistors, resistance decreases as temperature rises usually due to an increase in conduction electrons bumped up by thermal agitation from valency band. An NTC is commonly used as a temperature sensor, or in series with a circuit as an inrush current limiter.
- With PTC thermistors, resistance increases as temperature rises usually due to increased thermal lattice agitations particularly those of impurities and imperfections. PTC thermistors are commonly installed in series with a circuit, and used to protect against overcurrent conditions, as resettable fuses.

In this kit we use an NTC one. Each thermistor has a normal resistance. Here it is 10k ohm, which is measured under 25 degree Celsius.

Here is the relation between the resistance and temperature:

$$RT = RN * \exp(B(1/TK - 1/TN))$$

- **RT** is the resistance of the NTC thermistor when the temperature is TK.
- **RN** is the resistance of the NTC thermistor under the rated temperature TN. Here, the numerical value of RN is 10k.
- **TK** is a Kelvin temperature and the unit is K. Here, the numerical value of TK is 273.15 + degree Celsius.
- **TN** is a rated Kelvin temperature; the unit is K too. Here, the numerical value of TN is 273.15+25.
- And **B(beta)**, the material constant of NTC thermistor, is also called heat sensitivity index with a numerical value 3950.
- **exp** is the abbreviation of exponential, and the base number e is a natural number and equals 2.7 approximately.

Convert this formula $TK = 1 / (\ln(RT/RN) / B + 1 / TN)$ to get Kelvin temperature that minus 273.15 equals degree Celsius.

This relation is an empirical formula. It is accurate only when the temperature and resistance are within the effective range.

Example

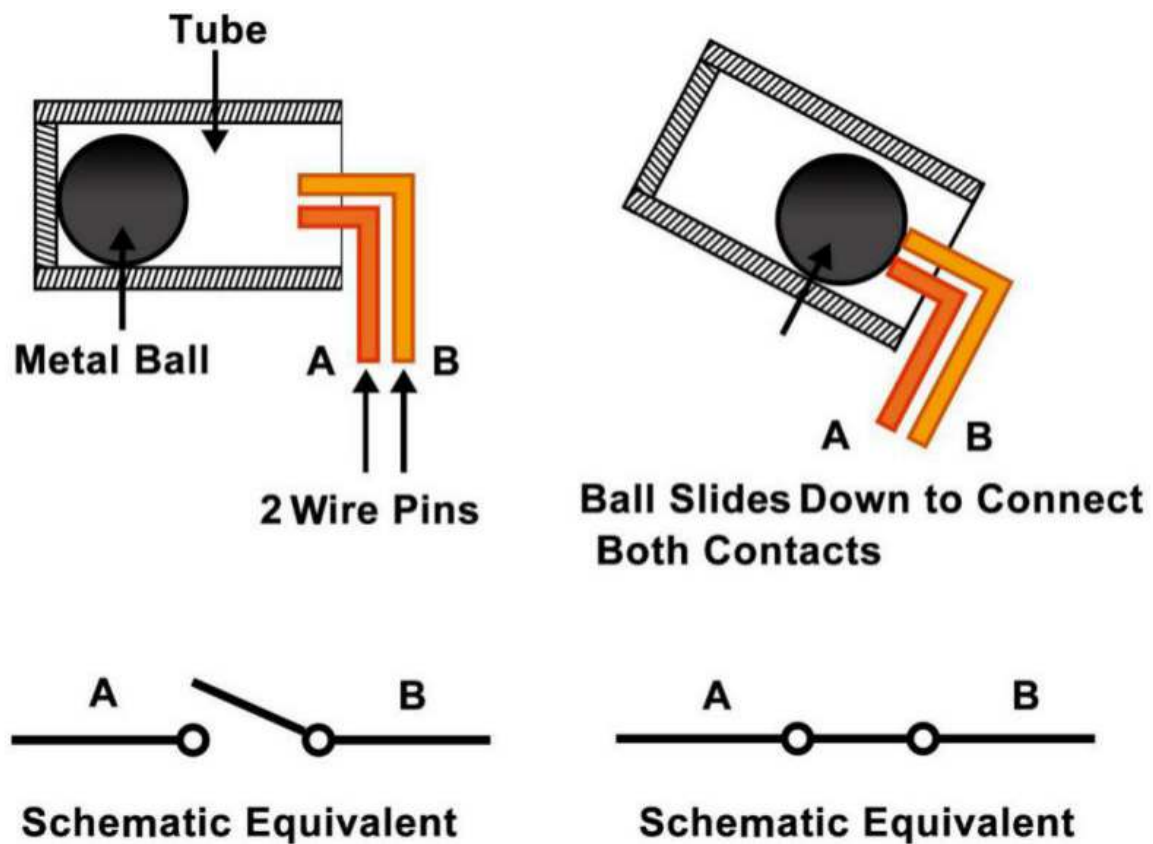
- [2.27 Thermistor](#) (Arduino Project)
- [2.8 Low Temperature Alarm](#) (Scratch Project)

1.34 Tilt Switch



The tilt switch used here is a ball one with a metal ball inside. It is used to detect inclinations of a small angle.

The principle is very simple. When the switch is tilted in a certain angle, the ball inside rolls down and touches the two contacts connected to the pins outside, thus triggering circuits. Otherwise the ball will stay away from the contacts, thus breaking the circuits.

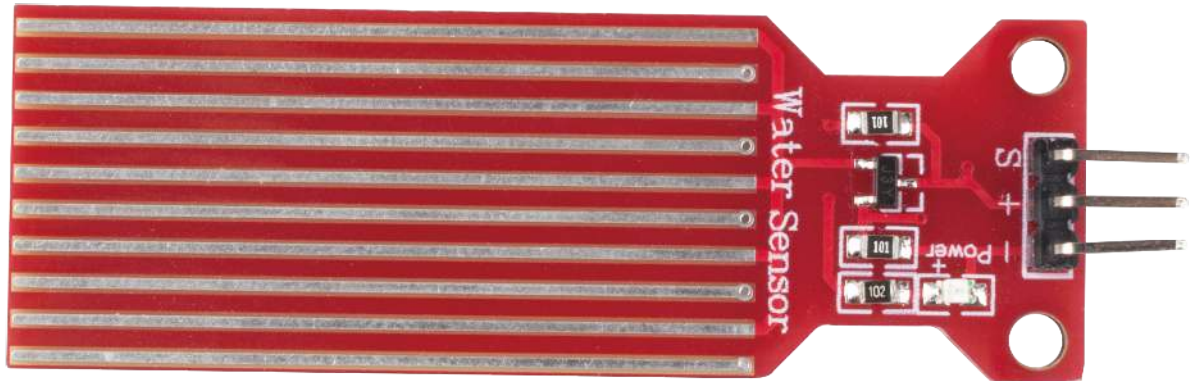


- [SW520D Tilt Switch Datasheet](#)

Example

- [2.17 Tilt Switch](#) (Arduino Project)
- [2.7 Tumbler](#) (Scratch Project)

1.35 Water Level Sensor Module



- S: Signal pin, the more water the sensor is immersed in, the greater the output value.
- +: Power supply, 3.3 ~ 5V DC.
- -: Ground.

The Water Level Sensor module is an easy-to-use, compact and cost effective water level/drop detection sensor that measures the water level by a series of exposed parallel wire traces to determine the size of the water drop/volume.

The water level sensor has ten exposed copper traces, five for the Power traces and five for the Sensor traces, which are crossed and bridged by water when flooded. The circuit board has a power LED that lights up when the board is energized.

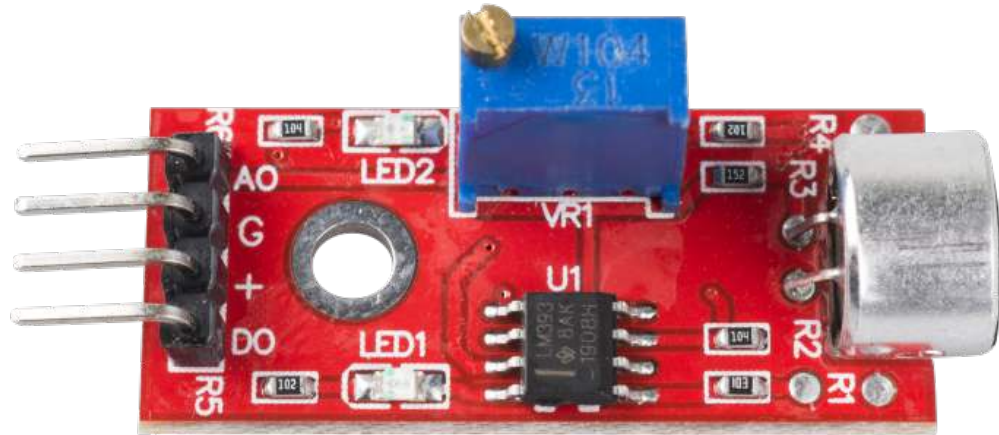
The combination of these traces acts like a variable resistor, changing the resistance value according to the water level. To be more precise, the more water the sensor is immersed in, the better the conductivity and the lower the resistance. Conversely, the less conductive it is, the higher the resistance. Next, the sensor will process the output signal voltage which will be sent to the microcontroller, thus helping us to determine the water level.

Warning: The sensor cannot be fully submerged in water, please only leave the part where the ten traces are located in contact with water. In addition, energizing the sensor in a humid environment will speed up the corrosion of the probe and cut the life of the sensor, so we recommend that you only supply power when taking readings.

Example

- [2.29 Water Sensor Module](#) (Arduino Project)
- [2.21 Catching Starfish](#) (Scratch Project)

1.36 Sound Sensor Module



A sound sensor is defined as a module that detects sound waves through its intensity and converting it to electrical signals.

This module can be used for security, switch, and monitoring applications. Its accuracy can be easily adjusted for the convenience of usage.

It uses a microphone which supplies the input to an amplifier, peak detector and buffer. When the sensor detects a sound, it processes an output signal voltage which is sent to a micro-controller then performs necessary processing.

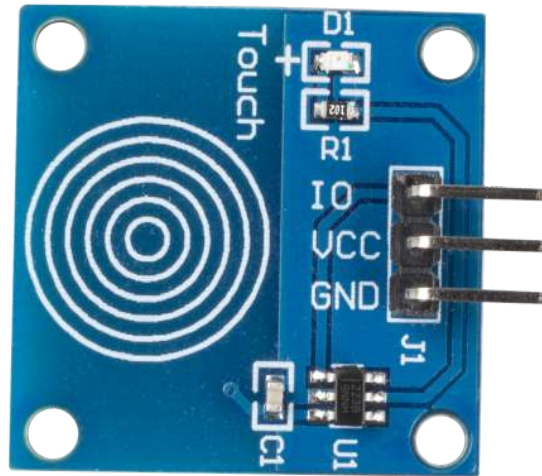
This module has two outputs:

- **AO:** analog output, real-time output voltage signal of microphone.
- **DO:** when the intensity of the sound reaches a certain threshold, the output is a high or low level signal. The threshold sensitivity can be achieved by adjusting the potentiometer.

Example

- [2.28 Sound Sensor Module](#) (Arduino Project)
- [2.13 Blow Ball](#) (Scratch Project)

1.37 Touch Switch Module

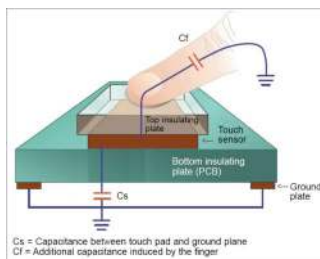


- **IO**: Signal pin, usually low level, will output high level after touch.
- **VCC**: Power supply, 3.3 ~ 5V DC.
- **GND**: Ground.

This module is a capacitive touch switch module based on a touch sensor IC (TTP223B). In the normal state, the module outputs a low level with low power consumption; when a finger touches the corresponding position, the module outputs a high level and becomes low level again after the finger is released.

Here is how the capacitive touch switch works:

A capacitive touch switch has different layers—top insulating face plate followed by touch plate, another insulating layer and then ground plate.



In practice, a capacitive sensor can be made on a double-sided PCB by regarding one side as the touch sensor and the opposite side as ground plate of the capacitor. When power is applied across these plates, the two plates get charged. In equilibrium state, the plates have the same voltage as the power source.

The touch detector circuit has an oscillator whose frequency is dependent on capacitance of the touchpad. When a finger is moved close to the touchpad, additional capacitance causes frequency of this internal oscillator to change. The detector circuit tracks oscillator frequency at timed intervals, and when the shift crosses the threshold change, the circuit triggers a key-press event.

Example

- [2.18 Touch Switch Module](#) (Arduino Project)
- [2.15 GAME - Inflating the Balloon](#) (Scratch Project)

1.38 Obstacle Avoidance Module

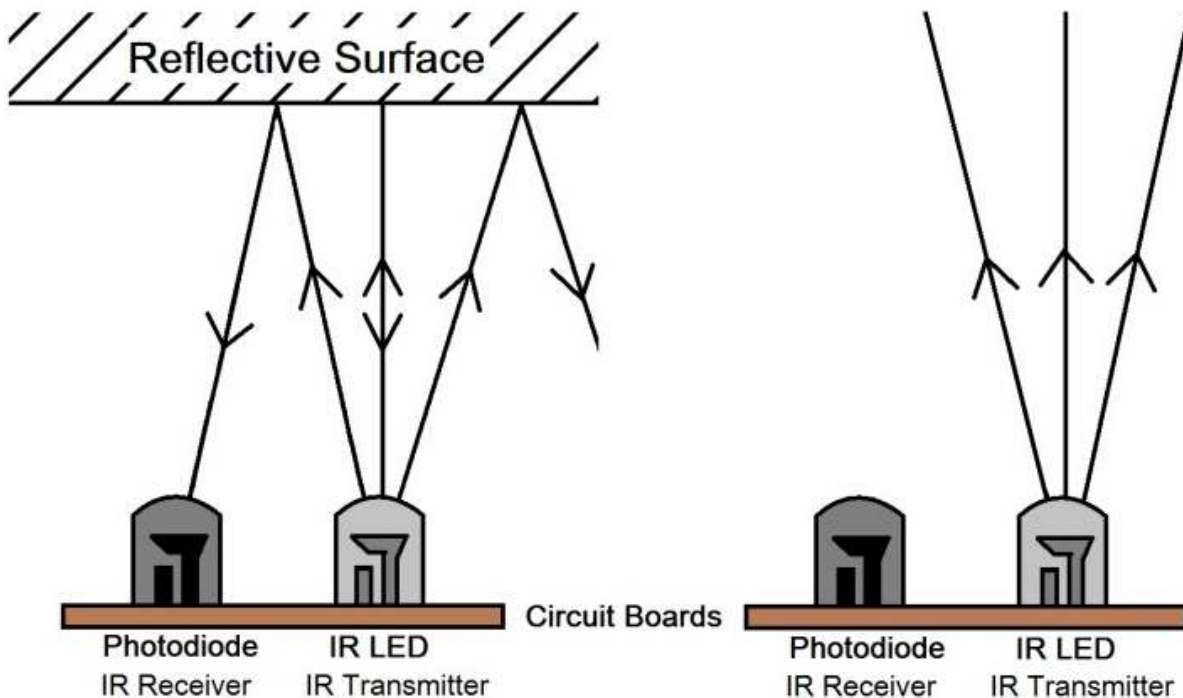


- **VCC:** Power supply, 3.3 ~ 5V DC.
- **GND:** Ground
- **OUT:** Signal pin, usually high level, and low level when an obstacle is detected.

The IR obstacle avoidance module has strong adaptability to environmental light, it has a pair of infrared transmitting and receiving tubes.

The transmitting tube emits infrared frequency, when the detection direction encounters an obstacle, the infrared radiation is received by the receiving tube, after the comparator circuit processing, the green indicator will light up and output low level signal.

The detection distance can be adjusted by potentiometer, the effective distance range 2-30cm.



Example

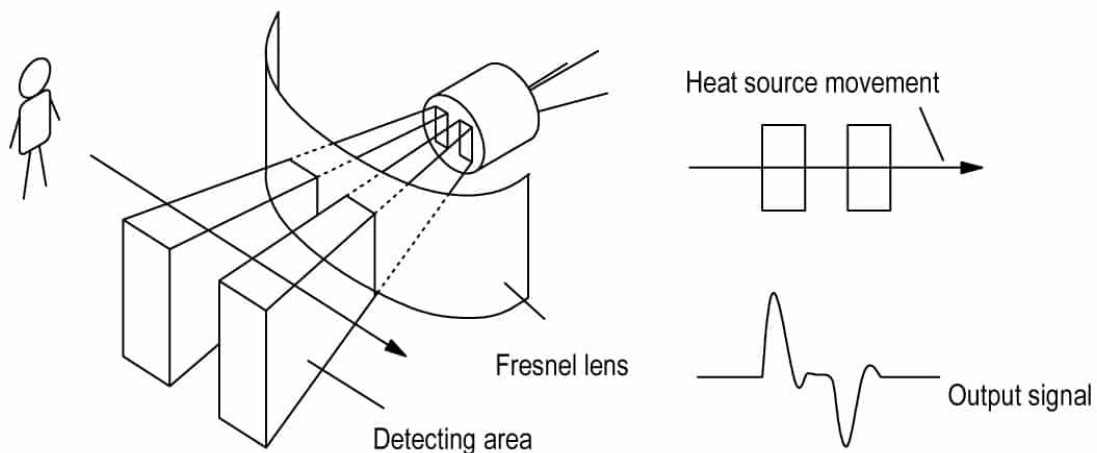
- [2.30 IR Obstacle Avoidance Sensor](#) (Arduino Project)
- [2.14 GAME - Shooting](#) (Scratch Project)

1.39 PIR Motion Sensor Module

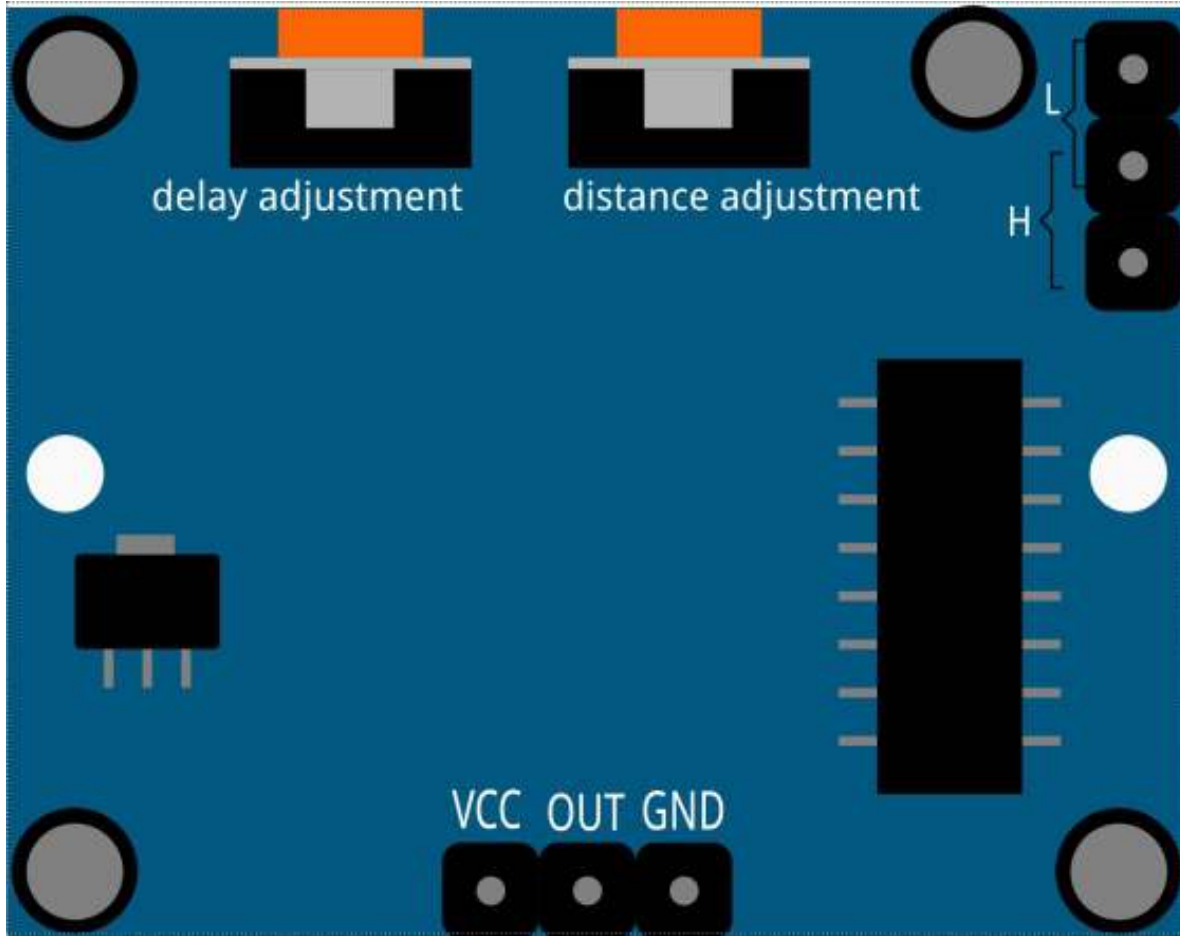


The PIR sensor detects infrared heat radiation that can be used to detect the presence of organisms that emit infrared heat radiation.

The PIR sensor is split into two slots that are connected to a differential amplifier. Whenever a stationary object is in front of the sensor, the two slots receive the same amount of radiation and the output is zero. Whenever a moving object is in front of the sensor, one of the slots receives more radiation than the other, which makes the output fluctuate high or low. This change in output voltage is a result of detection of motion.



After the sensing module is wired, there is a one-minute initialization. During the initialization, module will output for 0~3 times at intervals. Then the module will be in the standby mode. Please keep the interference of light source and other sources away from the surface of the module so as to avoid the misoperation caused by the interfering signal. Even you'd better use the module without too much wind, because the wind can also interfere with the sensor.



Distance Adjustment

Turning the knob of the distance adjustment potentiometer clockwise, the range of sensing distance increases, and the maximum sensing distance range is about 0-7 meters. If turn it anticlockwise, the range of sensing distance is reduced, and the minimum sensing distance range is about 0-3 meters.

Delay adjustment

Rotate the knob of the delay adjustment potentiometer clockwise, you can also see the sensing delay increasing. The maximum of the sensing delay can reach up to 300s. On the contrary, if rotate it anticlockwise, you can shorten the delay with a minimum of 5s.

Two Trigger Modes

Choosing different modes by using the jumper cap.

- **H**: Repeatable trigger mode, after sensing the human body, the module outputs high level. During the subsequent delay period, if somebody enters the sensing range, the output will keep being the high level.
- **L**: Non-repeatable trigger mode, outputs high level when it senses the human body. After the delay, the output will change from high level into low level automatically.

Example

- *2.31 PIR Module* (Arduino Project)

1.40 Ultrasonic Module

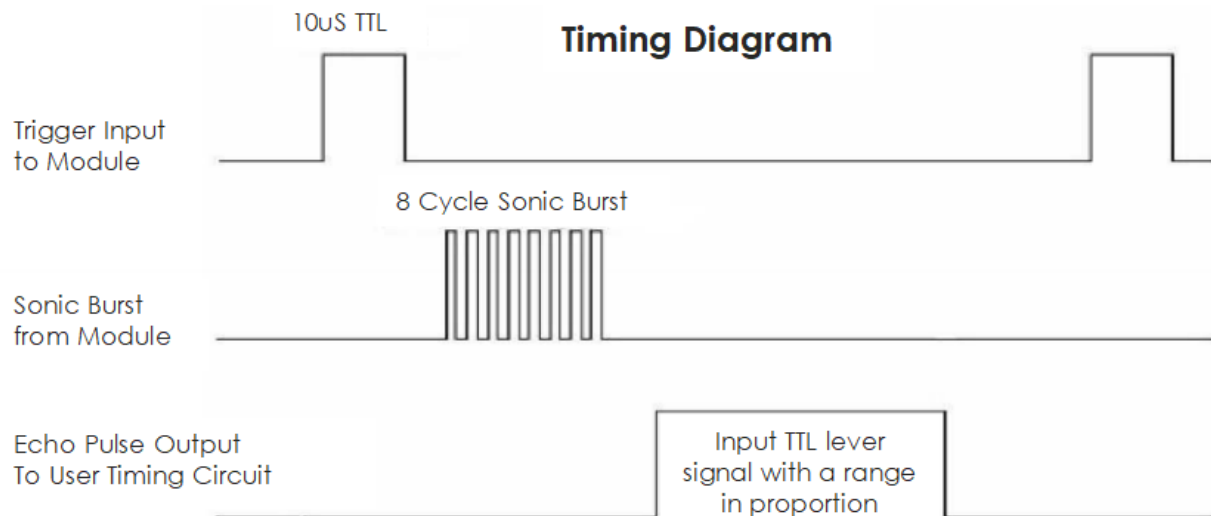


Ultrasonic ranging module provides 2cm - 400cm non-contact measurement function, and the ranging accuracy can reach to 3mm. It can ensure that the signal is stable within 5m, and the signal is gradually weakened after 5m, till the 7m position disappears.

The module includes ultrasonic transmitters, receiver and control circuit. The basic principles are as follows:

1. Use an IO flip-flop to process a high level signal of at least 10us.
2. The module automatically sends eight 40khz and detects if there is a pulse signal return.
3. If the signal returns, passing the high level, the high output IO duration is the time from the transmission of the ultrasonic wave to the return of it. Here, test distance = (high time x sound speed (340 m / s) / 2.

The timing diagram is shown below.



You only need to supply a short 10us pulse for the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. You can calculate the range through the time interval between sending trigger signal and receiving echo signal.

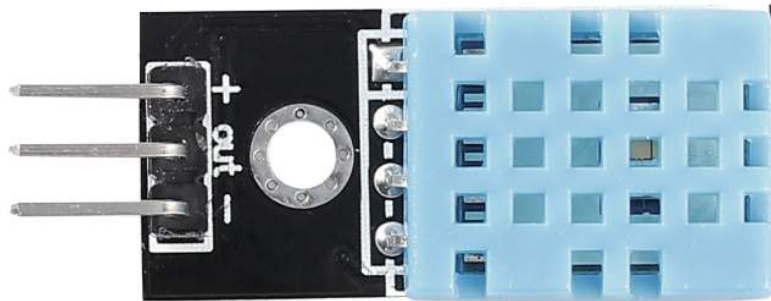
Formula: $us / 58 = \text{centimeters}$ or $us / 148 = \text{inch}$; or: the range = high level time * velocity (340M/S) / 2; you are

suggested to use measurement cycle over 60ms in order to prevent signal collisions of trigger signal and the echo signal.

Example

- [2.33 Ultrasonic Module](#) (Arduino Project)
- [3.1 Reversing Aid](#) (Arduino Project)
- [2.18 GAME - Flappy Parrot](#) (Scratch Project)

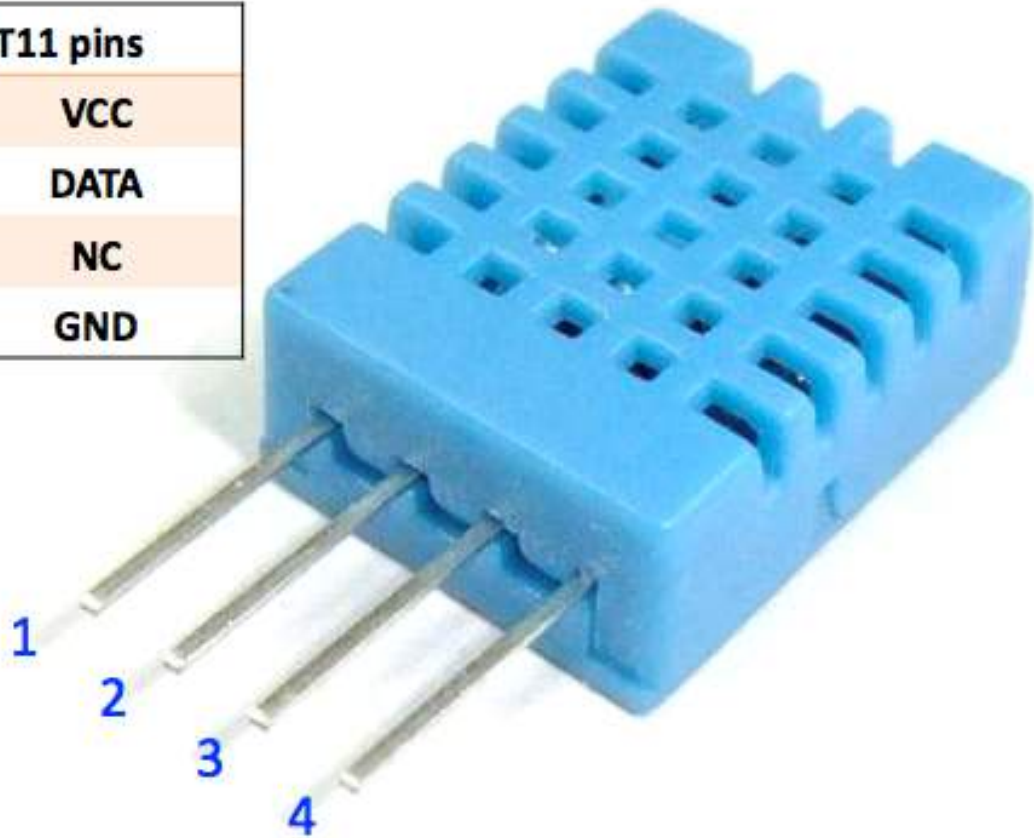
1.41 Humiture Sensor Module



The digital temperature and humidity sensor DHT11 is a composite sensor that contains a calibrated digital signal output of temperature and humidity. The technology of a dedicated digital modules collection and the temperature and humidity sensing technology are applied to ensure that the product has high reliability and excellent long-term stability.

Only three pins are available for use: VCC, GND, and DATA. The communication process begins with the DATA line sending start signals to DHT11, and DHT11 receives the signals and returns an answer signal. Then the host receives the answer signal and begins to receive 40-bit humiture data (8-bit humidity integer + 8-bit humidity decimal + 8-bit temperature integer + 8-bit temperature decimal + 8-bit checksum).

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND

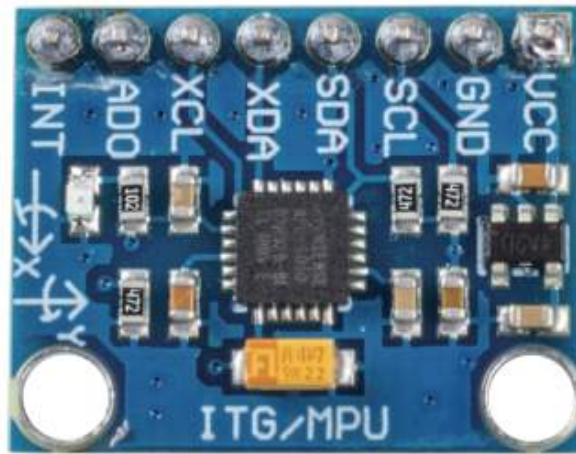


- [DHT11 Datasheet](#)

Example

- [2.32 DHT11 Module](#) (Arduino Project)
- [2.10 Read Temperature and Humidity](#) (Scratch Project)

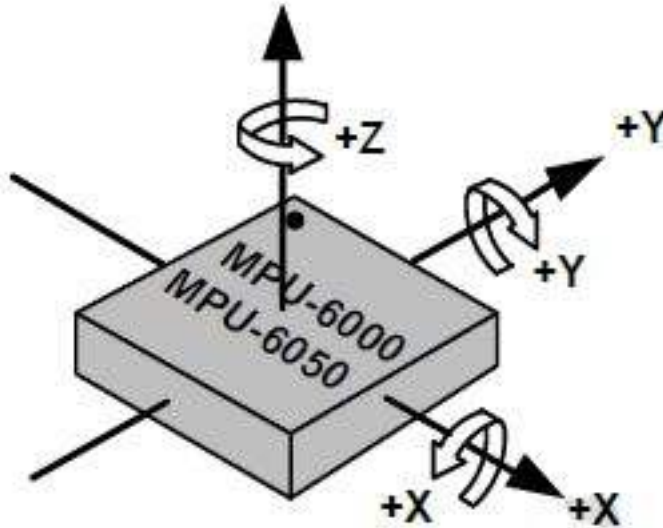
1.42 MPU6050 Module



The MPU-6050 is a 6-axis (combines 3-axis Gyroscope, 3-axis Accelerometer) motion tracking device.

Its three coordinate systems are defined as follows:

Put MPU6050 flat on the table, assure that the face with label is upward and a dot on this surface is on the top left corner. Then the upright direction upward is the z-axis of the chip. The direction from left to right is regarded as the X-axis. Accordingly the direction from back to front is defined as the Y-axis.

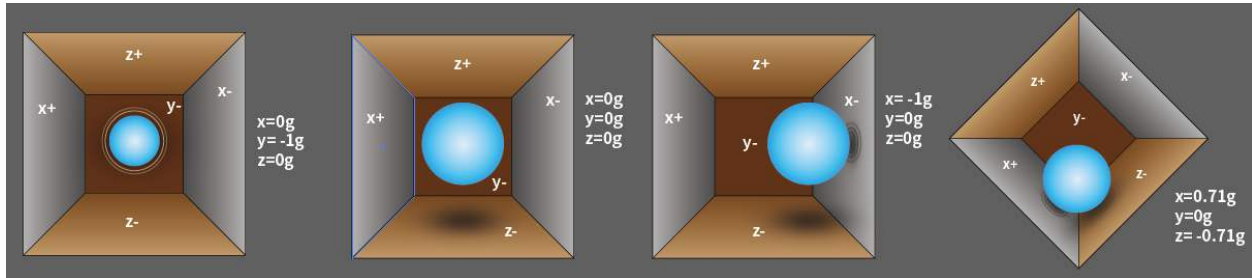


3-axis Accelerometer

The accelerometer works on the principle of piezo electric effect, the ability of certain materials to generate an electric charge in response to applied mechanical stress.

Here, imagine a cuboidal box, having a small ball inside it, like in the picture above. The walls of this box are made with piezo electric crystals. Whenever you tilt the box, the ball is forced to move in the direction of the inclination,

due to gravity. The wall with which the ball collides, creates tiny piezo electric currents. There are totally, three pairs of opposite walls in a cuboid. Each pair corresponds to an axis in 3D space: X, Y and Z axes. Depending on the current produced from the piezo electric walls, we can determine the direction of inclination and its magnitude.



We can use the MPU6050 to detect its acceleration on each coordinate axis (in the stationary desktop state, the Z-axis acceleration is 1 gravity unit, and the X and Y axes are 0). If it is tilted or in a weightless/overweight condition, the corresponding reading will change.

There are four kinds of measuring ranges that can be selected programmatically: +/-2g, +/-4g, +/-8g, and +/-16g (2g by default) corresponding to each precision. Values range from -32768 to 32767.

The reading of accelerometer is converted to an acceleration value by mapping the reading from the reading range to the measuring range.

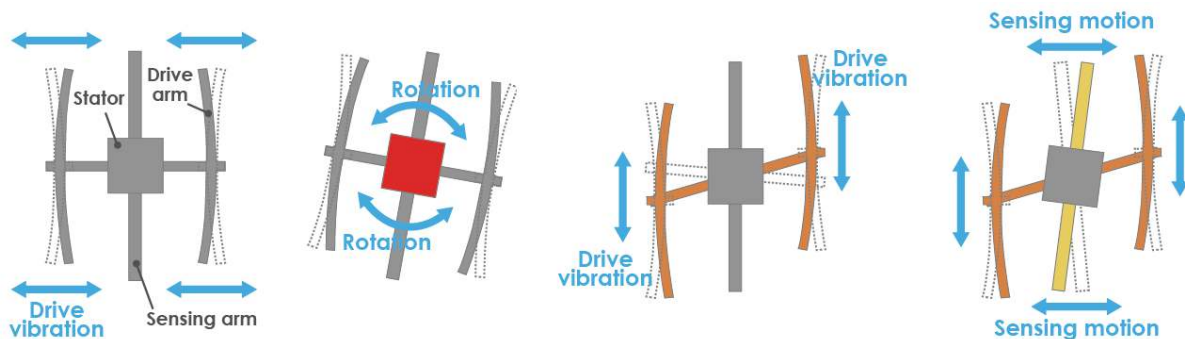
$$\text{Acceleration} = (\text{Accelerometer axis raw data} / 65536 * \text{full scale Acceleration range}) \text{ g}$$

Take the X-axis as an example, when Accelerometer X axis raw data is 16384 and the range is selected as +/-2g:

$$\text{Acceleration along the X axis} = (16384 / 65536 * 4) \text{ g} = 1\text{g}$$

3-axis Gyroscope

Gyroscopes work on the principle of Coriolis acceleration. Imagine that there is a fork like structure, that is in constant back and forth motion. It is held in place using piezo electric crystals. Whenever, you try to tilt this arrangement, the crystals experience a force in the direction of inclination. This is caused as a result of the inertia of the moving fork. The crystals thus produce a current in consensus with the piezo electric effect, and this current is amplified.



1. Normally, a drive arm vibrates in a certain direction.
2. Direction of rotation
3. When the gyro is rotated, the Coriolis force acts on the drive arms, producing vertical vibration.
4. The stationary part bends due to vertical drive arm vibration, producing a sensing motion in the sensing arms.

The Gyroscope also has four kinds of measuring ranges: +/- 250, +/- 500, +/- 1000, +/- 2000. The calculation method and Acceleration are basically consistent.

The formula for converting the reading into angular velocity is as follows:

$$\text{Angular velocity} = (\text{Gyroscope axis raw data} / 65536 * \text{full scale Gyroscope range}) \text{ } ^\circ/\text{s}$$

The X axis, for example, the Accelerometer X axis raw data is 16384 and ranges $\pm 250^\circ/s$:

Angular velocity along the X axis = $(16384 / 65536 * 500)^\circ/s = 125^\circ/s$

Example

- [2.34 MPU6050 Module](#) (Arduino Project)

1.43 MFRC522 Module

RFID

Radio Frequency Identification (RFID) refers to technologies that involve using wireless communication between an object (or tag) and an interrogating device (or reader) to automatically track and identify such objects. The tag transmission range is limited to several meters from the reader. A clear line of sight between the reader and tag is not necessarily required.

Most tags contain at least one integrated circuit (IC) and an antenna. The microchip stores information and is responsible for managing the radio frequency (RF) communication with the reader. Passive tags do not have an independent energy source and depend on an external electromagnetic signal, provided by the reader, to power their operations. Active tags contain an independent energy source, such as a battery. Thus, they may have increased processing, transmission capabilities and range.



MFRC522

MFRC522 is a kind of integrated read and write card chip. It is commonly used in the radio at 13.56MHz. Launched by the NXP Company, it is a low-voltage, low-cost, and small-sized non-contact card chip, a best choice of intelligent instrument and portable handheld device.

The MF RC522 uses advanced modulation and demodulation concept which fully presented in all types of 13.56MHz passive contactless communication methods and protocols. In addition, it supports rapid CRYPTO1 encryption algorithm to verify MIFARE products. MFRC522 also supports MIFARE series of high-speed non-contact communication, with a two-way data transmission rate up to 424kbit/s. As a new member of the 13.56MHz highly integrated reader card series, MF RC522 is much similar to the existing MF RC500 and MF RC530 but there also exists great differences. It communicates with the host machine via the serial manner which needs less wiring. You can choose between SPI, I2C and serial UART mode (similar to RS232), which helps reduce the connection, save PCB board space (smaller size), and reduce cost.

Example

- [2.35 RFID-RC522 Module](#) (Arduino Project)

PLAY WITH ARDUINO

1. Basic

2.1 1.1 Get Started with Arduino IDE

Arduino is an open source platform with simple software and hardware. You can pick it up in short time even if you are a beginner. It provides an integrated development environment (IDE) for code compiling, compatible with multiple control boards. So you can just download the Arduino IDE, upload the sketches (i.e. the code files) to the board, and then you can see relative experimental phenomena. For more information, refer to <http://www.arduino.cc>.

Please follow the tutorial below to learn how to install the Arduino IDE, add libraries and upload code.

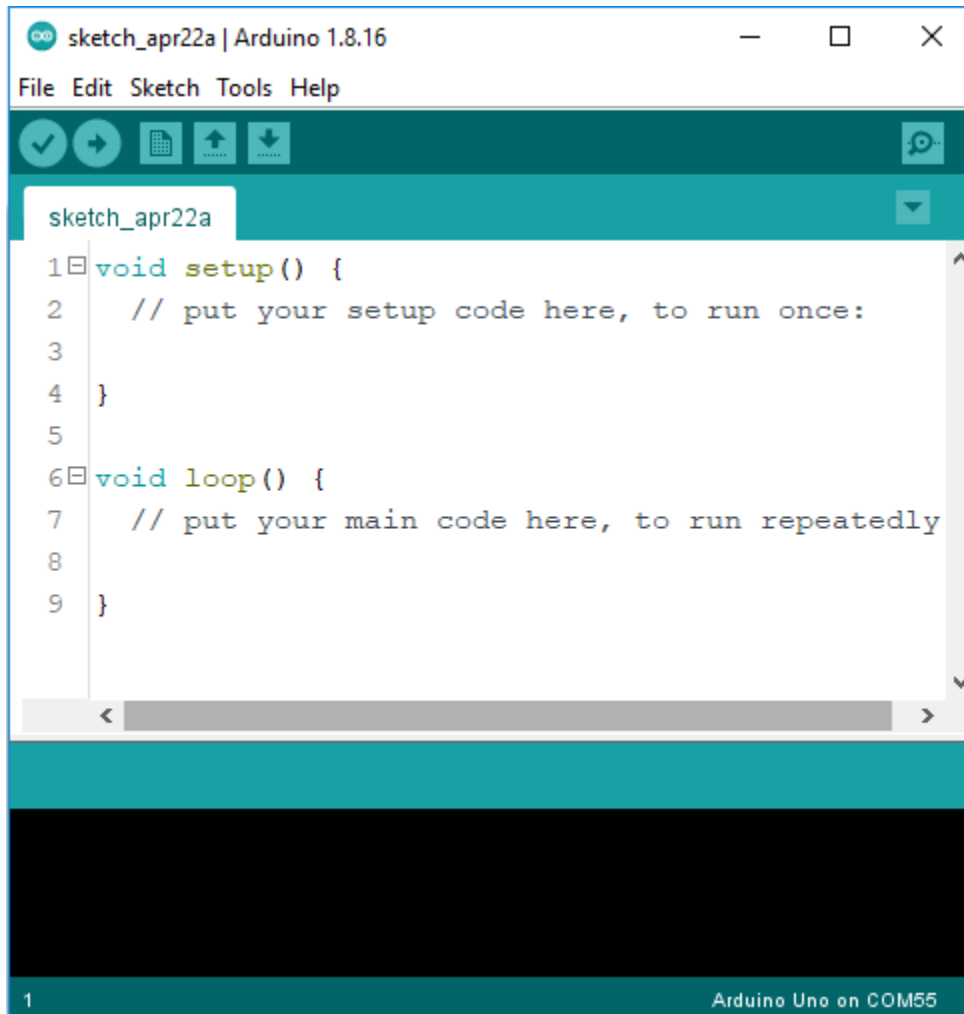
2.1.1 Install and Introduce Arduino IDE

Install Arduino IDE

It currently has three programming tools for you to choose from: Arduino IDE 1, Arduino IDE 2 and Arduino Web Editor.

- **Arduino IDE 1** is the standard, offline editor.
- **Arduino IDE 2** is the future version the Arduino IDE which is faster and even more powerful.
- **Arduino Web Editor** is an online development environment.

Arduino IDE 1



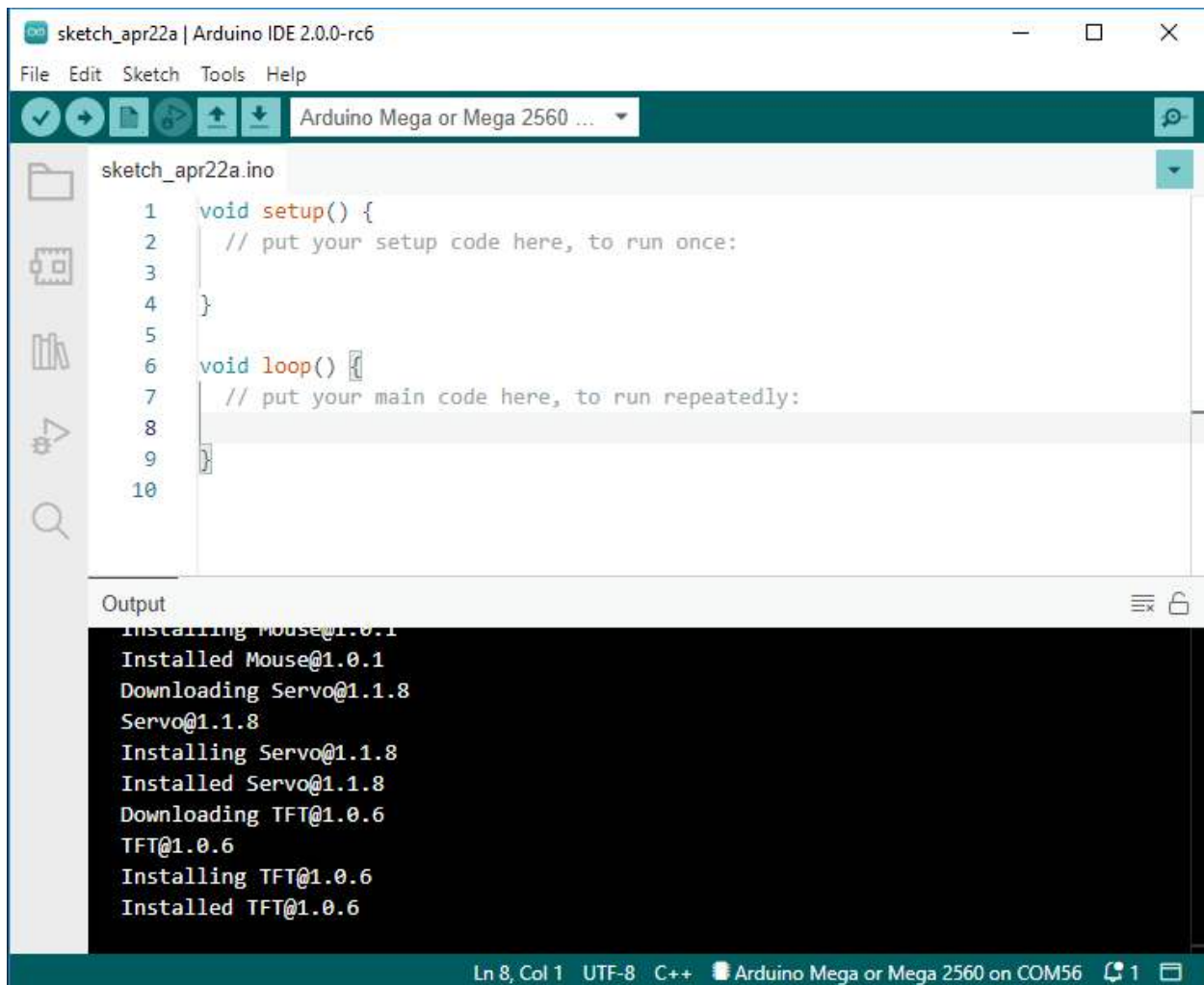
The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

You can find everything about **Arduino IDE 1** from: <https://docs.arduino.cc/software/ide-v1>.

The following links are tutorials for the installation of **Arduino IDE 1** for each operating system.

- Windows
- mac OS
- Linux

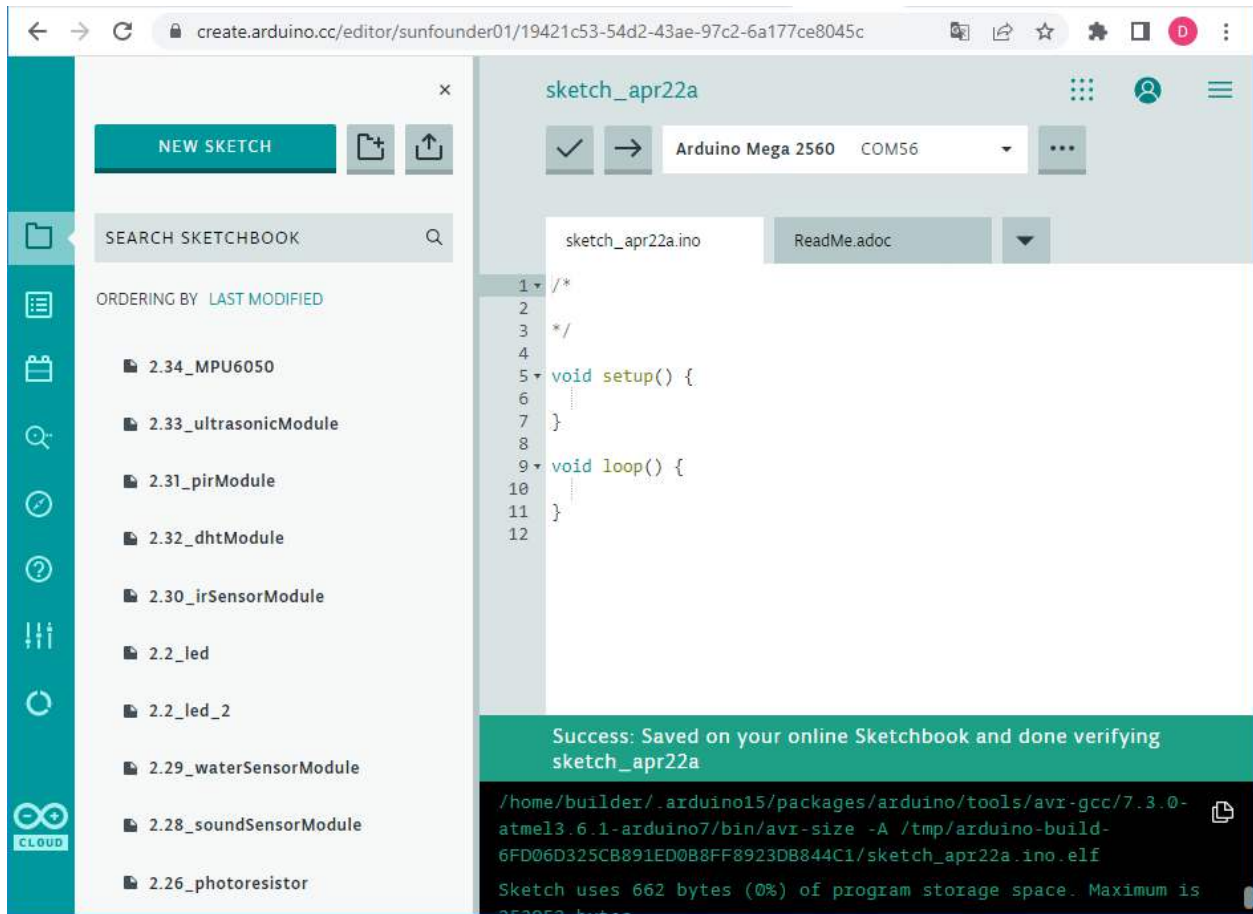
Arduino IDE 2



The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

You can find everything about **Arduino IDE 2** from: <https://docs.arduino.cc/software/ide-v2>

Arduino Web Editor



Arduino Web Editor is an online development environment, with online storage and thousands of libraries available, and also supports new Arduino boards.

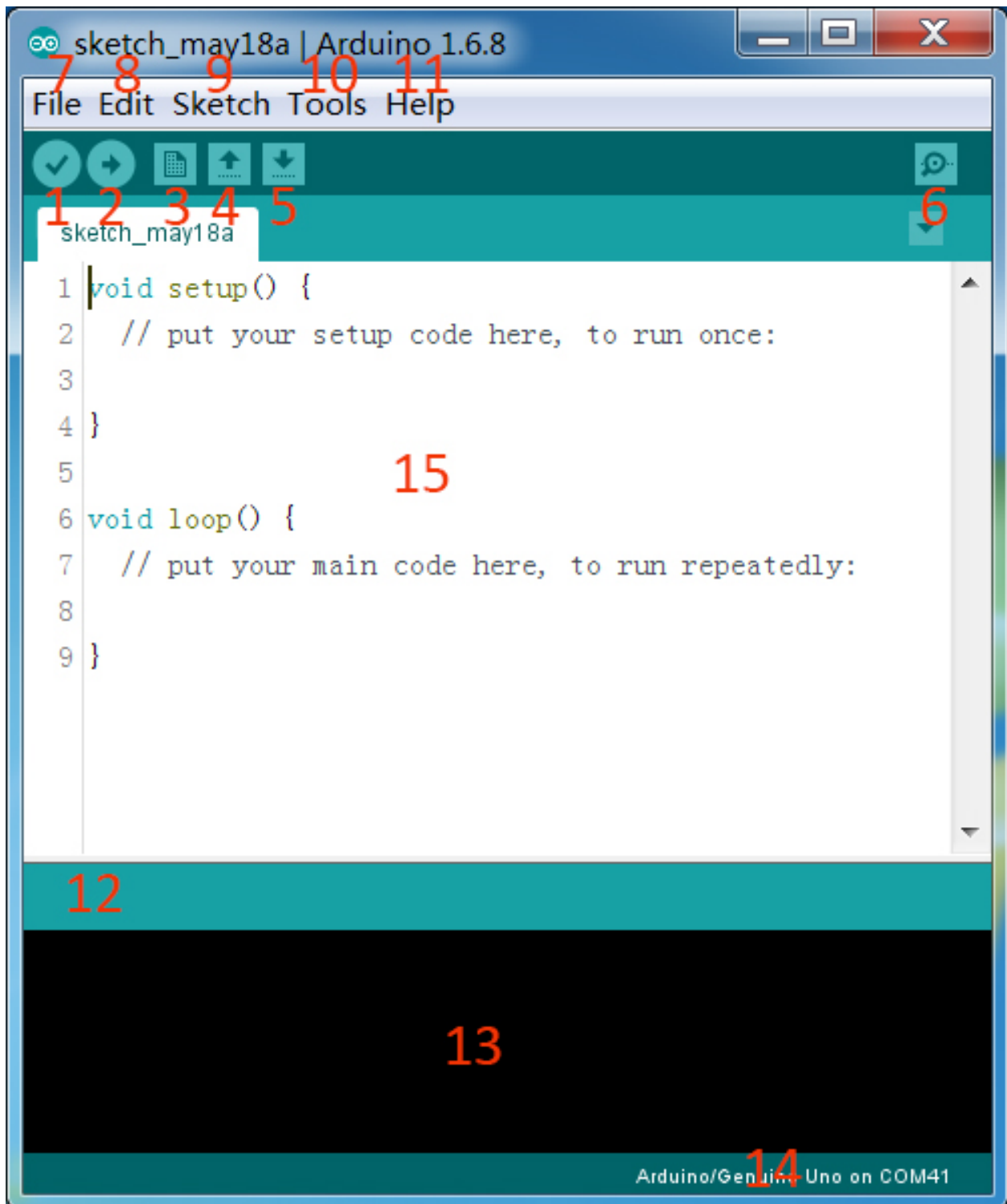
You can find everything about **Arduino Web Editor** from: <https://docs.arduino.cc/cloud/web-editor>

Introduce of Arduino IDE 1

Double-click the Arduino icon (arduino.exe) created by the installation process.



Then the Arduino IDE will appear. Let's check details of the software.



1. **Verify:** Compile your code. Any syntax problem will be prompted with errors.
2. **Upload:** Upload the code to your board. When you click the button, the RX and TX LEDs on the board will flicker fast and won't stop until the upload is done.
3. **New:** Create a new code editing window.
4. **Open:** Open an .ino sketch.

5. **Save:** Save the sketch.
6. **Serial Monitor:** Click the button and a window will appear. It receives the data sent from your control board. It is very useful for debugging.
7. **File:** Click the menu and a drop-down list will appear, including file creating, opening, saving, closing, some parameter configuring, etc.
8. **Edit:** Click the menu. On the drop-down list, there are some editing operations like Cut, Copy, Paste, Find, and so on, with their corresponding shortcuts.
9. **Sketch:** Includes operations like Verify, Upload, Add files, etc. More important function is Include Library – where you can add libraries.
10. **Tool:** Includes some tools – the most frequently used Board (the board you use) and Port (the port your board is at). Every time you want to upload the code, you need to select or check them.
11. **Help:** If you're a beginner, you may check the options under the menu and get the help you need, including operations in IDE, introduction information, troubleshooting, code explanation, etc.
12. In this message area, no matter when you compile or upload, the summary message will always appear.
13. Detailed messages during compile and upload. For example, the file used lies in which path, the details of error prompts.
14. **Board and Port:** Here you can preview the board and port selected for code upload. You can select them again by **Tools** -> **Board / Port** if any is incorrect.
15. The editing area of the IDE. You can write code here.

2.1.2 Download Code and Add Libraries

Download the Code

Download the relevant code from the link below.

- [SunFounder Vincent Kit for Arduino](#)
- Or check out the code at [Vincent Kit for Arduino - GitHub](#)

Add Libraries

What is Library?

A library, gathering some function definitions and header files, usually contains two files: .h (header file, including function statement, Macro definition, constructor definition, etc.) and .cpp (execution file, with function implementation, variable definition, and so on). When you need to use a function in some library, you just need to add a header file (e.g. `#include <dht.h>`), and then call that function. This can make your code more concise. If you don't want to use the library, you can also write that function definition directly. Though as a result, the code will be long and inconvenient to read.

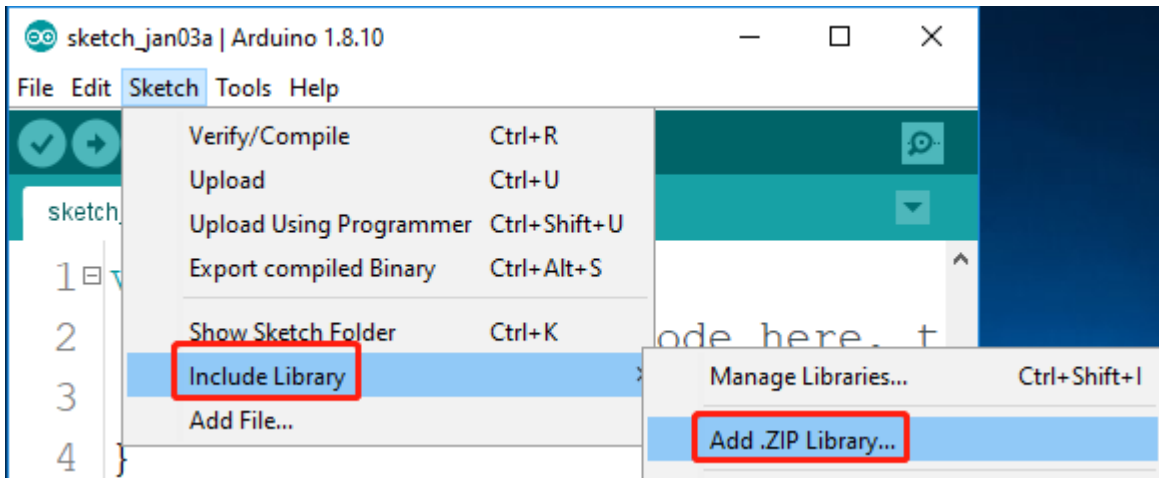
Add libraries

Some libraries are already built in the Arduino IDE, when some others may need to be added. So now let's see how to add one. There are 2 methods for that.

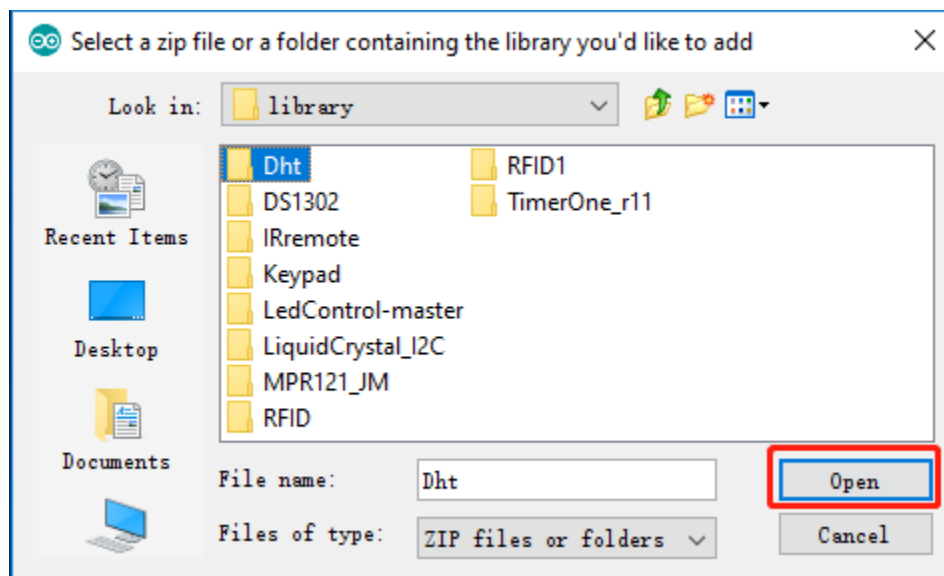
Method 1 Add .ZIP Library

Directly import the library in Arduino IDE (take Dht as an example below). The advantage of this method is easy to understand and operate, but on the other hand, only one library can be imported at a time. So it is inconvenient when you need to add quite a lot of libraries.

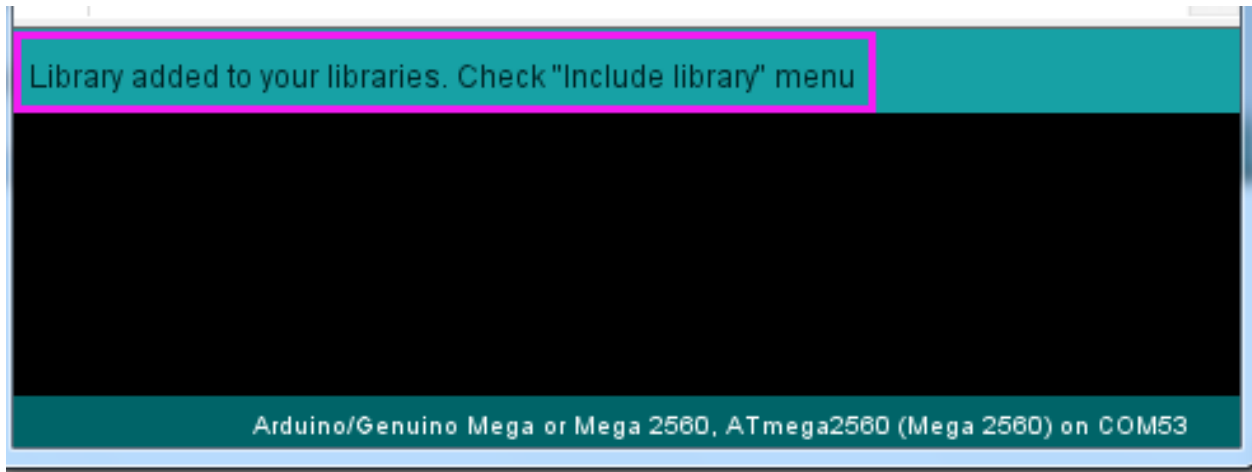
Step 1: Select **Sketch** -> **Include Library** -> **Add ZIP Library**.



Step 2: Find **Library** folder, Click **Open**.



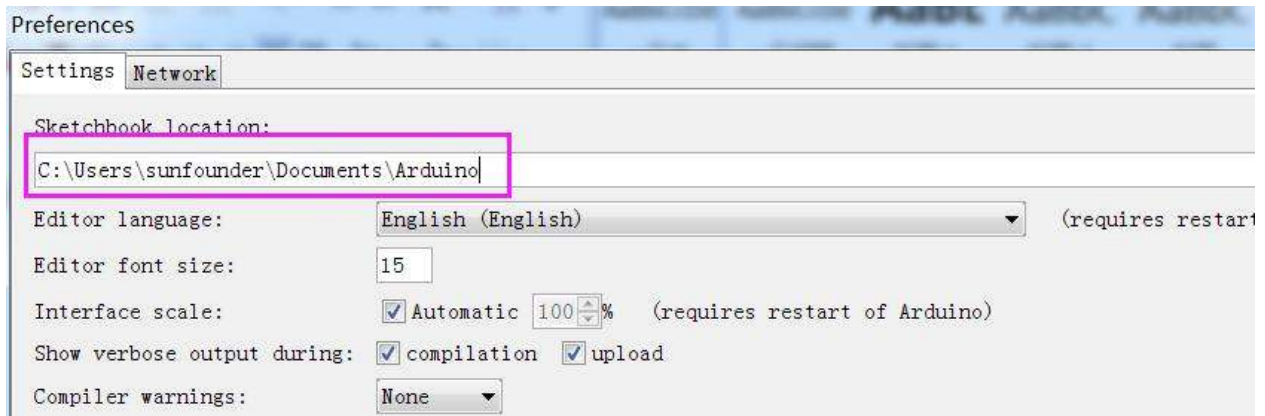
Step 3: When you see **Library added to your libraries**. Check **“Include library”** menu, it means you have added the library successfully. Please use the same method to add other libraries then.



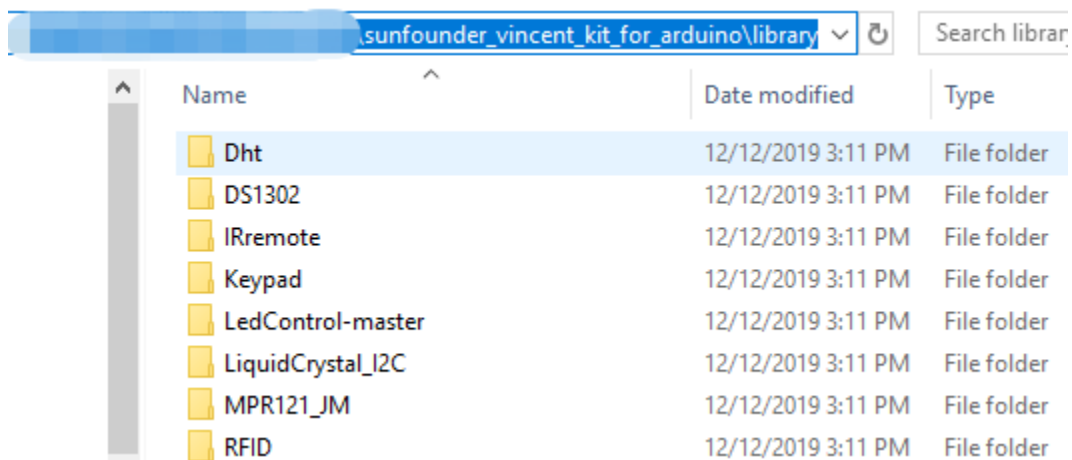
Method 2 Manually Add

You can also try to put the library in the `libraries` folder of your sketchbook by yourself. This method can copy all libraries and add them at a time, but first you need to find the path where the `libraries` folder is located.

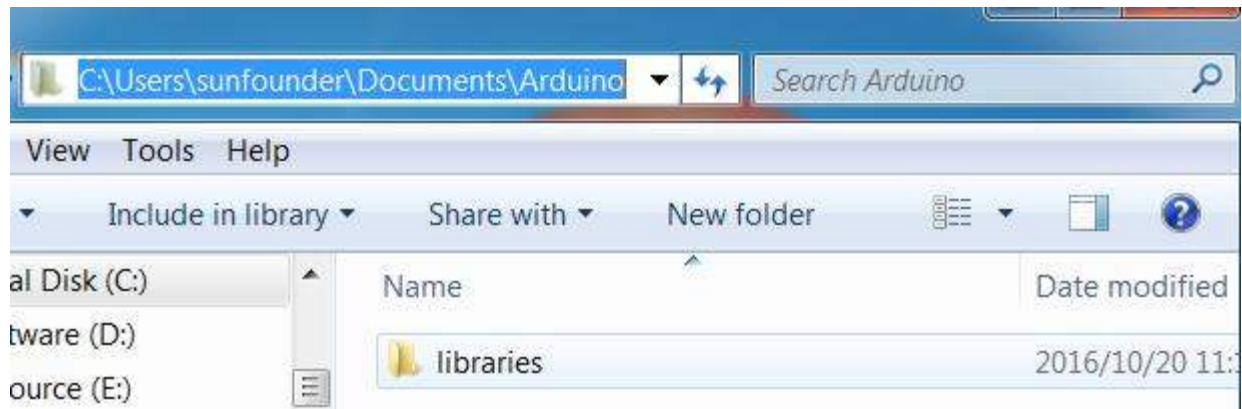
Step 1: You can find or change the location of your `libraries` folder at **File > Preferences > Sketchbook location**.



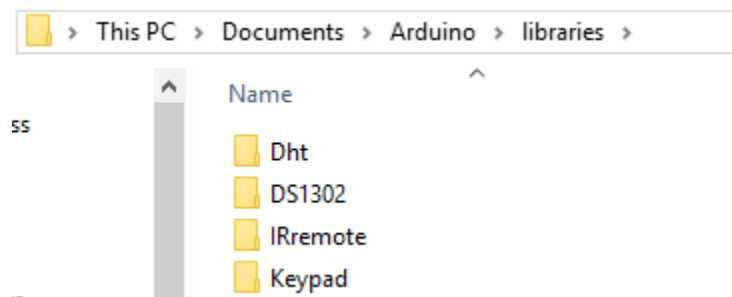
Step 2: Go to the directory `sunfounder_vincent_kit_for_arduino/library` and copy all the folders inside.



Step 3: Go to the location of your `libraries` folder (find from Arduino IDE), click to open it.



Step 4: Paste all the libraries copied before to the `libraries` folder.



There is also a way to use the **Library Manager**, but the libraries added using this method are not necessarily compatible with the code we provide, so it is not recommended.

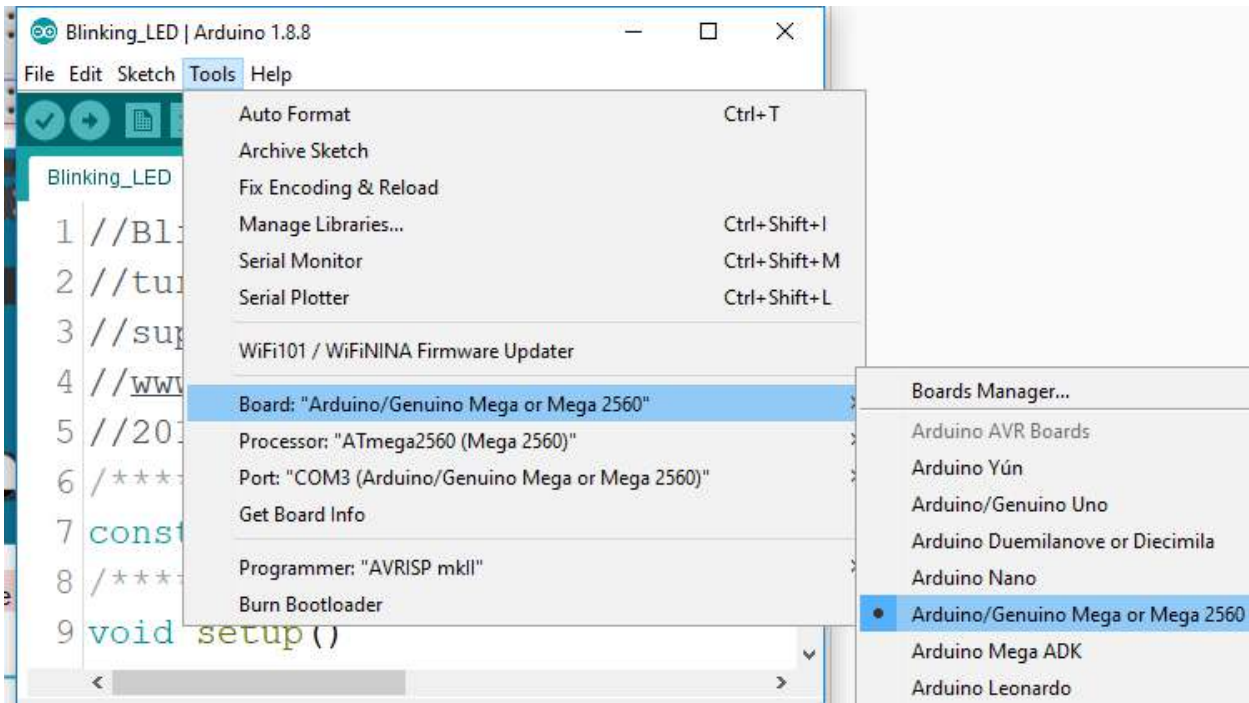
Here is the link to the official Arduino tutorial: <https://docs.arduino.cc/software/ide-v1/tutorials/installing-libraries>

2.1.3 Upload the Code

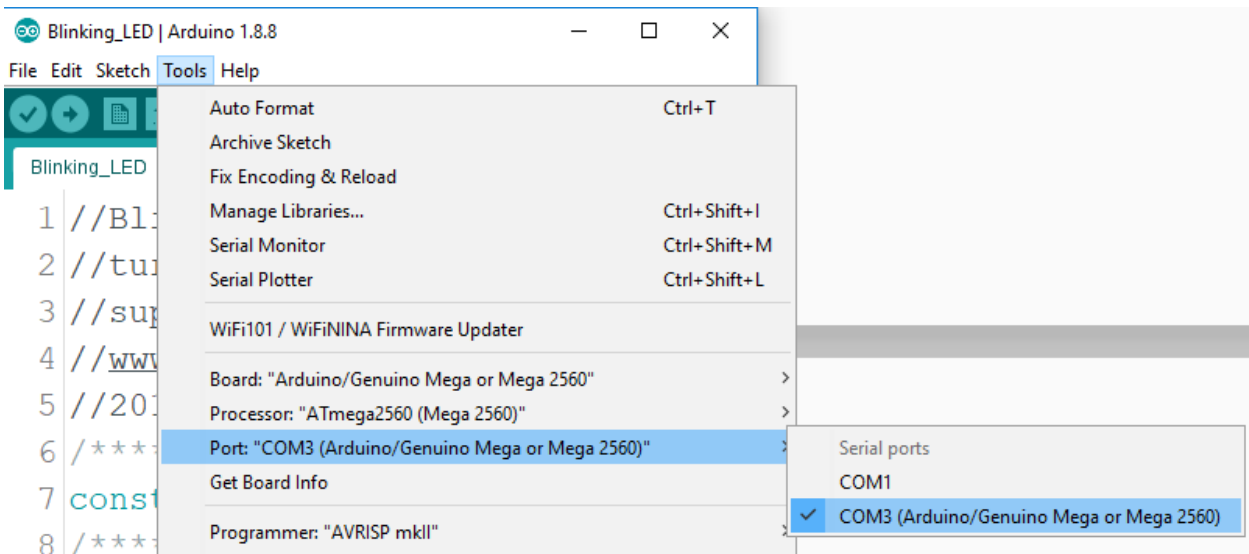
For Arduino IDE 1/2

Click **Tools** -> **Board** -> **Arduino AVR Boards** and select **Arduino/Genuino Mega or Mega 2560**.

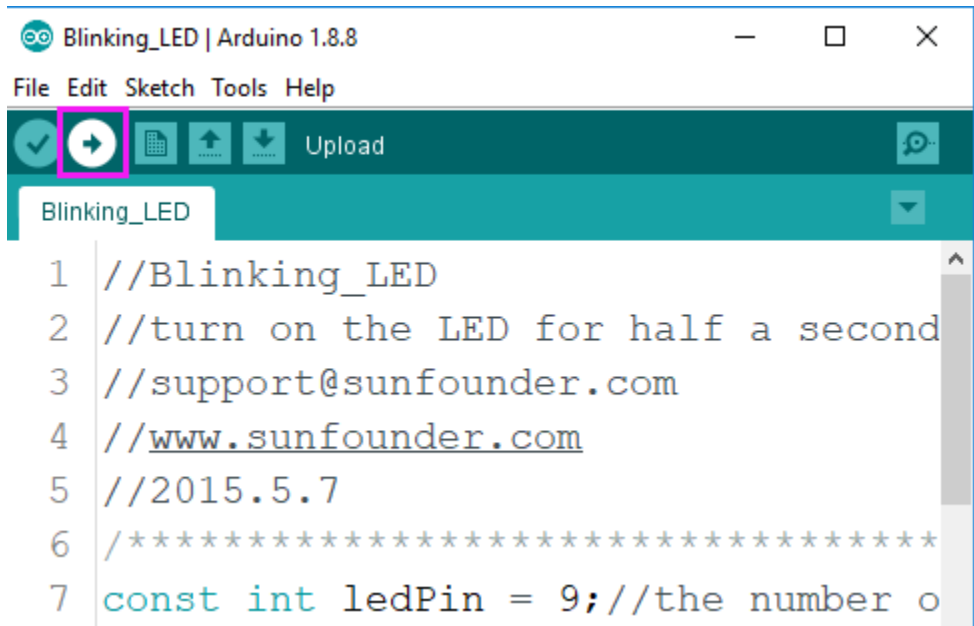
Note: For Arduino IDE 2, you may need to install **Arduino AVR Boards** first. Click **Tools** -> **Board** -> **Boards Managers**, find **Arduino AVR Boards**, and then click **INSTALL**.



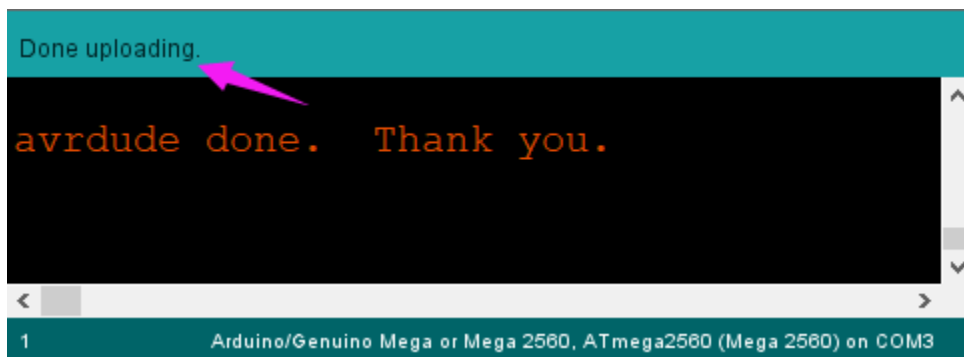
Then select **Tools** ->**Port**. Your port should be different from mine.



Click the **Upload** icon to upload the code to the control board.

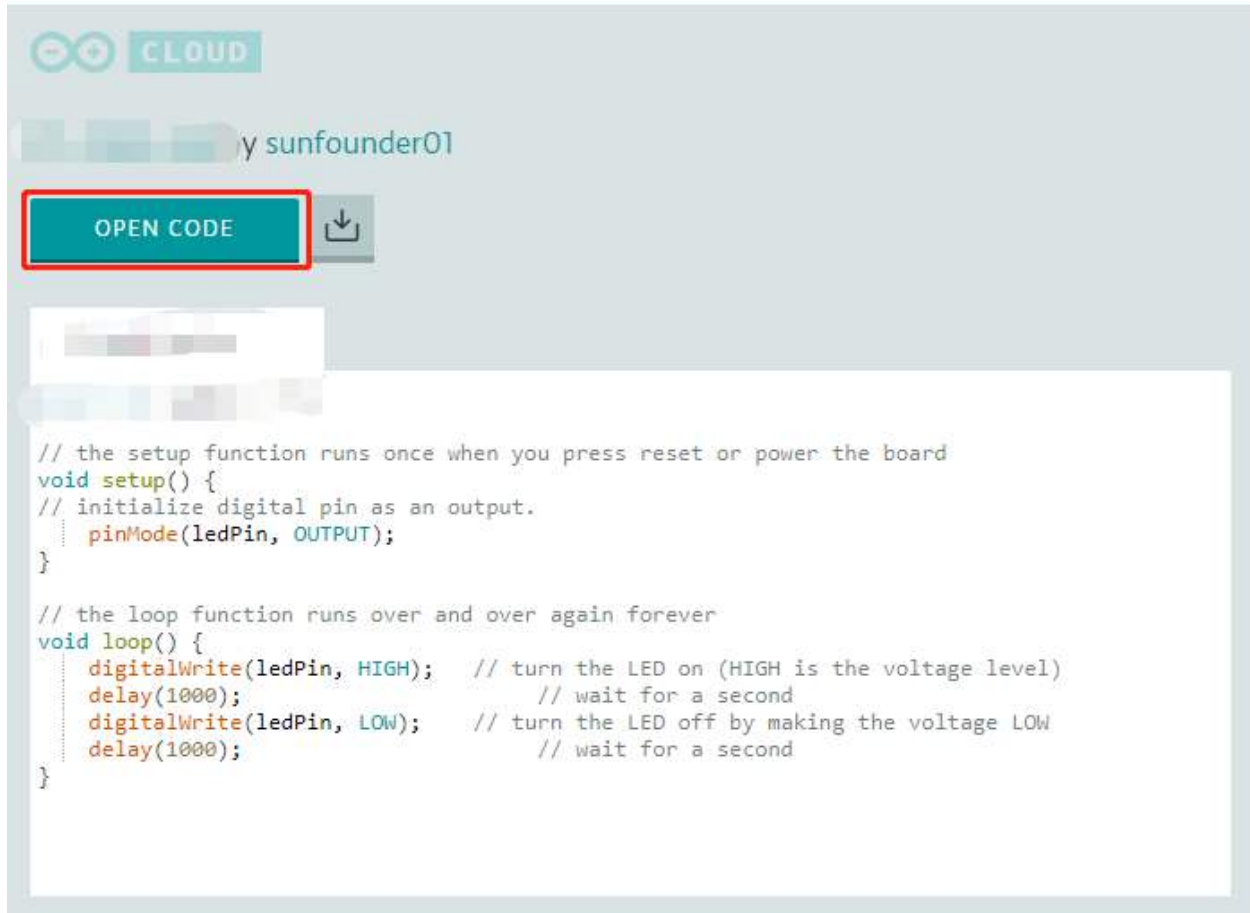


If "Done uploading" appears at the bottom of the window, it means the sketch has been successfully uploaded.



For Web Editor

In each project there is an Arduino code display window, click **Open Code**.



The screenshot shows the Arduino IDE interface. At the top left, there are navigation icons and a 'CLOUD' button. Below that, the sketch name 'y sunfounder01' is visible. A red box highlights the 'OPEN CODE' button. To the right of this button is a download icon. Below the buttons is a code editor containing the following C++ code:

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

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

Click on **Open in Web Editor** or **Add to MY Sketch**.

The screenshot shows the Arduino Cloud IDE interface. At the top, there is a teal header with the 'CLOUD' logo and a user profile icon labeled 'sunfounder01'. Below the header is a code editor with the following code:

```

1 const int ledPin = 
2 
3 // the setup function runs once when you press reset or power the board
4 void setup() {
5 // initialize digital pin as an output.
6   pinMode(ledPin, OUTPUT);
7 }
8 
9 // the loop function runs over and over again forever
10 void loop() {
11   digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
12   delay(1000); // wait for a second
13   digitalWrite(ledPin, LOW); // turn the LED off by making the voltage LOW
14   delay(1000); // wait for a second
15 }
16

```

To the right of the code editor is a sidebar with a red-bordered button labeled 'OPEN IN WEB EDITOR' and a download icon. Below this is a 'SKETCH INFO' section with the following details:

SKETCH INFO	
Size	654 bytes
Created	March 03 2022, 3:50:58
Modified	March 03 2022, 5:55:54

If you have registered and logged in to your account, you will be inside the Web Editor. However, you need to install Arduino Create Agent and run it before uploading the code in Web Editor.

For detailed tutorials, please refer to: <https://docs.arduino.cc/cloud/web-editor/tutorials/getting-started/getting-started-web-editor>.





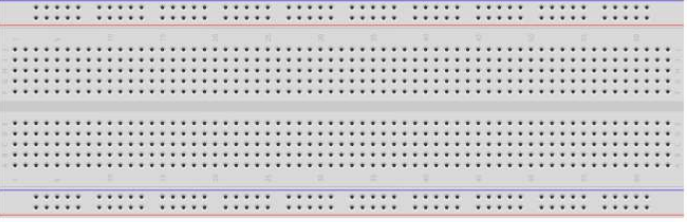
Warning: For projects that require additional libraries, they will not work in Web Editor.

2.2 1.2 Digital Write

2.2.1 Overview

The `digitalWrite()` statement here is used to write high level or low level to pins and to let LED and active buzzer work or stop. In this lesson, we will take LED as an example to introduce the experiment phenomenon.

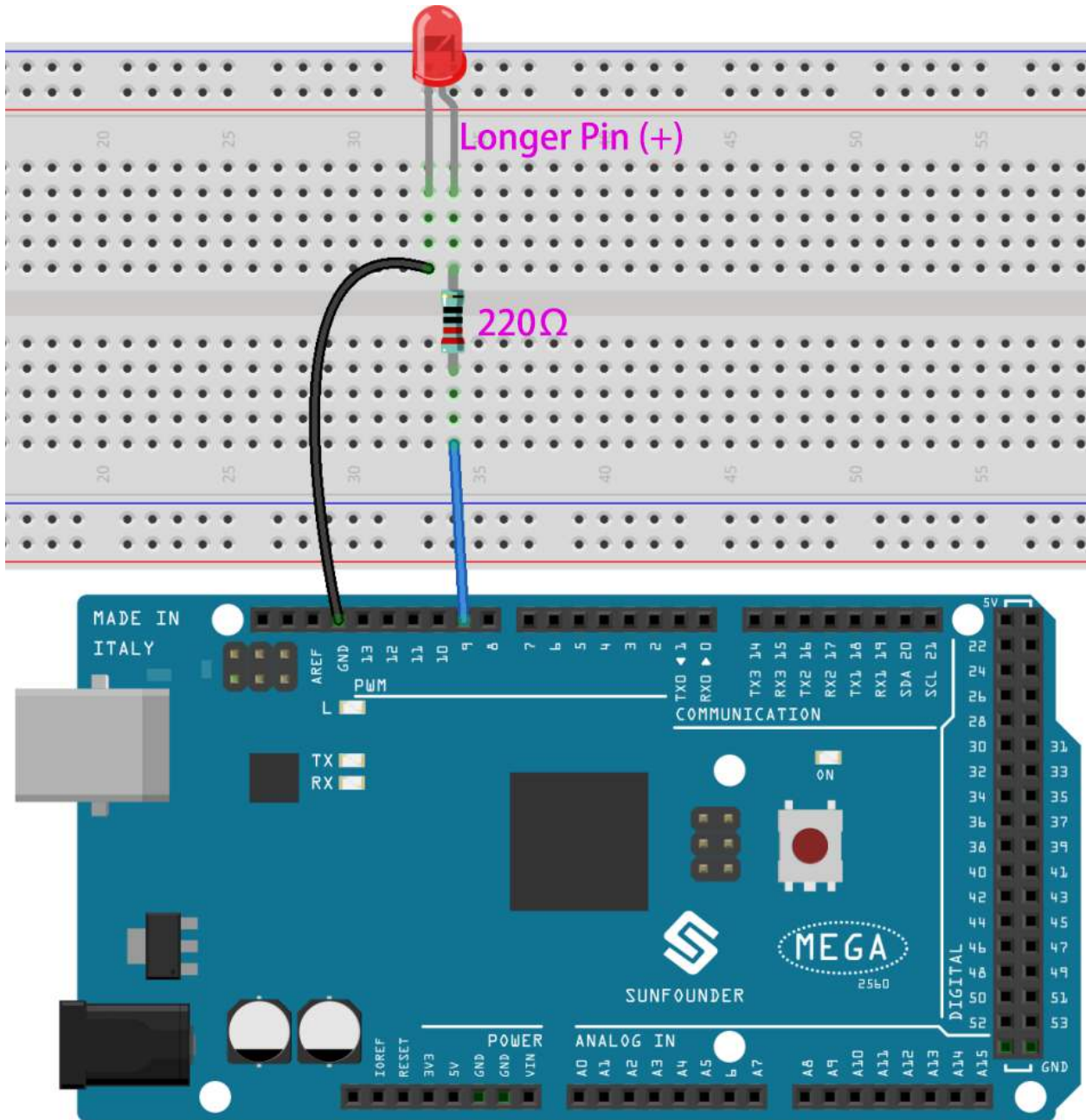
2.2.2 Components Required

1 * LED 	1 * 220ohm Resistor 	Several Jumper Wires 
1 * Mega 2560 Board 	1 * Breadboard 	

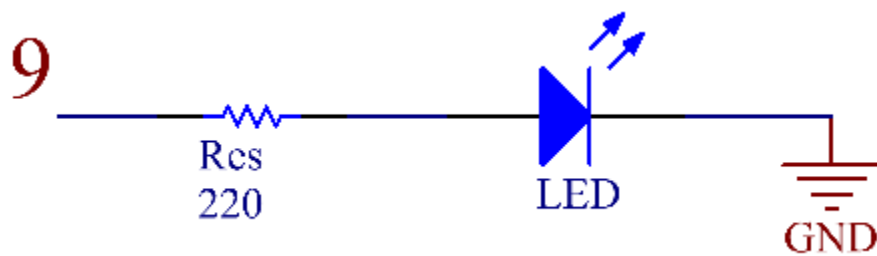
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *LED*
- *Resistor*

2.2.3 Fritzing Circuit

In this example, we use digital pin 9 to drive the LED. Attach one side of the resistor to the digital pin 9 and the longer pin (anode) of the LED to the other side of the resistor. Connect the shorter pin (cathode) of the LED to GND.



2.2.4 Schematic Diagram



2.2.5 Code

After finishing the circuit connection, connect the Mega2560 board to the computer. Run the Arduino software IDE and type in the following codes.

Note:

- You can open the file `1.2_digitalWrite.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\1.2_digitalWrite` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Or click **Open Code** to open it in [Web Editor](#).
 - Then *Upload the Code* to the board.
-

Upload the codes to the Mega2560 board, and you can see the blinking of LED.

2.2.6 Code Analysis

Here, we connect the LED to the digital pin 9, so we need to declare an int variable called `ledpin` at the beginning of the program and assign a value of 9.

```
const int ledPin = 9;
```

Now, initialize the pin in the `setup()` function, where you need to initialize the pin to OUTPUT mode.

```
pinMode(ledPin, OUTPUT);
```

In `loop()`, `digitalWrite()` is used to provide 5V high level signal for `ledpin`, which will cause voltage difference between LED pins and light LED up.

```
digitalWrite(ledPin, HIGH);
```

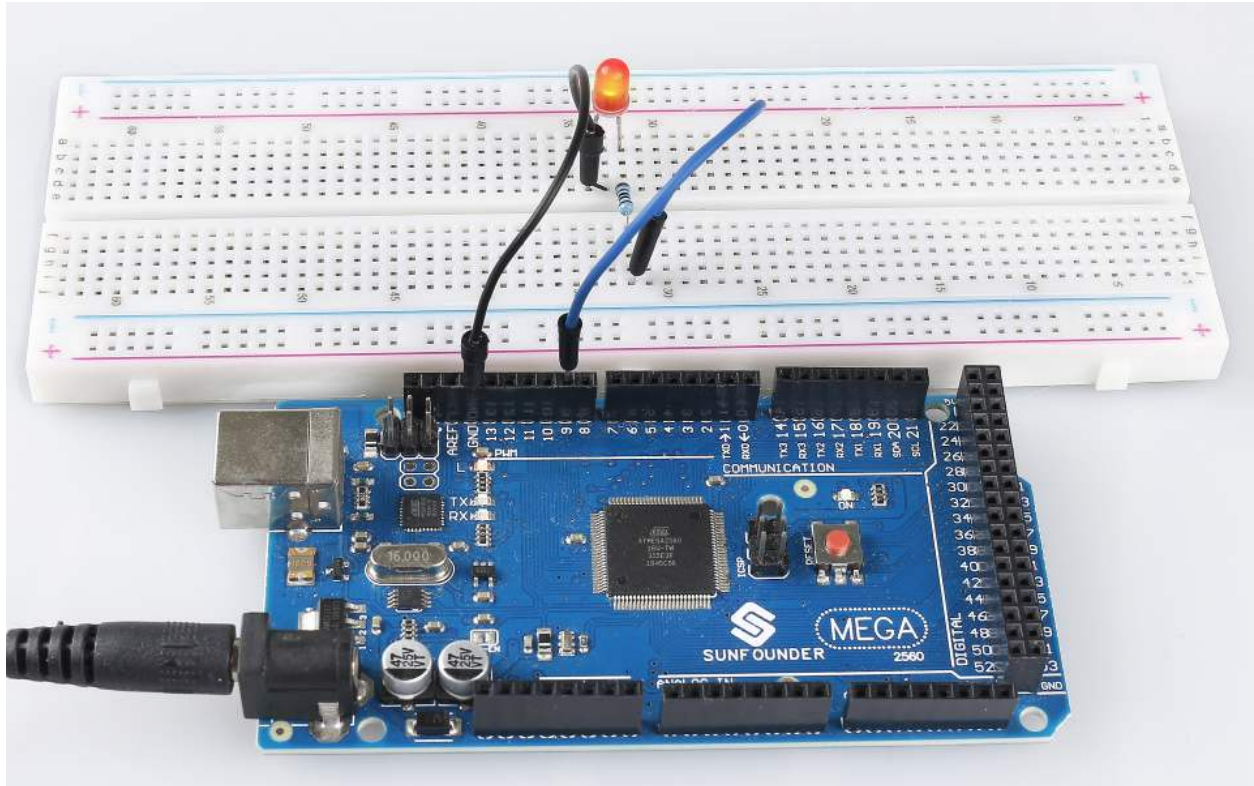
If the level signal is changed to LOW, the `ledPin`'s signal will be returned to 0 V to turn LED off.

```
digitalWrite(ledPin, LOW);
```

An interval between on and off is required to allow people to see the change, so we use a `delay(1000)` code to let the controller do nothing for 1000 ms.

```
delay(1000);
```

2.2.7 Phenomenon Picture





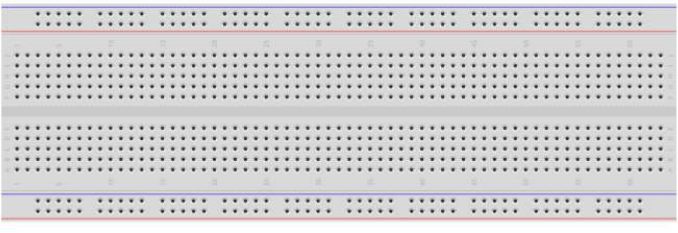


2.3 1.3 Analog Write

2.3.1 Overview

You can write the PWM wave to the pin by using `analogWrite()`. This method can be used to adjust the brightness of LED, change the color of RGB, or adjust the motor speed, etc. Here we will take LED as an example to get gradient brightness of LED.

2.3.2 Components Required

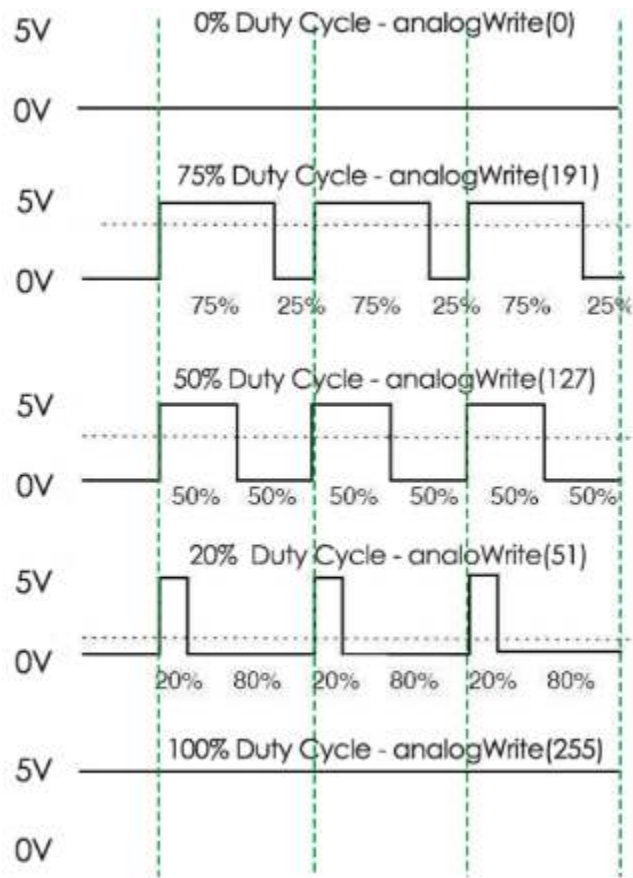
<p>1 * LED</p> 	<p>1 * 220ohm Resistor</p> 	<p>Several Jumper Wires</p> 
<p>1 * Mega 2560 Board</p> 	<p>1 * Breadboard</p> 	

- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *LED*
- *Resistor*

2.3.3 ※ Pulse Width Modulation

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of “on time” is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

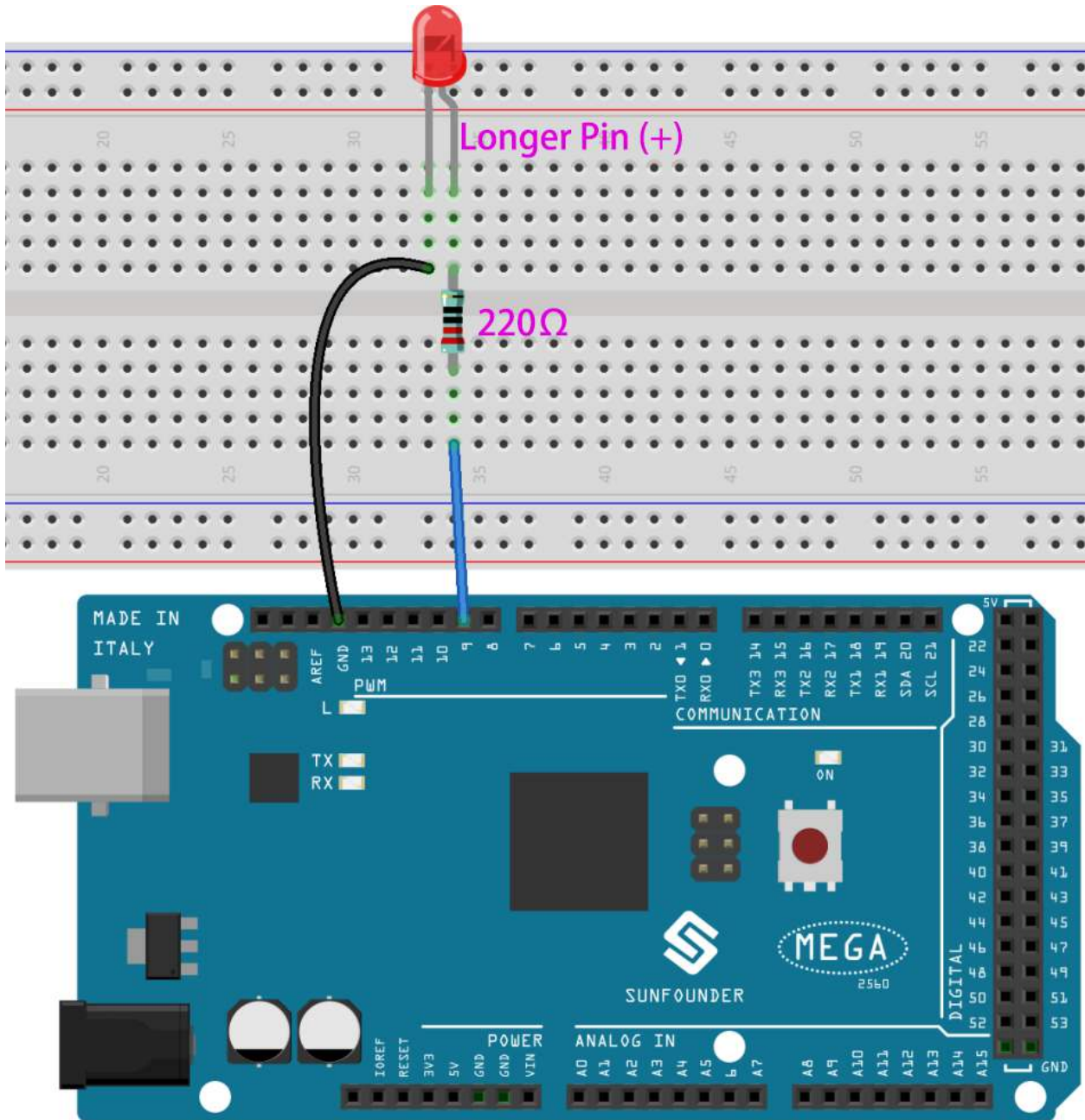
A call to `analogWrite()` is on a scale of 0 - 255, such that `analogWrite(255)` requests a 100% duty cycle (always on), and `analogWrite(127)` is a 50% duty cycle (on half the time) for example.



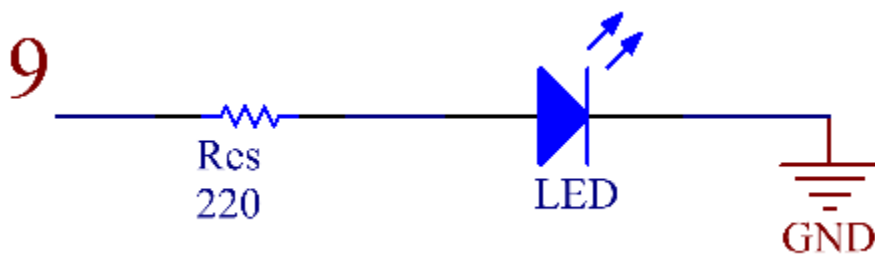
2.3.4 Fritzing Circuit

In this example, we use the PWM pin 9 to drive the LED. Connect one end of the resistor to pin 9. Connect the long pin (anode) of the LED to the other end of the resistor. Connect the short pin (negative, referred to as the cathode) of LED to GND.

Note: PWM pins of Mega2560 board are 2 -13, 44 - 46.



2.3.5 Schematic Diagram



2.3.6 Code

Note:

- You can open the file `1.3_analogWrite.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\1.3_analogWrite` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After uploading the code to the Mega2560 board, you can see that the LED gradually brightens out and turns off gradually.

2.3.7 Code Analysis

Declare pin 9 as ledPin.

```
int ledPin = 9;
```

`analogWrite()` in `loop()` assigns ledPin an analog value (PWM wave) between 0 and 255 to change the brightness of LED.

```
analogWrite(ledPin, value);
```

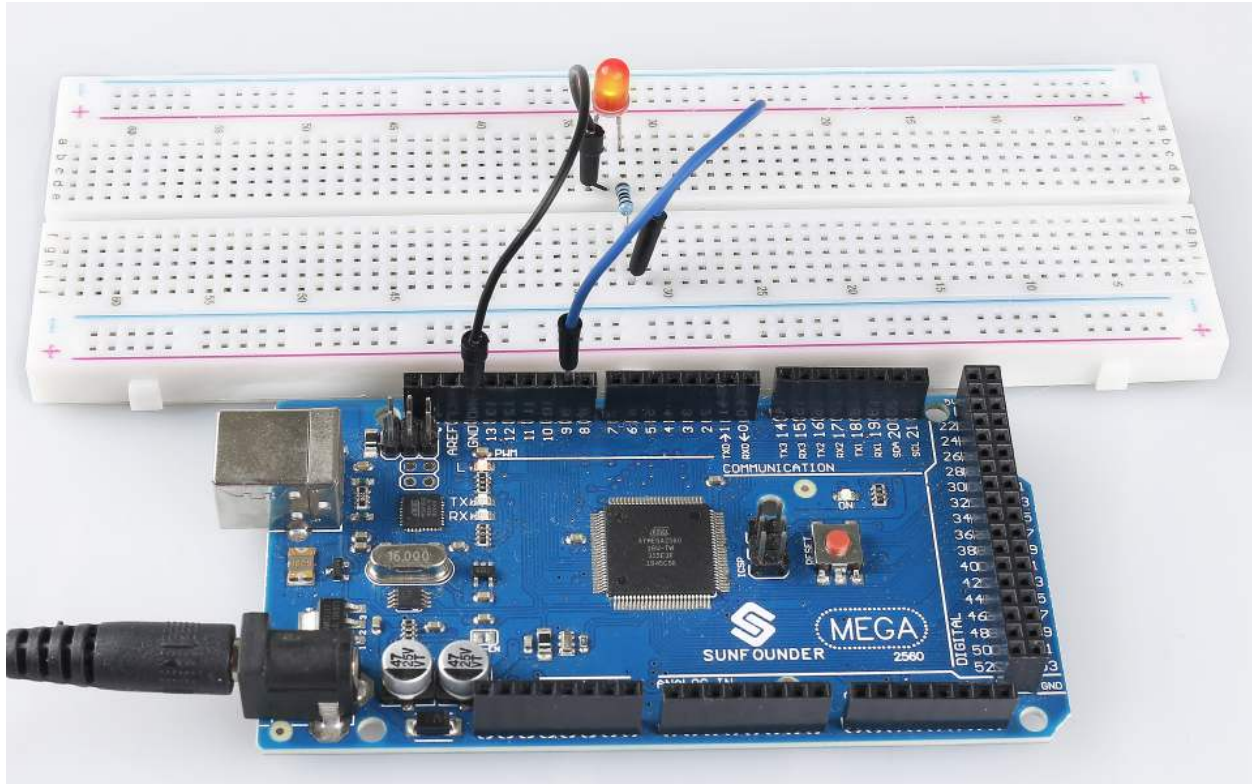
Using a for loop, the value of `analogWrite()` can be changed step by step between the minimum value (0) and the maximum value (255).

```
for (int value = 0 ; value <= 255; value += 5) {  
    analogWrite(ledPin, value);  
}
```

In order to see the experimental phenomenon clearly, a `delay(30)` needs to be added to the for cycle to control the brightness change time.

```
void loop() {  
for (int value = 0 ; value <= 255; value += 5) {  
    analogWrite(ledPin, value);  
    delay(30);  
}  
}
```

2.3.8 Phenomenon Picture








2.4 1.4 Digital Read

2.4.1 Overview

You can use the `digitalRead()` command to read the level status from a digital pin. The command is suitable for digital input elements such as Button, Touch sensor, infrared motion sensor, etc. This article will take Button as an example to read the level state.

This example also shows you how to monitor the state of a switch by using USB to establish serial communication between a control board and a computer.

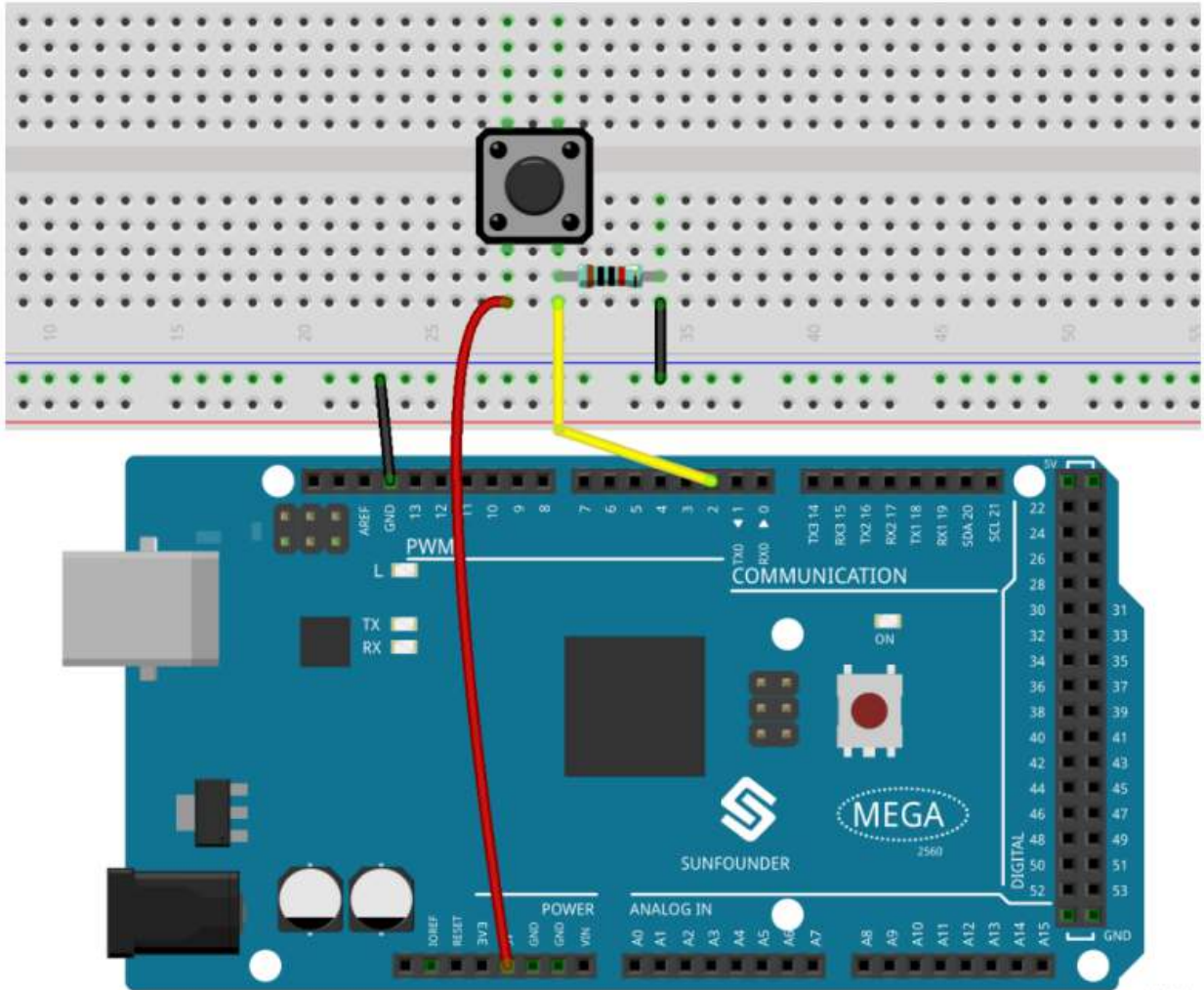
2.4.2 Components Required

<p>1 * Button</p> 	<p>1 * 10k ohm resistor</p> 	<p>Several Jumper Wires</p> 
<p>1 * Mega 2560 Board</p> 	<p>1 * Breadboard</p> 	

- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Button*
- *Resistor*

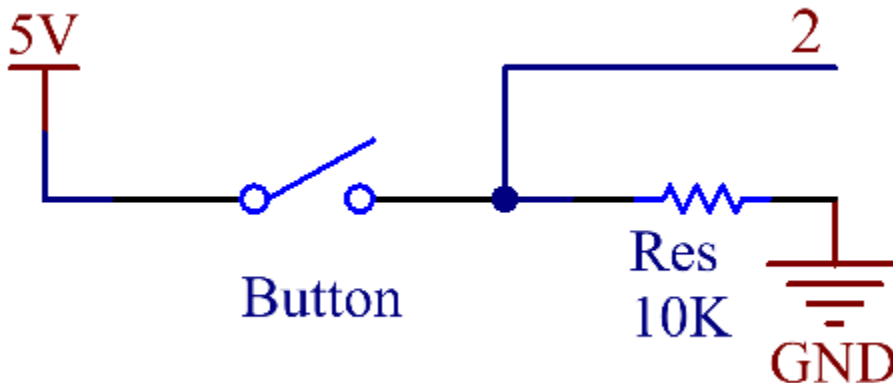
2.4.3 Fritzing Circuit

In this example, we read the signal of the button with the digital pin 2. When the button is not pressed, the digital pin 2 (through the drop-down resistor) is connected to ground to read the low level (0); when the button is pressed, the two pins are connected and when the pin is connected to the 5V power supply, the high level (1) is read.



Note: If you disconnect the digital I/O pin from everything, the LED may blink erratically. This is because the input is “floating” - that is, it doesn’t have a solid connection to voltage or ground, and it will randomly return either HIGH or LOW. That’s why you need a pull-down resistor in the circuit.

2.4.4 Schematic Diagram



2.4.5 Code

Note:

- You can open the file `1.4_digitalRead.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\1.4_digitalRead` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After uploading the code to the Mega2560 board, we can open the serial port monitor to see the reading value of the pin. When you press Button, the serial port monitor will display 1 and when Button is released, 0 will be displayed.

2.4.6 Code Analysis

Start the serial communication in `setup()` and set the data rate to 9600.

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

You also need to set the status of the digital pin 2 to INPUT to read the output status of Button.

```
pinMode(2, INPUT);
```

Use the `digitalRead()` statement in `loop()` to read the level state of the digital pin 2 and declare a variable to store the state.

```
int buttonState = digitalRead(2);
```

Print the value stored by the variable on the serial port monitor.

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

Use `delay()` statements to make printing results easy to observe.

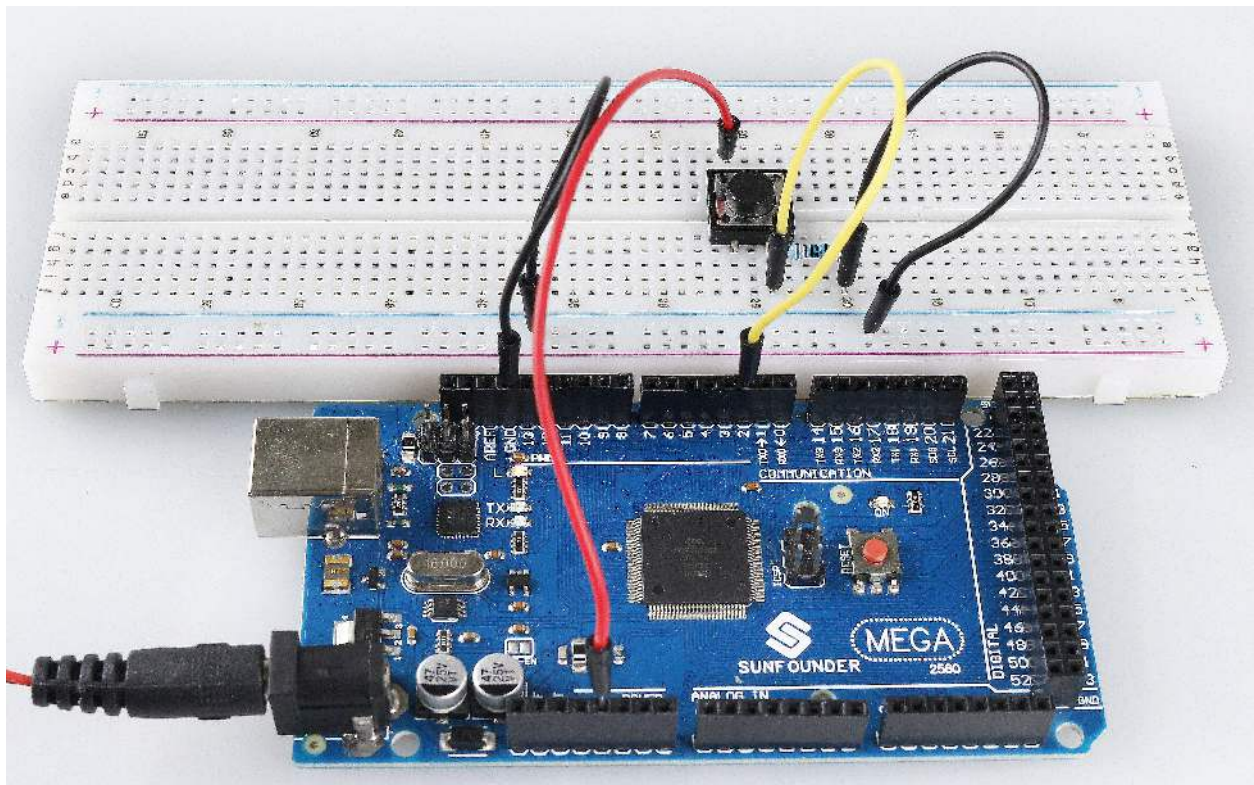
```
delay(1);
```

2.4.7 ※ How to turn on Serial Port Monitor

Click the magnifier icon at the top right of the Arduino IDE programming window to open the **Serial Monitor**.



2.4.8 Phenomenon Picture

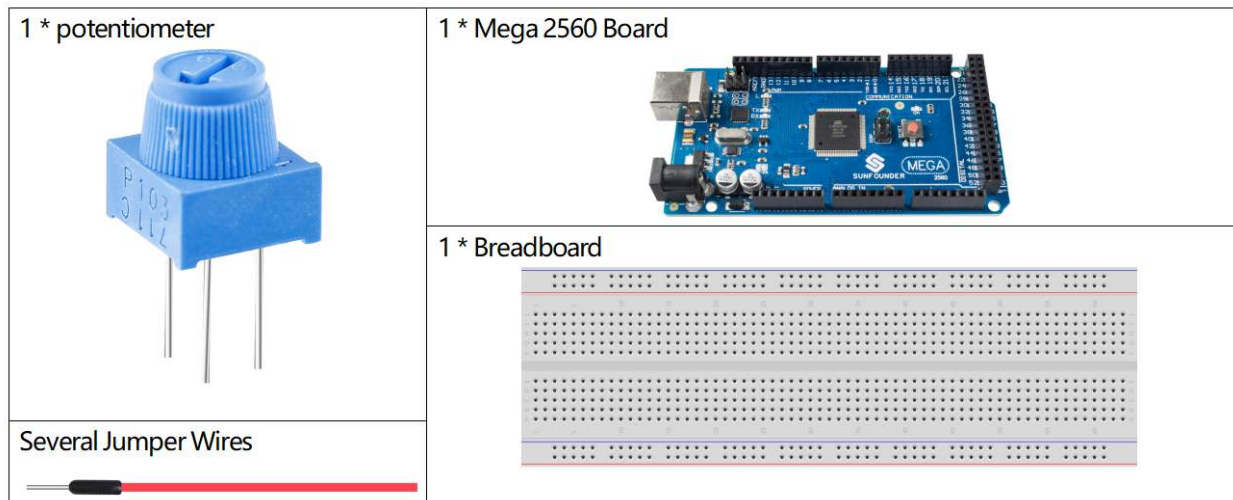


2.5 1.5 Analog Read

2.5.1 Overview

You can use the `analogRead()` command to read analog input from the physical world through an analog pin, which is suitable for analog input elements such as potentiometers, photoresistance, water level detection sensors, and so on. This article will take the potentiometer as an example to read the analog value of its output.

2.5.2 Components Required

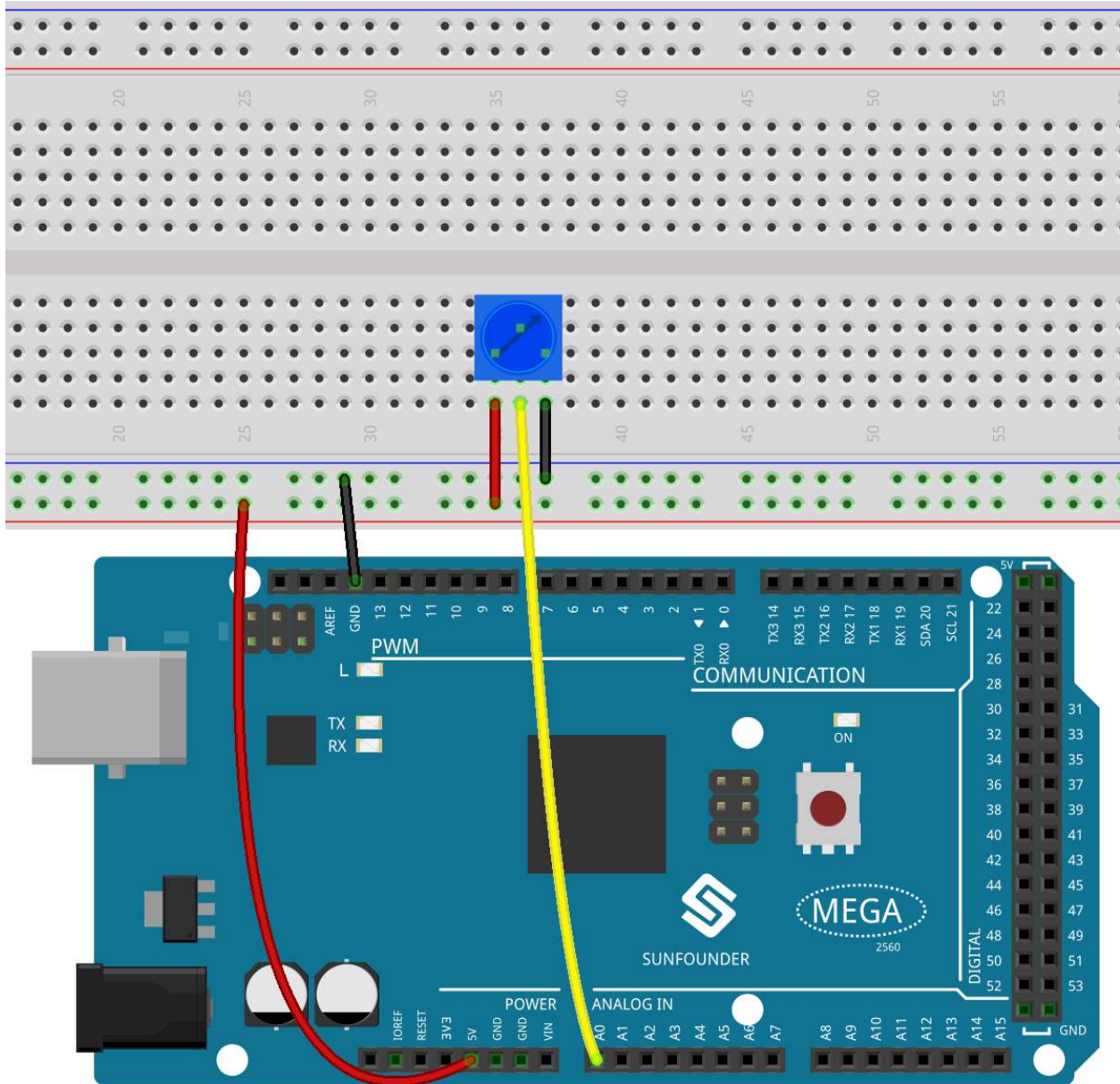


- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Potentiometer*

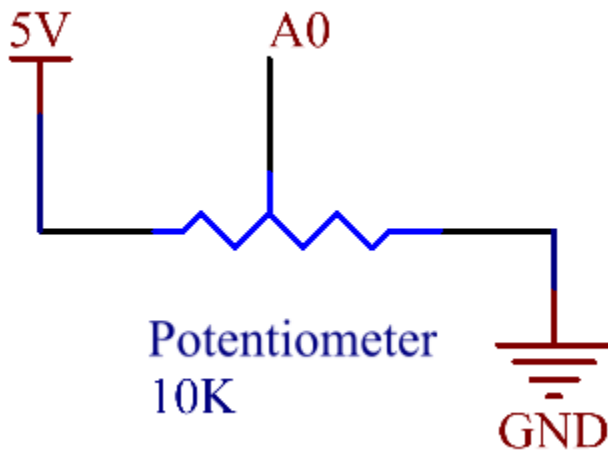
2.5.3 Fritzing Circuit

In this example, we use the analog pin (A0) to read the value of the potentiometer. Connect the pins at both ends of the potentiometer to 5V and GND respectively. Connect the middle pin to A0.

The voltage of the middle pin will be output to the Mega2560 board as an analog value. By rotating the axis of the potentiometer, you can change the voltage on the middle pin, thereby changing the analog value of the pin obtained by A0.



2.5.4 Schematic Diagram



2.5.5 Code

Note:

- You can open the file `1.5_analogRead.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\1.5_analogRead` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After the code is uploaded to the Mega2560 board, you can open the serial port monitor to view the reading value of the pin. When the shaft of the potentiometer is turned, the serial port monitor will print the value that changes between “0” and “1023”.

2.5.6 Code Analysis

To enable Arduino IDE to print the value transmitted from electronic component to the Mega2560 board, you need to start serial communication in `setup()` and set the data rate to 9600.

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

Use the `analogRead()` statement in `loop()` to read the level state acquired by analog pin A0 and declare a variable to store the level state.

```
int sensorValue = analogRead(A0);
```

Print the value stored in the variable on the serial monitor.

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

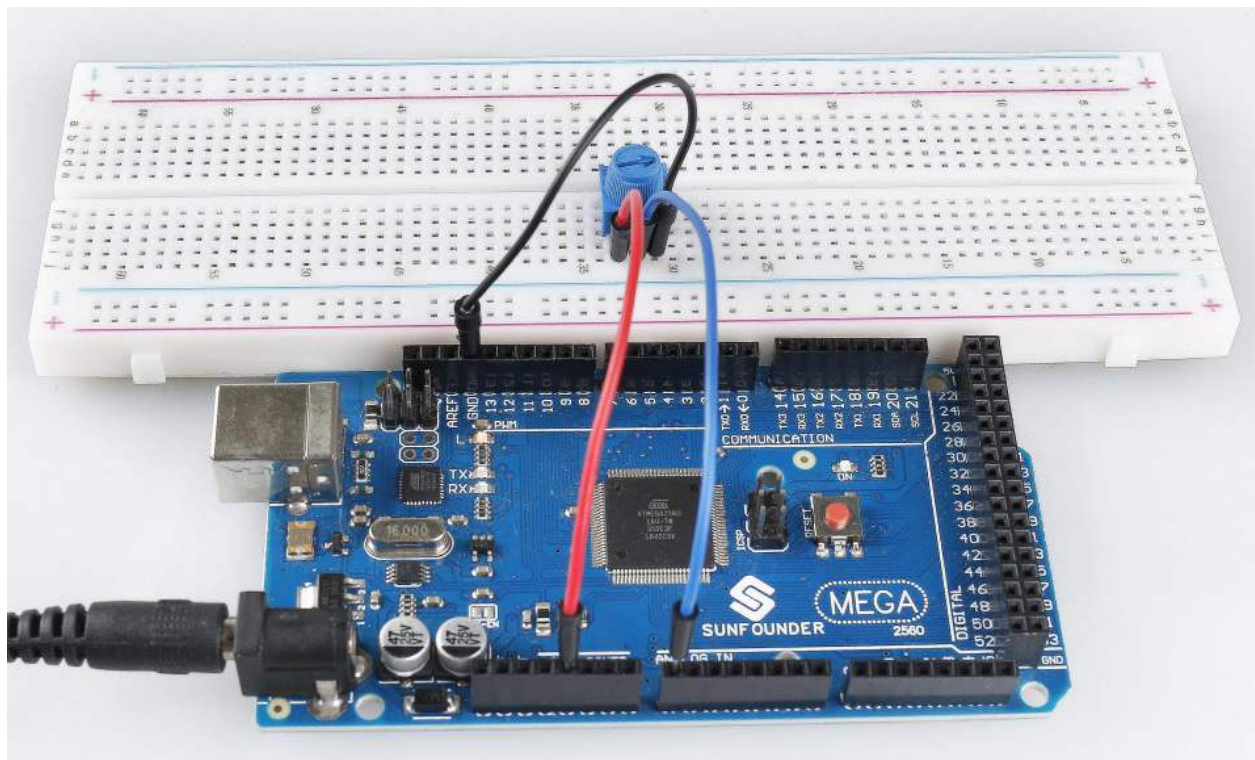
Use `delay()` statements to make printing results easy to observe.

```
delay(1);
```

2.5.7 ※ Analog-to-Digital Converter

The Arduino have a circuit inside called an analog-to-digital converter or ADC that reads this changing voltage and converts it to a number between 0 and 1023. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and the input value is 0. When the shaft is turned all the way in the opposite direction, there are 5 volts going to the pin and the input value is 1023. In between, `analogRead()` returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

2.5.8 Phenomenon Picture







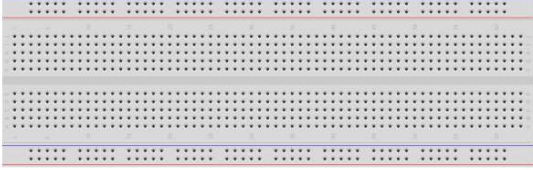


2.6 1.6 Digital Input Control Output

2.6.1 Overview

With the understanding of `digitalWrite()` and `digitalRead()`, we can build a complete I / O system to control the output device by obtaining the data from the input device. We can use this method to enable digital input components such as Button, Touch sensor, Infrared motion sensor to control digital output devices such as LED, active buzzer. This lesson will take Button and LED as examples to realize button control LED with the condition (if-else).

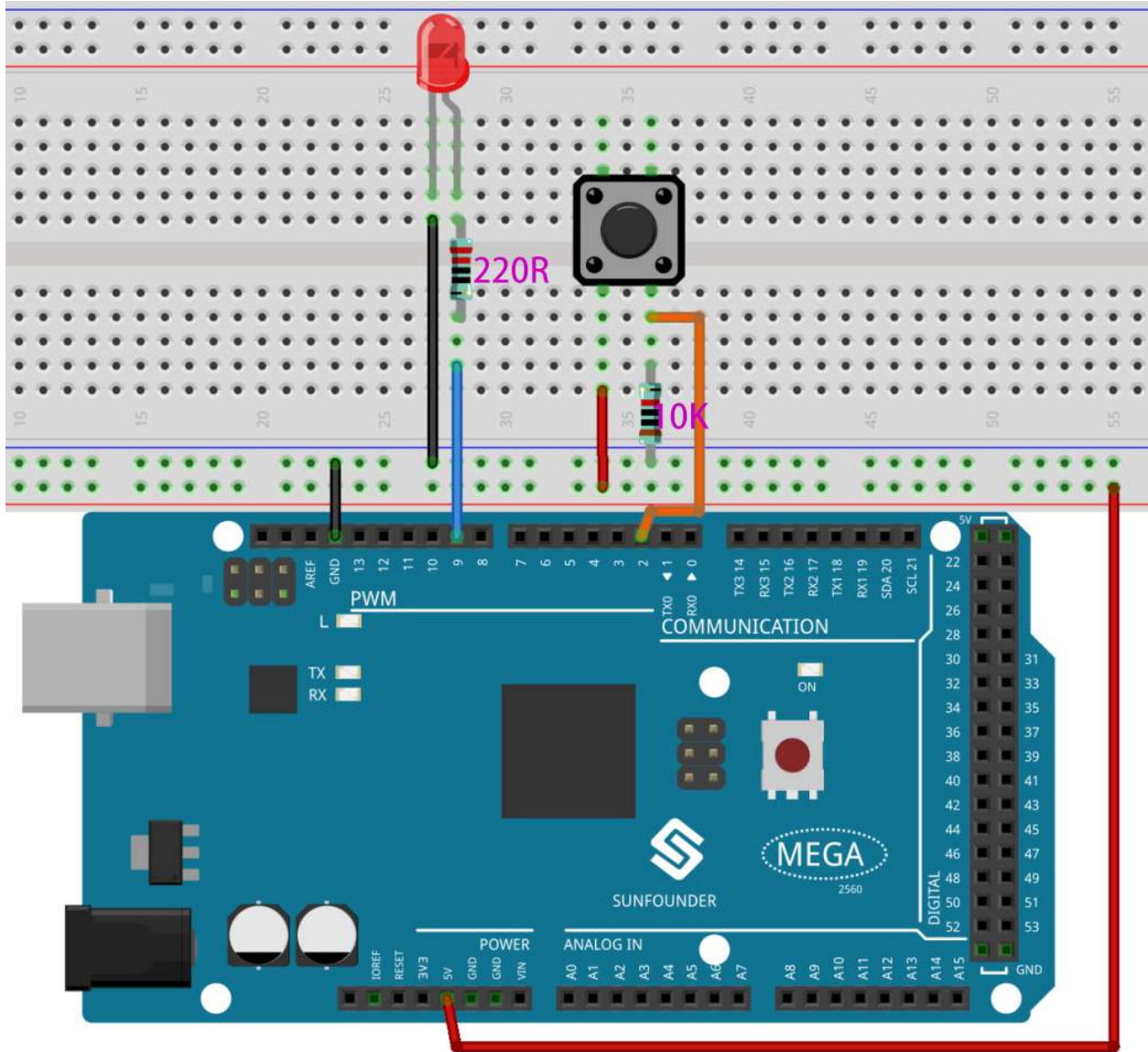
2.6.2 Components Required

<p>1 * Button</p> 	<p>1 * LED</p> 	<p>1 * 10k ohm resistor</p> 
	<p>Jumper wire</p> 	<p>1 * 220ohm Resistor</p> 
<p>1 * Mega 2560 Board</p> 	<p>1 * Breadboard</p> 	

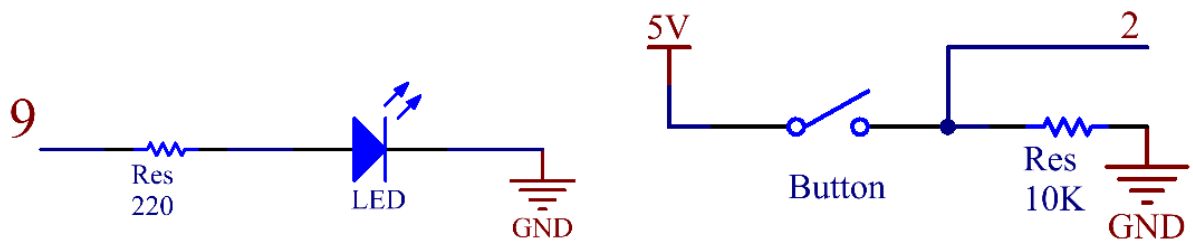
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Button*
- *Resistor*

2.6.3 Fritzing Circuit

In this example, we use pin 9 to drive LED. Use digital pin 2 to read the signal of Button. When the button is pressed, the LED lights up.



2.6.4 Schematic Diagram



2.6.5 Code

Note:

- You can open the file `1.6_digitalInputControlOutput.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\1.6_digitalInputControlOutput` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After uploading the code to the Mega2560 board, you can hold down Button to lighten the LED.

2.6.6 Code Analysis

Declare the pins of LED and Button and declare a variable to store the state of button.

```
const int buttonPin = 2;
const int ledPin = 9;
int buttonState = 0;
```

Initialize the pin mode in `setup()`.

```
pinMode(ledPin, OUTPUT);
pinMode(buttonPin, INPUT);
```

Read the status of the Button in `loop()` and assign it to the variable `buttonState`.

```
buttonState = digitalRead(buttonPin);
```

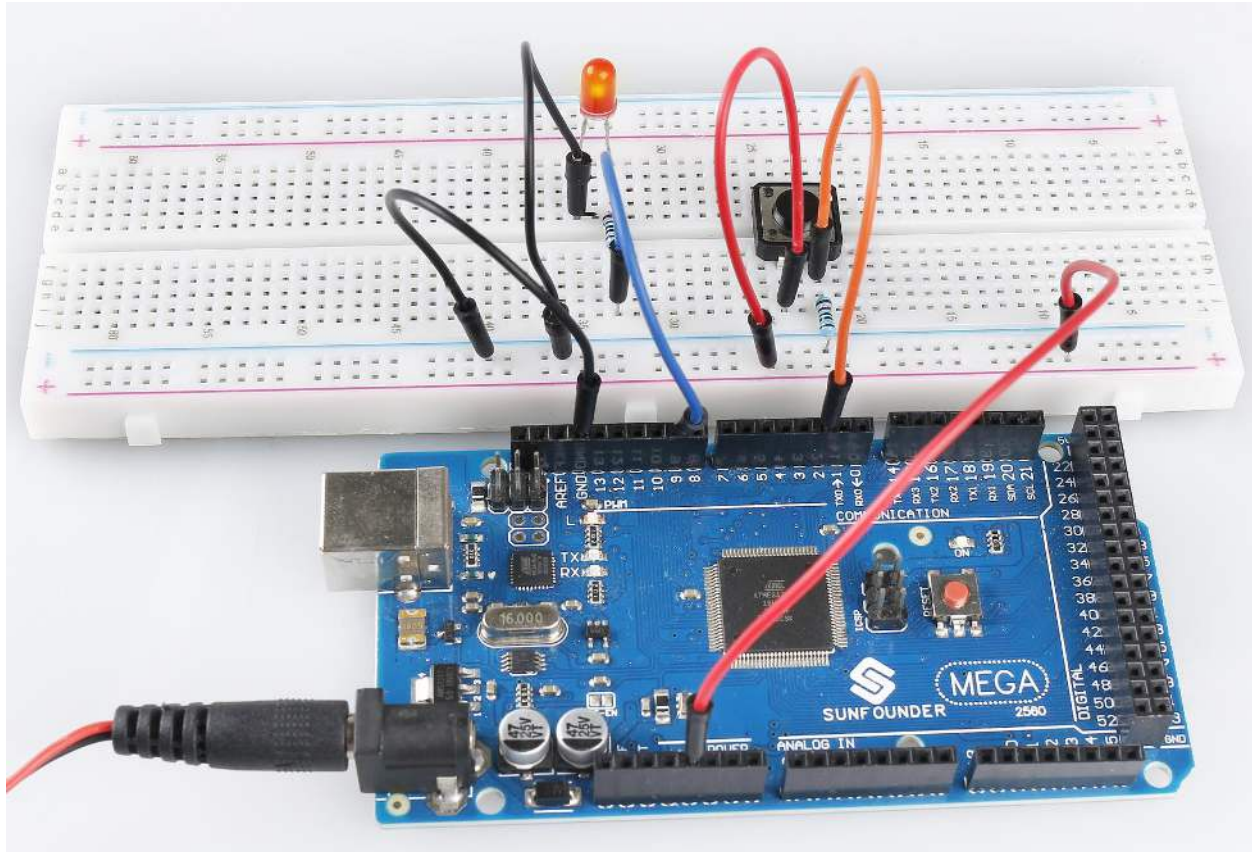
Use if condition to judge: if you get high level from a button, light up the LED.

```
if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
}
```

Otherwise, turn off the LED.

```
else {
    digitalWrite(ledPin, LOW);
}
```

2.6.7 Phenomenon Picture






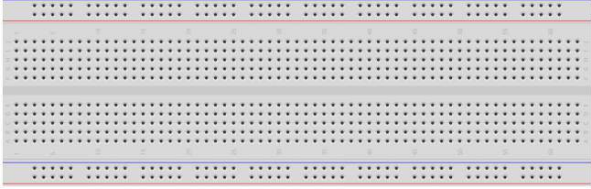


2.7 1.7 Analog Input Control Output

2.7.1 Overview

You can install an I/O system by using an analog input/ output device. For example, you can use potentiometer, photoresistor, water level sensor, etc., to control the brightness of LED, the speed of motor, and the like. In this lesson, potentiometer and LED are taken as examples to change the brightness of the LED when the potentiometer is turning.

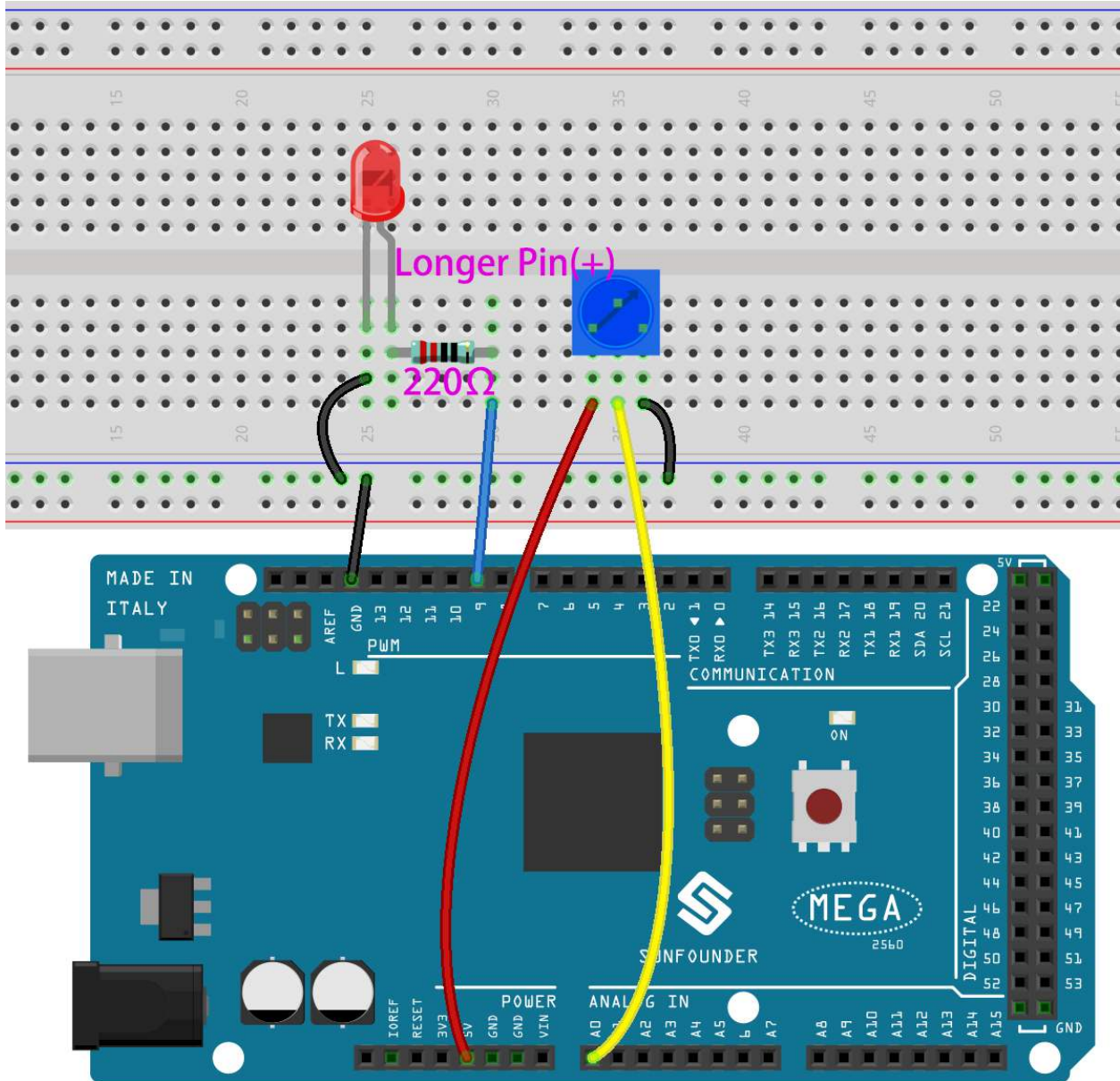
2.7.2 Components Required

<p>1 * Potentiometer</p> 	<p>1 * LED</p> 	<p>1 * 220 ohm resistor</p>  <p>Several Jumper Wires</p> 
<p>1 * Mega 2560 Board</p> 		<p>1 * Breadboard</p> 

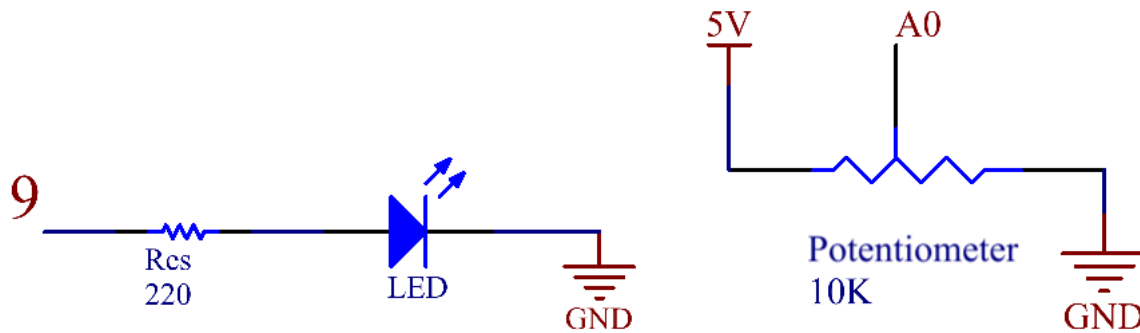
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *LED*
- *Resistor*
- *Potentiometer*

2.7.3 Fritzing Circuit

In this lesson, we use PWM pin 9 to drive LED. The analog pin (A0) is used to read the value of potentiometer. After uploading the code, you'll notice that the brightness of the LED changes as the potentiometer rotates.



2.7.4 Schematic Diagram



2.7.5 Code

Note:

- You can open the file `1.7_analogInputControlOutput.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\1.7_analogInputControlOutput` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

When the codes are uploaded to the Mega2560 board, you can see that the brightness of LED is changing with the turning of the knob of potentiometer.

2.7.6 Code Analysis

Declare the pins of LED and Button.

```
const int sensorPin = A0;
const int ledPin = 9;
```

In `setup()`, set the mode of `ledPin` to `OUTPUT`.

```
pinMode(ledPin, OUTPUT);
```

Read the readings of potentiometer in `loop()`.

```
int sensorValue=analogRead(sensorPin);
```

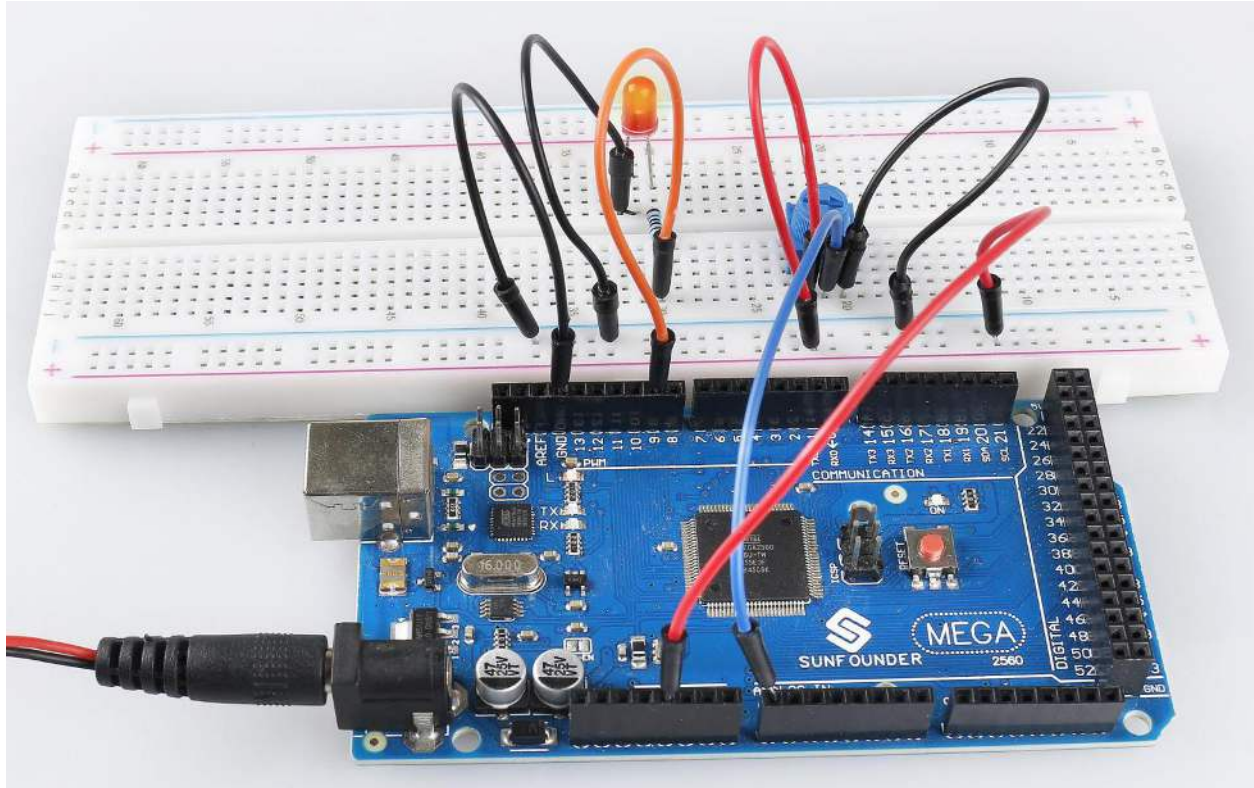
Map the potentiometer reading to the LED brightness value (0-1024 is mapped to 0-255).

```
int brightness = map(sensorValue,0,1024,0,255);
```

Write the brightness value to LED.

```
analogWrite(ledPin,brightness);
```

2.7.7 Phenomenon Picture





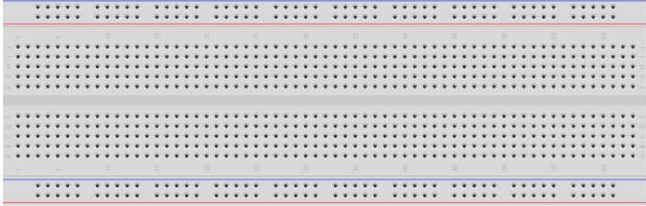


2.8 1.8 Serial Read

2.8.1 Overview

In addition to reading data from the electronic components, the Mega2560 board can read the data input in the serial port monitor, and you can use `Serial.read()` as the controller of the circuit experiment. Then we use LED to experiment with the `Serial. Read()` statement to control LED to turn on and off.

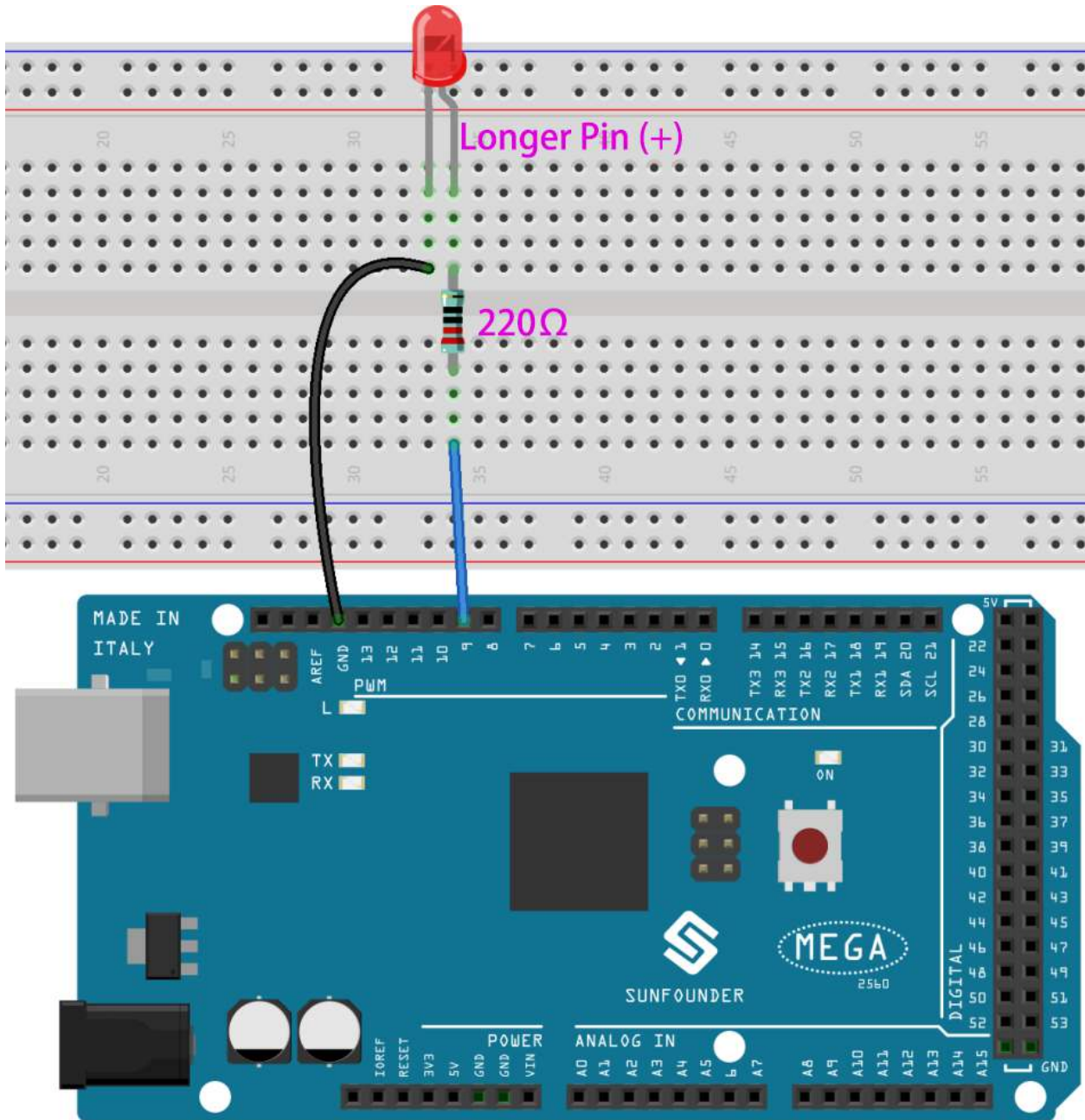
2.8.2 Components Required

<p>1 * LED</p> 	<p>1 * 220ohm Resistor</p> 	<p>Several Jumper Wires</p> 
<p>1 * Mega 2560 Board</p> 	<p>1 * Breadboard</p> 	

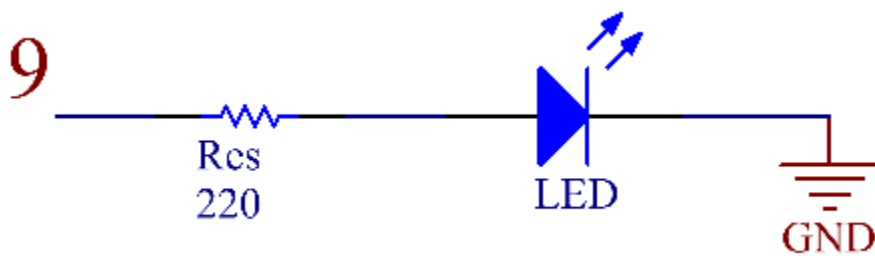
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *LED*
- *Resistor*

2.8.3 Fritzing Circuit

In this example, we use digital pin 9 to drive LED. When 1 is entered in serial monitor, the LED lights up. When 0 is entered, the LED turns off.



2.8.4 Schematic Diagram

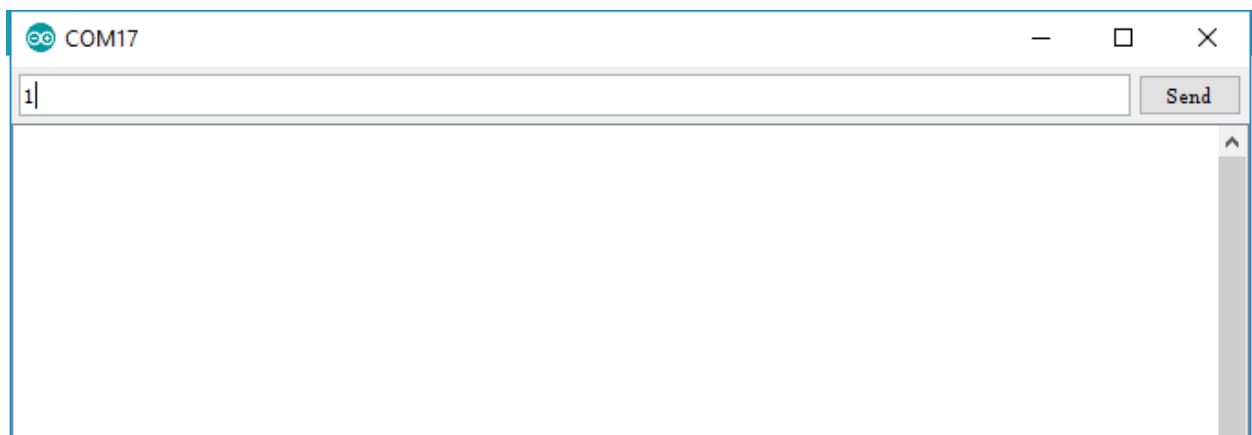


2.8.5 Code

Note:

- You can open the file `1.8_serialRead.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\1.8_serialRead` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After the codes are uploaded to the Mega2560 board, please turn on the serial port monitor. Typing in "1" can make LED turn on and typing in "0" can make it turn off.



2.8.6 Code Analysis

Declare digital pin 9 as ledPin.

```
const int ledPin = 9;
```

`Serial.read()` reads a single byte of ASCII value, and therefore you need to declare a int type variable, `incomingByte` to store the acquired data.

```
int incomingByte = 0;
```

Run the serial communication in `setup()` and set the data rate to 9600.

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

Set ledPin to OUTPUT mode.

```
pinMode(ledPin, OUTPUT);
```

The state of serial port monitor is judged in `loop()`, and the information processing will be carried out only when the data are received.

```
if (Serial.available() > 0){}
```

Reads the input value in the serial port monitor and stores it to the variable `incomingByte`.

```
incomingByte = Serial.read();
```

When the character '1' is obtained, the LED is lit; when '0' is obtained, the LED turns off.

```
if(incomingByte=='1'){digitalWrite(ledPin,HIGH);}  
else if(incomingByte=='0'){digitalWrite(ledPin,LOW);}
```

Note: Serial.read() takes the ASCII value in single character, which means that when you input '1', the obtained value is not the number '1', but the character '1' whose corresponding ASCII value is 49.

2.8.7 ※ ASCII chart

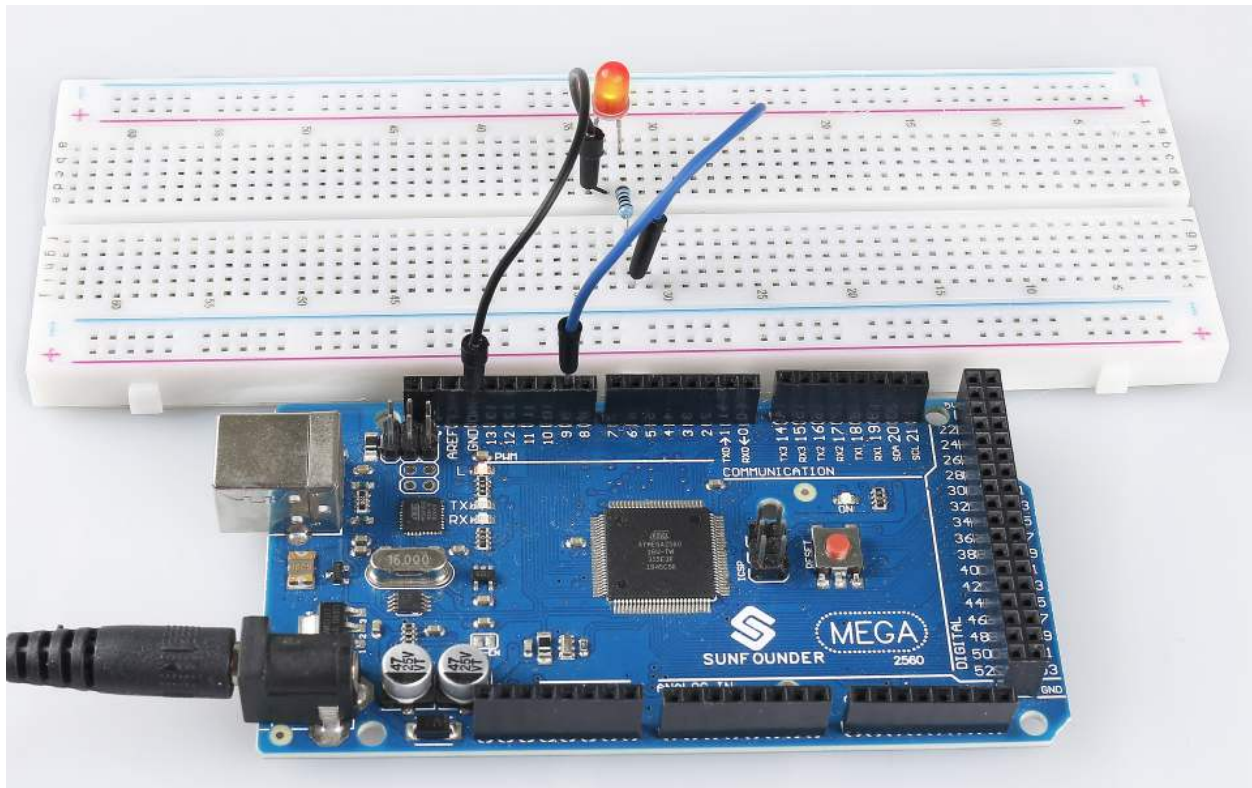
The ASCII (American Standard Code for Information Interchange) encoding dates to the 1960's. It is the standard way that text is encoded numerically.

Note that the first 32 characters (0-31) are non-printing characters, often called control characters. The more useful characters have been labeled.

ASCII Character	ASCII Character	ASCII Character	ASCII Character	ASCII Character	ASCII Character	ASCII Character	ASCII Character	ASCII Character	ASCII Character		
0	Null	23		46	.	69	E	92	/	115	s
1		24		47	/	70	F	93]	116	t
2		25		48	0	71	G	94	^	117	u
3		26		49	1	72	H	95	_	118	v
4		27		50	2	73	I	96	`	119	w
5		28		51	3	74	J	97	a	120	x
6		29		52	4	75	K	98	b	121	y
7		30		53	5	76	L	99	c	122	z
8		31		54	6	77	M	100	d	123	{
9	tab	32	space	55	7	78	N	101	e	124	
10	line feed	33	!	56	8	79	O	102	f	125	}

11		34	"	57	9	80	P	103	g	126	`
12		35	#	58	:	81	Q	104	h	127	
13	carriage return	36	\$	59	;	82	R	105	i		
14		37	%	60	<	83	S	106	j		
15		38	&	61	=	84	T	107	k		
16		39	,	62	>	85	U	108	l		
17		40	(63	?	86	V	109	m		
18		41)	64	@	87	W	110	n		
19		42	*	65	A	88	X	111	o		
20		43	+	66	B	89	Y	112	p		
21		44	,	67	C	90	Z	113	q		
22		45	-	68	D	91	[114	r		

2.8.8 Phenomenon Picture




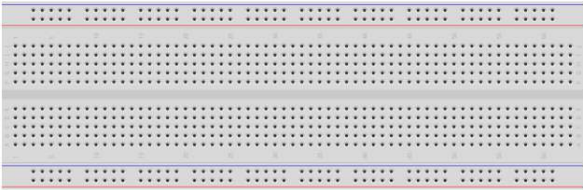


2.9 1.9 Digital Input Pull-Up

2.9.1 Overview

When using some switch input devices, some pull-up or pull-down resistors are often used to keep the level of corresponding pins at certain value on the condition that the device is not working. Such as in 1.4 Digital Read, a 10k resistor is used to make the pin be connected to GND under the condition that the button is not pressed down. If we have used a lot of switch input components and want to simplify the circuit, we can set the pin mode to `INPUT_PULLUP` in the code so that the pin reads the high level in the suspended state.

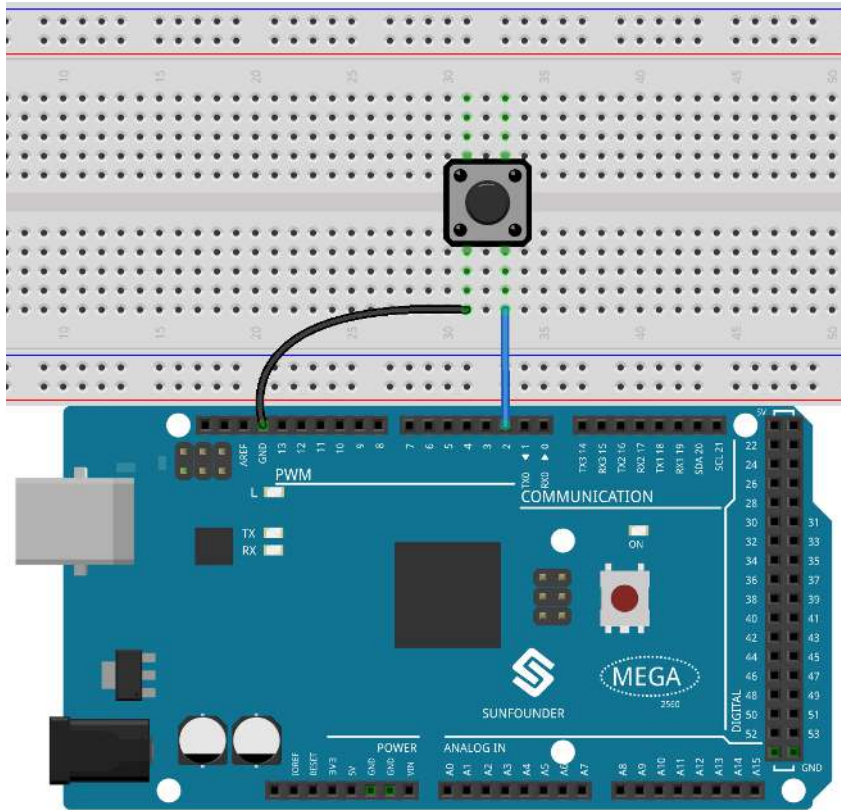
2.9.2 Components Required

1 * potentiometer 	Several Jumper Wires 
1 * Mega 2560 Board 	1 * Breadboard 

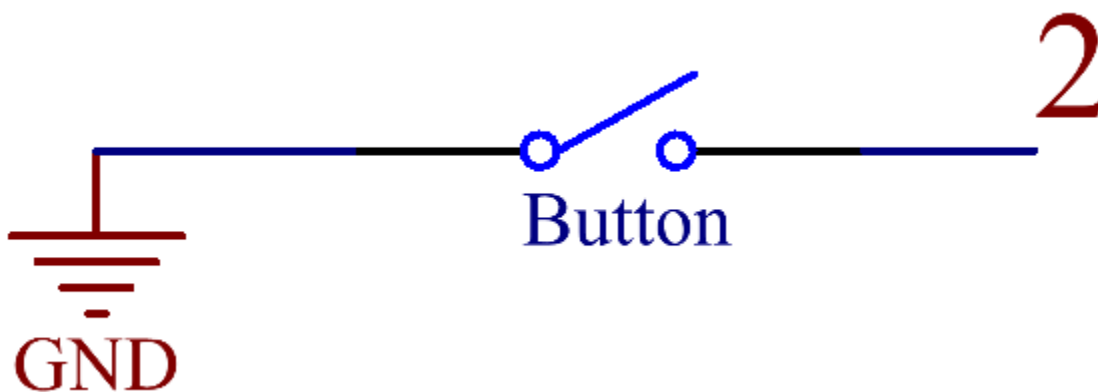
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Button*

2.9.3 Fritzing Circuit

In this example, we use pin 2 to read the signal of button. The internal pull-up in pin 2 is valid, so if the button isn't pressed, HIGH is read in pin 2; when the button is pressed, LOW is read.



2.9.4 Schematic Diagram



2.9.5 Code

Note:

- You can open the file `1.9_digitalInputPullup.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\1.9_digitalInputPullup` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Or click **Open Code** to open it in [Web Editor](#).
 - Then *Upload the Code* to the board.
-

After the codes are uploaded to the Mega2560 board, you can open the serial port monitor to view the read values of the pin. When the Button is pressed, the serial port monitor will display “0”, and the “1” will be displayed when the button is released.

2.9.6 Code Analysis

Run the serial communication in `setup()` and set the data rate to 9600.

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

Configure pin 2 as an input and enable the internal pull-up resistor.

```
pinMode(2, INPUT_PULLUP);
```

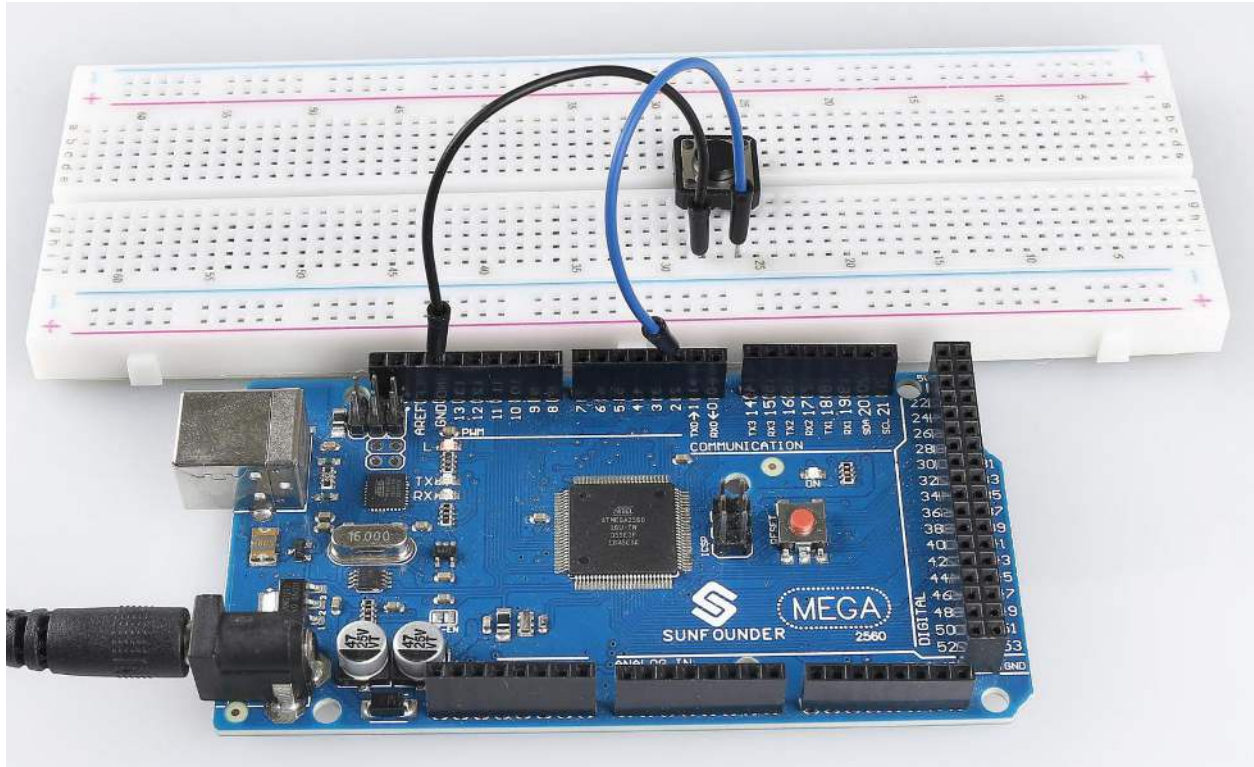
Read the level state from the digital pin 2 by using `digitalRead()` statement in `loop()` and declare a variable to store it.

```
int buttonState = digitalRead(2);
```

Print the values stored by variables on the serial port monitor.

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


2.9.7 Phenomenon Picture





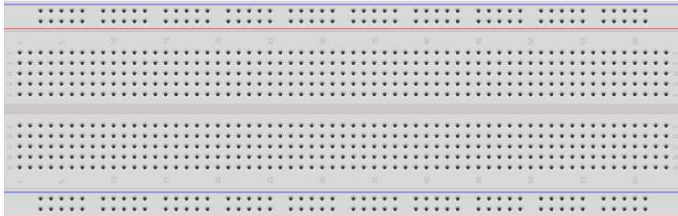


2.10 1.10 State Change Detection

2.10.1 Overview

When you use button, you can not only press down the button, light on the LED, release the button, turn off the LED, but also can switch the working state of the LED every time the button is pressed. In order to achieve this effect, you need to know when the state of the button changes from **off** to **on**, that is, “state change detection”. In this lesson, we will print the results of state change detection of the button in the serial monitor.

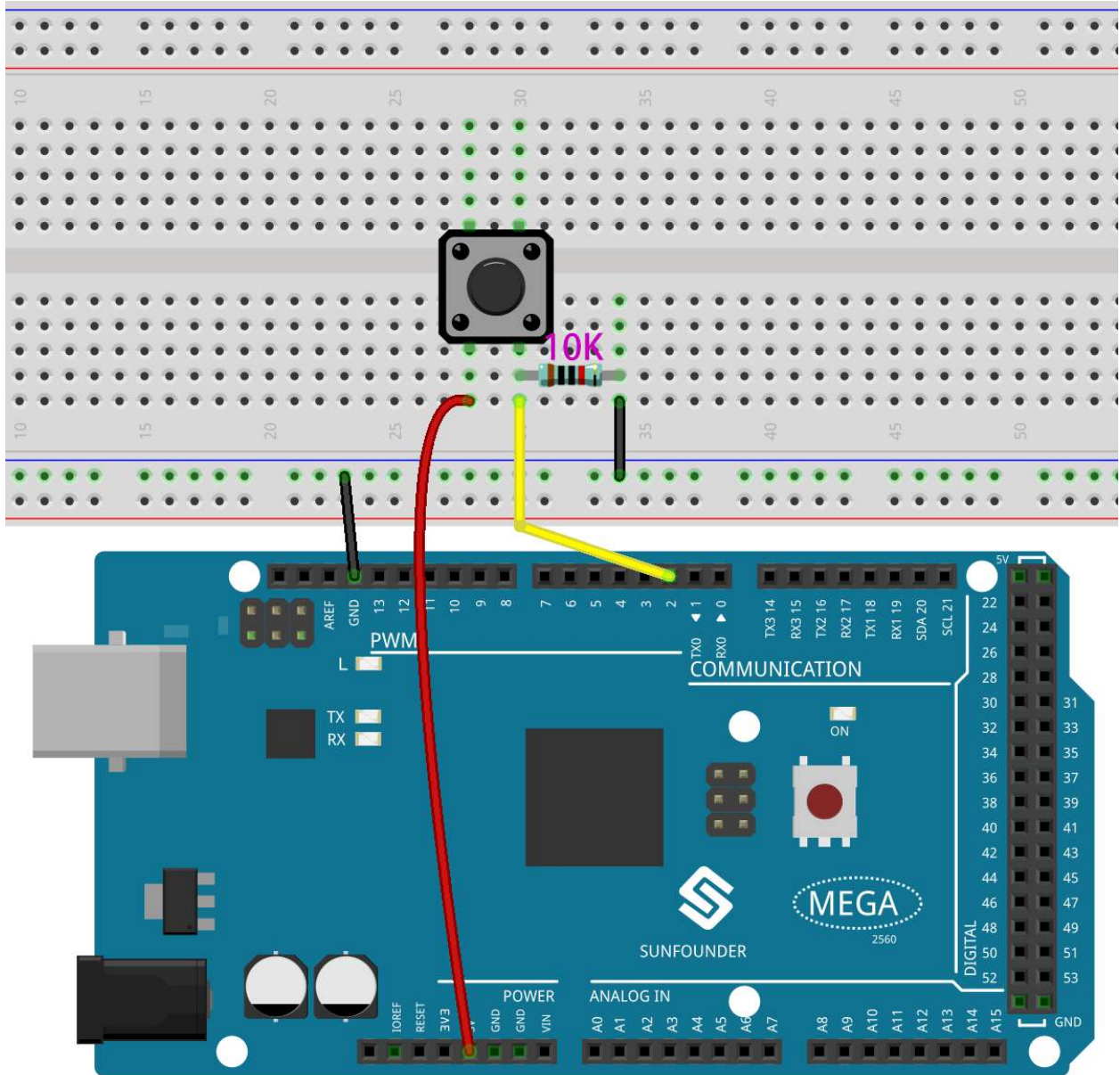
2.10.2 Components Required

1 * Button 	1 * 10k ohm resistor 	Several Jumper Wires 
1 * Mega 2560 Board 	1 * Breadboard 	

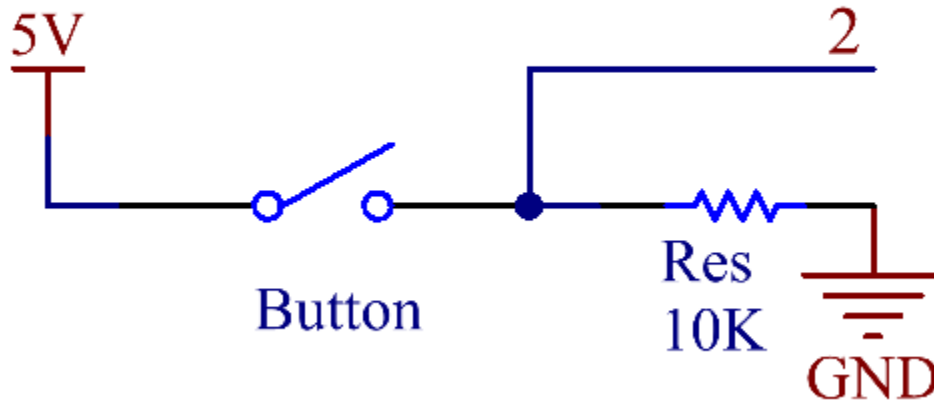
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Button*
- *Resistor*

2.10.3 Fritzing Circuit

In this example, we use pin 2 to read the signal of the button.



2.10.4 Schematic Diagram



2.10.5 Code

Note:

- You can open the file `1.10_stateChangeDetection.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\1.10_stateChangeDetection` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Or click **Open Code** to open it in [Web Editor](#).
 - Then *Upload the Code* to the board.
-

After the codes are uploaded into the Mega2560 board, the output number will switch between 0 and 1 every time you press the button.

2.10.6 Code Analysis

Declare a pin connected to Button.

```
const int buttonPin = 2;
```

Declare a variable called `detectionState` to record every state of state change detection.

```
int detectionState = 0;
```

Declare two variables to read the state of the button for state change detection.

```
int buttonState = 0;  
int lastButtonState = 0;
```

In `setup()`, initialize the pins and then start up the serial monitor.

```
pinMode(buttonPin, INPUT);  
Serial.begin(9600);
```

In loop(), read the value of buttonPin and then assign to the variable buttonState.

```
buttonState = digitalRead(buttonPin);
```

Compare buttonState with lastButtonState, if they are not equal, it indicates that the state is changed. A delay(50) is needed to realize debouncing during the changing detection. After comparison, assign the buttonState to lastButtonState to make the next round of judgment.

```
if (buttonState != lastButtonState) {  
  ...  
  delay(50);  
}  
lastButtonState = buttonState;
```

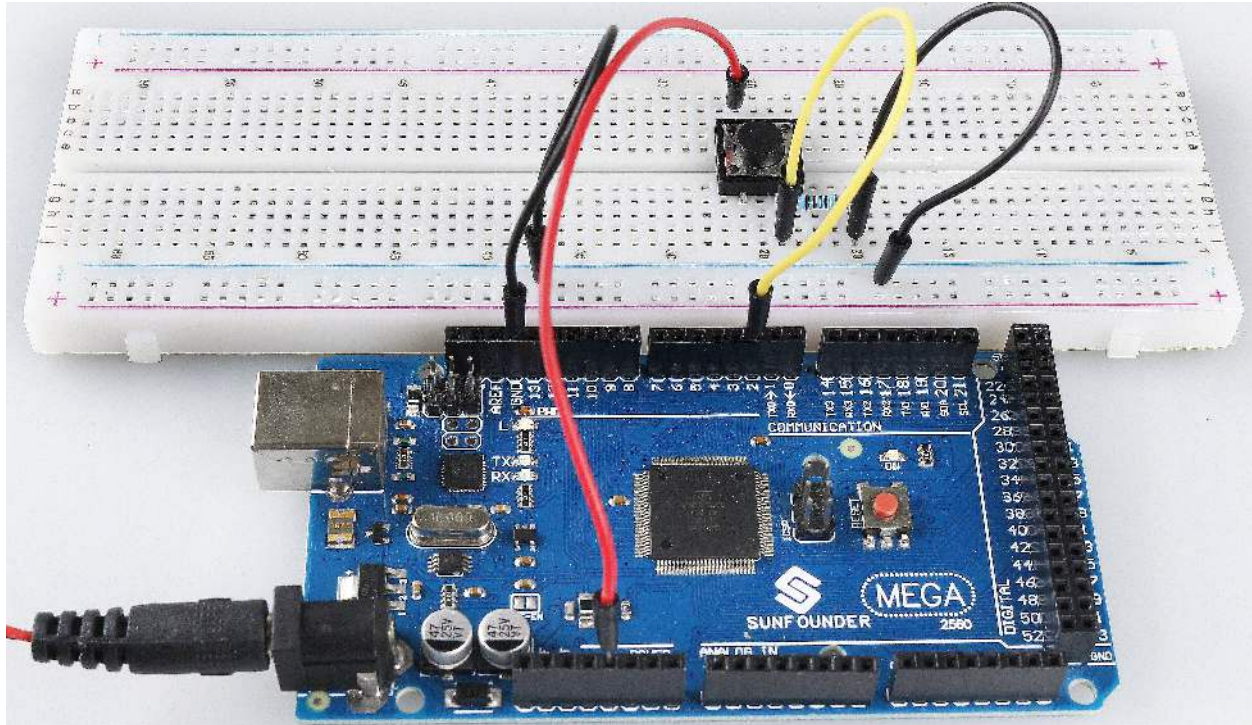
The state change judgment installed (buttonState != lastButtonState), the further judgment is made to get the conditionPress the button.

```
if (buttonState == HIGH) {  
  ...  
}
```

Under the statePress the button, detectionState is being operated and it switches between 1 and 0. Meanwhile, the value of detectionState is printed.

```
detectionState=(detectionState+1)%2;  
Serial.print("The detection state is:");  
Serial.println(detectionState);
```

2.10.7 Phenomenon Picture



2.11 1.11 Interval

2.11.1 Overview

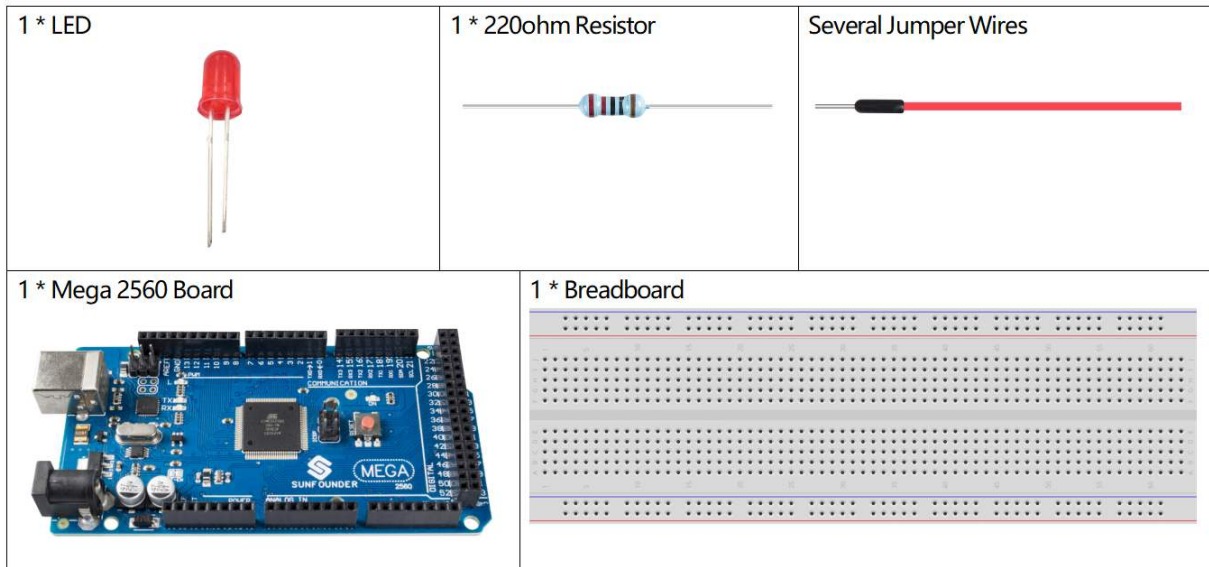
Sometimes you need to do two things at once. For example you might want to blink an LED while reading a button press. In this case, you can't use `delay()`, because Arduino pauses your program during the `delay()`. If the button is pressed while Arduino is paused waiting for the `delay()` to pass, your program will miss the button press.

This sketch demonstrates how to blink an LED without using `delay()`. It turns the LED on and then makes note of the time. Then, each time through `loop()`, it checks to see if the desired blink time has passed. If it has, it toggles the LED on or off and makes note of the new time. In this way the LED blinks continuously while the sketch execution never lags on a single instruction.

An analogy would be warming up a pizza in your microwave, and also waiting some important email. You put the pizza in the microwave and set it for 10 minutes. The analogy to using `delay()` would be to sit in front of the microwave watching the timer count down from 10 minutes until the timer reaches zero. If the important email arrives during this time you will miss it.

What you would do in real life would be to turn on the pizza, and then check your email, and then maybe do something else (that doesn't take too long!) and every so often you will come back to the microwave to see if the timer has reached zero, indicating that your pizza is done.

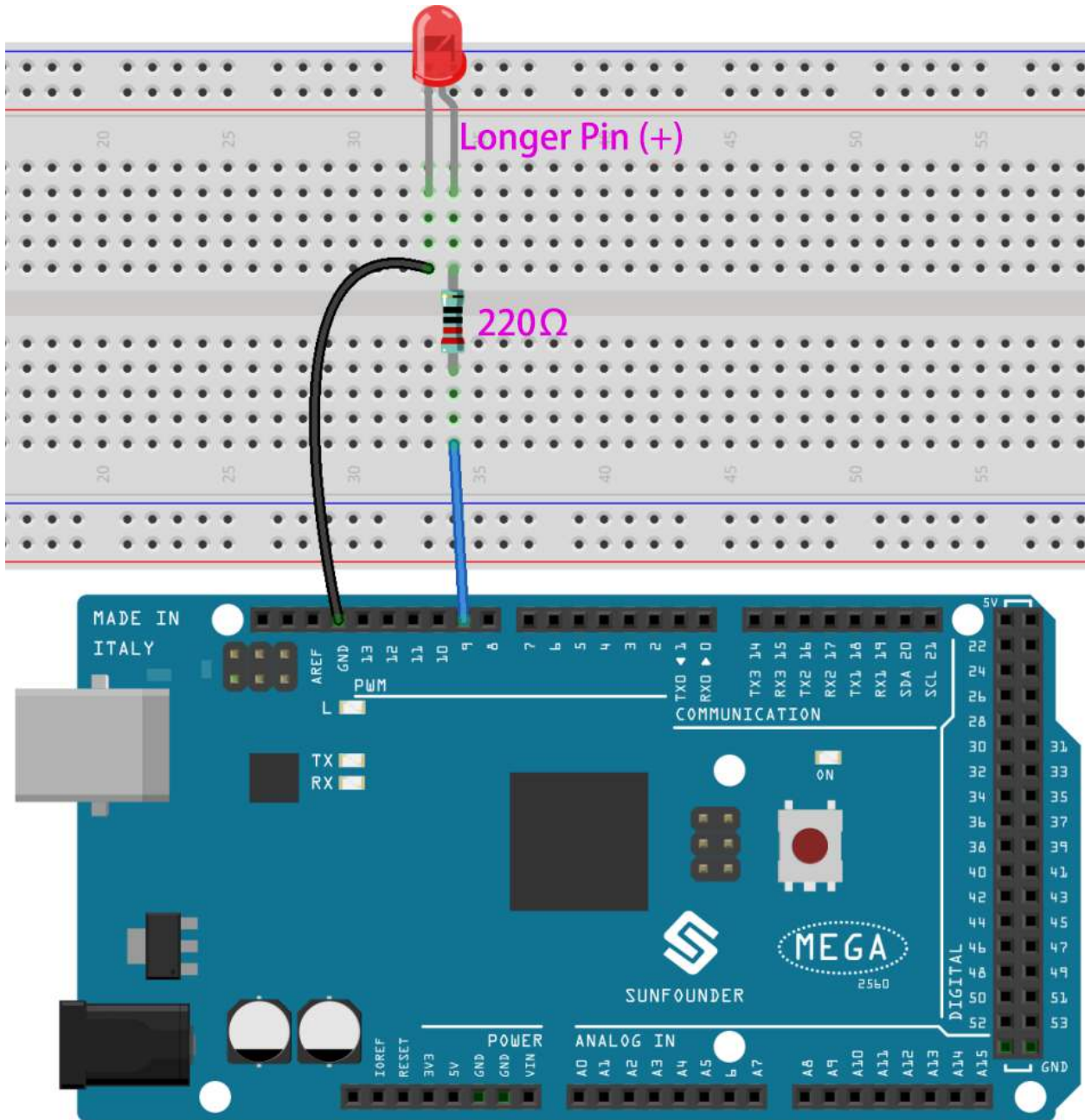
2.11.2 Components Required



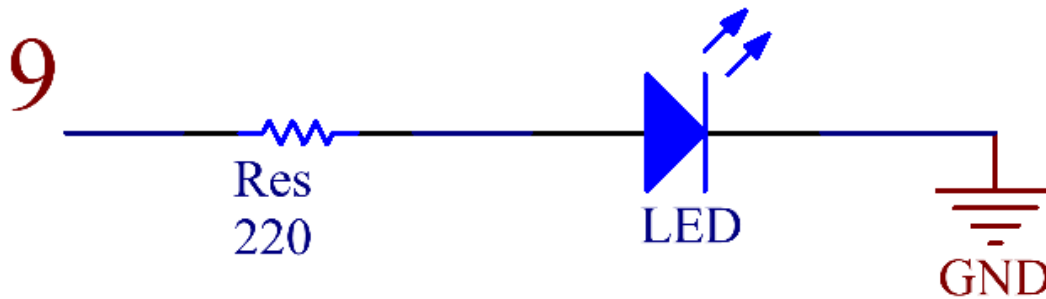
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *LED*
- *Resistor*

2.11.3 Fritzing Circuit

In this example, we use digital pin 9 to drive the LED, and we attach one side of the resistor to the corresponding digital pins. The longer pin of LED (a positive electrode, referred to as anode) is connected to the other side of the resistor. The shorter pin (a negative electrode, referred to as cathode) of LED is attached to GND.



2.11.4 Schematic Diagram



2.11.5 Code

Note:

- You can open the file `1.11_interval.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\1.11_interval` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

When you finish uploading the codes to the Mega2560 board, you can see the LED uploading.

2.11.6 Code Analysis

Declare the digital pin 9 as ledPin.

```
const int ledPin = 9;
```

Set the state of ledState to LOW to turn off the LED.

```
int ledState = LOW;
```

Initial a variable named previousMillis to store previous operating time of microcontroller.

```
unsigned long previousMillis = 0;
```

Set the interval time to 1000ms (milliseconds).

```
const long interval = 1000;
```

Set ledPin to OUTPUT mode.

```
pinMode(ledPin, OUTPUT);
```

In loop(), declare currentMillis to store the current time.

```
unsigned long currentMillis = millis();
```

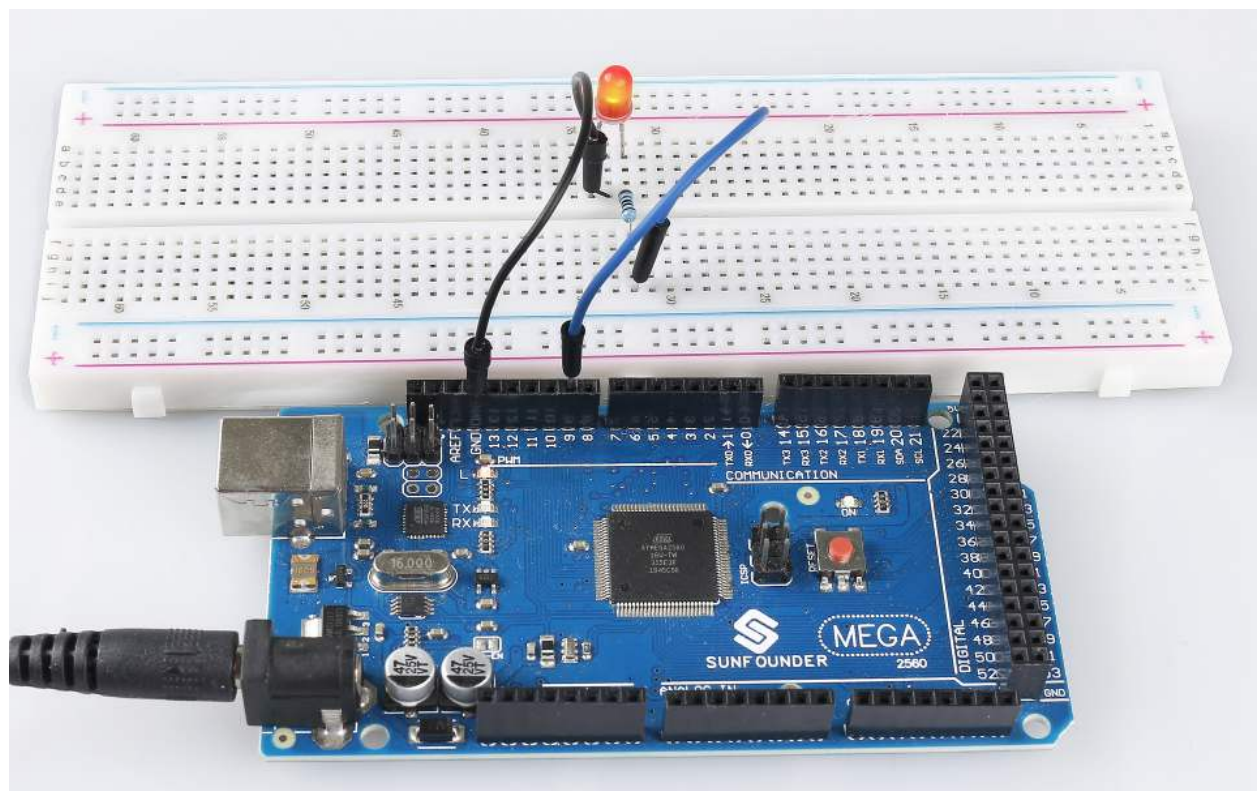
When the interval between the current operating time and last updating time is larger than 1000ms, certain functions are triggered. Meanwhile, update the previousMillis to the current time for the next triggering that is to happen 1 second latter.

```
if (currentMillis - previousMillis >= interval) {  
    previousMillis = currentMillis; // save the last time you blinked the LED  
    //...  
}
```

Here, certain functions executed at intervals are to change the state of LED.

```
if (ledState == LOW)  
{ledState = HIGH;}  
else  
{ledState = LOW;}  
digitalWrite(ledPin, ledState);
```

2.11.7 Phenomenon Picture



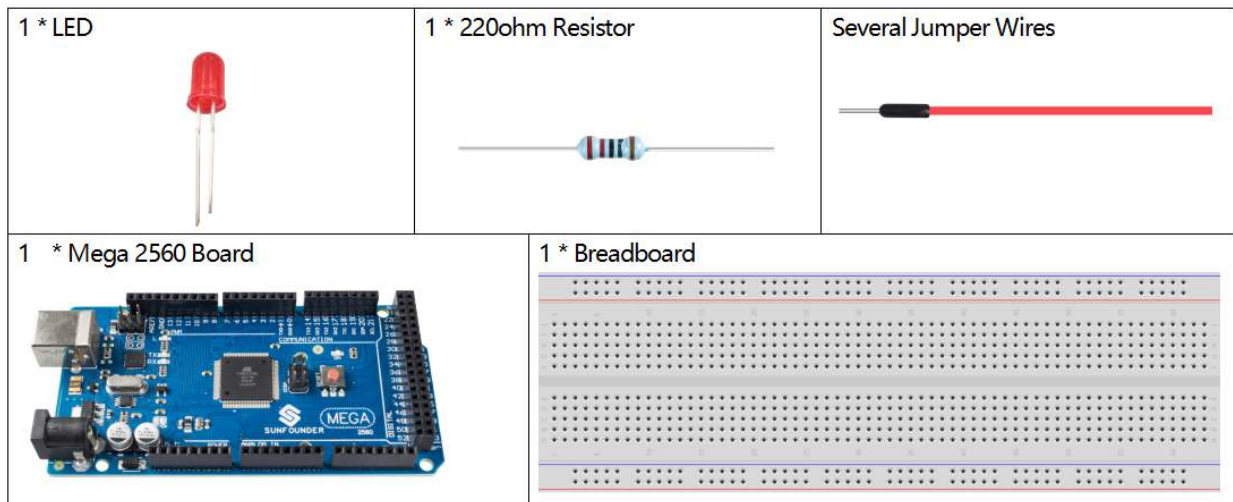
2. Projects for Each Component

2.12 2.2 LED

2.12.1 Overview

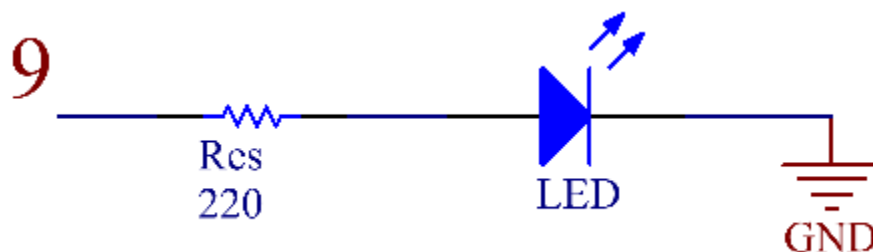
In this lesson, you will learn how to use LED. LED is a kind of common light-emitting device that works according to the principle that the recombination of electron and hole releases energy to give out light. This component is used widely in the current society, such as illumination, panel display, medical device and so on.

2.12.2 Components Required



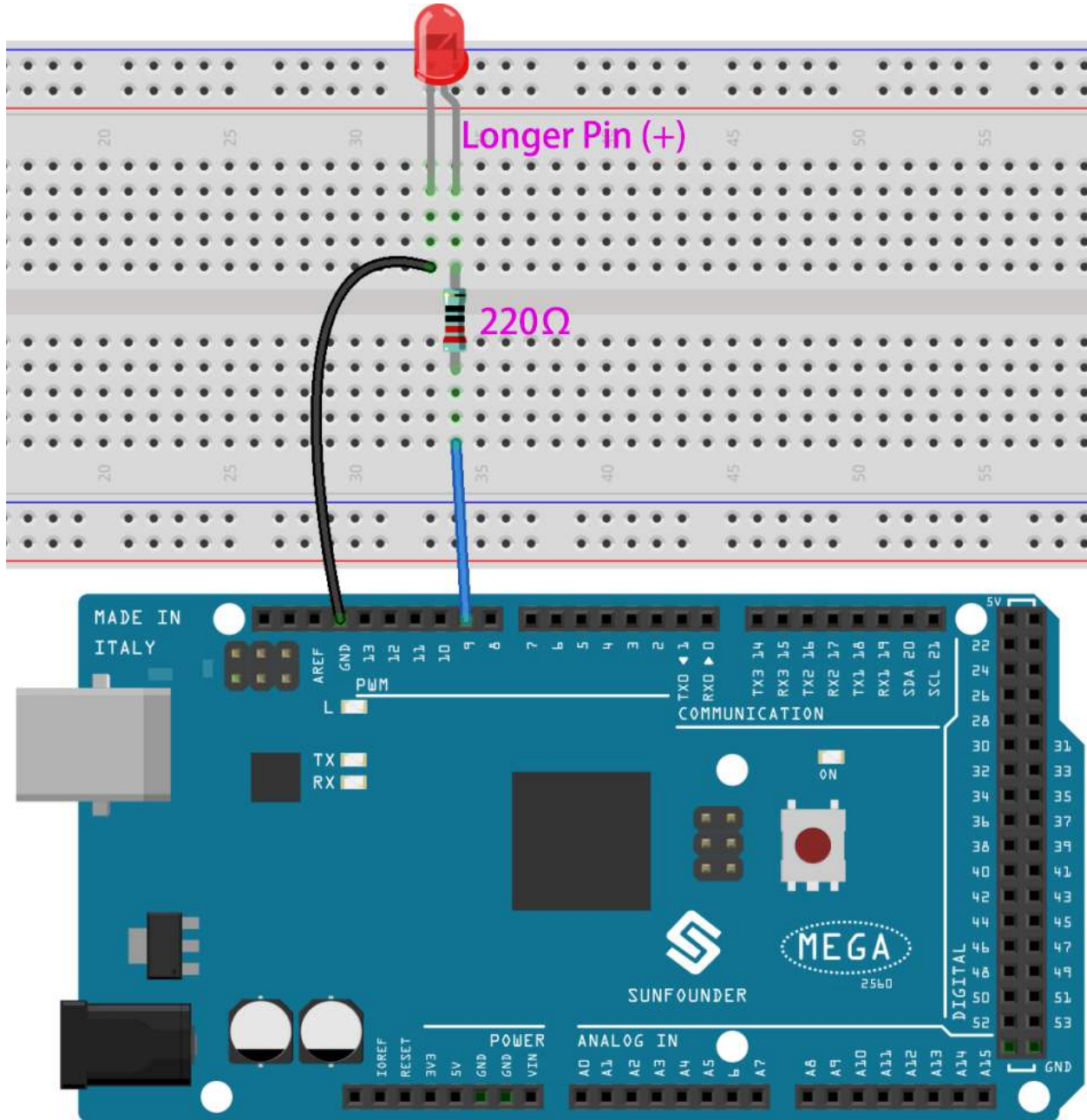
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *LED*
- *Resistor*

2.12.3 Schematic Diagram



2.12.4 Fritzing Circuit

In this example, we use pin 9 to drive LED. Insert one side of the resistor in the digital pin 9 and connect the longer pin (a positive electrode, referred to as anode) of the LED with the other side of the resistor. Extend the shorter pin (a negative electrode, referred to as cathode) of the LED to GND.



2.12.5 Code

Note:

- You can open the file `2.2_led.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.2_led` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Or click **Open Code** to open it in [Web Editor](#).
 - Then *Upload the Code* to the board.
-

Example 1

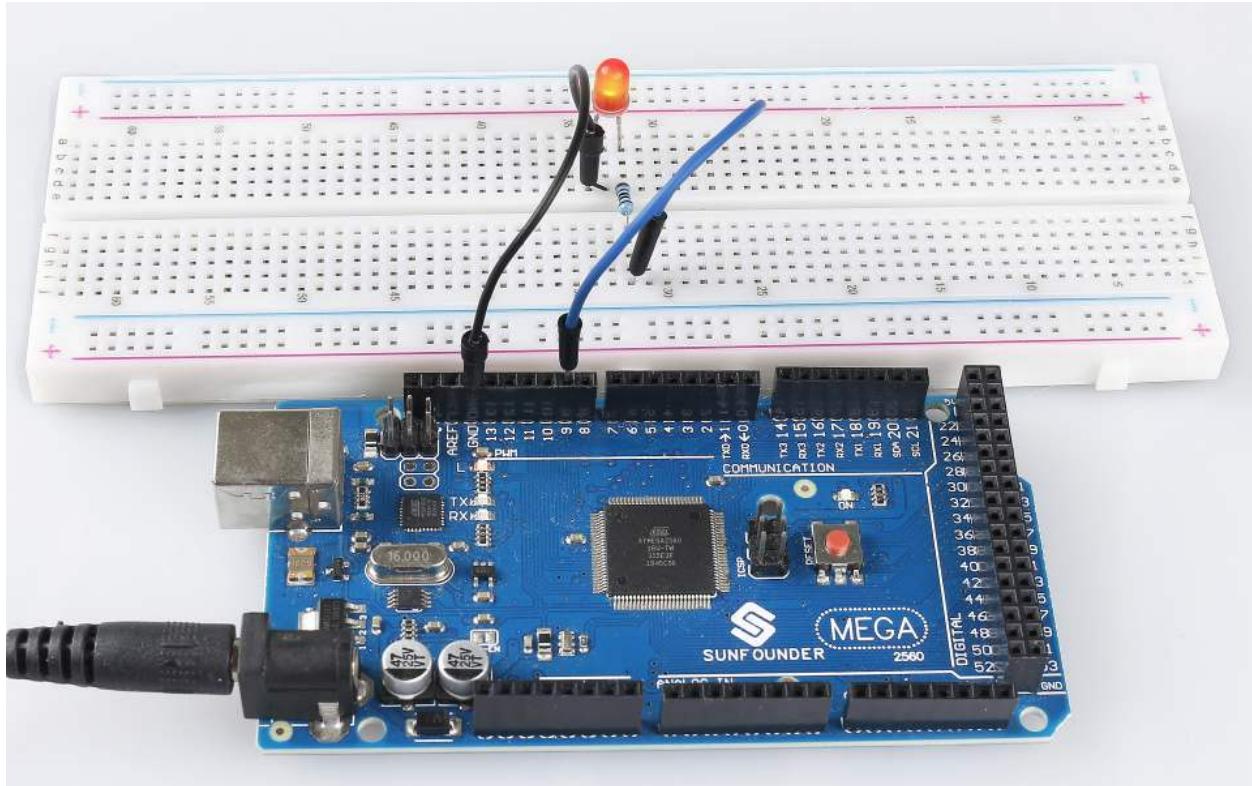
Finished uploading the codes to the Mega2560 board, you can see the LED flashing. Refer to [1.2 Digital Write](#) to check the detail code explanation.

Example 2:**Note:**

- You can open the file `2.2_led_2.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.2_led_2` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Or click **Open Code** to open it in [Web Editor](#).
 - Then *Upload the Code* to the board.
-

After uploading the codes to the Mega2560 board, you can see the LED getting brighter, then turning off, getting brighter, then turning off again. . . this loop will continue in this way. About the detail code explanation, please refer to [1.3 Analog Write](#).

2.12.6 Phenomenon Picture





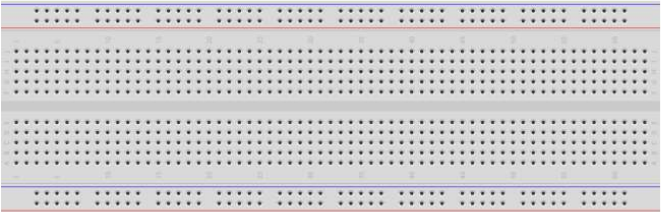


2.13 2.3 RGB LED

2.13.1 Overview

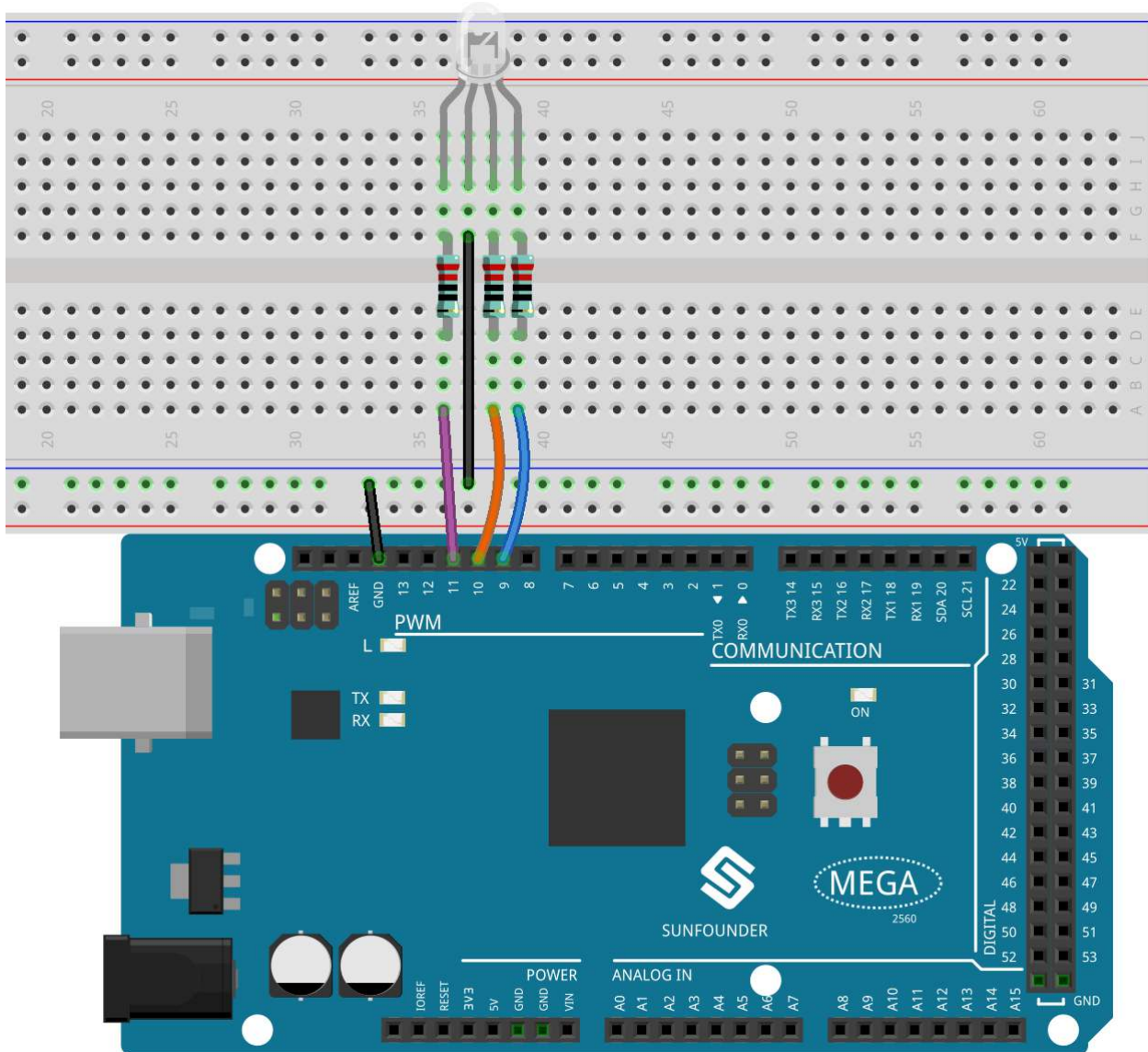
In this lesson, you will learn about how to use RGB LED. A RGB LED packages three LEDs in red, green and blue into one transparent or semitransparent plastic shell. It displays a broad array of colors by changing the input voltage of three pins and adding the three colors together in different ways. As is said in a statistic, RGB LED can create 16,777,216 different colors.

2.13.2 Components Required

<p>1 * RGB LED</p> 	<p>3 * 220 ohm resistor</p> 	<p>Several Jumper Wires</p> 
<p>1 * Mega 2560 Board</p> 	<p>1 * Breadboard</p> 	

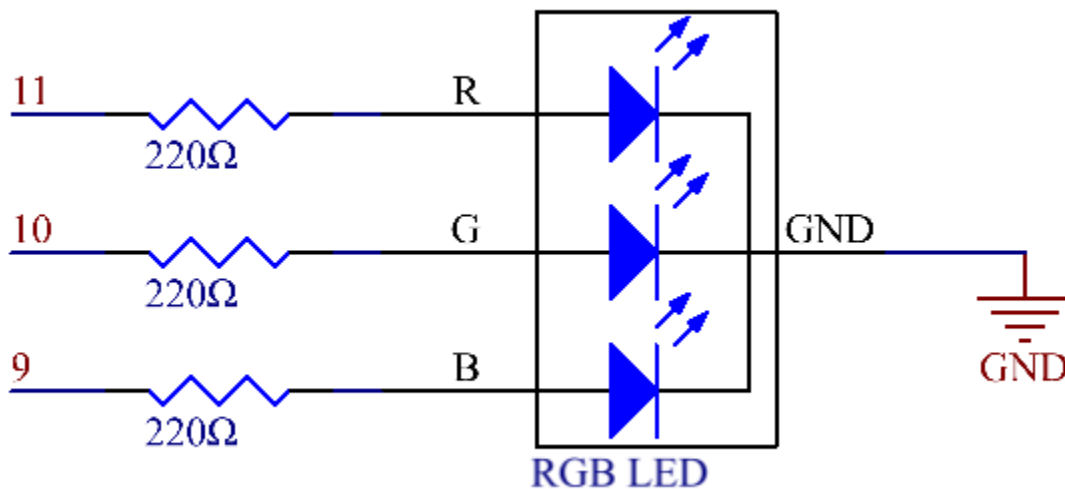
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *RGB LED*
- *Resistor*

2.13.3 Fritzing Circuit



Here we input a value between 0 and 255 to the three pins of the RGB LED to make it display different colors. After connecting the pins of R, G, and B to a current limiting resistor, connect them to the pin 9, pin 10, and pin 11 respectively. The longest pin (GND) of the LED connects to the GND of the Mega 2560. When the three pins are given different PWM values, the RGB LED will display different colors.

2.13.4 Schematic Diagram



2.13.5 Code

Note:

- You can open the file `2.3_rgbLed.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.3_rgbLed` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

2.13.6 Code Analysis

In this example, the function used to assign values to the three pins of RGB is packaged in an independent subfunction `color()`.

```
void color (unsigned char red, unsigned char green, unsigned char blue)// the color_
→generating function
{
analogWrite(redPin, red);
analogWrite(greenPin, green);
analogWrite(bluePin, blue);
}
```

In `loop()`, RGB value works as an input argument to call the function `color()` to realize that the RGB can emit different colors.

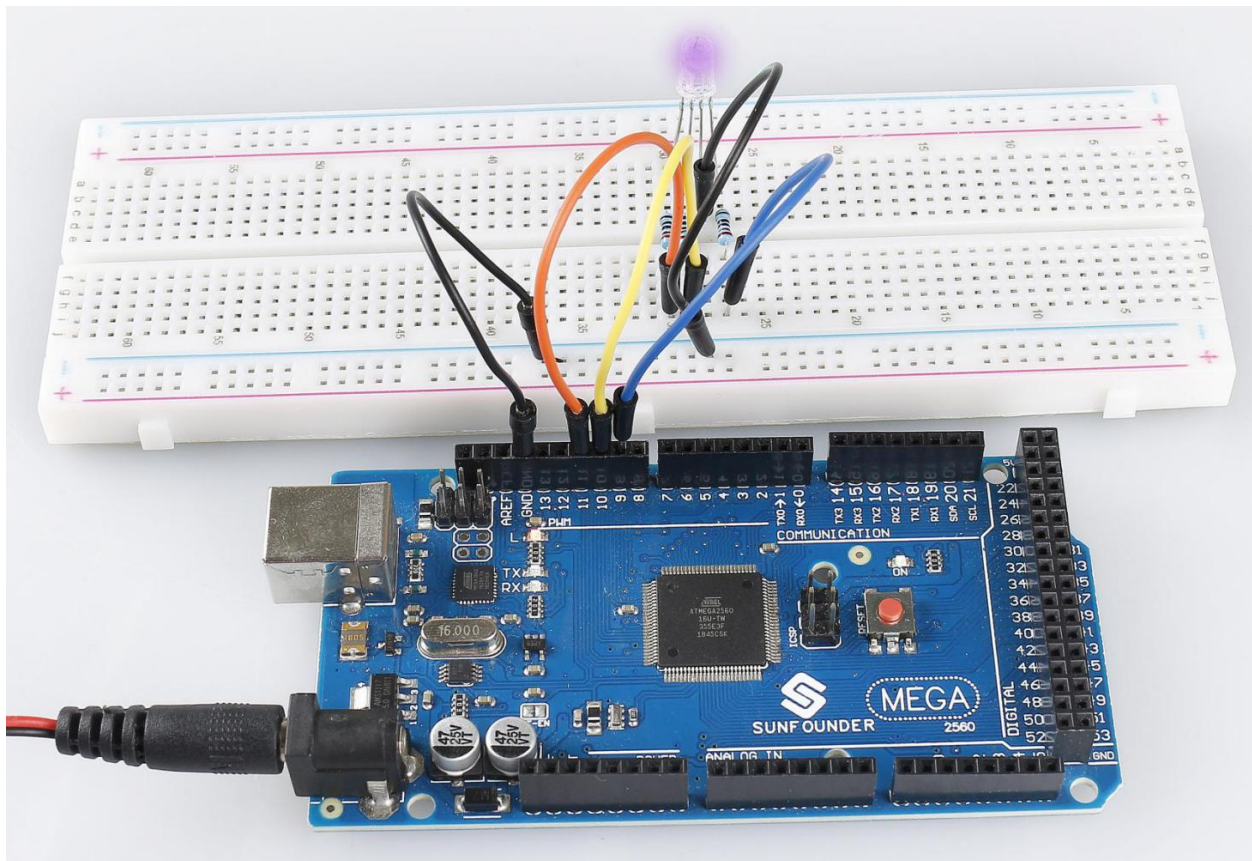
```
void loop() // run over and over again
{
```

(continues on next page)

(continued from previous page)

```
color(255, 0, 0); // turn the RGB LED red
delay(1000); // delay for 1 second
color(0,255, 0); // turn the RGB LED green
delay(1000); // delay for 1 second
color(0, 0, 255); // turn the RGB LED blue
delay(1000); // delay for 1 second
// ...
}
```

2.13.7 Phenomenon Picture

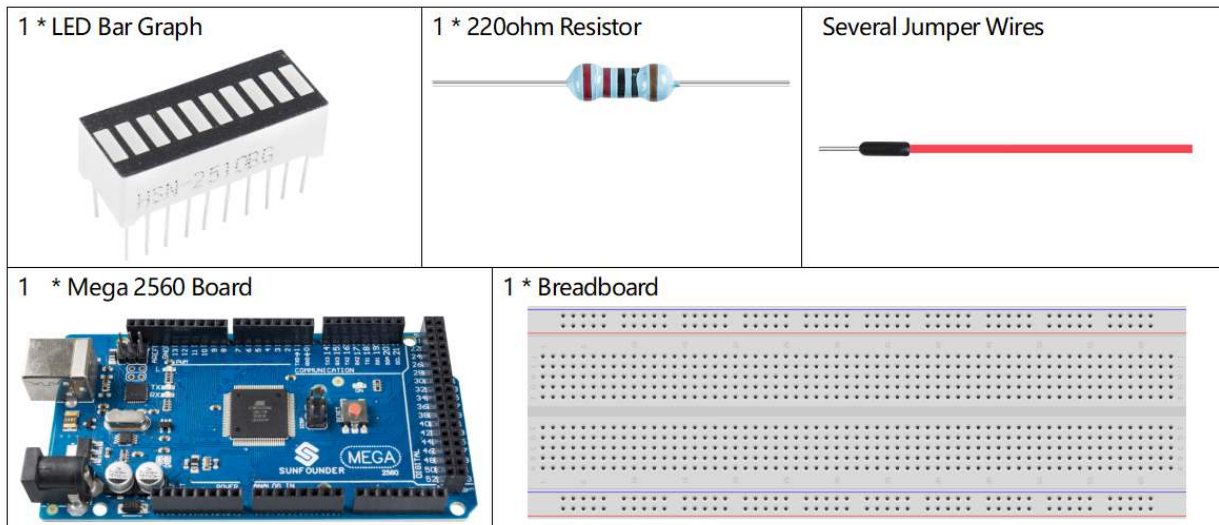


2.14 2.4 LED Bar Graph

2.14.1 Overview

In this lesson, you will learn something about LED Bar Graph. Generally, LED Bar Graph works as a battery level indicator, Audio equipment, industrial control panel. If we want, we can also find its other application.

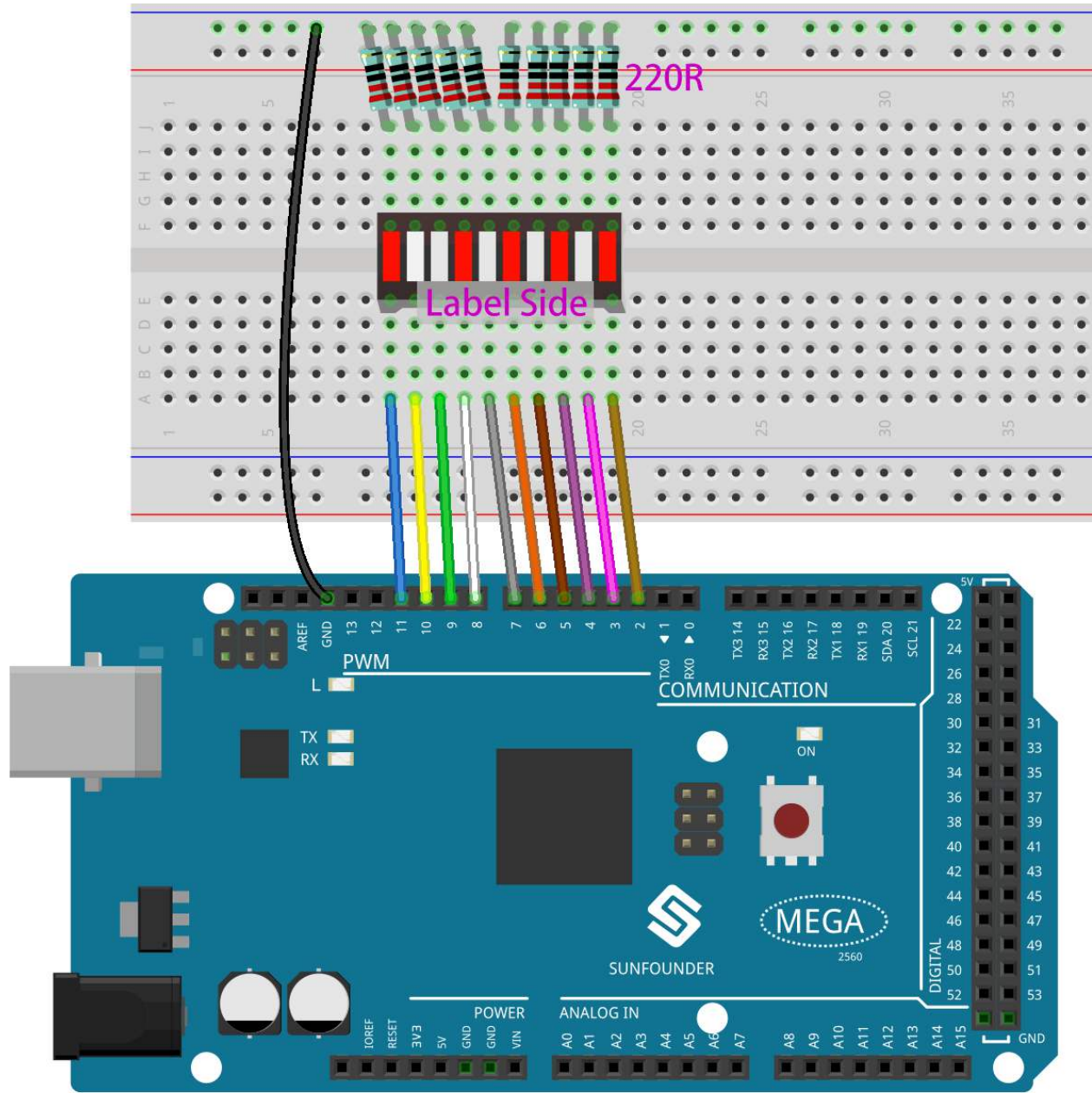
2.14.2 Components Required



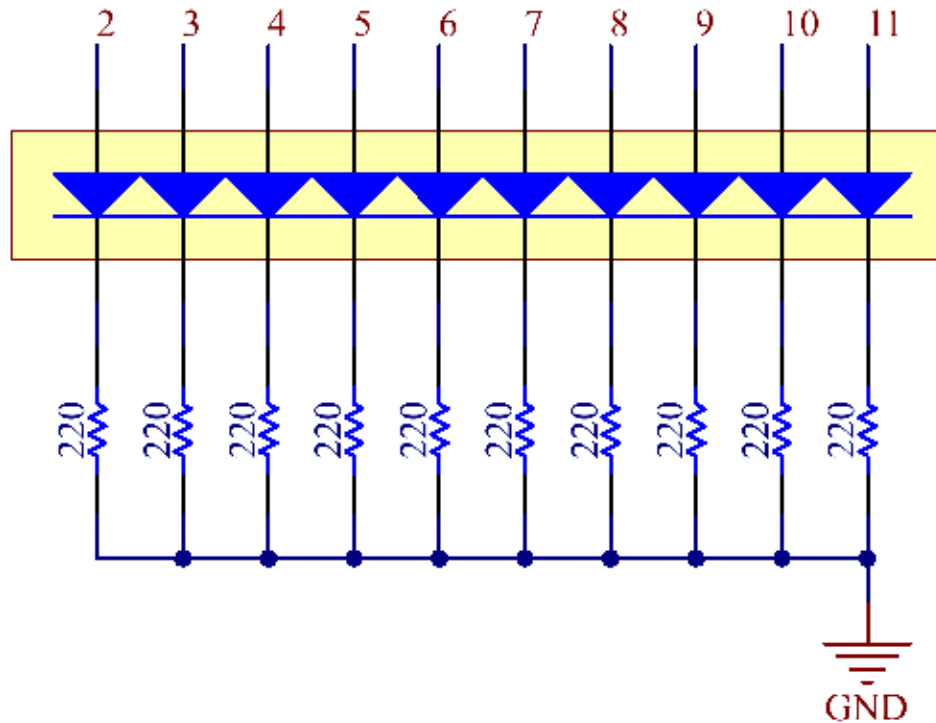
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *LED Bar Graph*
- *Resistor*

2.14.3 Fritzing Circuit

In this example, we use digital pins 2~11 to drive the LED Bar Graph. LED Bar Graph has ten separate LEDs inside and each LED has two pins. The left pins 1~10 of LED Bar Graph are connected with the digital pins 2~11 respectively; the right side pins 11~20 are separately extended to same side of these 220ohm resistors whose other sides are identically connected to GND.



2.14.4 Schematic Diagram



2.14.5 Code

Note:

- You can open the file `2.4_ledBarGraph.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.4_ledBarGraph` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

Uploaded the codes to the Mega2560 board, you can see that the LEDs on the LED Bar Graph flash in sequence.

2.14.6 Code Analysis

The codes in setup() use the for loop to initialize pins 2~11 to output mode in turn.

```
for(int i=2;i<=11;i++)
{
    pinMode(i,OUTPUT);
}
```

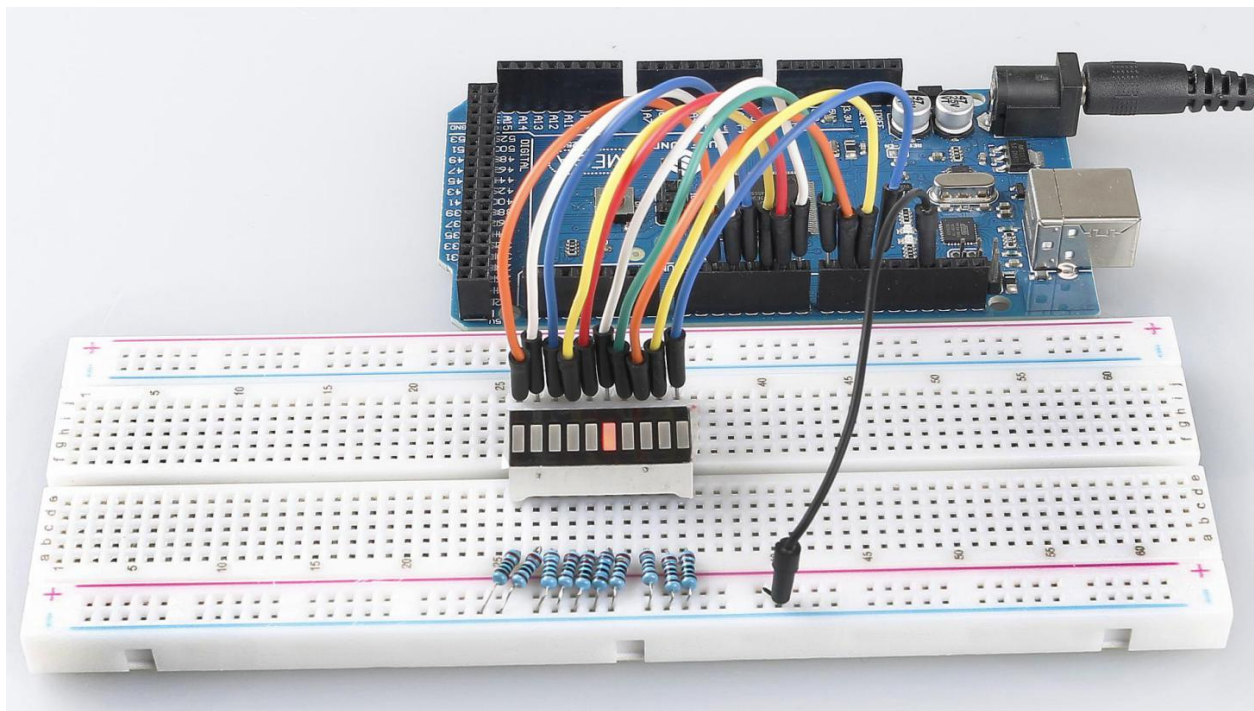
The for loop is used in loop() to make the LED flash(turn on 0.5s, then turn off 0.5s) in sequence.

```
for(int i=2;i<=11;i++)
{
    digitalWrite(i,HIGH);
    delay(500);
    digitalWrite(i,LOW);
    delay(500);
}
```

Refer to **Part 1-1.2 Digital Write** for more details about controlling the LED by using digital pins.

1.2 Digital Write

2.14.7 Phenomenon Picture





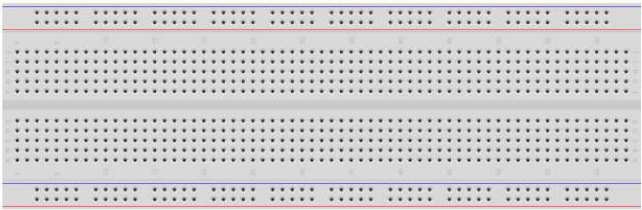


2.15 2.5 7-Segment Display

2.15.1 Overview

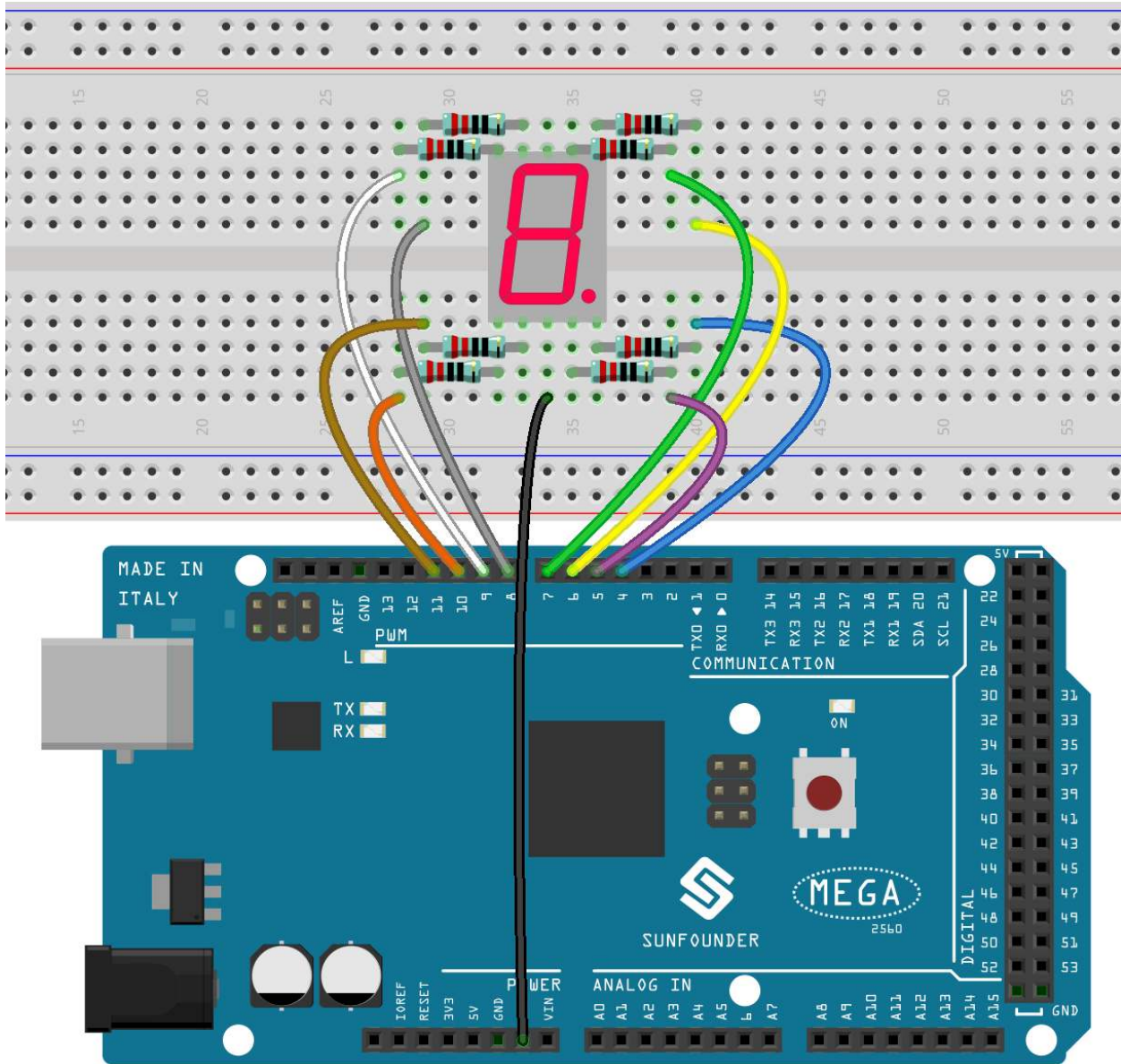
In this lesson, you will learn something about 7-Segment Display. 7-Segment Display has so many advantages that it is widely used in electrical equipments, especially in household appliances that display numerical information, such as display, air conditioner, water heater, refrigerator and so on. LEDs on the 7-Segment Display emit light by the input of different electrical signals to the different pins of it. The numerical information it can display includes time, date, temperature and so on.

2.15.2 Components Required

1 * 7-Segment Display 	8 * 220 ohm resistor 	Several Jumper Wires 
1 * Mega 2560 Board 	1 * Breadboard 	

- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *7-segment Display*
- *Resistor*

2.15.3 Fritzing Circuit

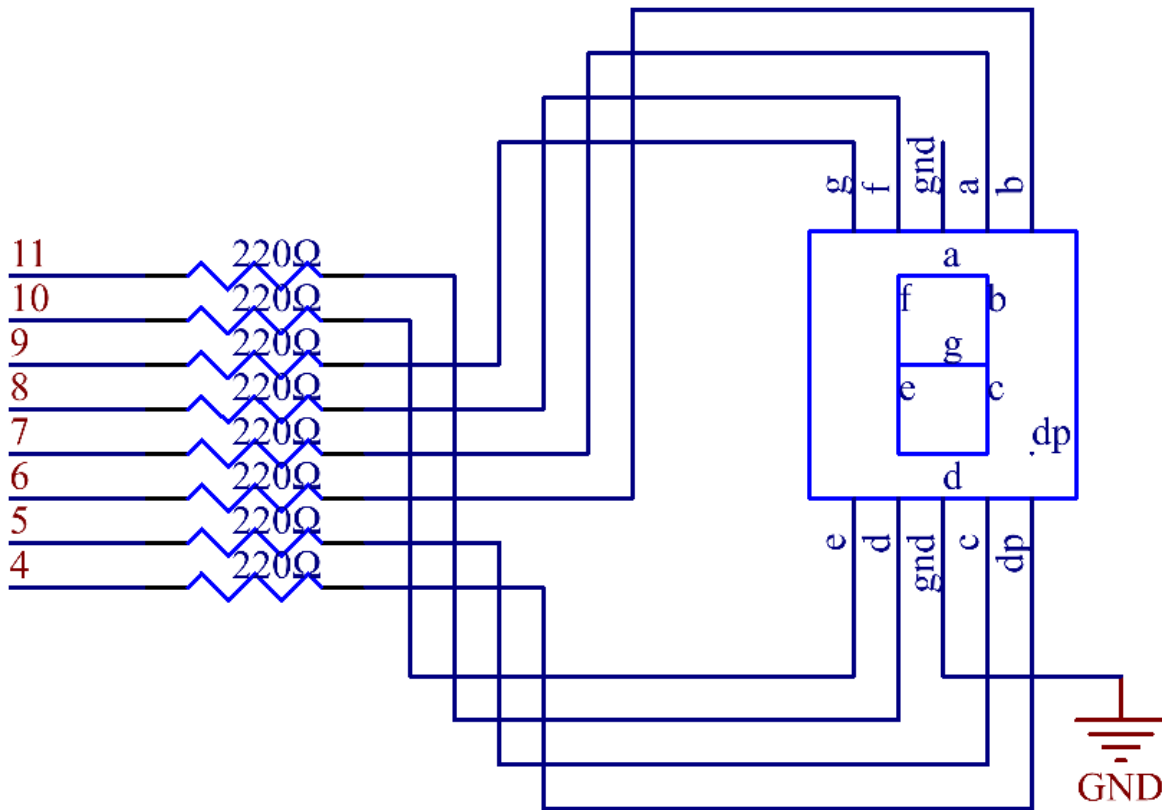


Connect each of pin a-g of the 7-Segment Display to one 220ohm current limiting resistor respectively and then to pin 4–11. GND connects to GND.

The wiring between the 7-segment display and the Mega2560 board as shown below :

7-Segment	Mega2560 Board
a	7
b	6
c	5
d	11
e	10
f	8
g	9
dp	4
" - "	GND

2.15.4 Schematic Diagram



2.15.5 Code

Note:

- You can open the file `2.5_7segment.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.5_7segment` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

Once upload the codes, you can see the 7-segment display displaying 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F in sequence.

2.15.6 Code Analysis

Take the pin numbers on 7-segment as names, and declare the pins on the Mega2560 board.

```
const int a=7; //a of 7-segment attach to digital pin 7
const int b=6; //b of 7-segment attach to digital pin 6
const int c=5; //c of 7-segment attach to digital pin 5
const int d=11; //d of 7-segment attach to digital pin 11
const int e=10; //e of 7-segment attach to digital pin 10
const int f=8; //f of 7-segment attach to digital pin 8
const int g=9; //g of 7-segment attach to digital pin 9
const int dp=4; //dp of 7-segment attach to digital pin 4
```

Install a series of subfunctions to package the level state at each block during the number display of the 7-segment. For example, when the character 2 is displayed, the block F and the block c are turn off; the other blocks are lit up.



First we need to know how it looks like when display the numeral **2** on the 7-Segment display. It's actually the segments a, b, d, e and g are power on, which generates the display of **2**. In programming, pins connected to these segments are set High level when c and f are Low level. Here we use a *for()* statement to set these pins as High level respectively (the braces after *for()* are deleted as there is only one line). Connect pin dp to pin 4; it's already defined as LOW in *setup()*.

After running this part, the 7-segment will display **2**. Similarly, the display of other characters are the same. Since the letters b and d in upper case, namely **B** and **D**, would look the same with **8** and **0** on the display, they are displayed in lower case instead.

```
...
void digital_2(void) //display 2 to the 7-segment
{
digitalWrite(b,HIGH);
digitalWrite(a,HIGH);
for(int j = 9; j <= 11; j++)
digitalWrite(j,LOW);
digitalWrite(c,LOW);
digitalWrite(f,LOW);
```

(continues on next page)

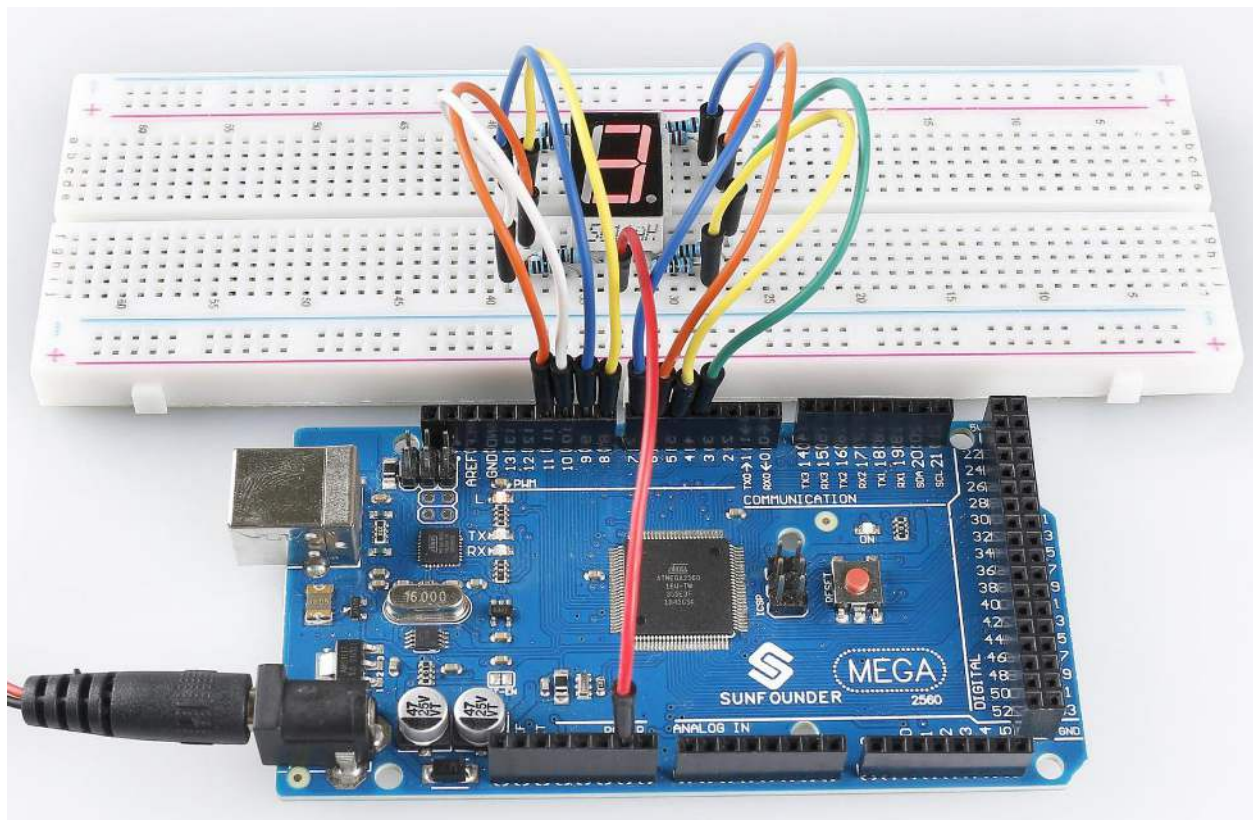
(continued from previous page)

```
}  
...
```

In loop(), call the function that displays the number.

```
void loop()  
{  
digital_1();//diaplay 1 to the 7-segment  
delay(1000);//wait for a second  
digital_2();//diaplay 2 to the 7-segment  
delay(1000); //wait for a second  
digital_3();//diaplay 3 to the 7-segment  
//...  
}
```

2.15.7 Phenomenon Picture






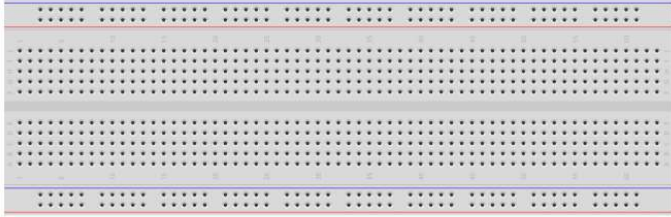


2.16 2.6 74HC595

2.16.1 Overview

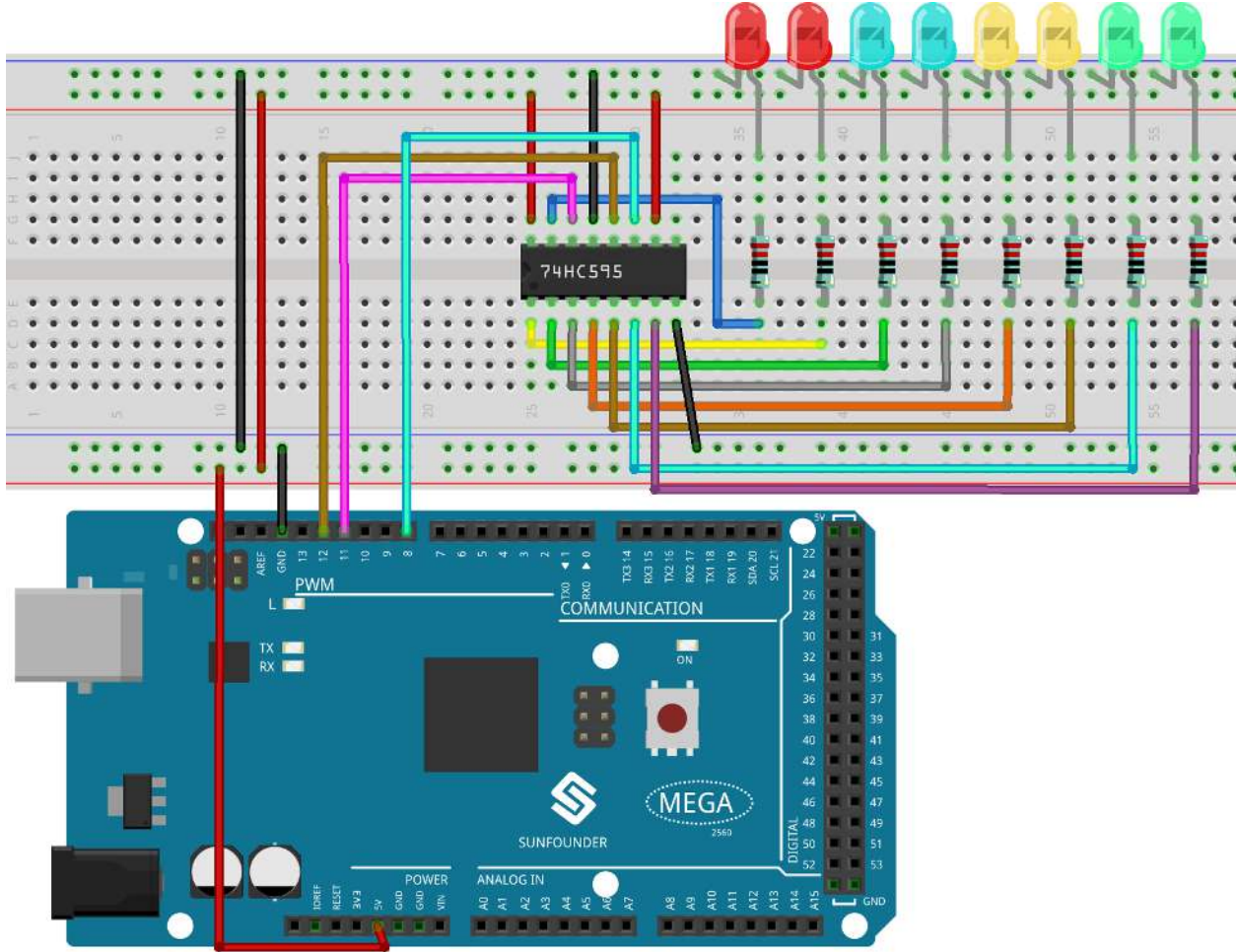
In this lesson, you will learn how to use 74HC595. 74HC595 consists of an 8bit shift register and a storage register with threestate parallel outputs. It converts serial input into parallel output so you can save IO ports of an MCU.

2.16.2 Components Required

<p>1 * 74HC595</p> 	<p>8 * LED</p> 	<p>8 * 220 ohm resistor</p> 	<p>Several Jumper Wires</p> 
<p>1 * Mega 2560 Board</p> 	<p>1 * Breadboard</p> 		

- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *LED*
- *Resistor*
- *74HC595*

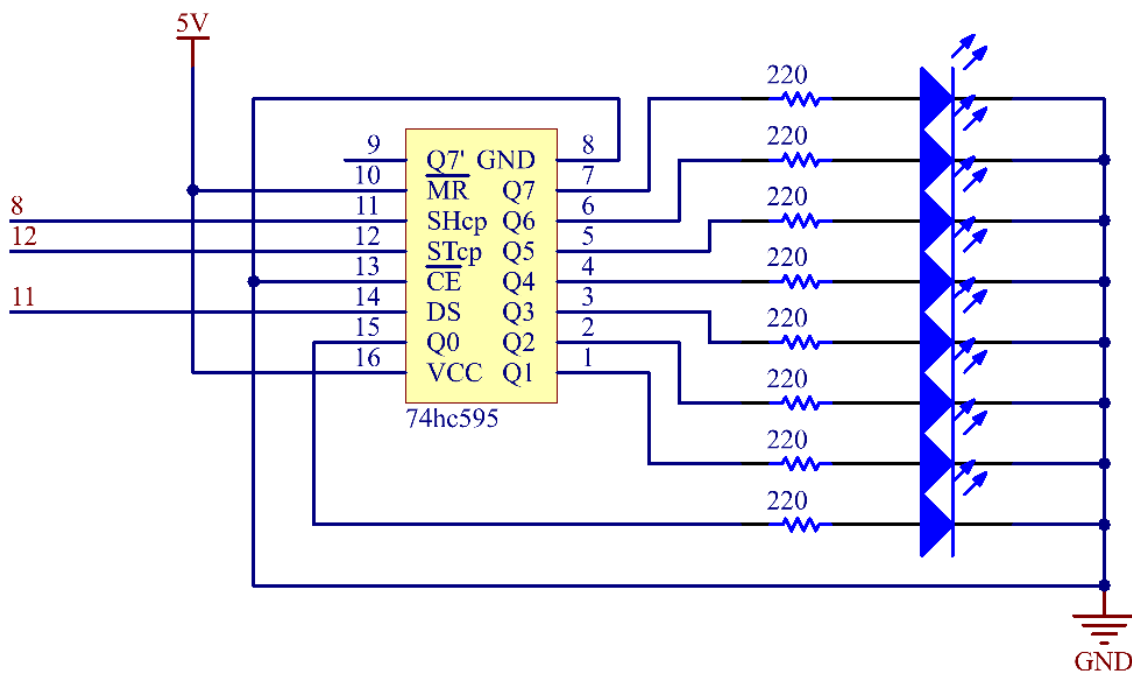
2.16.3 Fritzing Circuit



In this example, we use 74HC595 to control LED. Give each data output pin Q0-Q7 a 220 ohm resistor then connect them to LED. The wiring diagram is as follows:

74HC595	Mega Board
MR(10)	5V
SHcp(11)	8
STcp(12)	12
OE(13)	GND
DS(14)	11
VCC(16)	5V
GND(8)	GND

2.16.4 Schematic Diagram



2.16.5 Code

Note:

- You can open the file `2_6_74HC595.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2_6_74HC595` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Or click **Open Code** to open it in [Web Editor](#).
 - Then *Upload the Code* to the board.
-

When you finish uploading the codes to the Mega2560 board, you can see the LEDs turning on one after another.

2.16.6 Code Analysis

Declare an array, store several 8 bit binary numbers that are used to change the working state of the eight LEDs controlled by 74HC595.

```
int dataArray[] = {B00000000, B00000001, B00000011, B00000111, B00001111, B00011111, B00111111, B01111111, B11111111};
```

Set STcp to low level first and then high level. It will generate a rising edge pulse of STcp.

```
digitalWrite(STcp, LOW);
```

`shiftOut()` is used to shift out a byte of data one bit at a time, which means to shift a byte of data in `dataArray[num]` to the shifting register with the DS pin. `MSBFIRST` means to move from high bits.

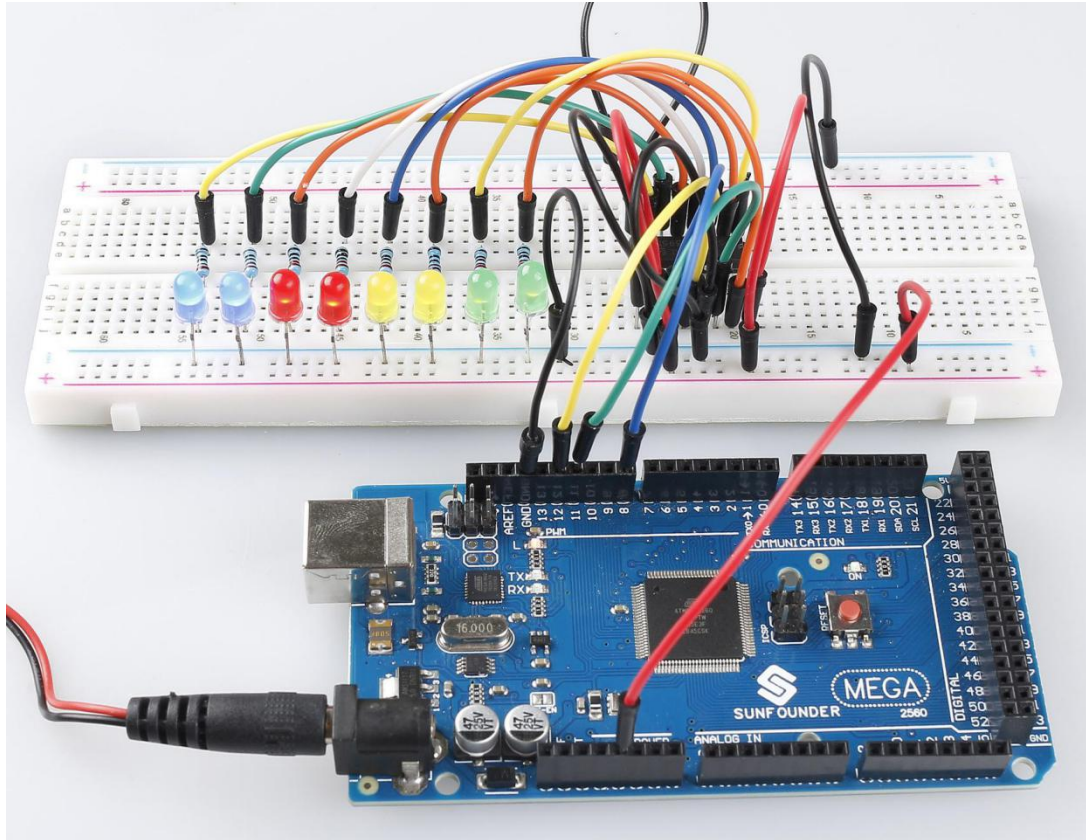
```
shiftOut(DS, SHcp, MSBFIRST, dataArray[num]);
```

After `digitalWrite(STcp, HIGH)` is run, the STcp will be at the rising edge. At this time, the data in the shift register will be moved to the memory register.

```
digitalWrite(STcp, HIGH);
```

A byte of data will be transferred into the memory register after 8 times. Then the data of memory register are output to the bus (Q0-Q7). For example, `shiftoutB00000001` will light up the LED controlled by Q0 and turn off the LED controlled by Q1~Q7.

2.16.7 Phenomenon Picture





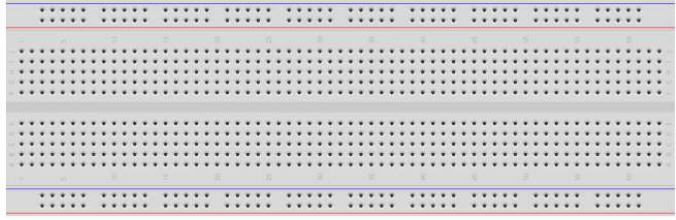


2.17 2.7 4-Digital 7-Segment Display

2.17.1 Overview

In this lesson, you will learn about the 4-Digital 7-Segment Display. It consists of four 7-segment displays working together so as to display 4 digit numbers.

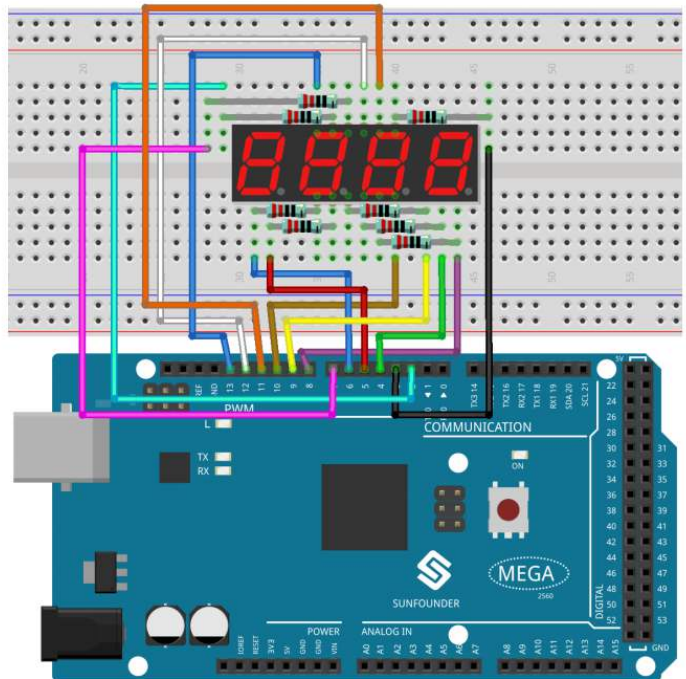
2.17.2 Components Required

<p>1 * 4-Digital 7-Segment Display</p> 	<p>8 * 220 ohm resistor</p> 	<p>Several Jumper Wires</p> 
<p>1 * Mega 2560 Board</p> 	<p>1 * Breadboard</p> 	

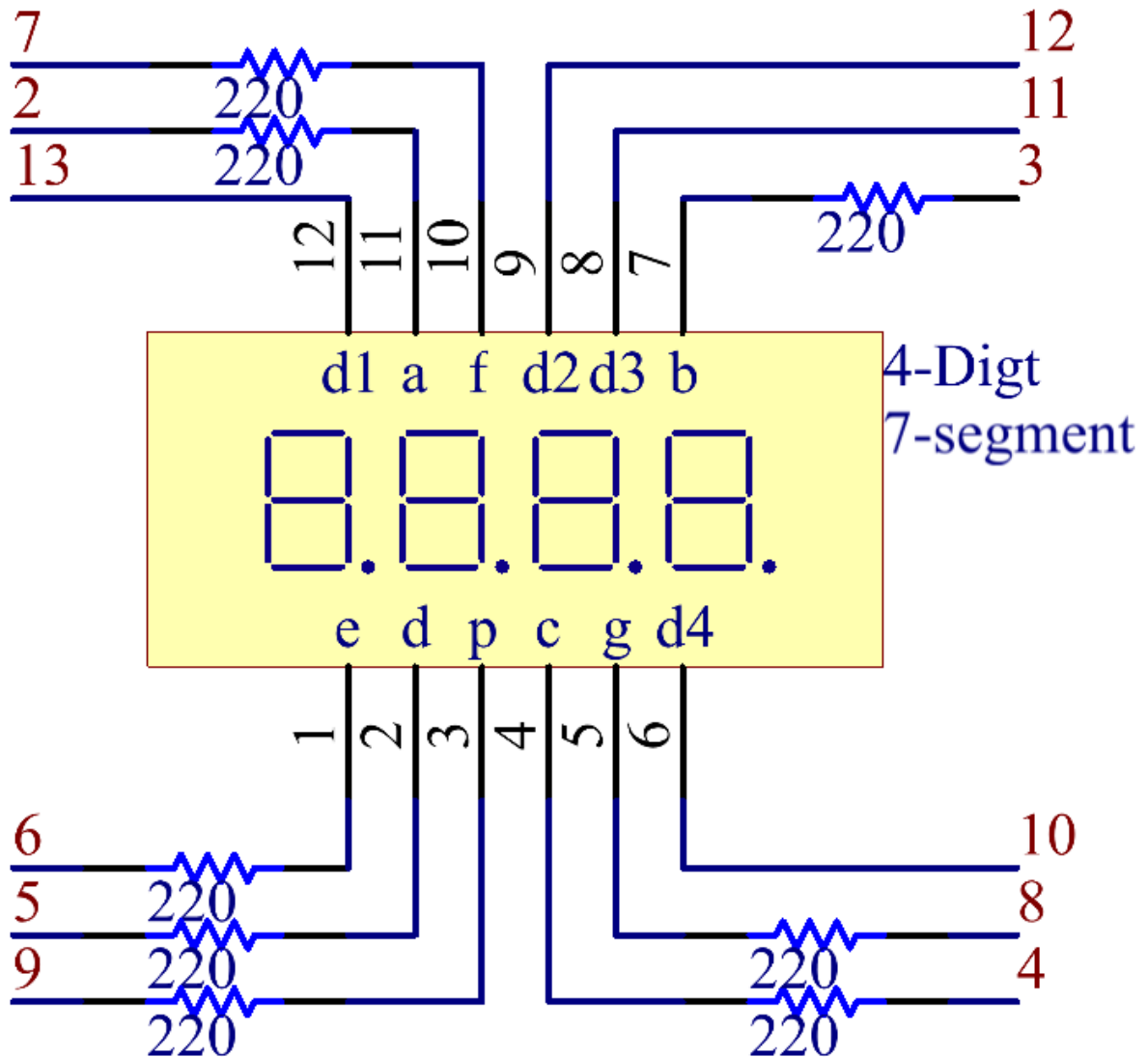
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Resistor*
- *4-Digit 7-Segment Display*

2.17.3 Fritzing Circuit

4-Digit Display	Mega 2560 Board
A(11)	2
B(7)	3
C(4)	4
D(2)	5
E(1)	6
F(10)	7
G(5)	8
dp(3)	9
D1(12)	13
D2(9)	12
D3(8)	11
D4(6)	10



2.17.4 Schematic Diagram

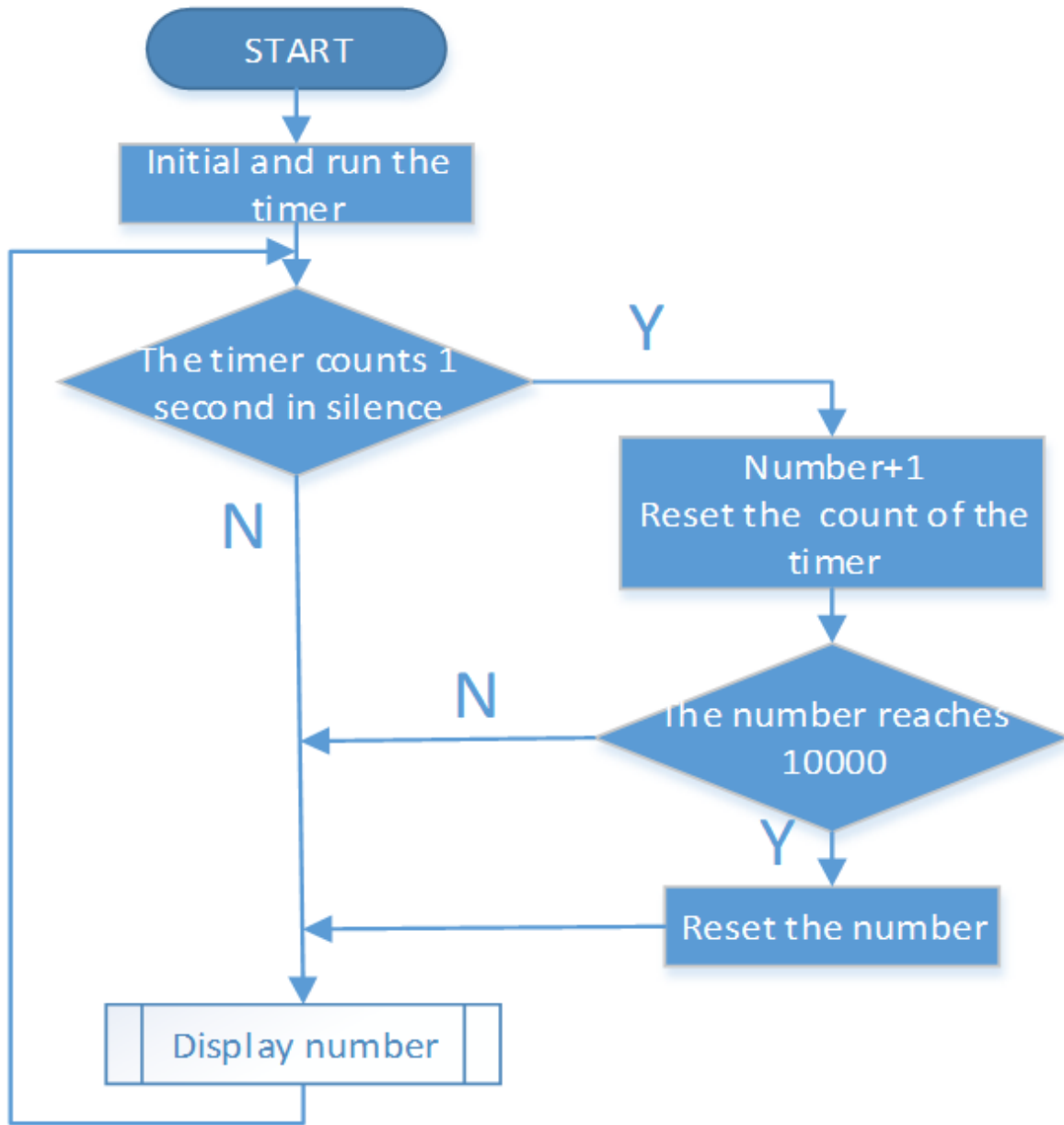


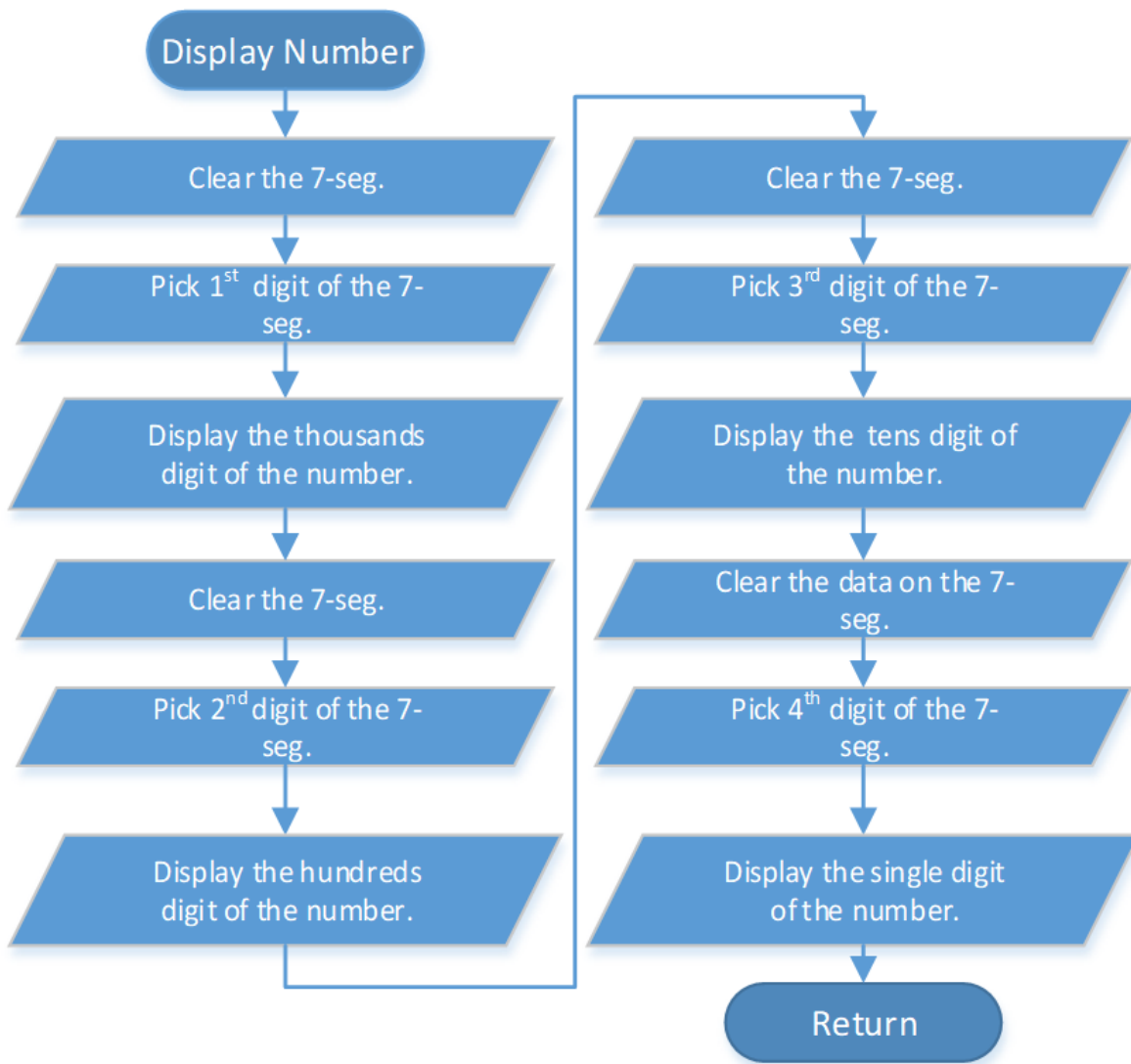
2.17.5 Code

Note:

- You can open the file `2.8_ledMatrix.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.8_ledMatrix` directly.
- Or copy this code into Arduino IDE 1/2.
- Then *Upload the Code* to the board.
- Please make sure you have added the library called `TimerOne`, detailed tutorials refer to *Add Libraries*.

2.17.6 Code Analysis





There are two points needing your attention:

1. Because every segment display works independently in the 4-Digital 7-Segment Display, the principle of visual persistence is applied to quickly display every 7 segment character in turn to form a continuous character string.

Refer to *2.5 7-Segment Display* to check the details of the number display of the 4-Digital 7-Segment Display.

2. In this example, a library `TimerOne.h` is used to realize the function of counting.

```
#include "TimerOne.h"
```

Library Functions

```
void initialize(long microseconds=1000000)
```

You must call this method first to use any of the other methods. You can optionally specify the timer's period here (in microseconds), by default it is set at 1 second.

Note: This breaks analogWrite() for digital pins 9 and 10 on Arduino.

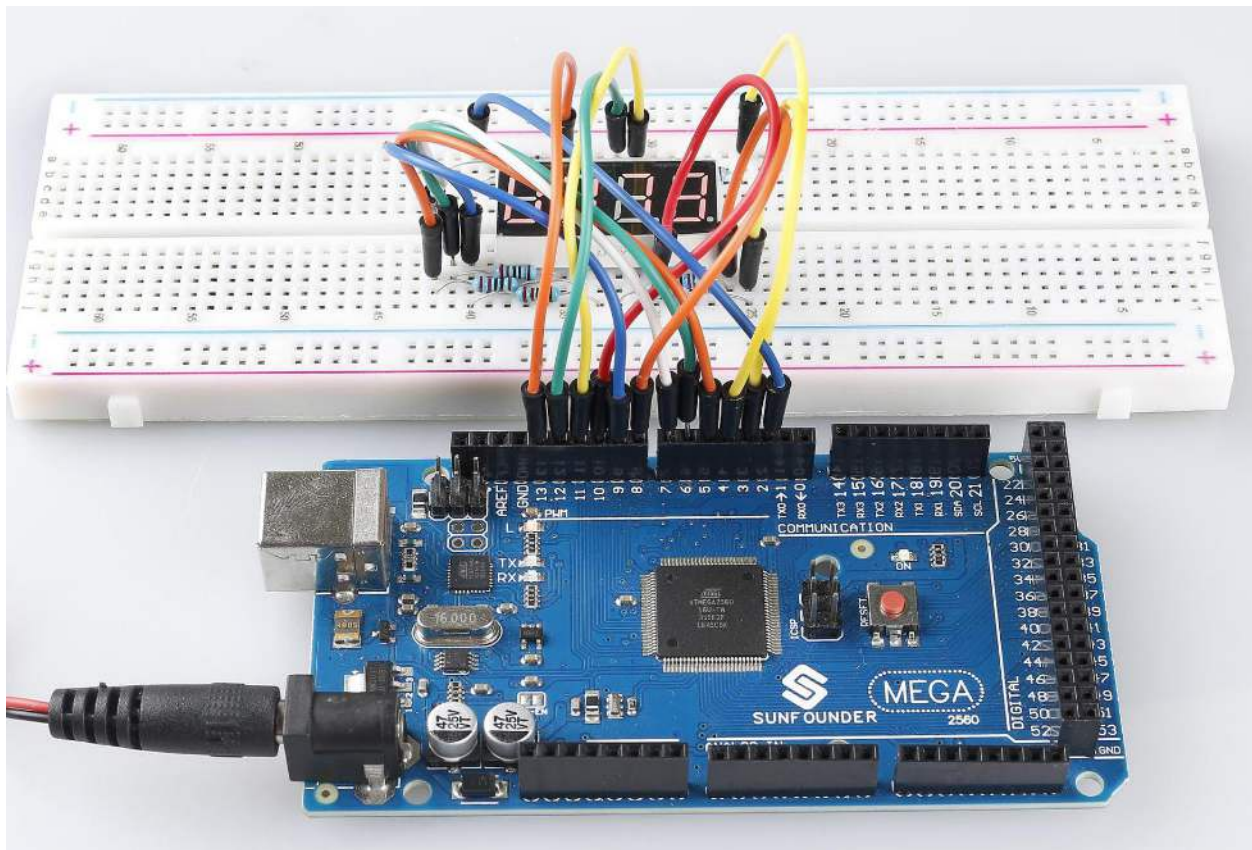
```
void attachInterrupt(void (*isr)(), long microseconds=-1);
```

Calls a function at the specified interval in microseconds. Be careful about trying to execute too complicated of an interrupt at too high of a frequency, or the CPU may never enter the main loop and your program will 'lock up'. Note that you can optionally set the period with this function if you include a value in microseconds as the last parameter when you call it.

```
void detachInterrupte();
```

Disables the attached interrupt.

2.17.7 Phenomenon Picture

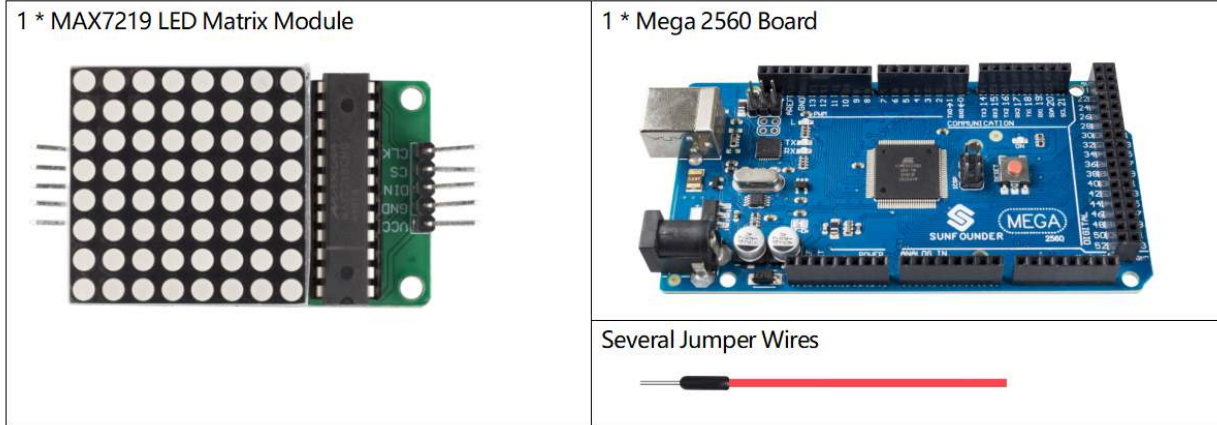


2.18 2.8 LED Matrix Module

2.18.1 Overview

In this lesson, you will learn about LED Matrix Module. LED Matrix Module uses the MAX7219 driver to drive the 8 x 8 LED Matrix.

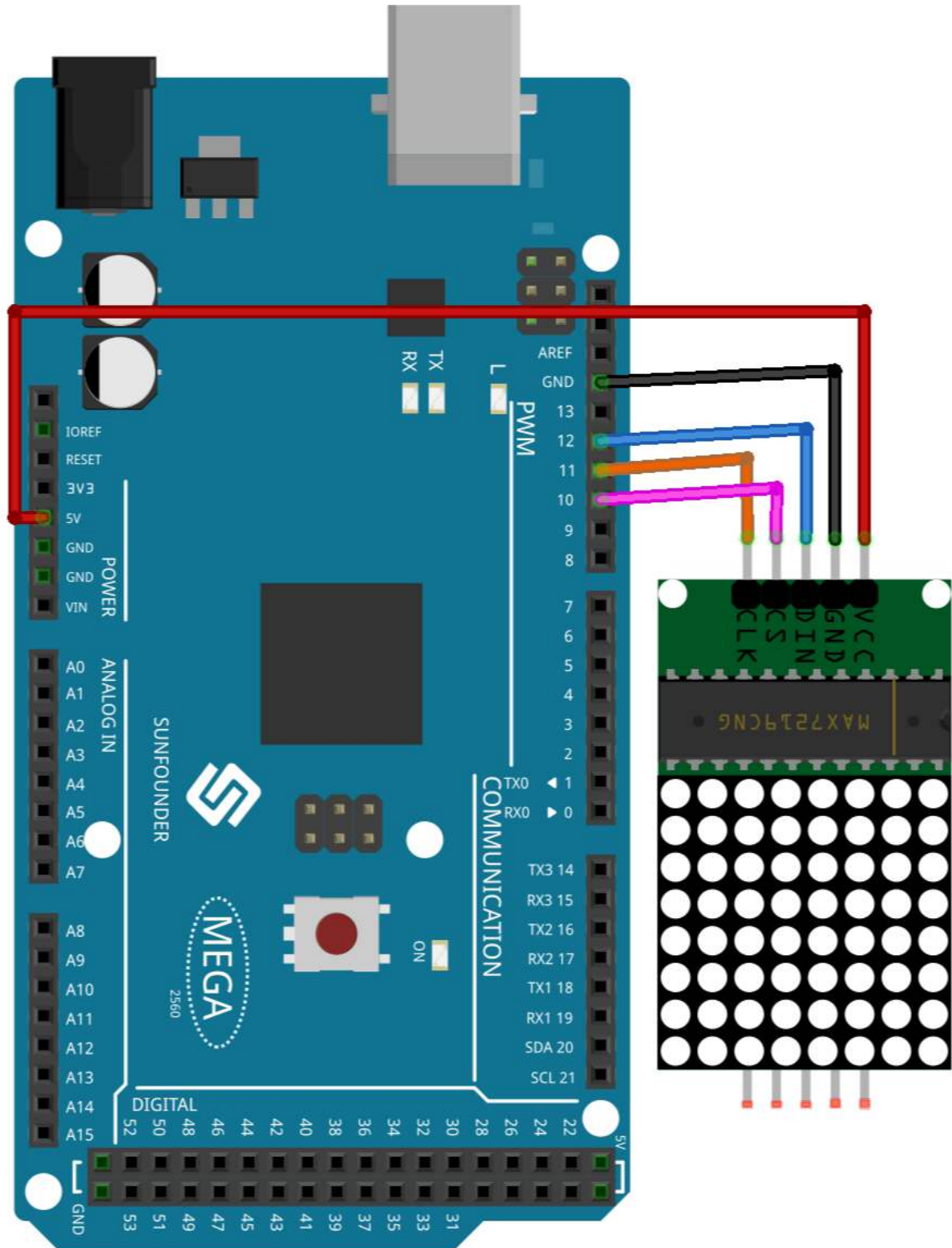
2.18.2 Components Required



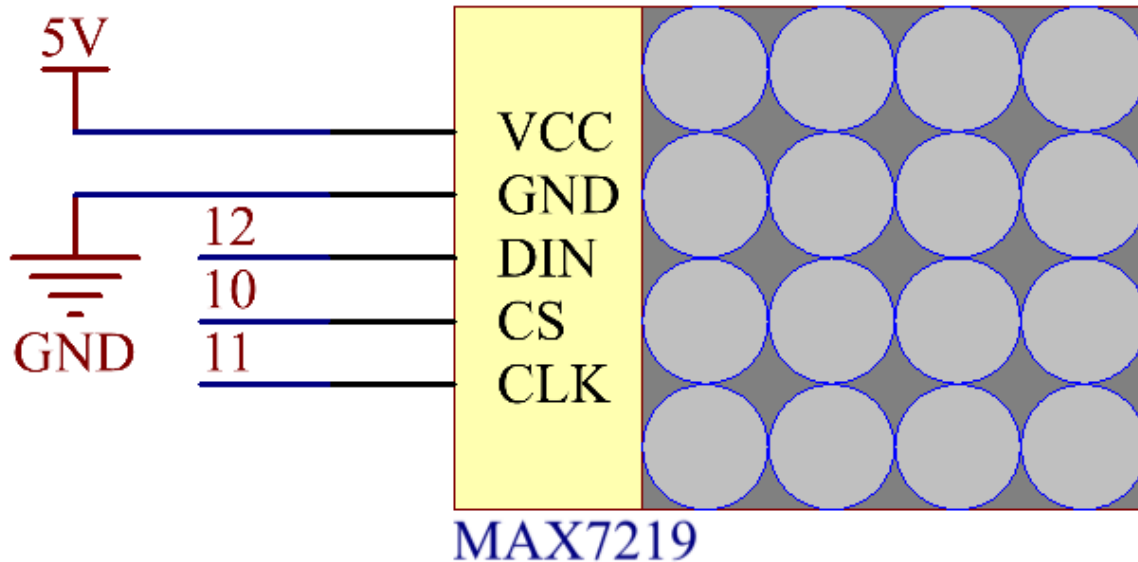
- *SunFounder Mega Board*
- *Jumper Wires*
- *LED Matrix Module*

2.18.3 Fritzing Circuit

In this example, we get the VCC pin of MAX7219 connected to 5V, GND to ground, DIN to digital pin 12, CS to digital pin 10, CLK to digital pin 11.



2.18.4 Schematic Diagram



2.18.5 Code

Note:

- You can open the file `2.8_ledMatrix.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.8_ledMatrix` directly.
- Or copy this code into Arduino IDE 1/2.
- Then *Upload the Code* to the board.
- Please make sure you have added the library called `LedControl`, detailed tutorials refer to *Add Libraries*.

After the codes are uploaded, you can see that the LEDs turn on in the sequence of a column, a row or a dot or there is a picture appearing on the LED matrix.

2.18.6 Code Analysis

By calling the library `LedControl.h`, you can easily use the LED matrix.

```
#include "LedControl.h"
```

Library Functions

```
LedControl(int dataPin,int clockPin,int csPin,int numDevices)
```

Create an instance of type `LedControl` through which we talk to the MAX7219 devices. The initialization of an `LedControl` takes 4 arguments.

- `dataPin`, `clockPin`, `csPin`: The first 3 arguments are the pin-numbers on the Arduino that are connected to the MAX7219. You are free to choose any of the digital IO-pins on the arduino, but since some of the pins are also used for serial communication or have a led attached to them its best to avoid pin 0,1 and 13.
- `numDevices`: The fourth argument is the number of cascaded MAX7219 devices you're using with this LedControl. The library can address up to 8 devices from a single LedControl-variable.

```
void shutdown(int addr, bool b)
```

- `addr`: The address of the display to control.
- `b`: If true the device goes into power-down mode. If false device goes into normal operation.

```
void setIntensity(int addr, int intensity)
```

The method lets you control brightness in 16 discrete steps. Larger values make the display brighter up to the maximum of 15.

- `addr`: The address of the display to control.
- `intensity`: the brightness of the display. Only values between 0(darkest) and 15(brightest) are valid.

```
void clearDisplay(int addr)
```

All LEDs off after this one.

- `addr`: The address of the display to control.

```
void setLed(int addr, int row, int col, boolean state)
```

Set the status of a single Led.

- `addr`: The address of the display to control.
- `row`: The row of the Led (0..7).
- `col`: The column of the Led (0..7).
- `state`: If true the led is switched on, if false it is switched off.

```
void setRow(int addr, int row, byte value)
```

Set all 8 LEDs in a row to a new state.

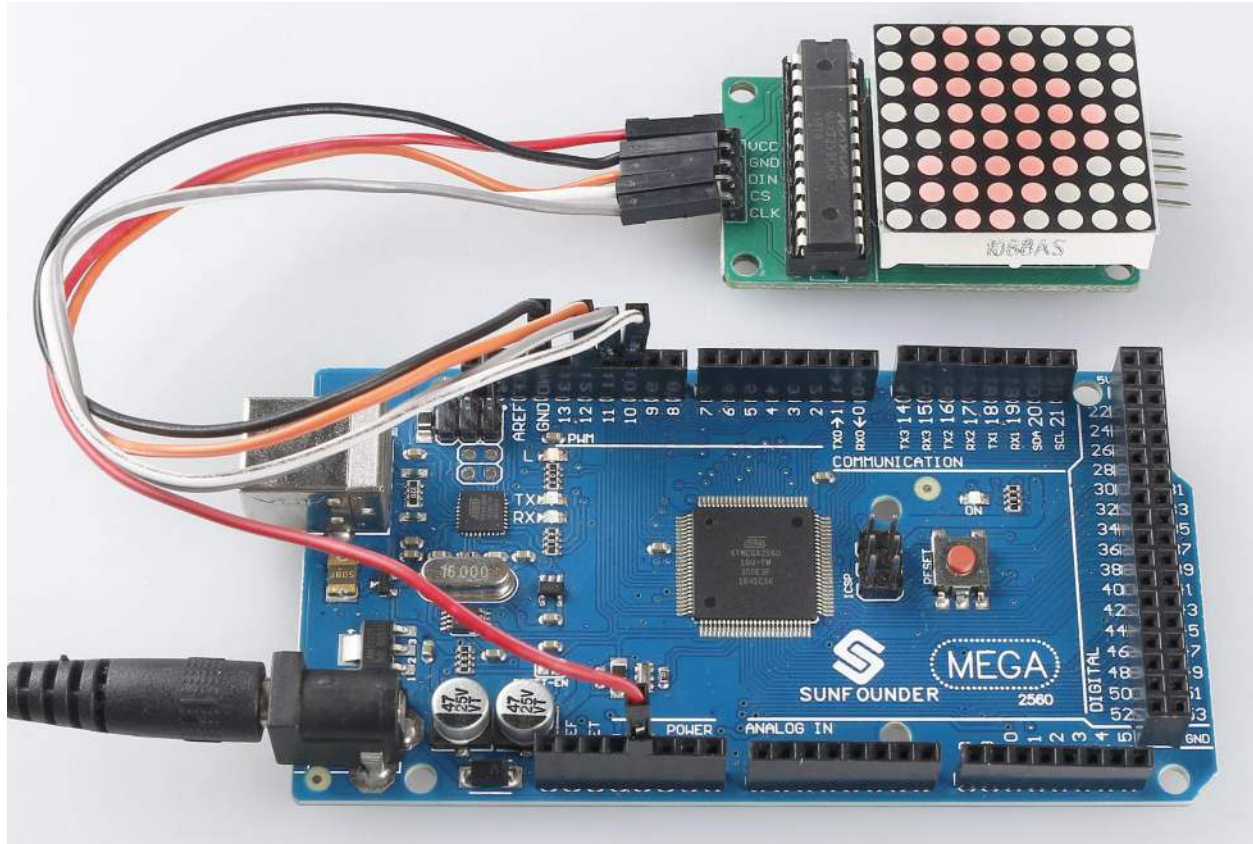
- `addr`: The address of the display to control.
- `row`: Row which is to be set (0..7).
- `value`: Each bit set to 1 will light up the corresponding Led.(e.g. B01000000 will light up the 2nd).

```
void setColumn(int addr, int col, byte value)
```

Set all 8 LEDs in a row to a new state.

- `addr`: The address of the display to control.
- `col`: Column which is to be set (0..7).
- `value`: Each bit set to 1 will light up the corresponding Led.(e.g. B01000000 will light up the 2nd).

2.18.7 Phenomenon Picture

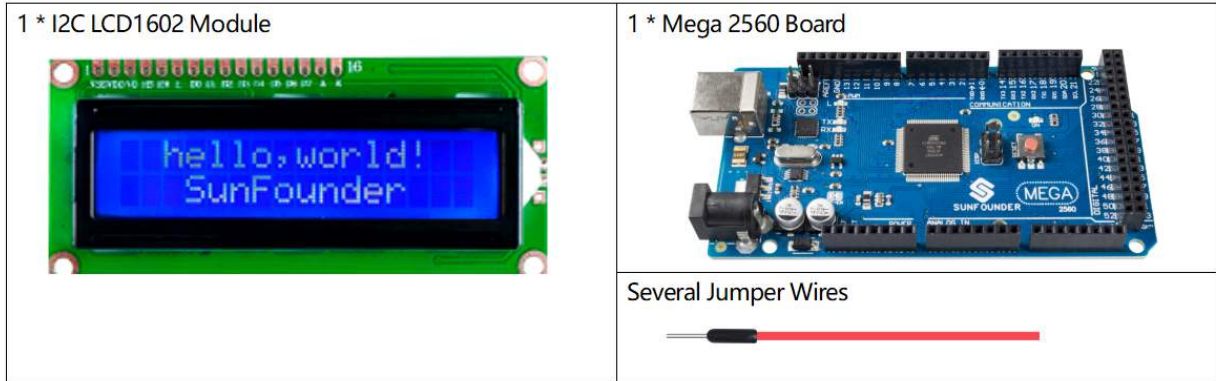


2.19 2.9 I2C LCD1602 Module

2.19.1 Overview

In this lesson, you will learn about LCD1602. LCD1602, or 1602 character-type liquid crystal display, a kind of dot matrix module to show letters, numbers, characters and so on.

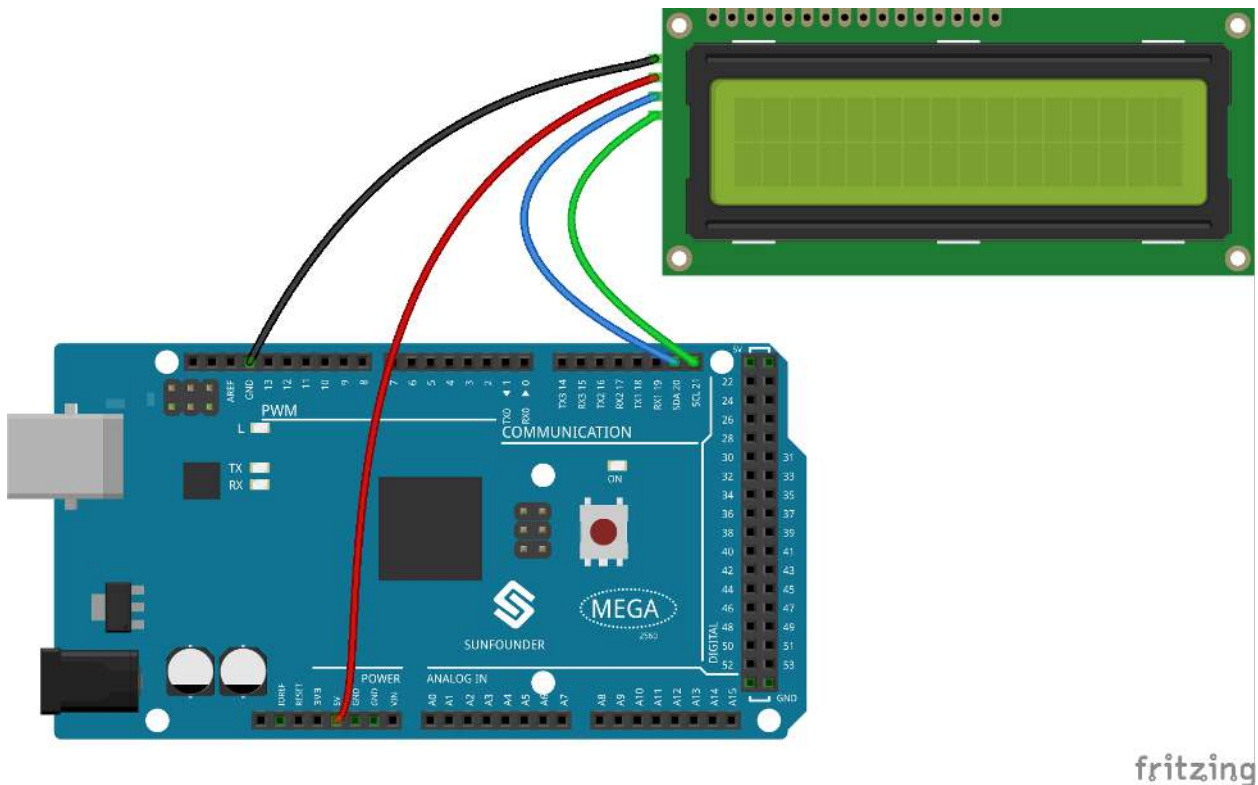
2.19.2 Components Required



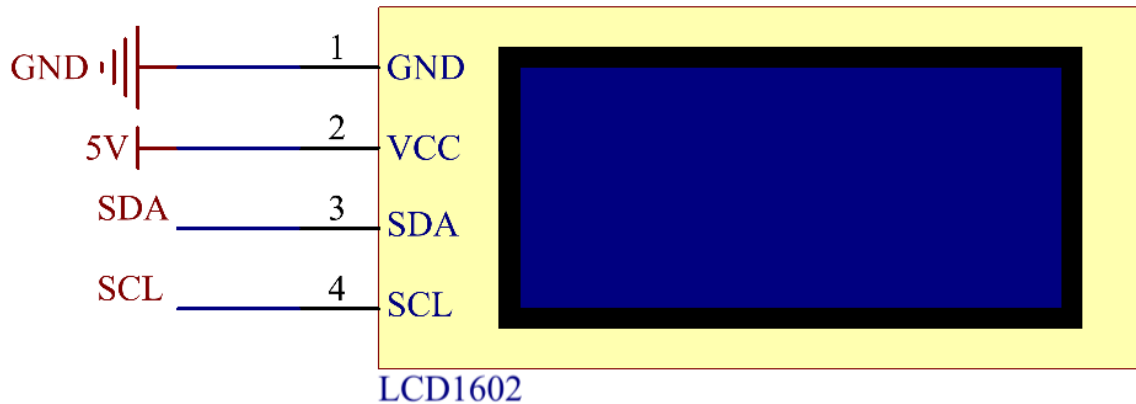
- *SunFounder Mega Board*
- *Jumper Wires*
- *I2C LCD1602*

2.19.3 Fritzing Circuit

In this example, we will get the first pin GND of LCD1602 connected to GND, the second pin VCC to 5V, the third pin SDA to the pin SDA 20 and the forth pin SCL to the pin SCL 21.



2.19.4 Schematic Diagram



Note: The SDA and SCL of the Mega2560 board are the pins 20 and 21.

2.19.5 Code

Note:

- You can open the file `2.9_i2clcd1602.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.9_i2clcd1602` directly.
- Or copy this code into Arduino IDE 1/2.
- Then *Upload the Code* to the board.
- Please make sure you have added the library called `LiquidCrystal_I2C`, detailed tutorials refer to [Add Libraries](#).

Upload the codes to the Mega2560 board, the content that you input in the serial monitor will be printed on the LCD.

Note: About the ASCII code and the character input in the serial monitor, please refer to [1.8 Serial Read](#).

2.19.6 Code Analysis

By calling the library `LiquidCrystal_I2C.h`, you can easily drive the LCD.

```
#include "LiquidCrystal_I2C.h"
```

Library Functions

```
LiquidCrystal_I2C(uint8_t lcd_Addr, uint8_t lcd_cols, uint8_t lcd_rows)
```

Creates a new instance of the `LiquidCrystal_I2C` class that represents a particular LCD attached to your Arduino board.

- `lcd_Addr`: The address of the LCD defaults to `0x27`.

- `lcd_cols`: The LCD1602 has 16 columns.
- `lcd_rows`: The LCD1602 has 2 rows.

```
void init ()
```

Initialize the lcd.

```
void backlight ()
```

Turn the (optional) backlight on.

```
void nobacklight ()
```

Turn the (optional) backlight off.

```
void display ()
```

Turn the LCD display on.

```
void nodisplay ()
```

Turn the LCD display off quickly.

```
void clear ()
```

Clear display, set cursor position to zero.

```
void setCursor (uint8_t col, uint8_t row)
```

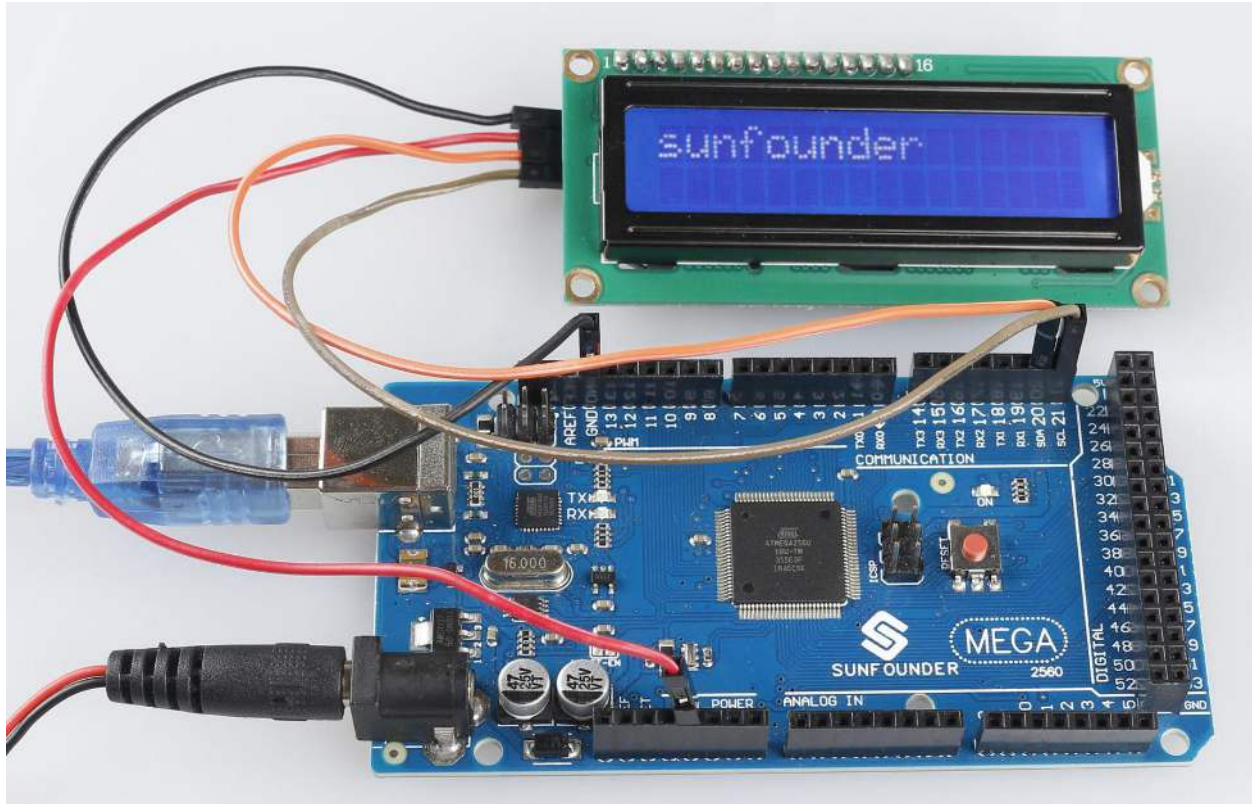
Set the cursor position to col,row.

```
void print (data, BASE)
```

Prints text to the LCD.

- `data`: The data to print (char, byte, int, long, or string).
- `BASE` (optional): The base in which to print numbers: BIN for binary (base 2), DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

2.19.7 Phenomenon Picture



2.20 2.10 Active Buzzer

2.20.1 Overview

In this lesson, you will get to know about active buzzer. As a type of electronic buzzer with an integrated structure, active buzzer is supplied by DC power, widely used in computer, alarm, electronic toy, telephone, timer and other electronic products or voice devices.

2.20.2 Components Required

1 * Active Buzzer



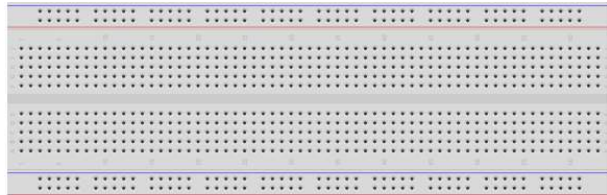
Several Jumper Wires



1 * Mega 2560 Board

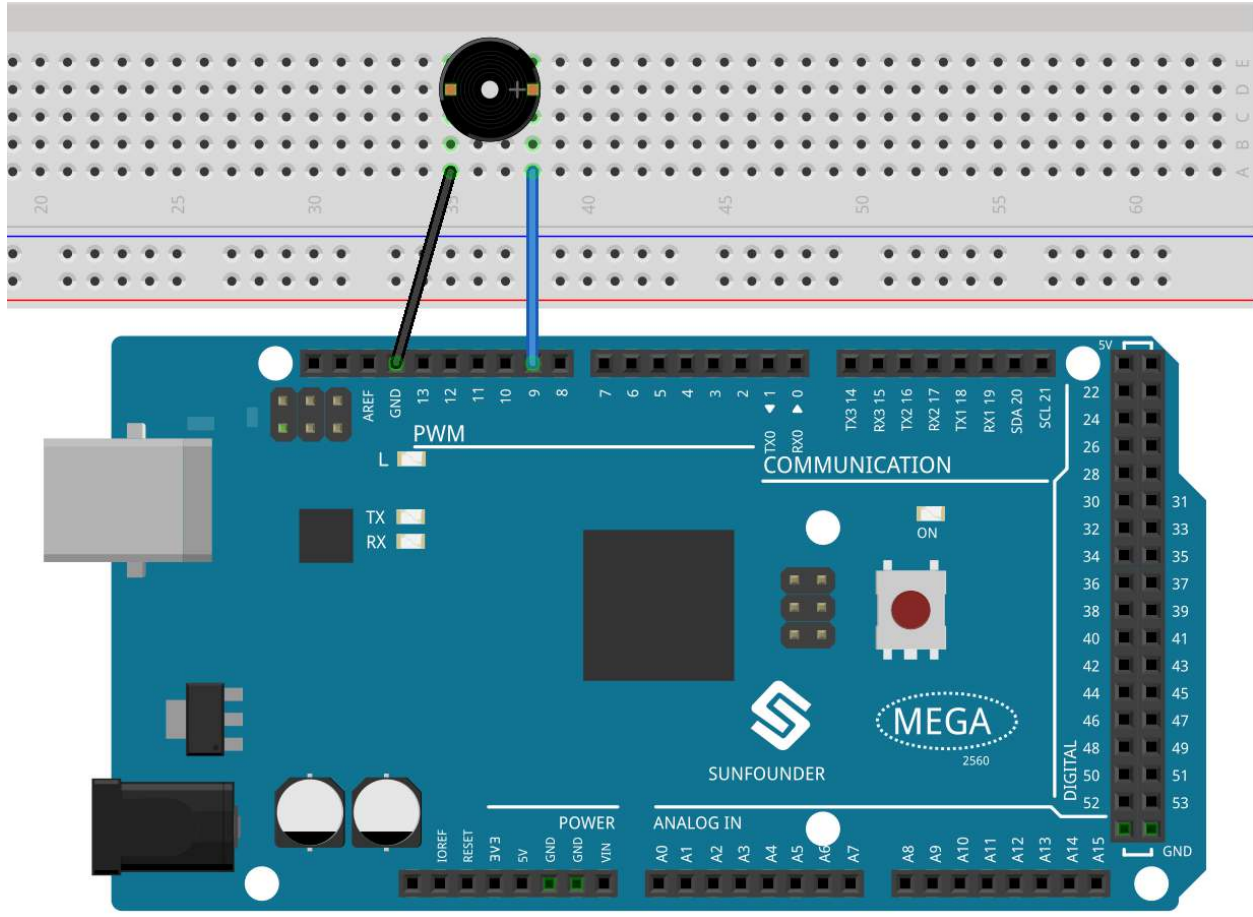


1 * Breadboard



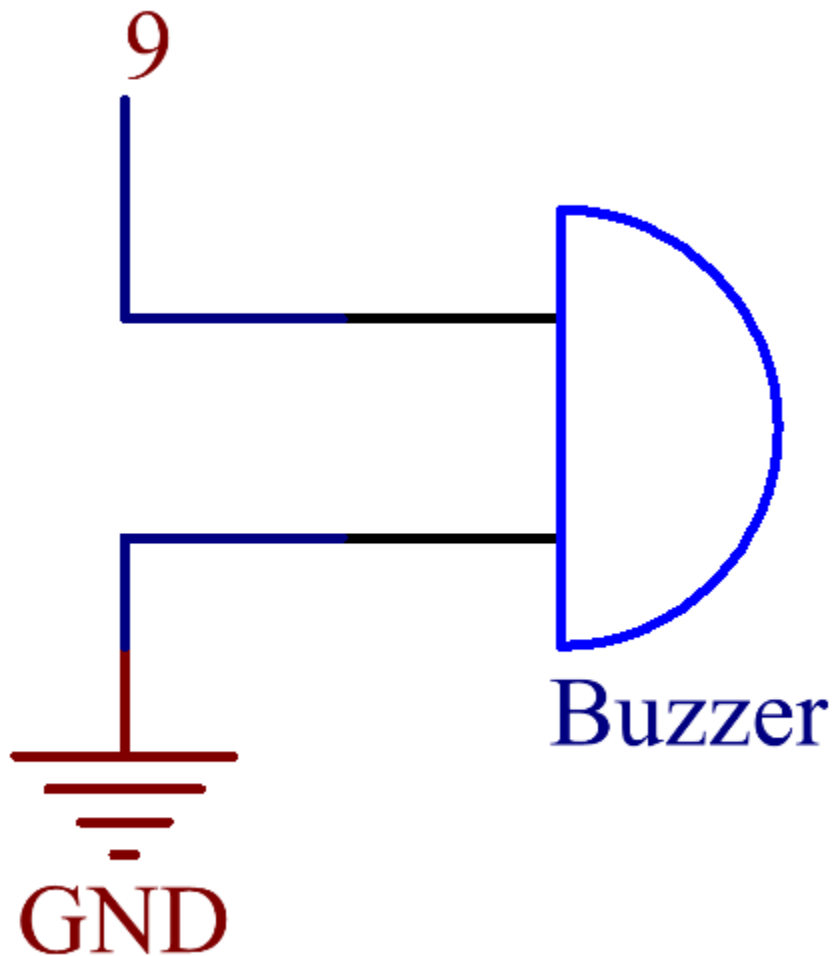
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Buzzer*

2.20.3 Fritzing Circuit



In this example, we use the digital pin 9 to drive the buzzer and extend the cathode of the Buzzer to GND and its anode to the digital pin 9.

2.20.4 Schematic Diagram



2.20.5 Code

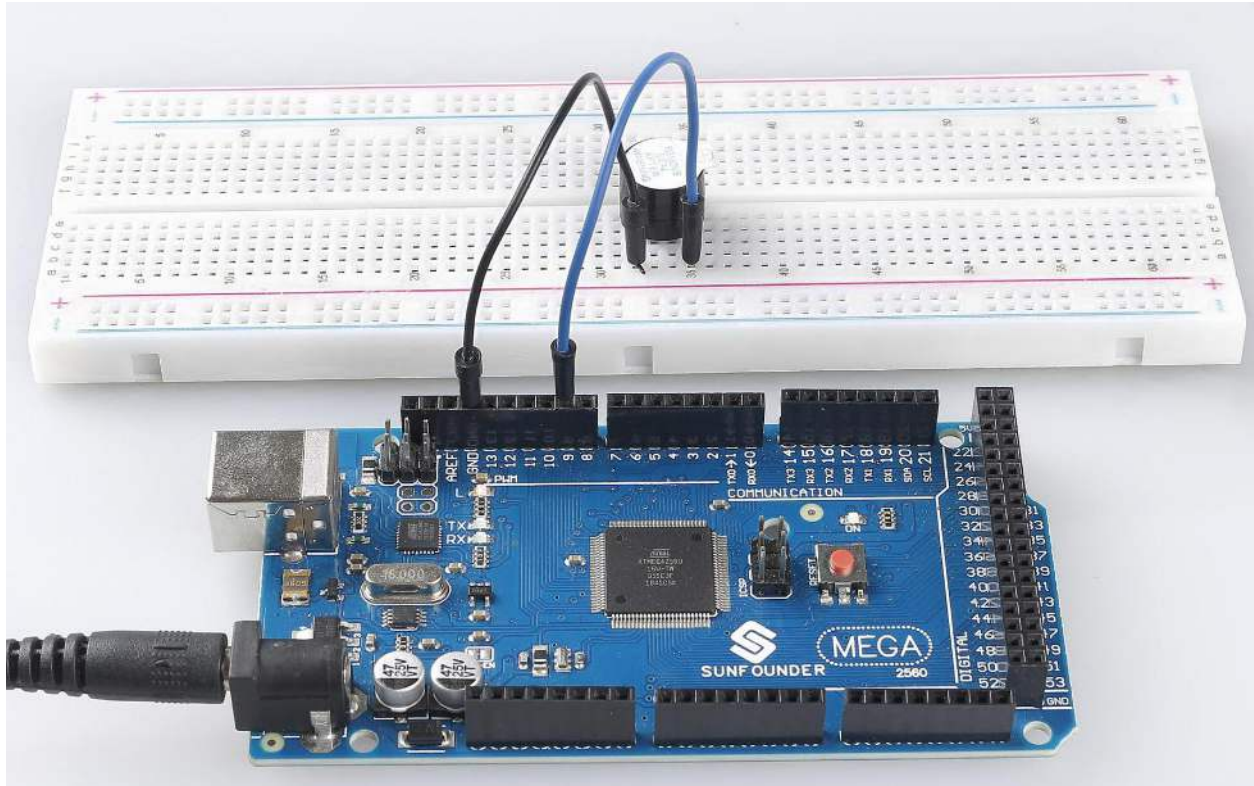
Note:

- You can open the file `2.10_activeBuzzer.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.10_activeBuzzer` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Or click **Open Code** to open it in [Web Editor](#).
 - Then *Upload the Code* to the board.
-

When you finish uploading the codes to the Mega2560 board, you can hear the beep—beep emitted from the buzzer. If you want to know more about the detail code explanation, please refer to **Part 1-1.2 Digital Write**.

1.2 Digital Write

2.20.6 Phenomenon Picture




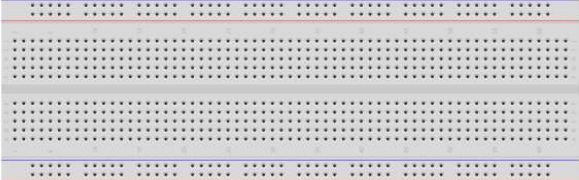


2.21 2.11 Passive Buzzer

2.21.1 Overview

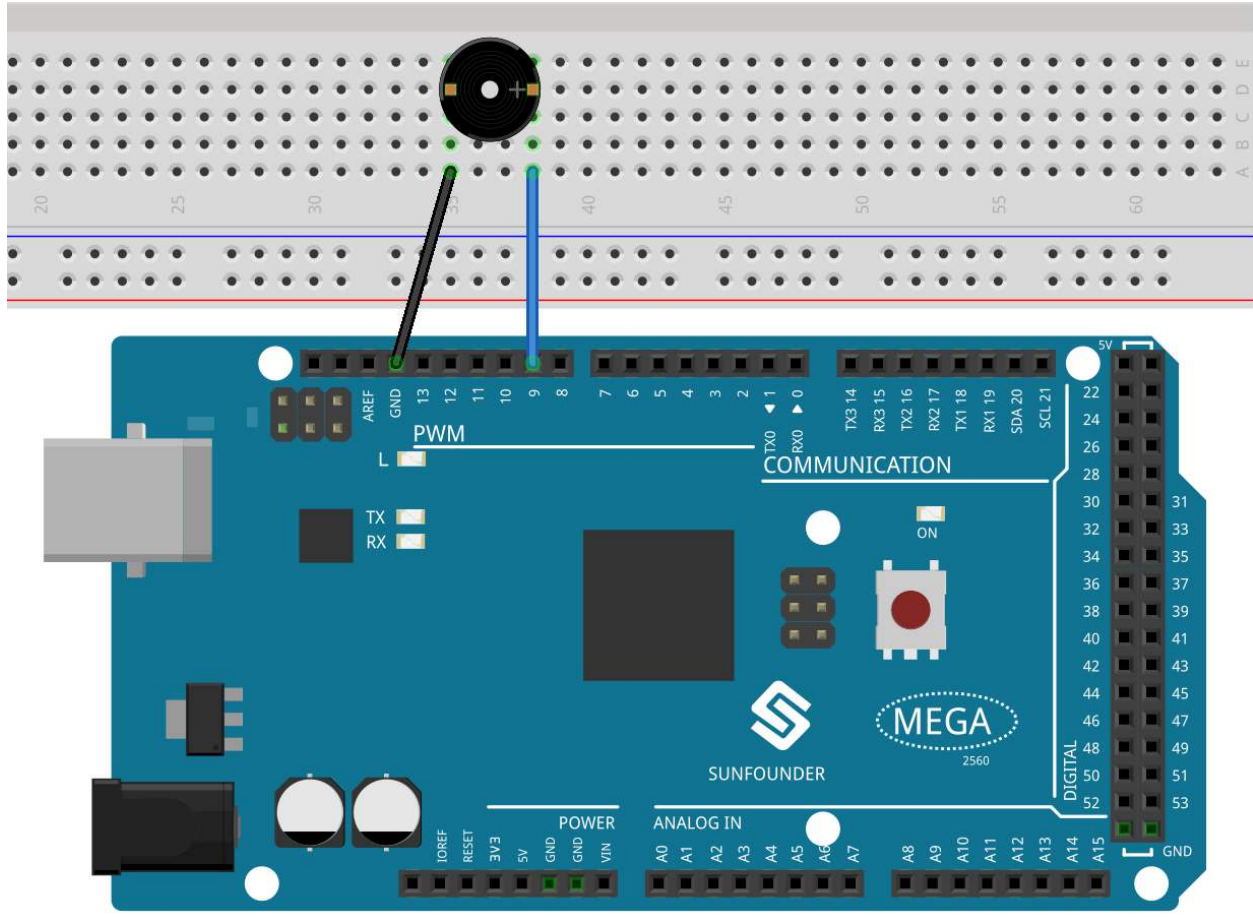
In this lesson, you will get to know about passive buzzer. As a type of electronic buzzer with an integrated structure, passive buzzer is supplied by DC power, widely used in computer, alarm, electronic toy, telephone, timer and other electronic products or voice devices.

2.21.2 Components Required

<p>1 * Passive Buzzer</p> 	<p>1 * Mega 2560 Board</p> 
<p>Several Jumper Wires</p> 	<p>1 * Breadboard</p> 

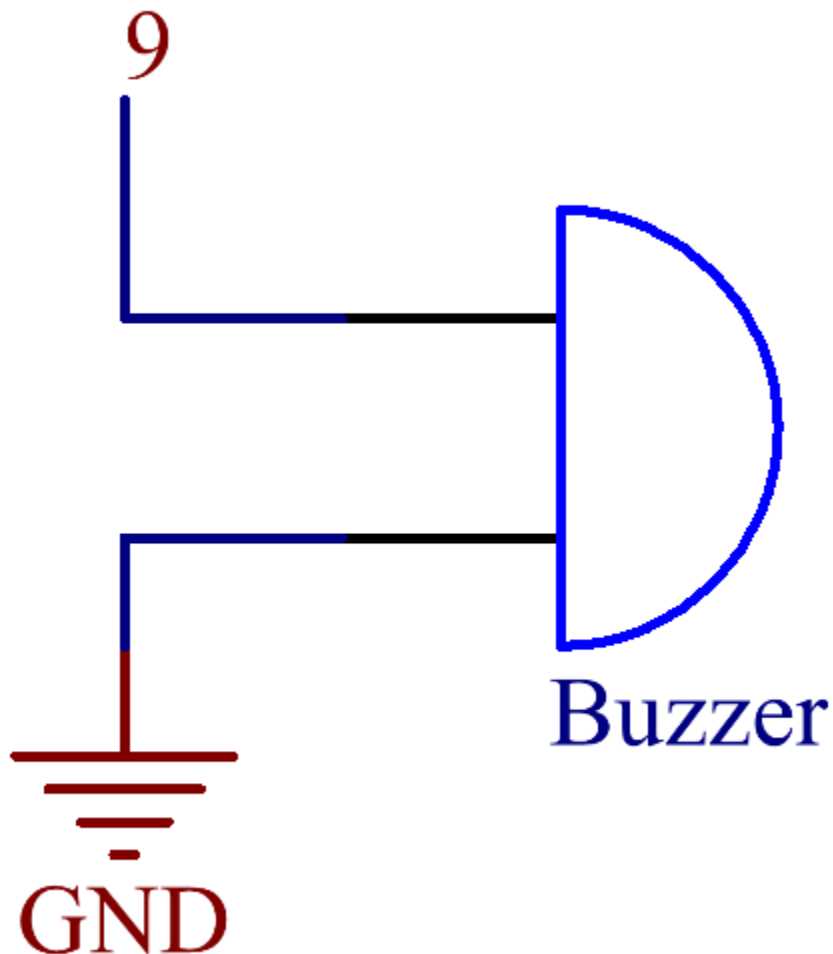
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Buzzer*

2.21.3 Fritzing Circuit



In this example, what we use to drive the buzzer is the pin 9. We get the cathode of the Buzzer to GND, and the anode to the digital pin 9.

2.21.4 Schematic Diagram



2.21.5 Code

Note:

- You can open the file `2.11_passiveBuzzer.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.11_passiveBuzzer` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Or click **Open Code** to open it in [Web Editor](#).
 - Then *Upload the Code* to the board.
-

At the time when you finish uploading the codes to the Mega2560 board, you can hear a melody containing seven notes.

2.21.6 Code Analysis

There are two points needing your attention:

1. `tone()` & `noTone()`: This function is used to control the sound of the passive buzzer directly and its prototype is as follows:

```
void tone(int pin, unsigned int frequency)
void tone(int pin, unsigned int frequency, unsigned long duration)
```

Generates a square wave of the specified frequency (and 50% duty cycle) on a pin (so as to make the passive buzzer vibrate to make sound). A duration can be specified, otherwise the wave continues until a call to `noTone()`. The pin can be connected to a piezo buzzer or other speaker to play tones.

Only one tone can be generated at a time. If a tone is already playing on a different pin, the call to `tone()` will have no effect. If the tone is playing on the same pin, the call will set its frequency.

Use of the `tone()` function will interfere with PWM output on pins 3 and 11 (on boards other than the Mega).

It is not possible to generate tones lower than 31Hz.

pin: The Arduino pin on which to generate the tone.

frequency: The frequency of the tone in hertz.

duration: The duration of the tone in milliseconds (optional)

```
void noTone(int pin)
```

Stops the generation of a square wave triggered by `tone()`. Has no effect if no tone is being generated.

pin: The Arduino pin on which to generate the tone.

Having known the two functions, you may grasp the codes—the installation of the array `melody[]` and the array `noteDurations[]` is the preparation of the subsequently several times of calling of the function `tone()` and the changing of tone and duration in the loop for better effect of music play.

2. `pitches.h`: The code uses an extra file, `pitches.h`. This file contains all the pitch values for typical notes. For example, `NOTE_C4` is middle C. `NOTE_FS4` is F sharp, and so forth. This note table was originally written by Brett Hagman, on whose work the `tone()` command was based. You may find it useful whenever you want to make musical notes.

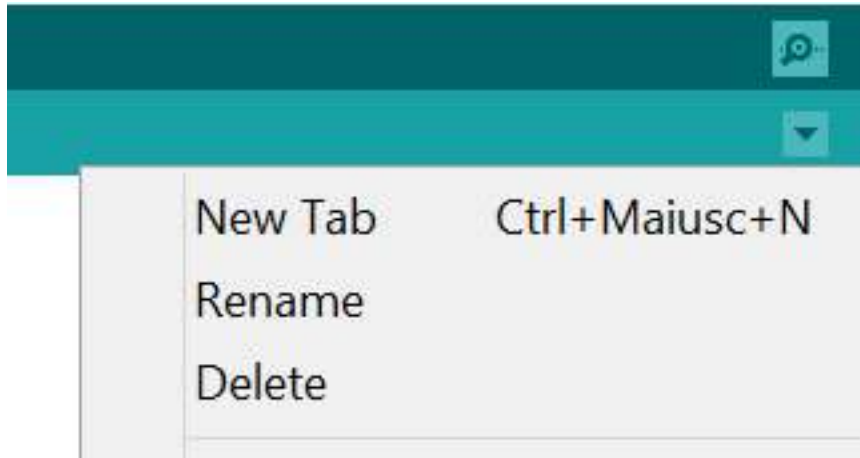
```
#include "pitches.h"
```

Note: There is already a `pitches.h` file in this sample program. If we put it together with the main code in one folder, the successive steps of installing `pitches.h` can be omitted.



After you open the code 2.11 passiveBuzzer, if you cannot open the `pitches.h` code, you can just install one manually. The steps are as follows:

To make the `pitches.h` file, either click on the button just below the serial monitor icon and choose “New Tab”, or use `Ctrl+Shift+N`.



Then paste in the following code and save it as `pitches.h`:

```
/*  
Public Constants  
*/  
#define NOTE_B0 31  
#define NOTE_C1 33  
#define NOTE_CS1 35  
#define NOTE_D1 37  
#define NOTE_DS1 39  
#define NOTE_E1 41  
#define NOTE_F1 44  
#define NOTE_FS1 46  
#define NOTE_G1 49  
#define NOTE_GS1 52  
#define NOTE_A1 55  
#define NOTE_AS1 58  
#define NOTE_B1 62  
#define NOTE_C2 65  
#define NOTE_CS2 69  
#define NOTE_D2 73  
#define NOTE_DS2 78  
#define NOTE_E2 82  
#define NOTE_F2 87  
#define NOTE_FS2 93  
#define NOTE_G2 98  
#define NOTE_GS2 104  
#define NOTE_A2 110  
#define NOTE_AS2 117  
#define NOTE_B2 123  
#define NOTE_C3 131  
#define NOTE_CS3 139  
#define NOTE_D3 147  
#define NOTE_DS3 156  
#define NOTE_E3 165  
#define NOTE_F3 175
```

(continues on next page)

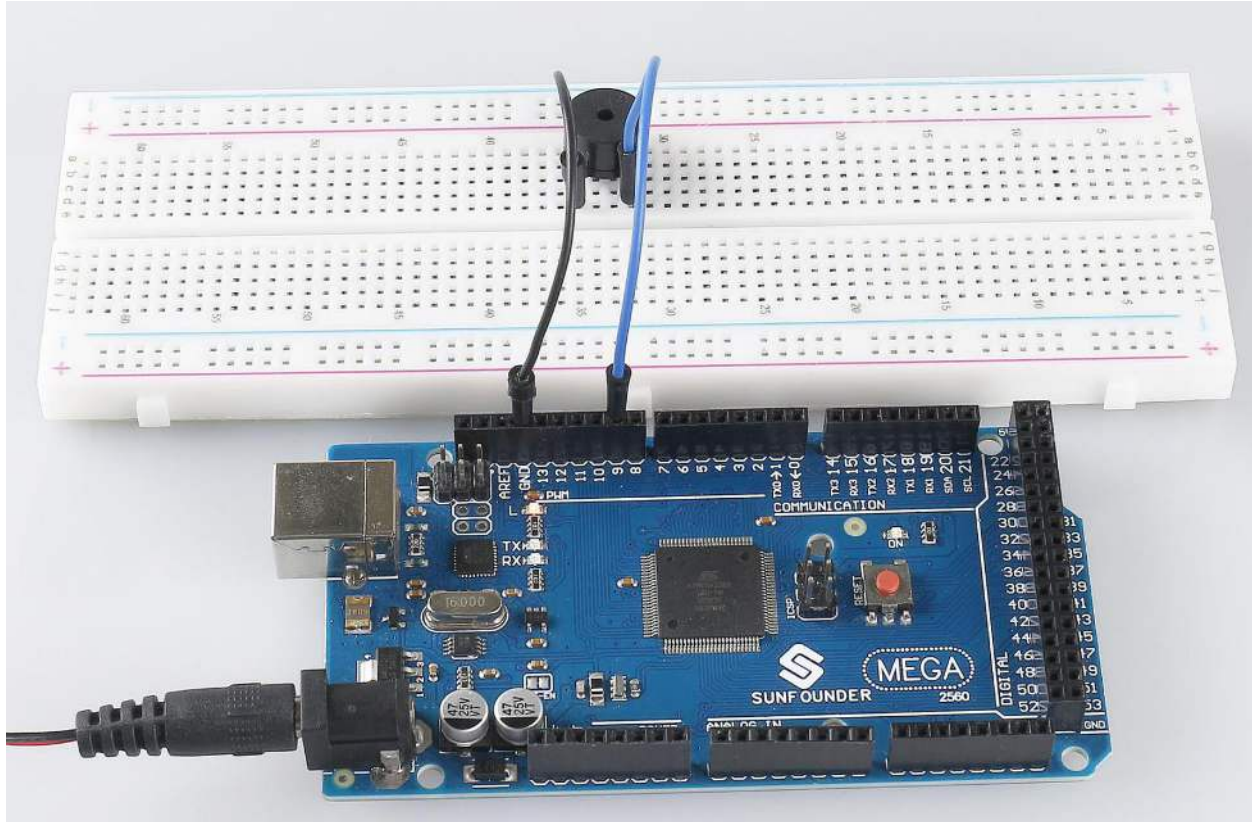
(continued from previous page)

```
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
```

(continues on next page)

```
#define NOTE_DS8 49
```

2.21.7 Phenomenon Picture



2.22 2.12 Servo

2.22.1 Overview

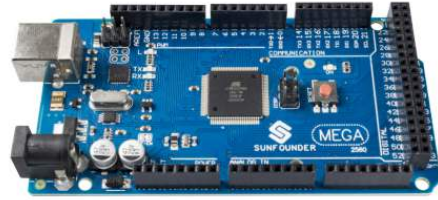
In this lesson, you will learn something about Servo. Servo is a kind of driver whose position (angular) can be adjustable and kept or a rotary actuator that allows for precise control of angular position. Currently, it is widely used in upscale remote control toys, such as airplane, submarine, telerobot and so on.

2.22.2 Components Required

1 * Servo



1 * Mega 2560 Board



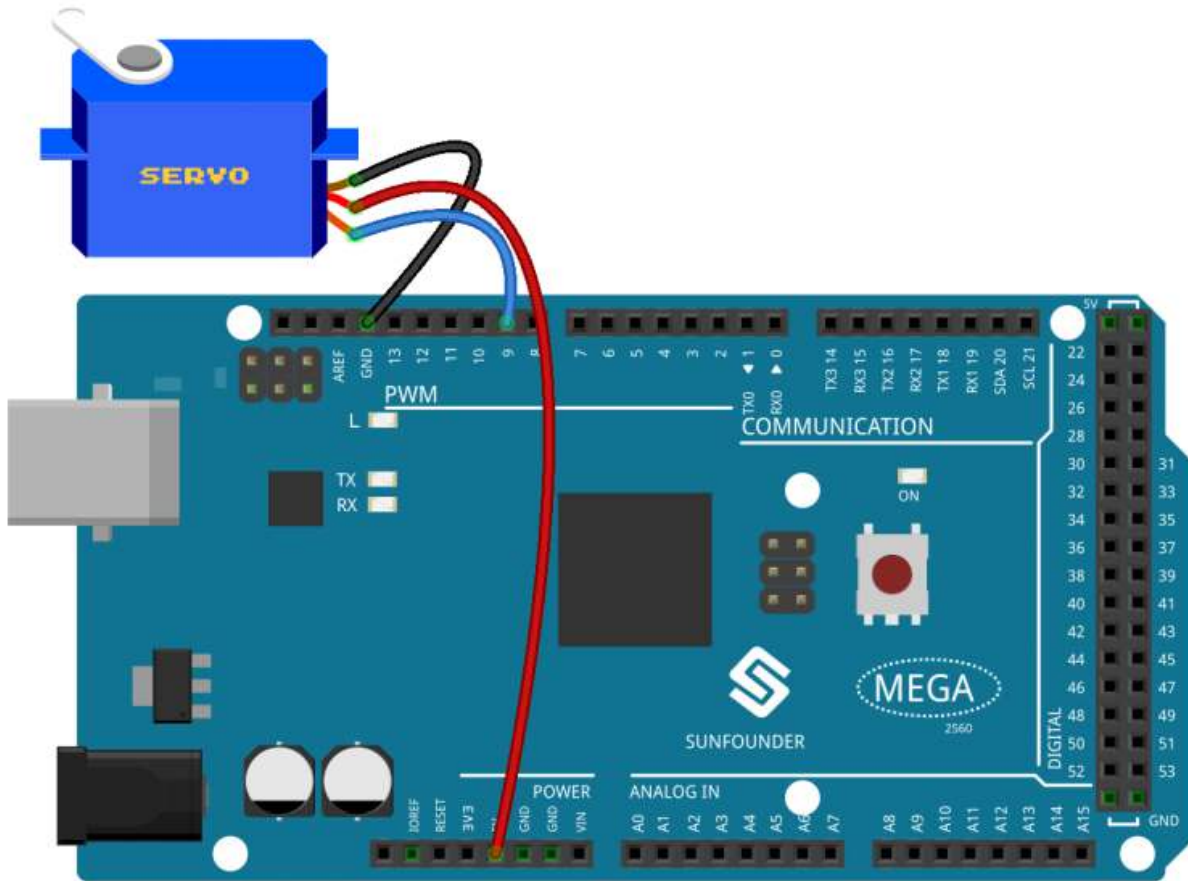
Several Jumper Wires



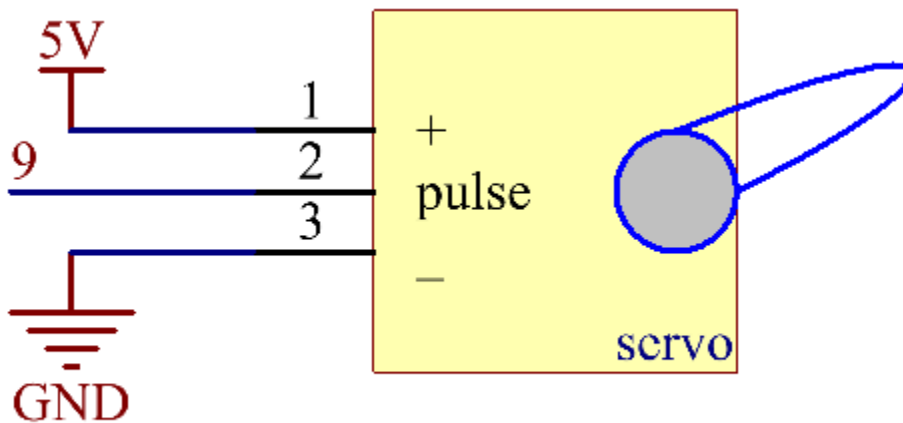
- *SunFounder Mega Board*
- *Jumper Wires*
- *Servo*

2.22.3 Fritzing Circuit

In this example, we use PWM pin 9 to drive the Servo, and get the orange wire of the servo connected to the PWM pin 9, the red one to 5V, and the brown one to GND.



2.22.4 Schematic Diagram



2.22.5 Code

Note:

- You can open the file `2.12_servo.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.12_servo` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Or click **Open Code** to open it in [Web Editor](#).
 - Then *Upload the Code* to the board.
-

Once you finish uploading the codes to the Mega2560 board, you can see the servo arm rotating in the range 0°~180°.

2.22.6 Code Analysis

By calling the library `Servo.h`, you can drive the servo easily.

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

Library Functions

```
Servo
```

Create **Servo** object to control a servo.

```
uint8_t attach(int pin);
```

Turn a pin into a servo driver. Calls `pinMode`. Returns 0 on failure.

```
void detach();
```

Release a pin from servo driving.

```
void write(int value);
```

Set the angle of the servo in degrees, 0 to 180.

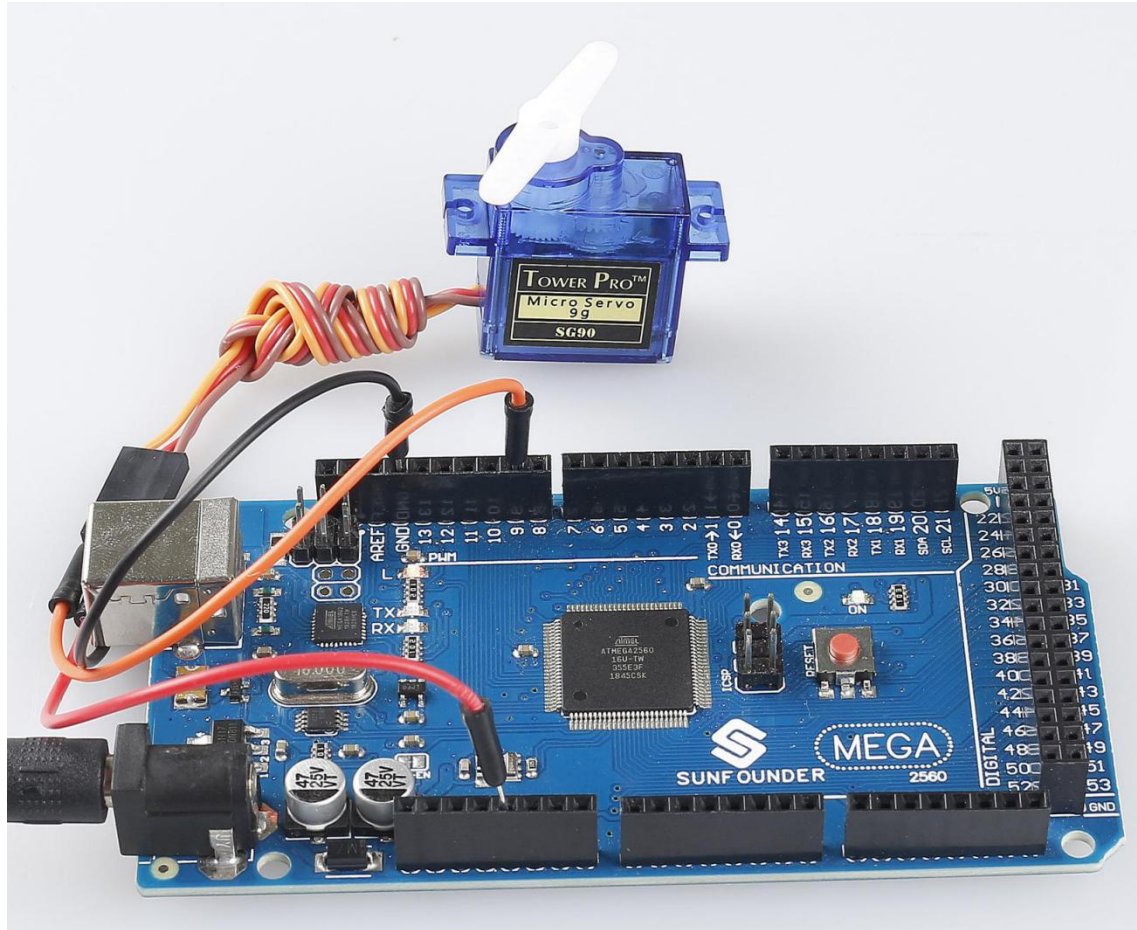
```
int read();
```

Return that value set with the last `write()`.

```
bool attached();
```

Return 1 if the servo is currently attached.

2.22.7 Phenomenon Picture

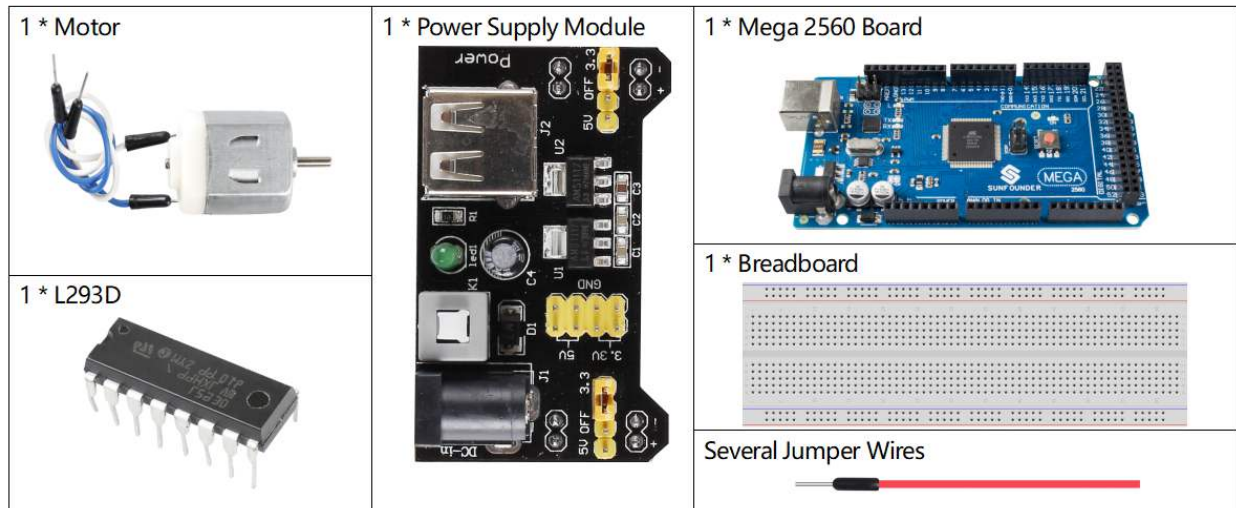


2.23 2.13 Motor

2.23.1 Overview

In this lesson, you will learn how to use Motor, the working principle of which is that the energized coil is forced to rotate in the magnetic field then the rotor of the motor rotates accordingly on which the pinion gear drives the engine flywheel to rotate.

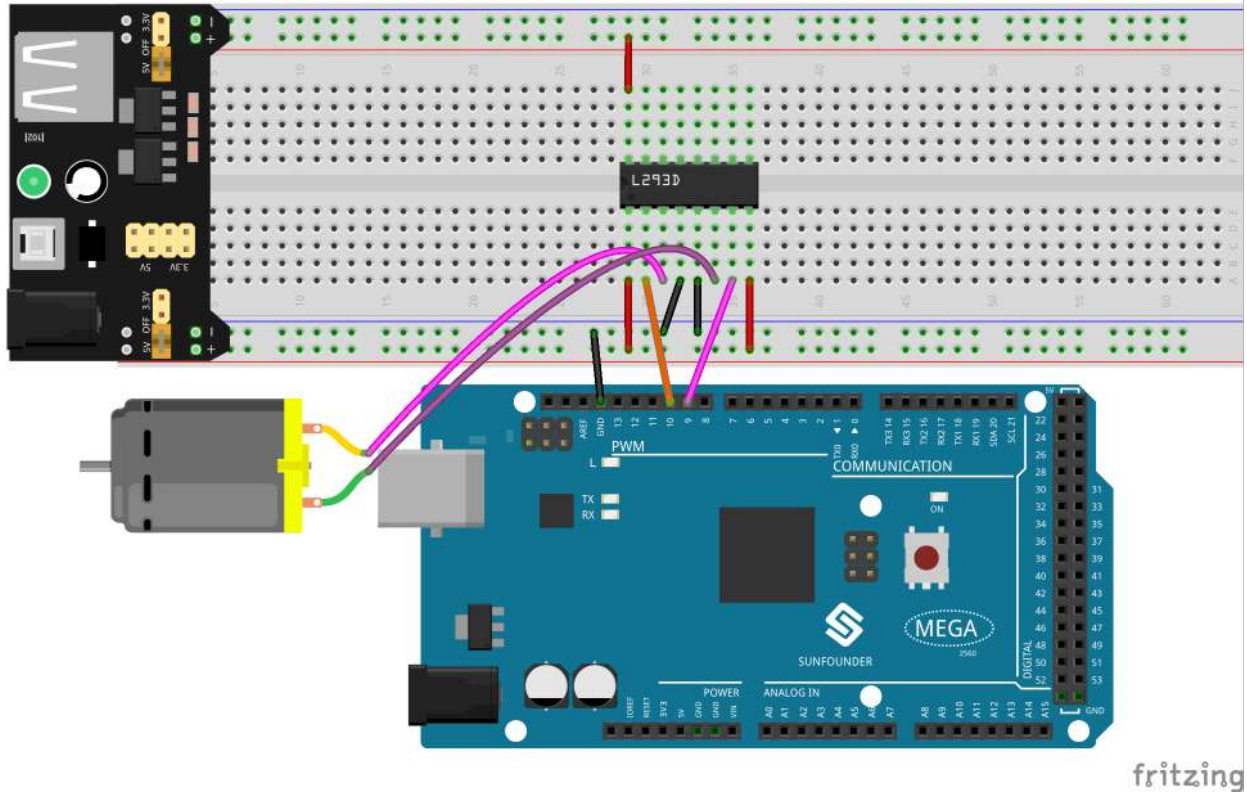
2.23.2 Components Required



- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *L293D*
- *DC Motor*
- *Power Supply Module*

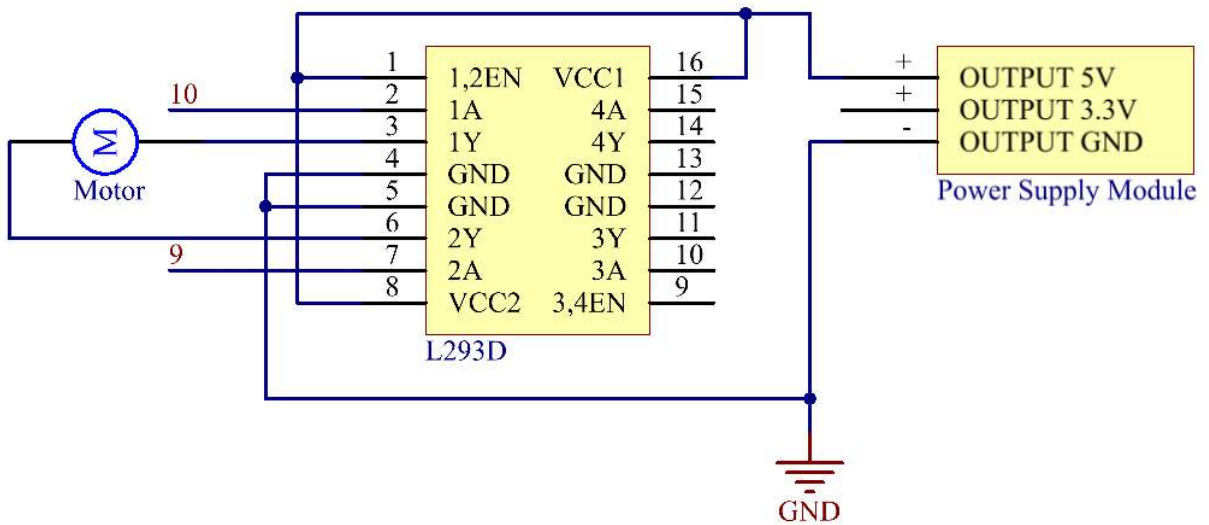
2.23.3 Fritzing Circuit

In this example, we use Power Supply Module to power the anode and cathode of breadboard. GND of Mega 2560 Board is connected to the cathode.



fritzing

2.23.4 Schematic Diagram



2.23.5 Code

Note:

- You can open the file `2.13_motor.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.13_motor` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After uploading the codes to the Mega2560 board, you can select the rotating direction of motor by typing A or B in the serial monitor.

2.23.6 Code Analysis

The motor can be driven by providing a voltage difference between the copper sheets at both sides of the motor. Therefore, you only need to write 0 for the voltage of one side of the copper sheet and 5V for the other side. Modify the written analog signal value to adjust the direction and speed.

```
void clockwise(int Speed)
{
  analogWrite(motor1A, 0);
  analogWrite(motor2A, Speed);
}

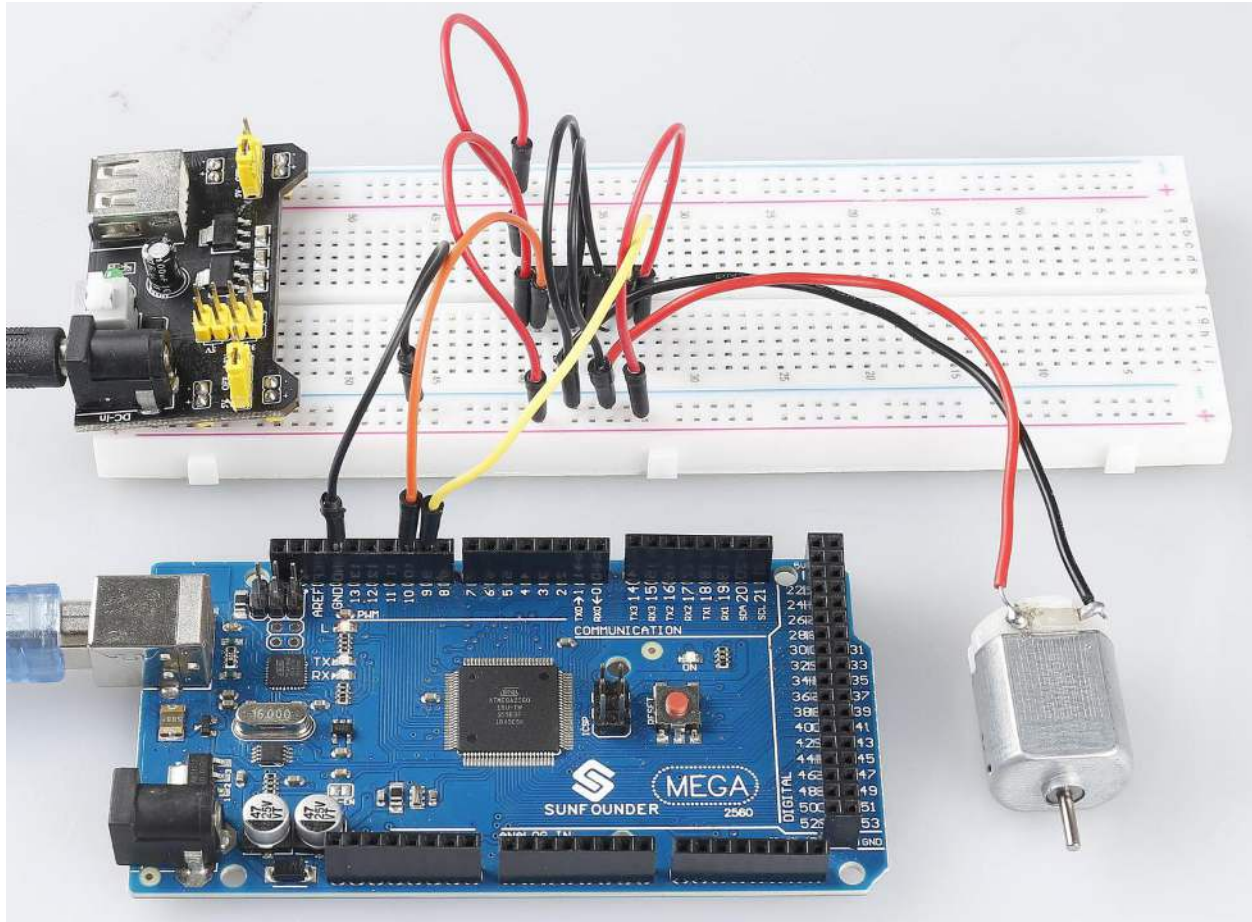
void anticlockwise(int Speed)
{
  analogWrite(motor1A, Speed);
  analogWrite(motor2A, 0);
}
```

In this example, `Serial.Read()` is used to control the direction of motor.

When you type 'A' in serial monitor, there calls the clockwise (255) function to make the motor rotate with the speed of 255. Input 'B', and the motor will rotate in reverse direction.

```
void loop() {
  if (Serial.available() > 0) {
    int incomingByte = Serial.read();
    switch(incomingByte){
      case 'A':
        clockwise(255);
        Serial.println("The motor rotate clockwise.");
        break;
      case 'B':
        anticlockwise(255);
        Serial.println("The motor rotate anticlockwise.");
        break;
    }
  }
  delay(3000);
  stopMotor();
}
```

2.23.7 Phenomenon Picture




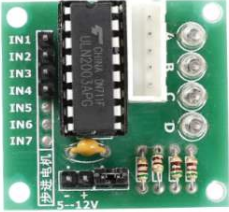
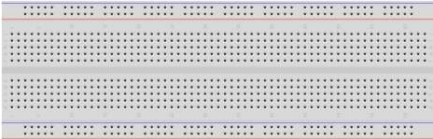



2.24 2.14 Stepper Motor

2.24.1 Overview

In this lesson, you will learn about Stepper Motor.

2.24.2 Components Required

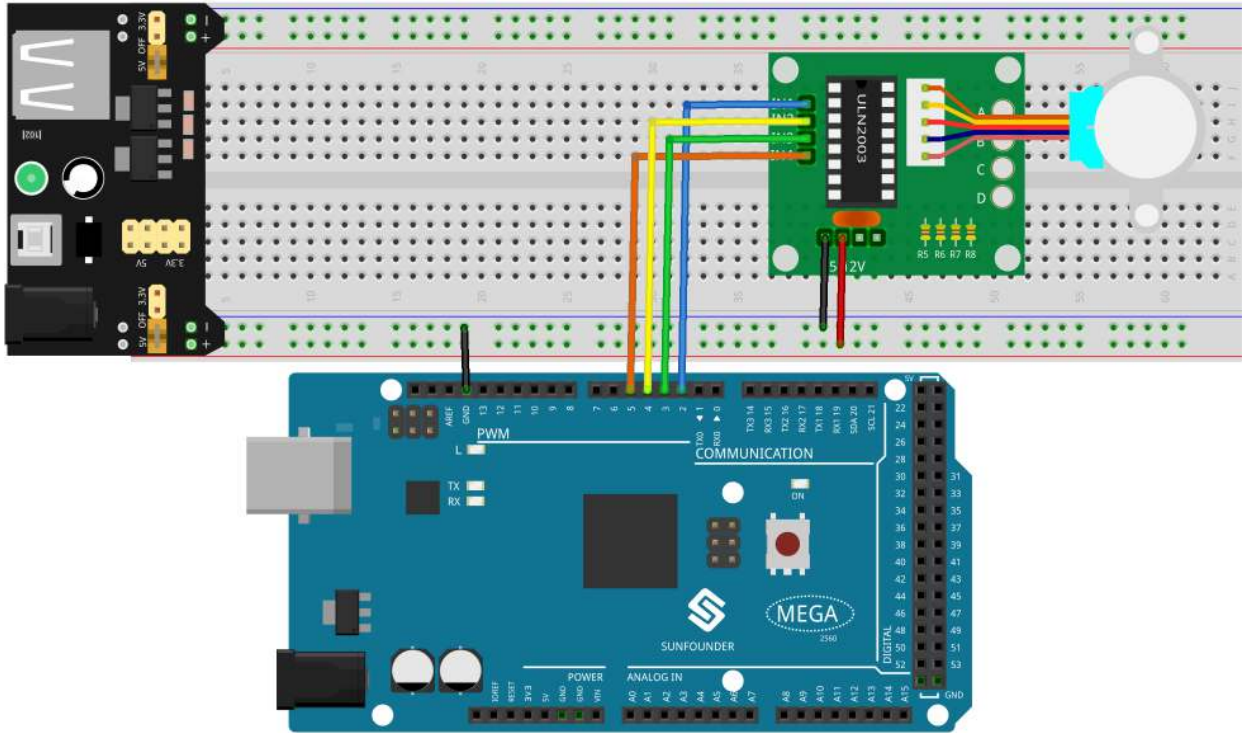
<p>1 * Stepper Motor</p> 	<p>1 * Power Supply Module</p> 	<p>1 * Mega 2560 Board</p> 
<p>1 * ULN2003</p> 	<p>1 * Breadboard</p> 	
<p>Several Jumper Wires</p> 		

- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Stepper Motor*
- *Power Supply Module*

2.24.3 Fritzing Circuit

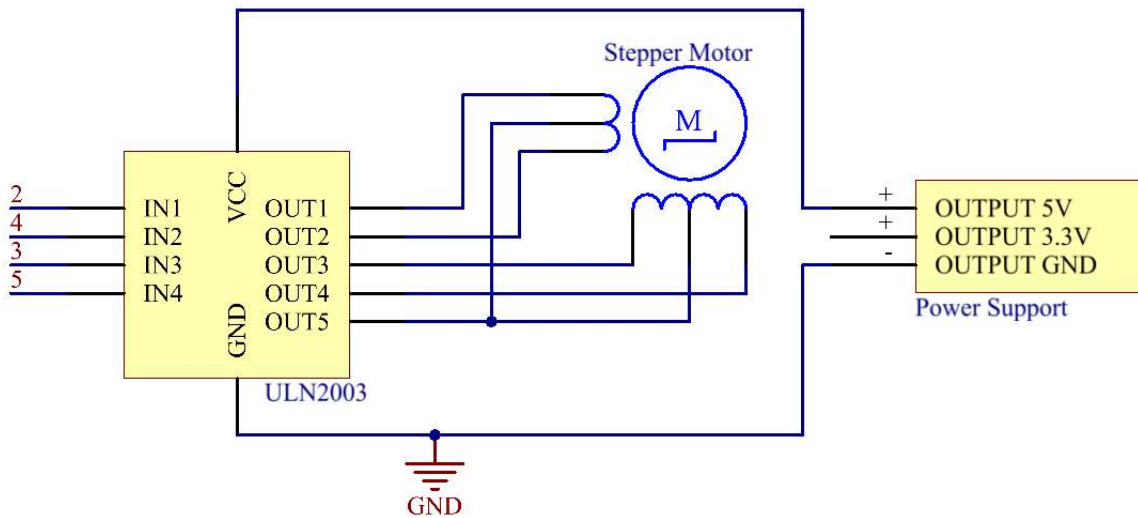
Power Supply Module is used to power the stepper motor. Get the GND of Mega 2560 Board and GND of ULN2003 connected to the cathode of the breadboard, and connect the VCC of ULN2003 to 5V OUTPUT of Power Supply.

The wiring of ULN2003 and Mega2560 is shown as follows:



fritzing

2.24.4 Schematic Diagram



2.24.5 Code

Note:

- You can open the file `2.14_stepperMotor.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.14_stepperMotor` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After uploading the codes to the Mega2560 board, you will be able to see that the stepper motor rotates one circle with an interval of a second and each circle takes 3.75s.

2.24.6 Code Analysis

By calling the library `Stepper.h`, you can easily drive the stepper motor.

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

Library Functions

```
Stepper(steps, pin1, pin2, pin3, pin4)
```

Creates a new instance of the `Stepper` class that represents a particular stepper motor attached to your Arduino board.

- `steps`: the number of steps in one revolution of your motor. If your motor gives the number of degrees per step, divide that number into 360 to get the number of steps (e.g. $360 / 3.6$ gives 100 steps). (int)

Note: Every circle of the stepper motor takes 2048 steps.

```
setSpeed(rpm)
```

Sets the motor speed in rotations per minute. This function doesn't make the motor turn, just sets the speed at which it will when you call `step()`.

- `rpm`: the speed at which the motor should turn in rotations per minute - a positive number. (long)

Note: The stepper motor we use here rotates 17 circles a minute at most.

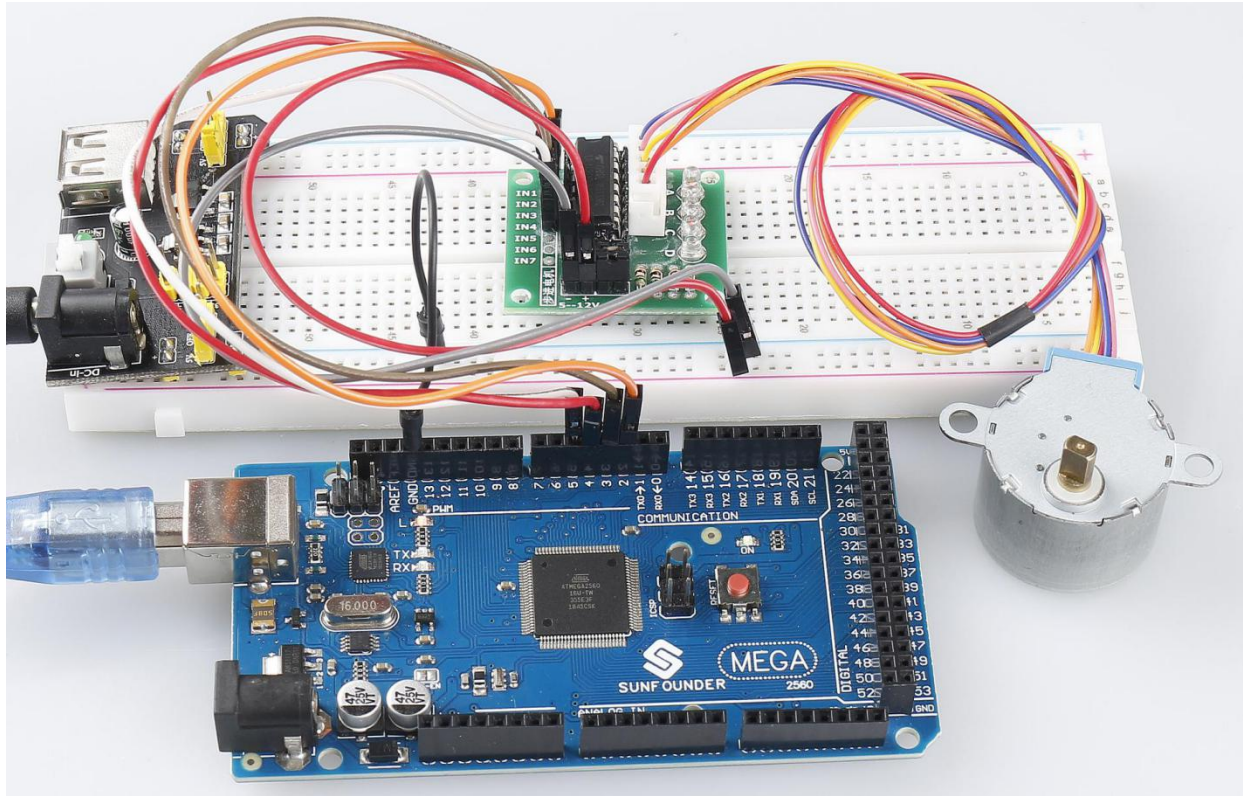
```
step(steps)
```

Turns the motor a specific number of steps, at a speed determined by the most recent call to `setSpeed()`.

This function is blocking; that is, it will wait until the motor has finished moving to pass control to the next line in your sketch. For example, if you set the speed to, say, 1 RPM and called `step(2048)` on a 2048-step motor, this function would take a full minute to run. For better control, keep the speed high and only go a few steps with each call to `step()`.

- `steps`: the number of steps to turn the motor - positive to turn one direction, negative to turn the other. (int)

2.24.7 Phenomenon Picture





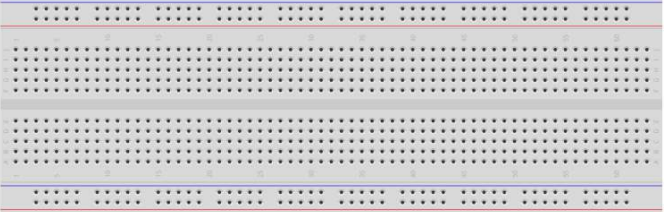


2.25 2.15 Button

2.25.1 Overview

In this lesson, you will learn about Button. Button is a common component used to control electronic devices. It is usually used as a switch to connect or break circuits.

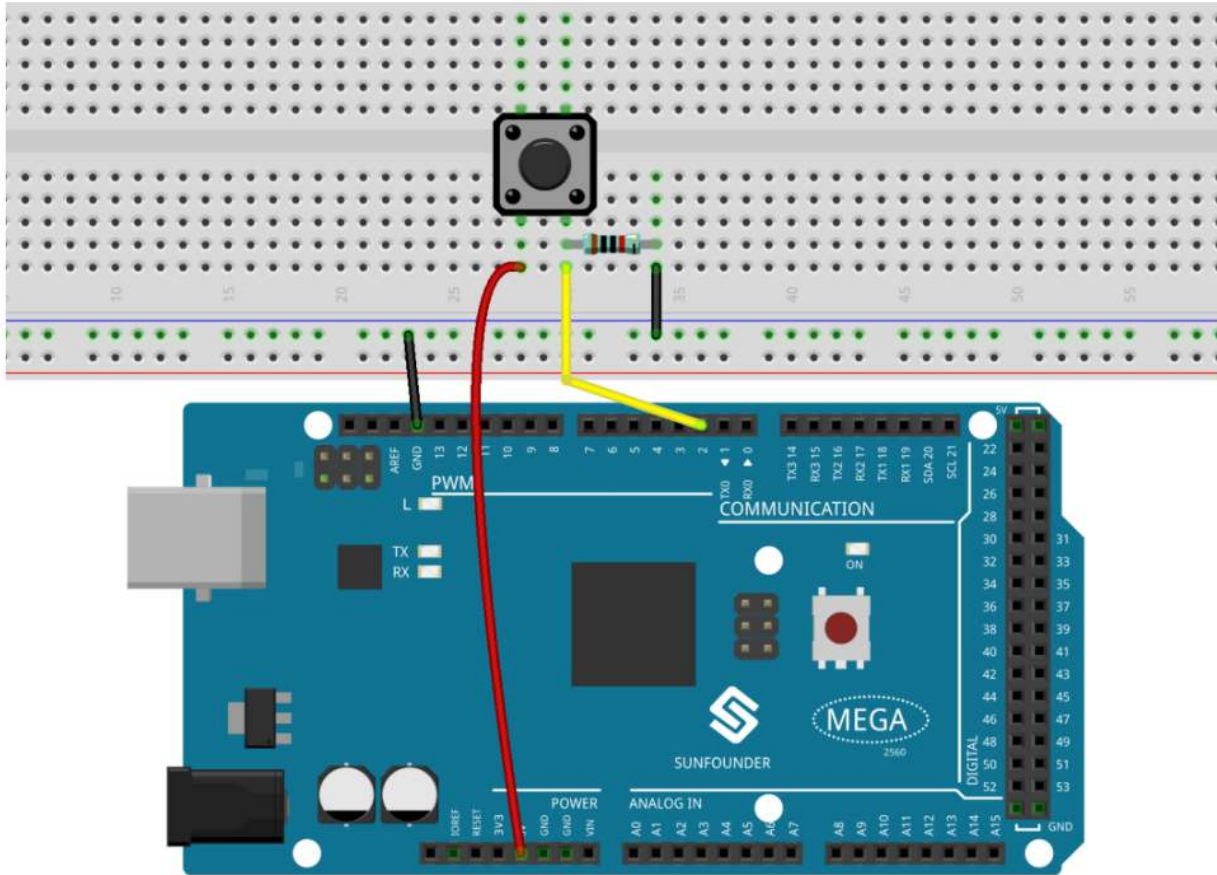
2.25.2 Components Required

<p>1 * Button</p> 	<p>1 * 10k ohm resistor</p> 	<p>Several Jumper Wires</p> 
<p>1 * Mega 2560 Board</p> 	<p>1 * Breadboard</p> 	

- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Resistor*
- *Button*

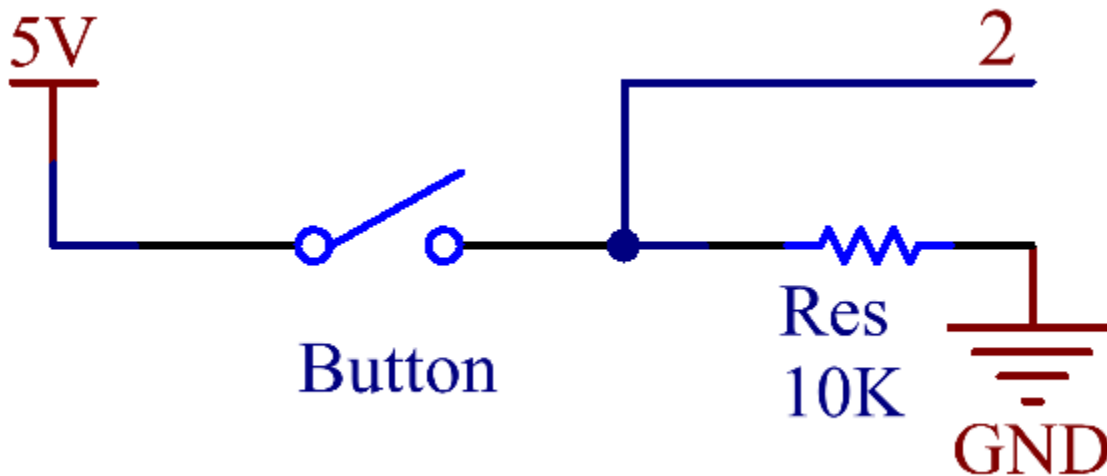
2.25.3 Fritzing Circuit

In this example, we read the signal of the button with the digital pin. When the button is not pressed, the digital pin 2 (through the drop-down resistor) is connected to ground to read the low level (0); when the button is pressed, the two pins are connected and when the pin is connected to the 5V power supply, the high level (1) is read.



Note: If you disconnect the digital I/O pin from anything, the LED may blink erratically. The input is “floating” or it doesn’t have a solid connection to voltage or ground, so it will randomly return either HIGH or LOW. That’s why there needs a pull-down resistor in the circuit.

2.25.4 Schematic Diagram



2.25.5 Code

Example 1

Note:

- You can open the file `2.15_button.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.15_button` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

Uploaded the codes to the Mega2560 board, you can see the readings of the pins on the serial monitor. When you press down the Button, there will display 1 on the serial monitor, and once you release it, there will display 0. As for the detail code explanation, please refer to [1.4 Digital Read](#).

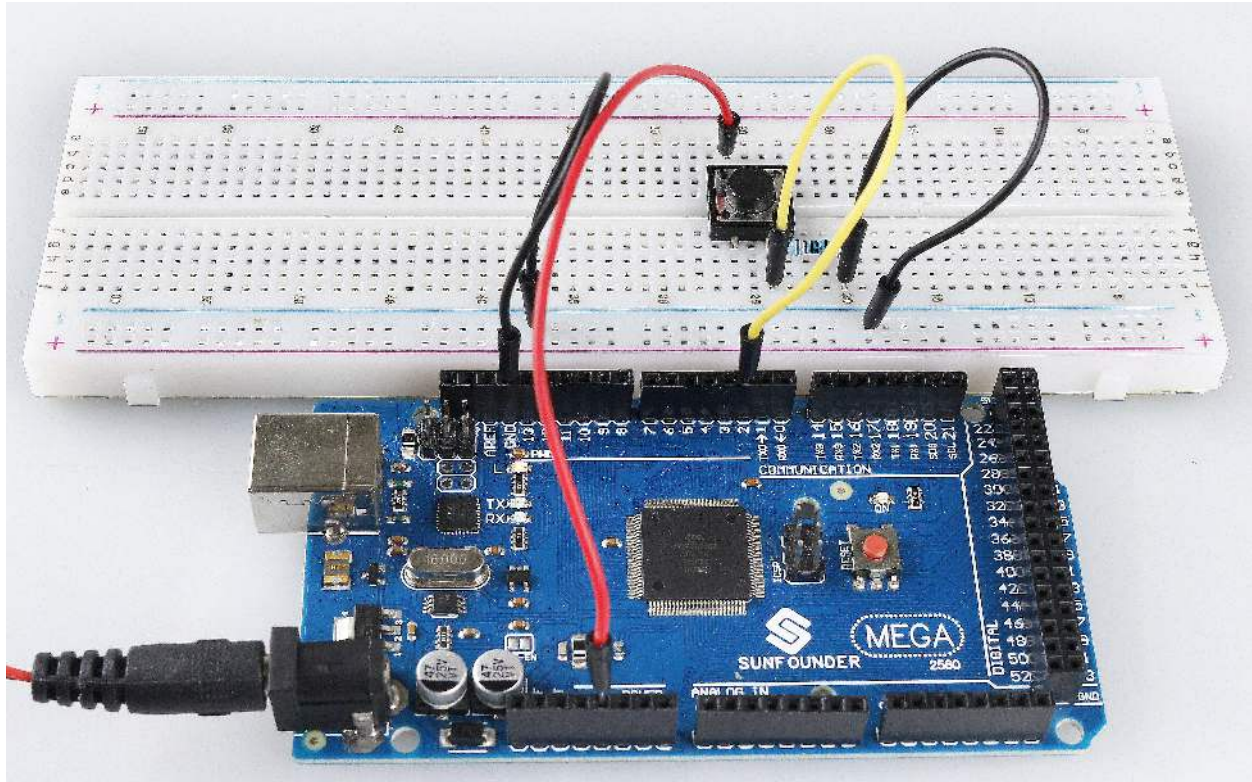
Example 2

Note:

- You can open the file `2.15_button_2.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.15_button_2` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

Uploaded the codes to the Mega2560 board, every time you press the button, the output value will switch between 0 and 1. If you want to know more about the code explanation, you can turn to [1.10 State Change Detection](#).

2.25.6 Phenomenon Picture

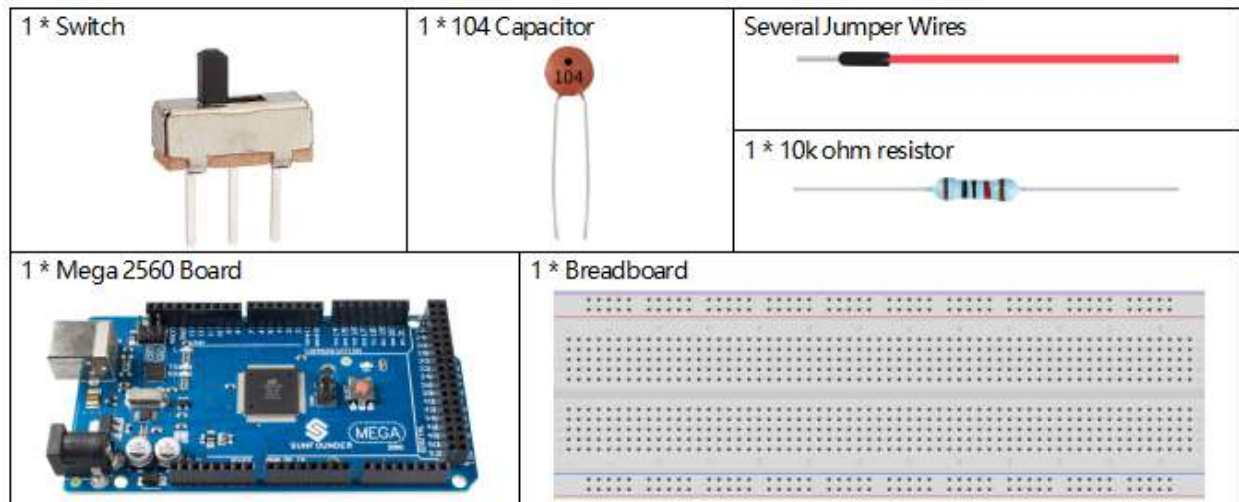


2.26 2.16 Slide Switch

2.26.1 Overview

In this lesson, you will get to know something about Switch. A slide switch, just as its name implies, is to slide the switch bar to connect or break the circuit, and further switch circuits. The slide switch is commonly used in low-voltage circuit. It has the features of flexibility and stability, and applies in electric instruments and electric toys widely.

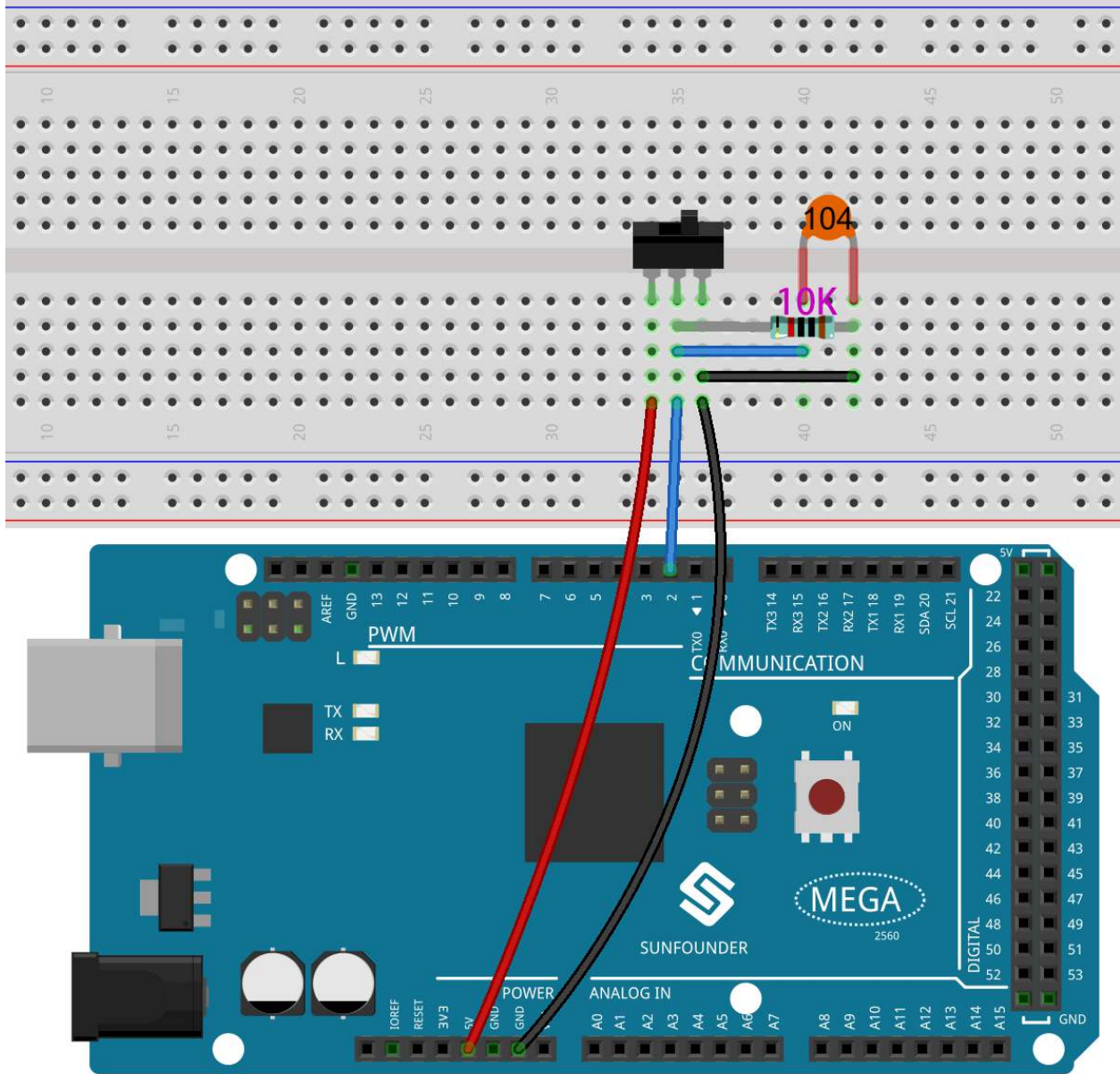
2.26.2 Components Required



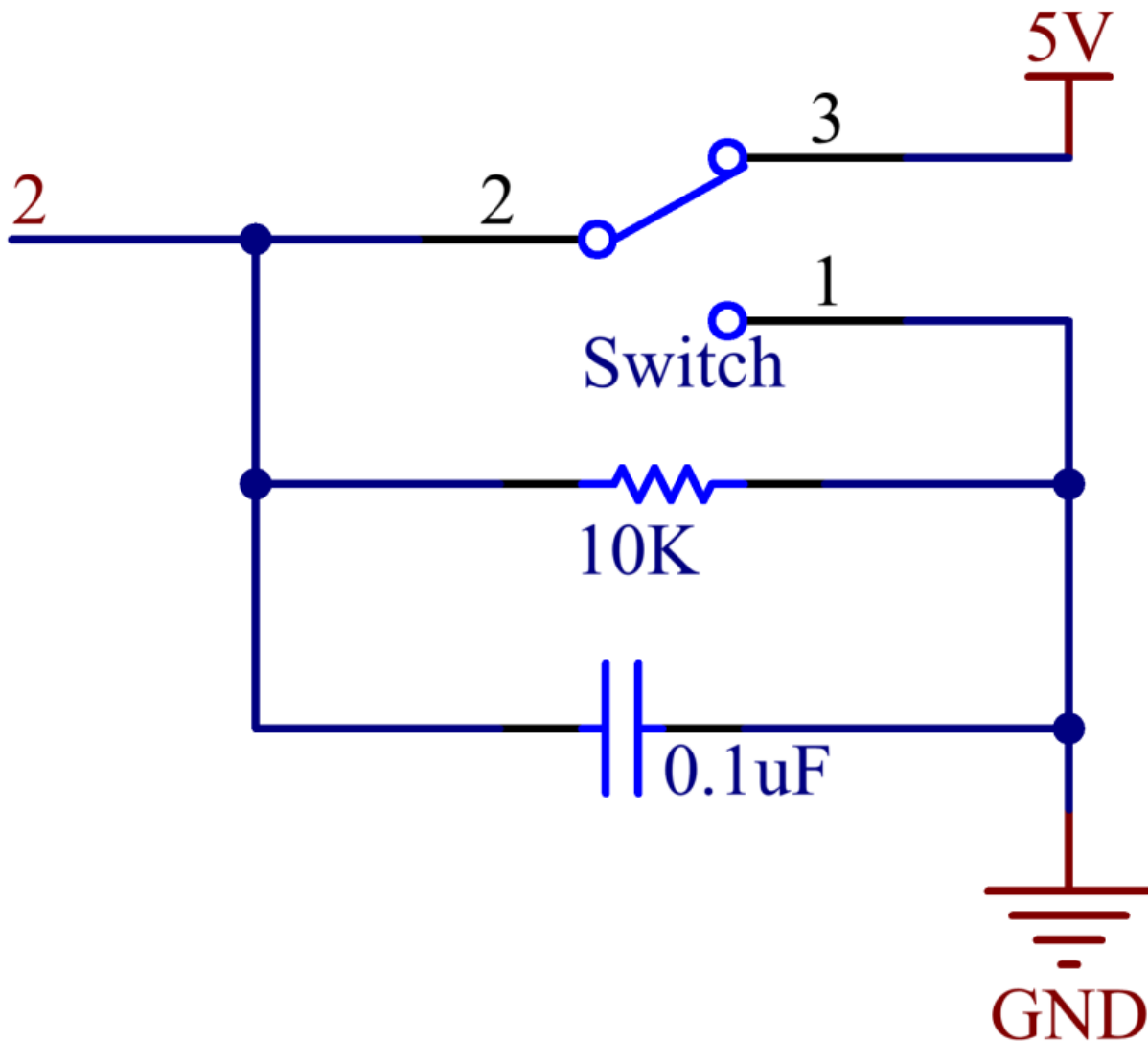
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Resistor*
- *Slide Switch*
- *Capacitor*

2.26.3 Fritzing Circuit

In this example, digital pin 2 is used to read the signal of Switch. In addition, you need to connect a 10 k resistor and a 104 capacitor in parallel to form a RC circuit (Resistor - Capacitance circuit) which is set between pin 2 and GND to realize debounce that may arise from your toggle of switch.



Schematic Diagram



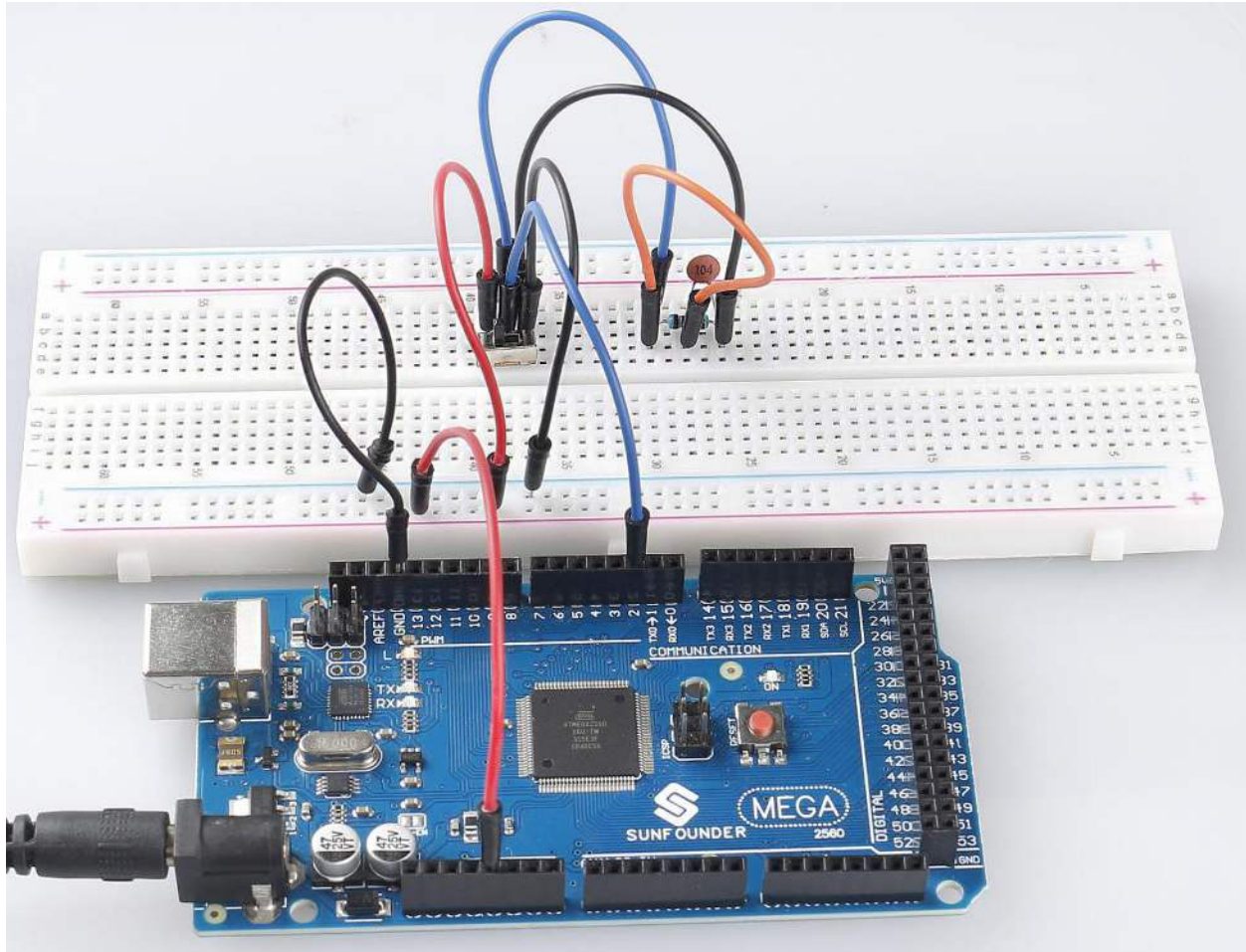
2.26.4 Code

Note:

- You can open the file `2.16_switch.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.16_switch` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After the codes are uploaded to the Mega2560 board, you can open the serial monitor to check the readings of the pin. When the Switch toggles to the left, the serial monitor displays 1; when toggles to the right, the serial monitor displays 0. Refer to [1.4 Digital Read](#) to check the code explanation.

2.26.5 Phenomenon Picture

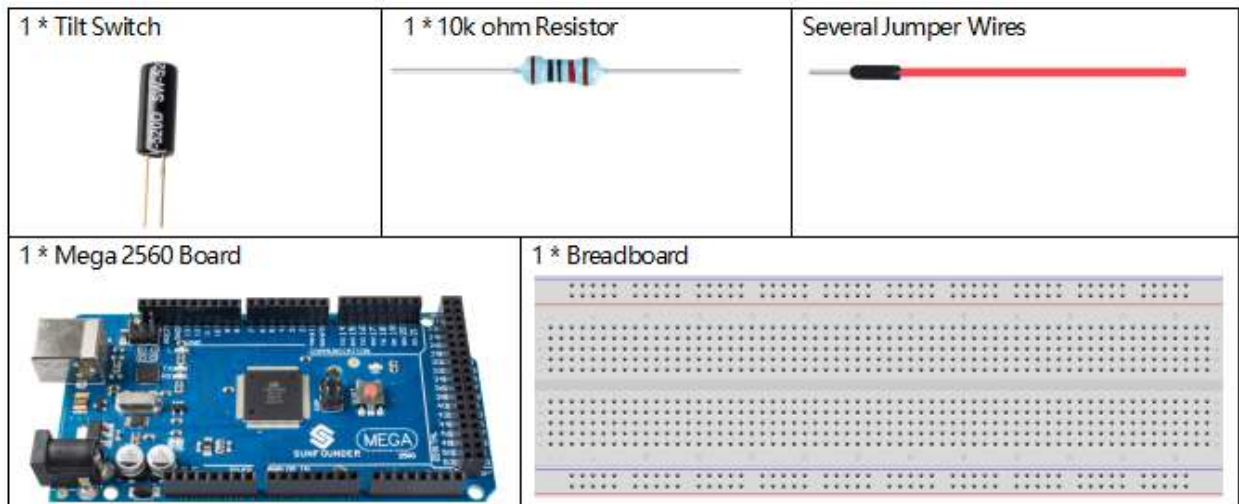


2.27 2.17 Tilt Switch

2.27.1 Overview

In this lesson, you will learn about tilt switch. Tilt switch can be used to detect whether objects tilt, which is of great value in practical applications. It can be used to judge the tilt of bridges, buildings, transmission line tower and so on, so it has an important guiding function in carrying out maintenance work.

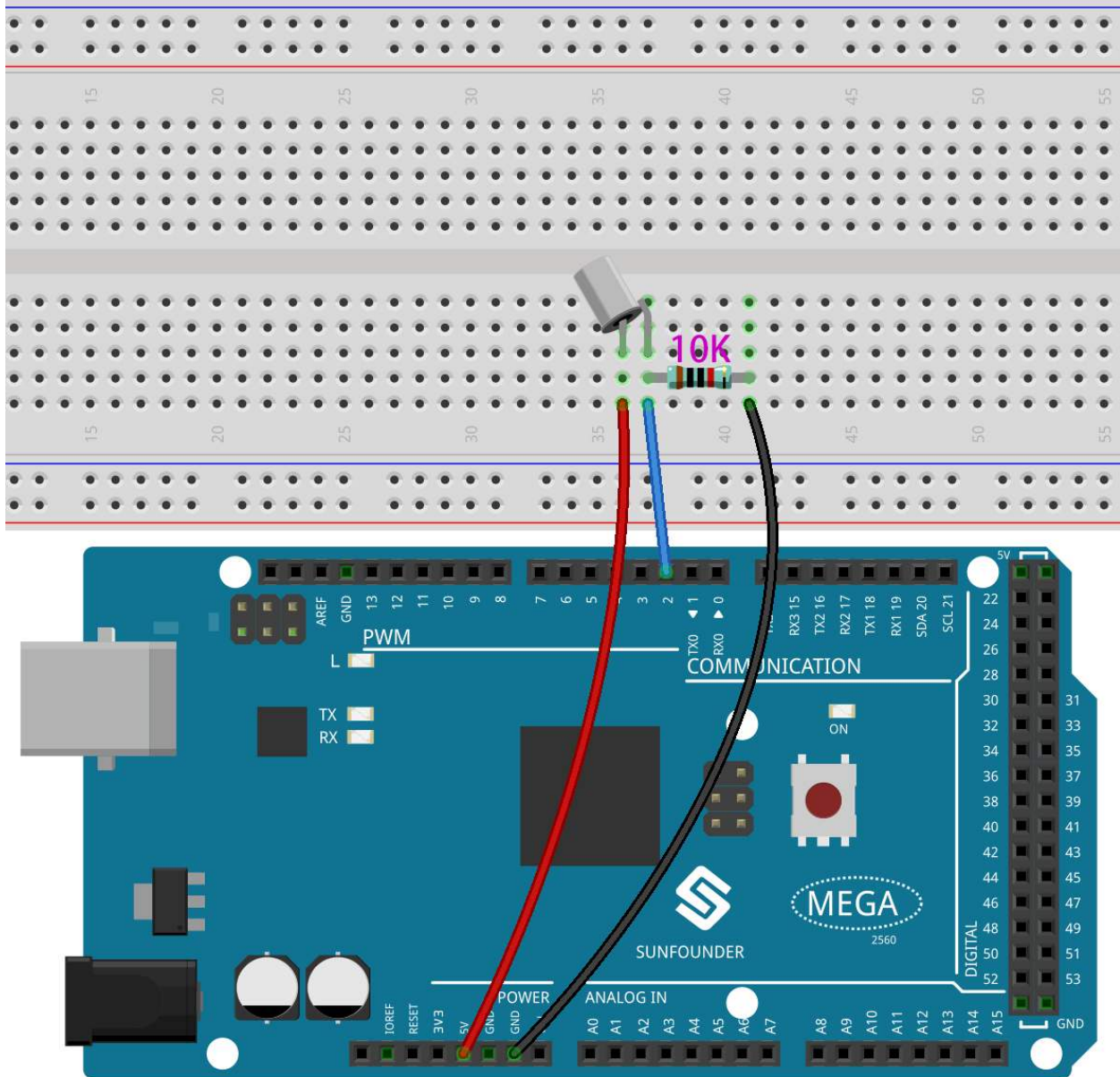
2.27.2 Components Required



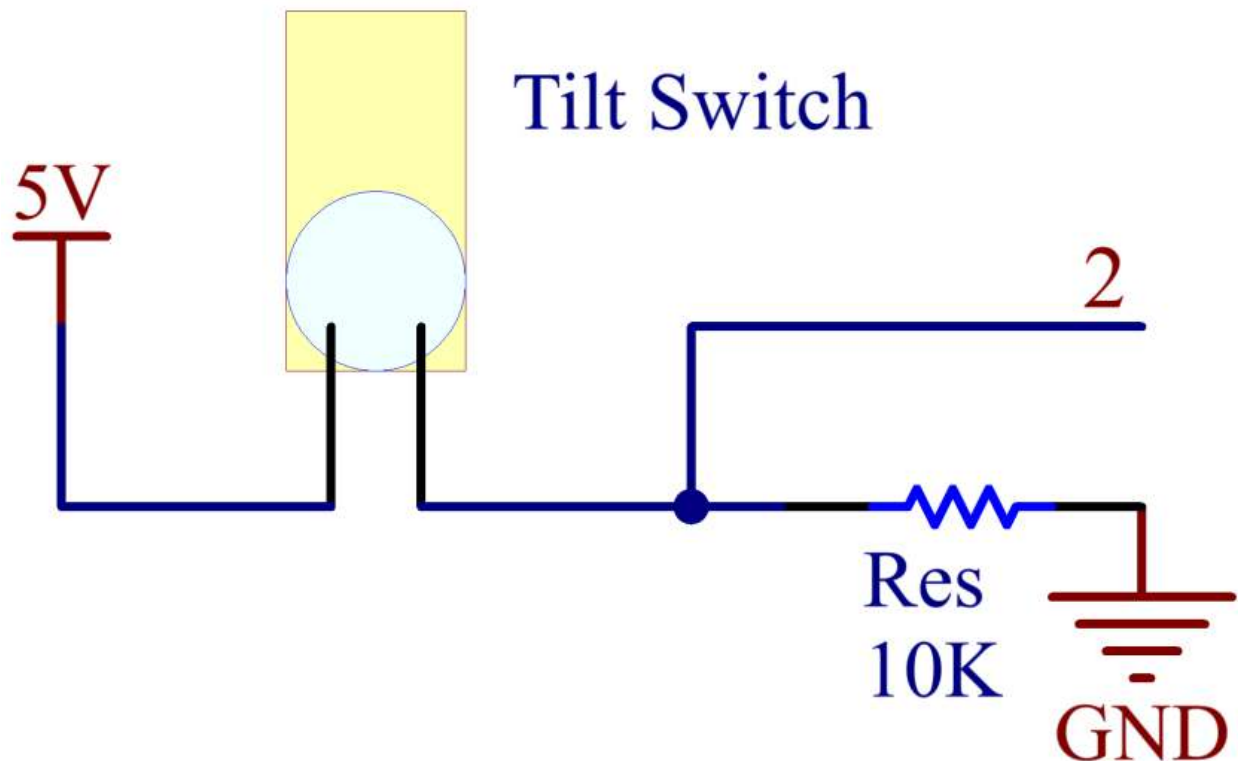
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Resistor*
- *Tilt Switch*

2.27.3 Fritzing Circuit

In this example, digital pin 2 is used to read the signal of Tilt Switch.



2.27.4 Schematic Diagram



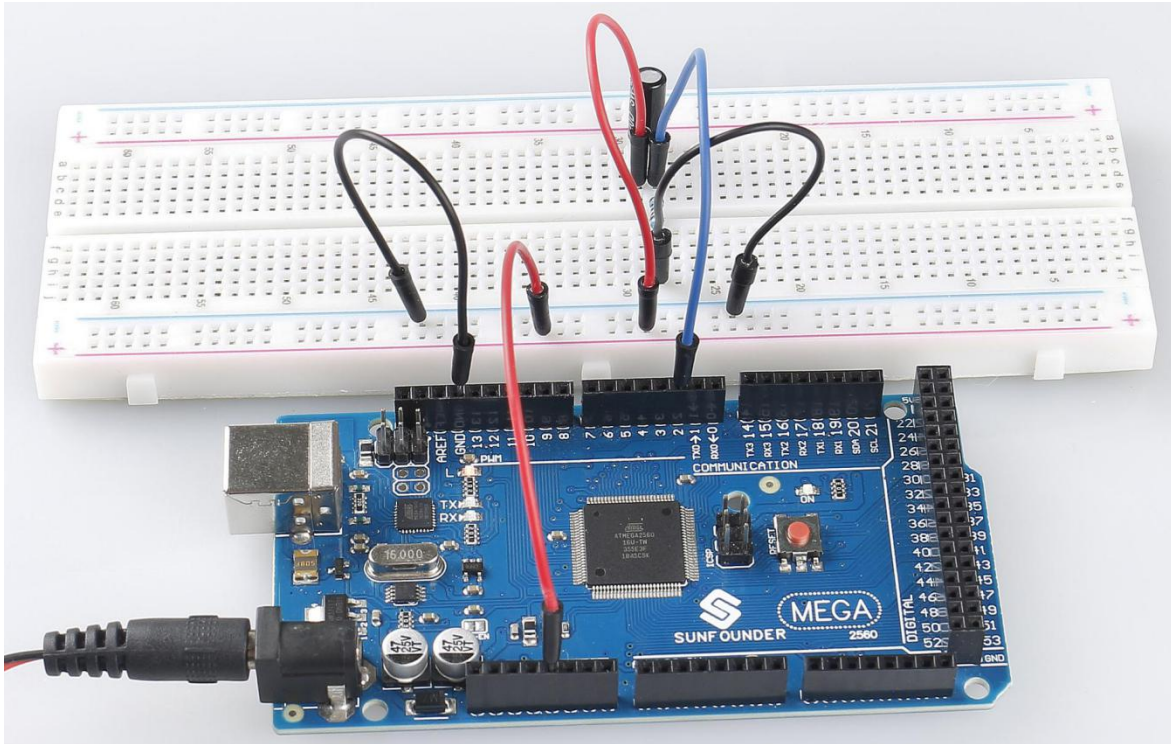
2.27.5 Code

Note:

- You can open the file `2.17_tiltSwitch.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.17_tiltSwitch` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After the codes are uploaded to the Mega2560 board, you can open the serial monitor to see the readings of pins, which displays 1 or 0 when Tilt Switch is vertical (bringing the internal metal ball into contacting with the Wire Pins) or tilted. For detailed explanation of codes, you can turn to [1.4 Digital Read](#).

2.27.6 Phenomenon Picture

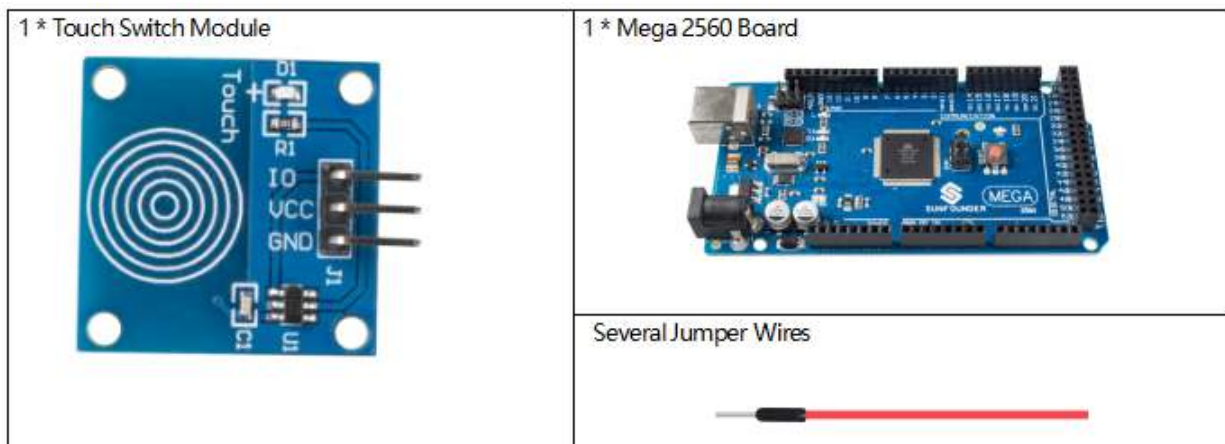


2.28 2.18 Touch Switch Module

2.28.1 Overview

In this lesson, you will learn about touch switch module. It can replace the traditional kinds of switch with these advantages: convenient operation, fine touch sense, precise control and least mechanical wear.

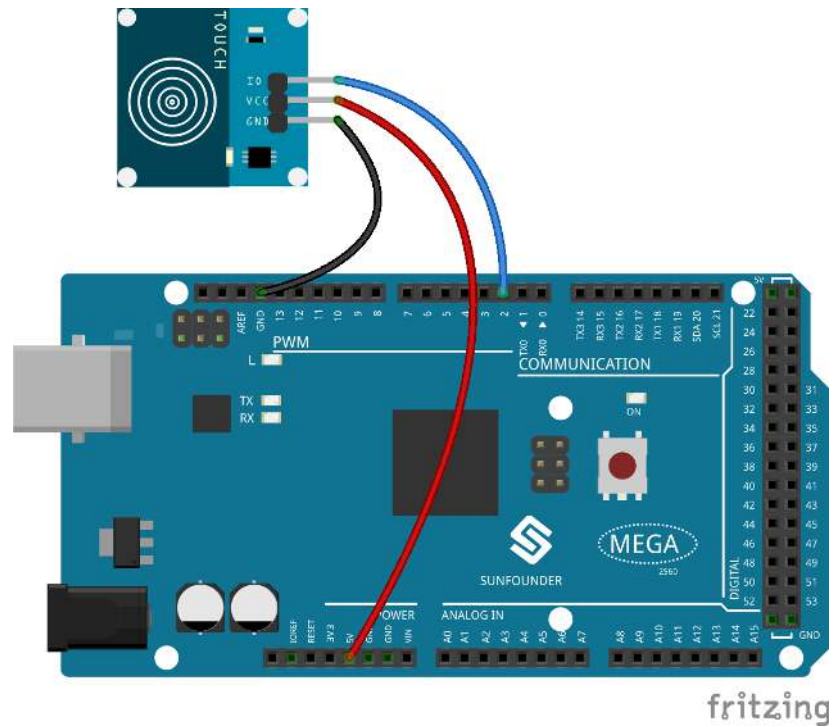
2.28.2 Components Required



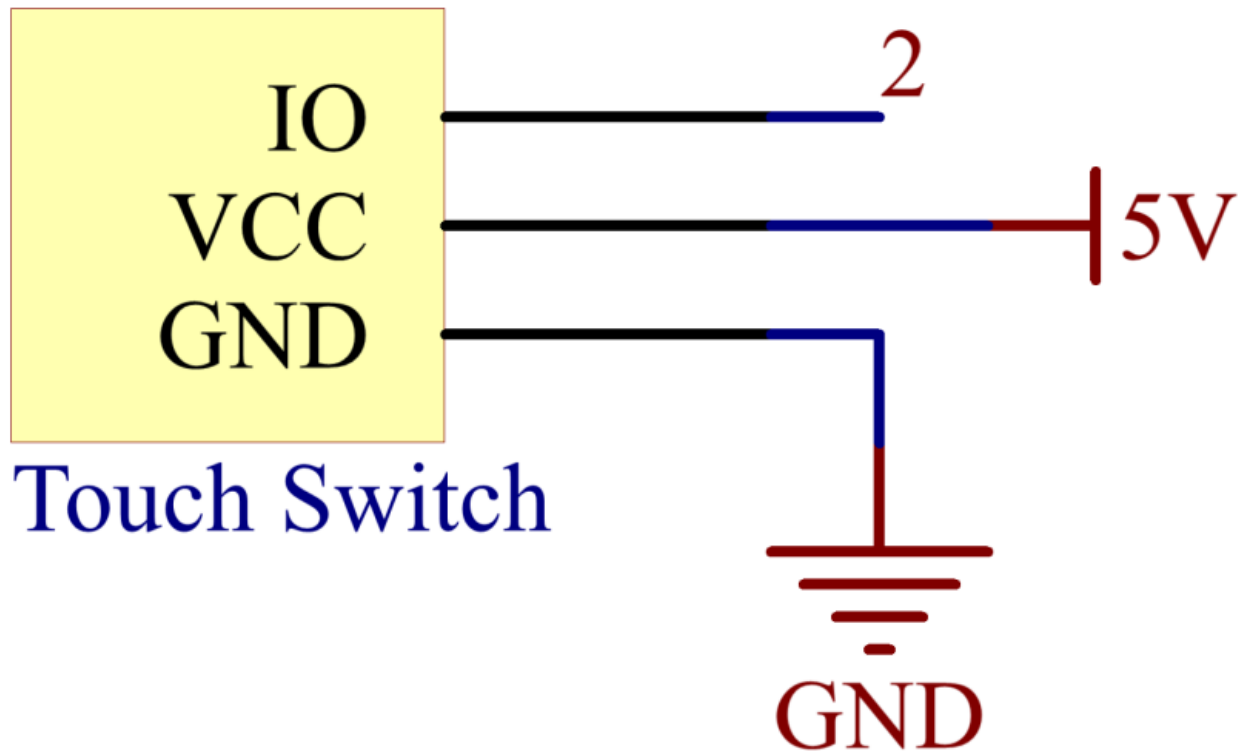
- *SunFounder Mega Board*
- *Jumper Wires*
- *Touch Switch Module*

2.28.3 Fritzing Circuit

In this example, pin 2 is used to read the signal of Touch Switch Module.



2.28.4 Schematic Diagram



2.28.5 Code

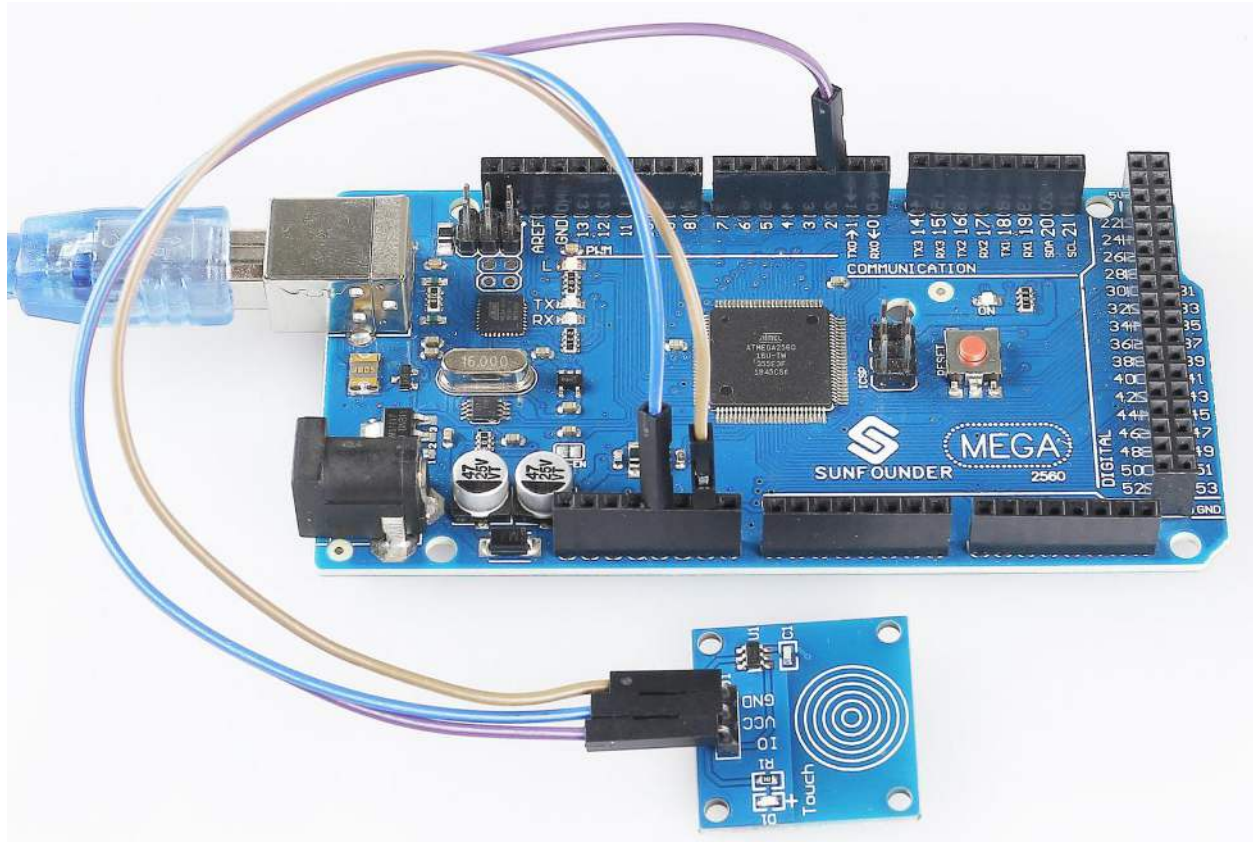
Note:

- You can open the file `2.18_touchSwitch.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.18_touchSwitch` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Or click **Open Code** to open it in [Web Editor](#).
 - Then *Upload the Code* to the board.
-

Uploaded the codes to the Mega2560 board, you can see the readings of pins displaying on the serial monitor.

When your finger tip touches the Touch switch module, 1 will be displayed on the serial monitor; and when you remove your finger, 0 will be displayed. As for the detailed code explanation, you need to turn to [1.4 Digital Read](#).

2.28.6 Phenomenon Picture

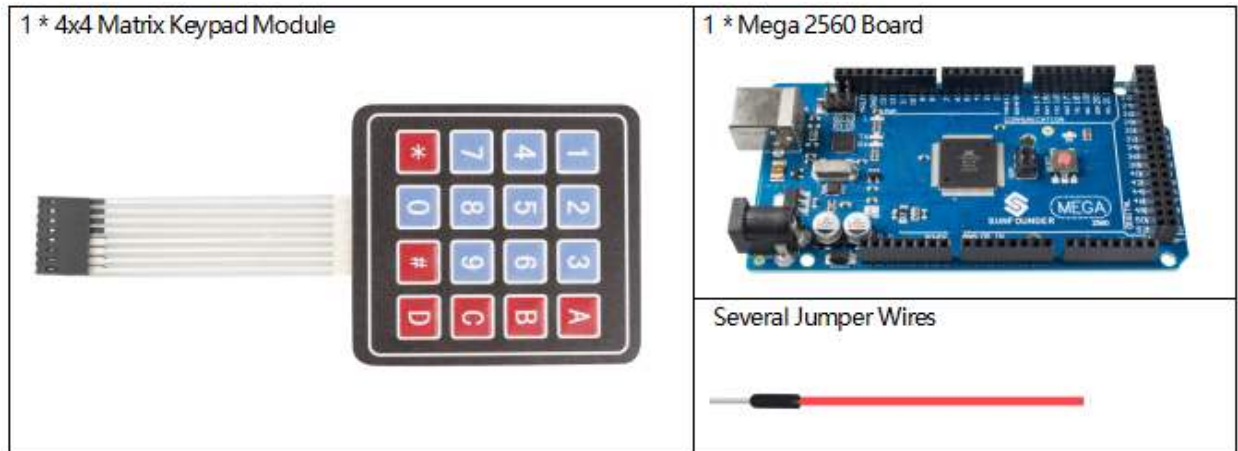


2.29 2.19 Keypad

2.29.1 Overview

In this lesson, you will learn to use Keypad. Keypad can be applied into various kinds of devices, including mobile phone, fax machine, microwave oven and so on. It is commonly used in user input.

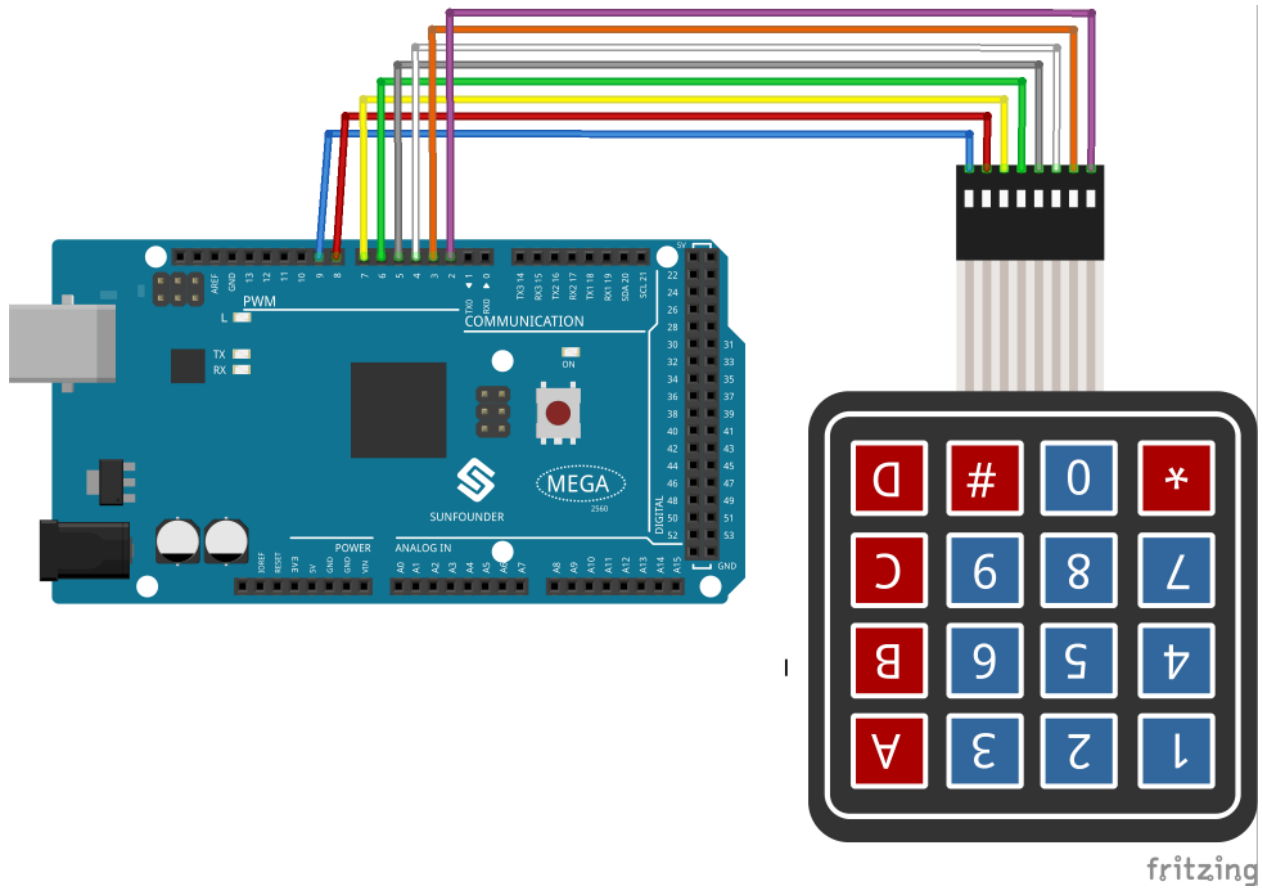
2.29.2 Components Required



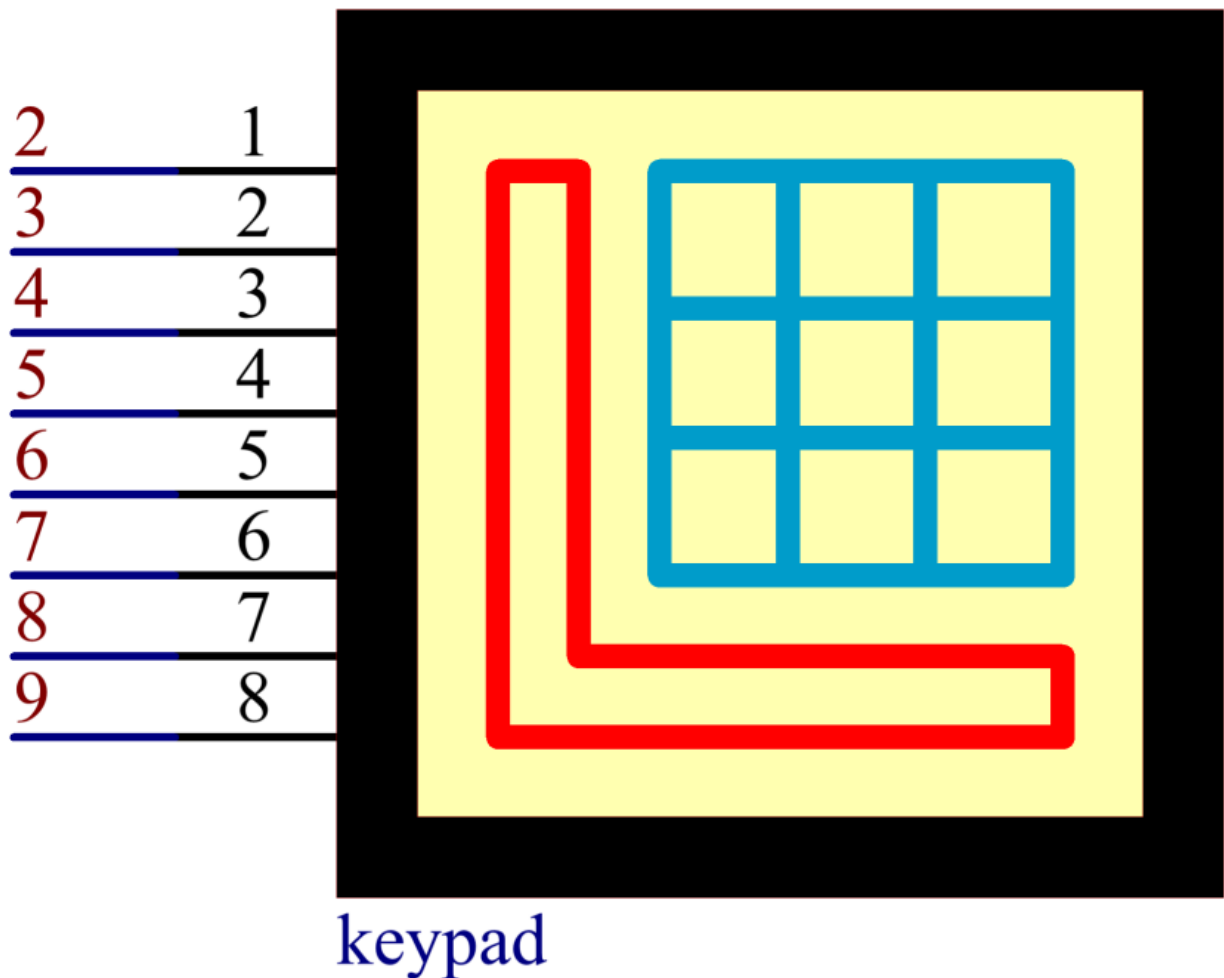
- *SunFounder Mega Board*
- *Jumper Wires*
- *Keypad*

2.29.3 Fritzing Circuit

In this example, we extend the pins 1~8 of Keypad to connect to the digital pins 2~9.



2.29.4 Schematic Diagram



2.29.5 Code

Note:

- You can open the file `2.19_keypad.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.19_keypad` directly.
- Or copy this code into Arduino IDE 1/2.
- Then *Upload the Code* to the board.
- Please make sure you have added the library called `Keypad`, detailed tutorials refer to *Add Libraries*.

After uploading the codes to the Mega2560 board, on the serial monitor, you can see the value of the key currently pressed on the Keypad.

2.29.6 Code Analysis

By calling the Keypad.h library, you can easily use Keypad.

```
#include <Keypad.h>
```

Library Functions

```
Keypad(char *userKeymap, byte *row, byte *col, byte numRows, byte numCols)
```

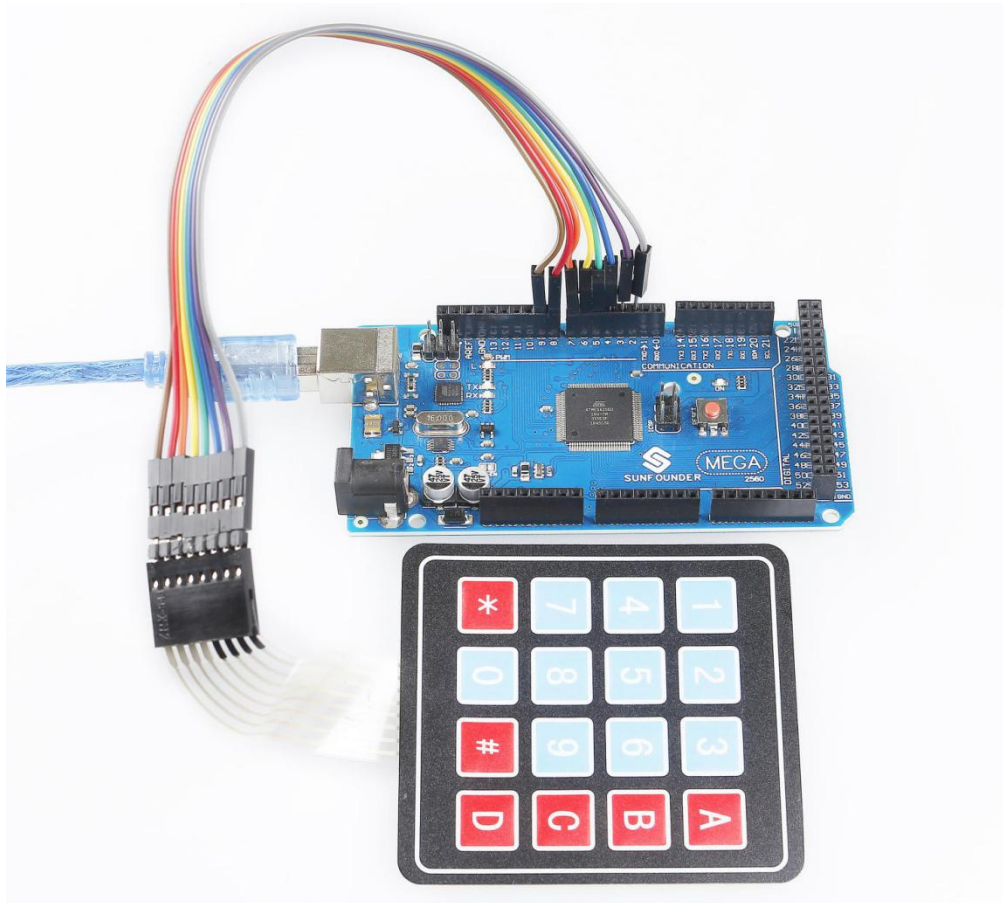
Initializes the internal keymap to be equal to userKeymap.

- userKeymap: The symbols on the buttons of the keypads.
- row, col: Pin configuration.
- numRows, numCols: Keypad sizes.

```
char getKey()
```

Returns the key that is pressed, if any. This function is non-blocking.

2.29.7 Phenomenon Picture







2.30 2.20 IR Receiver Module

2.30.1 Overview

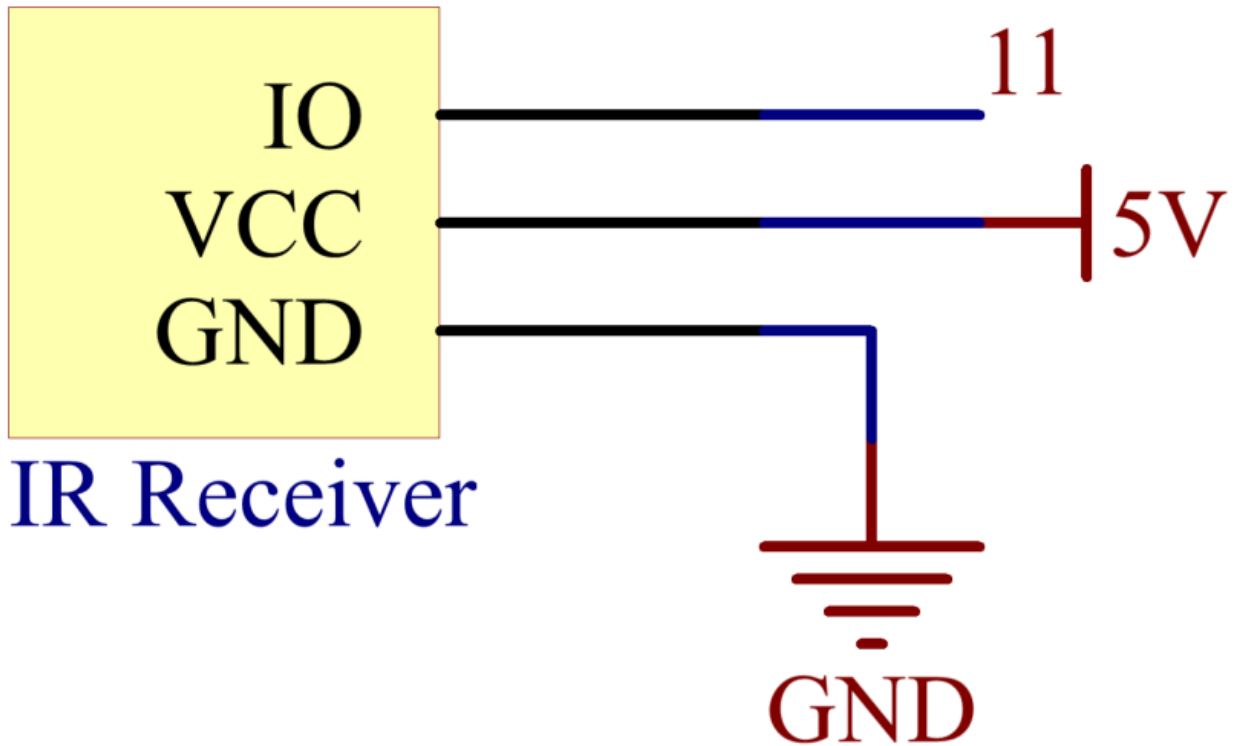
In this lesson, you will learn to use IR Receiver Module. IR Receiver is a component with photocell that is tuned to receive to infrared light. It is almost always used for remote control detection - every TV and DVD player has one of these in the front to receive for the IR signal from the clicker. Inside the remote control is a matching IR LED, which emits IR pulses to tell the TV to turn on, off or change channels.

2.30.2 Components Required

<p>1 * IR Receiver Module</p>  <p>The image shows a black rectangular IR Receiver Module. It has a small antenna on the left side. On the right side, there are three pins labeled DAT, VCC, and GND. The model number HX-M121 is printed at the bottom.</p>	<p>1 * IR Remote Controller</p>  <p>The image shows a standard grey IR remote control with various colored buttons (green, blue, red, yellow, purple) and a numeric keypad.</p>
<p>1 * Mega 2560 Board</p>  <p>The image shows a blue SunFounder Mega 2560 board. It features a USB Type-B port, a DC power jack, and a USB Type-A port. The board is populated with various components like resistors, capacitors, and a microcontroller chip.</p>	<p>Several Jumper Wires</p>  <p>The image shows a single red jumper wire with black plastic insulation on both ends.</p>

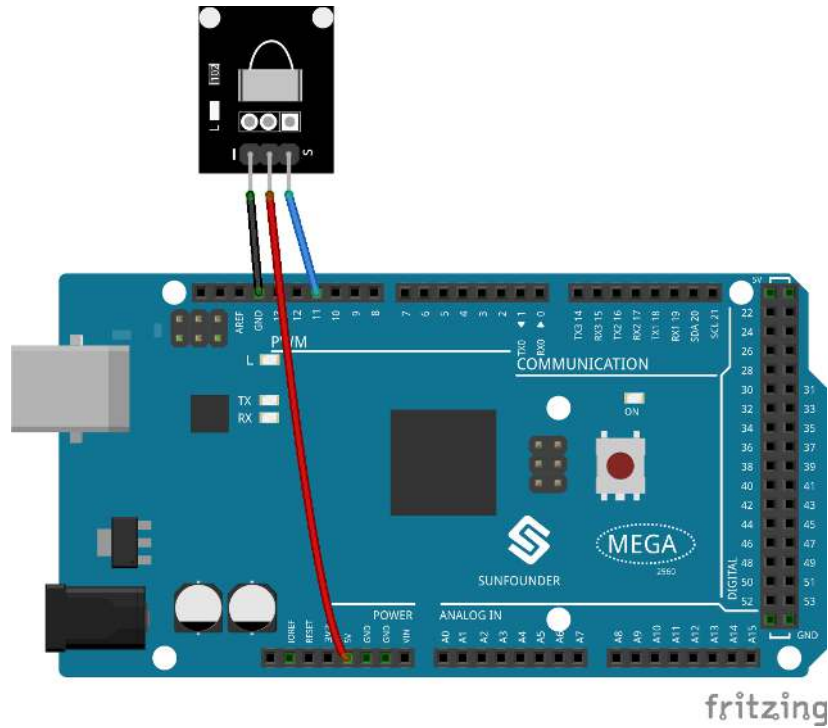
- *SunFounder Mega Board*
- *Jumper Wires*
- *IR Receiver Module*

2.30.3 Schematic Diagram



2.30.4 Fritzing Circuit

In this example, we wire up the left pin (-) of IR Receiver Module to GND, the middle pin to 5V, and the right pin (S) to pin 11.



2.30.5 Code

Note:

- You can open the file `2.20_irReceiver.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.20_irReceiver` directly.
- Or copy this code into Arduino IDE 1/2.
- Then *Upload the Code* to the board.
- Please make sure you have added the library called `IRremote`, detailed tutorials refer to *Add Libraries*.

After uploading the codes to the Mega2560 board, you can see that the current value of the pressed button of IR Remote Controller displays on the serial monitor.

2.30.6 Code Analysis

There are two important parts to notice in this program.

1. The code uses an extra file `decodeKeyValue.ino` to decode the values in class `decode_result` into key value. The file will be opened together with the main file.
2. IR Remote function is achieved by calling `IRremote.h` library related functions.

```
#include <IRremote.h>
```

Library Functions

```
IRrecv(int recvpin)
```

Create IRrecv object to control a IR Receiver module.

```
decode_result
```

In this kit, results are usually 8-digit hexadecimal numbers starting with 00FF. You can check decodeKeyValue.ino file in the sample file.

```
void enableIRIn()
```

Initialize the IR receiver module.

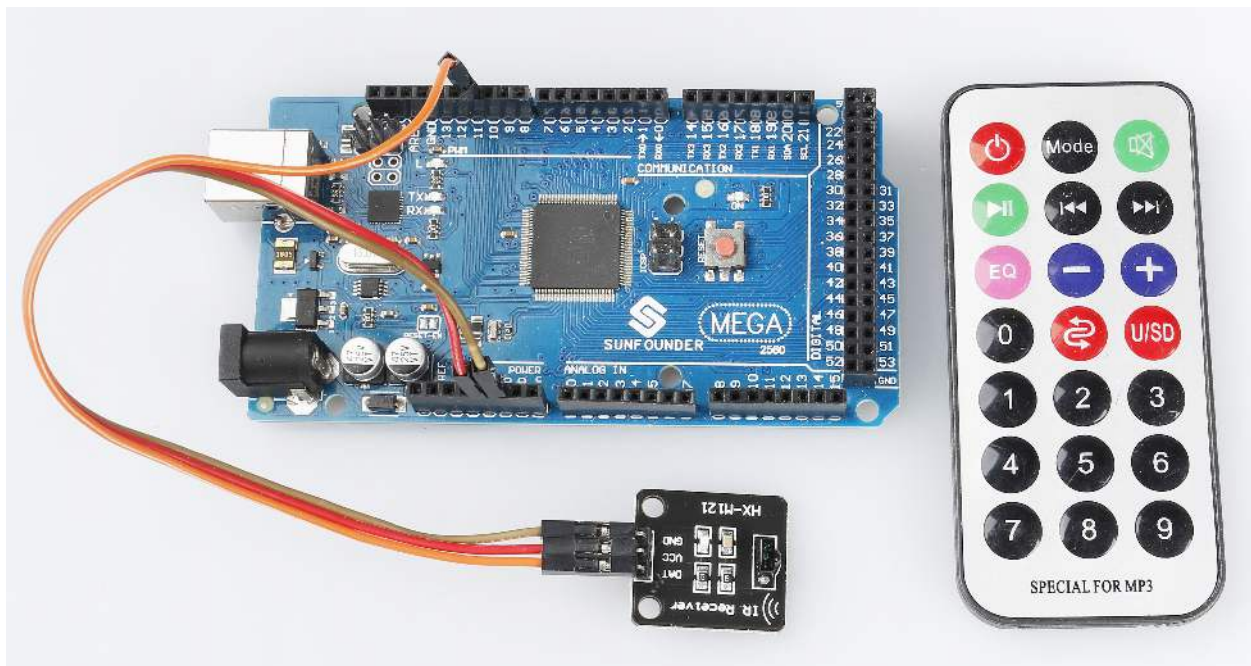
```
int decode(decode_results *results);
```

Decodes the received IR message. Returns 0 if no data ready, 1 if data ready. Results of decoding are stored in results.

```
void resume()
```

Restart for receiving an other value.

2.30.7 Phenomenon Picture

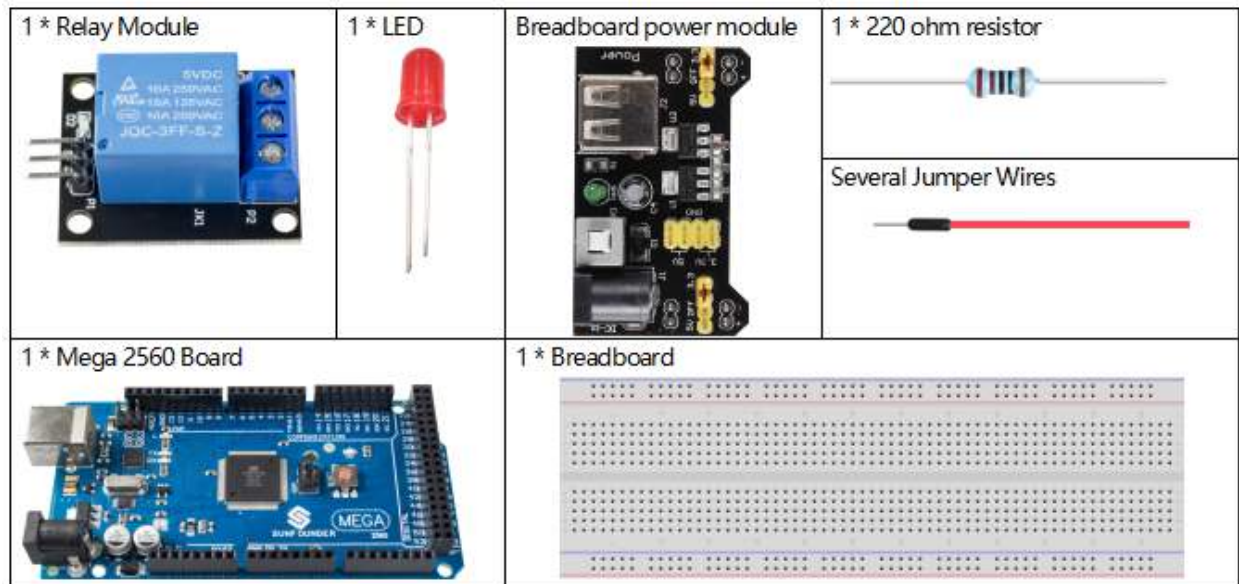


2.31 2.21 Relay Module

2.31.1 Overview

In this lesson, you will learn about Relay Module.

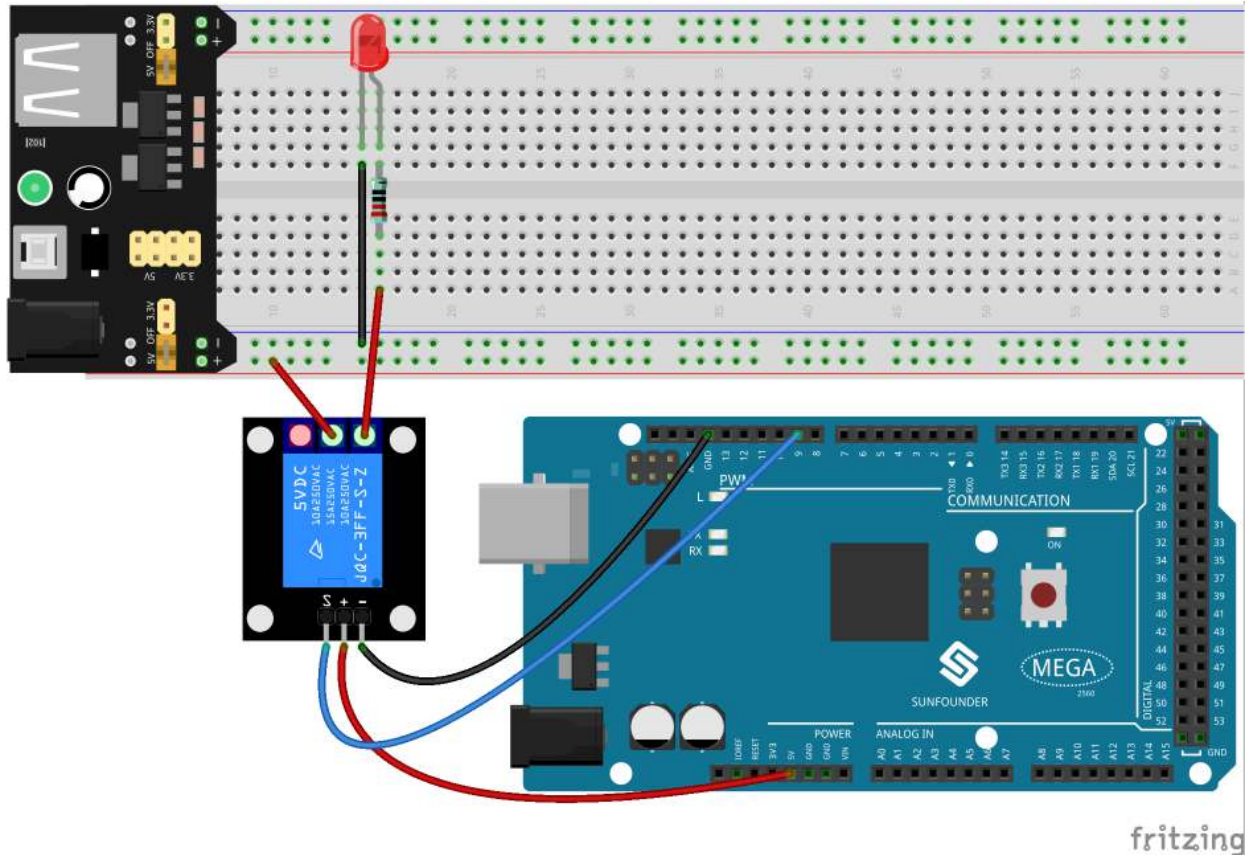
2.31.2 Components Required



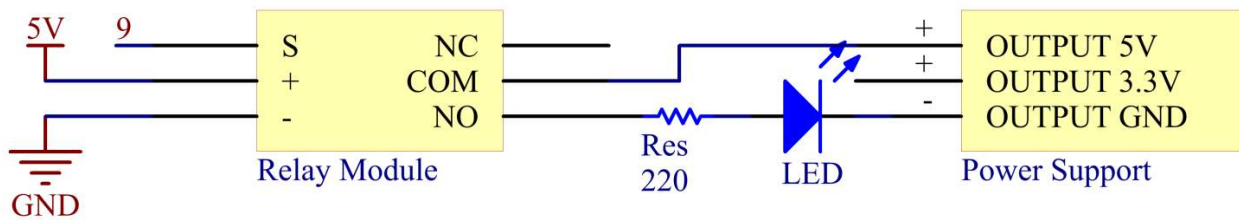
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *LED*
- *Resistor*
- *Power Supply Module*

2.31.3 Fritzing Circuit

In this example, we use Power Supply Module to power the loaduse LED as an example.



2.31.4 Schematic Diagram



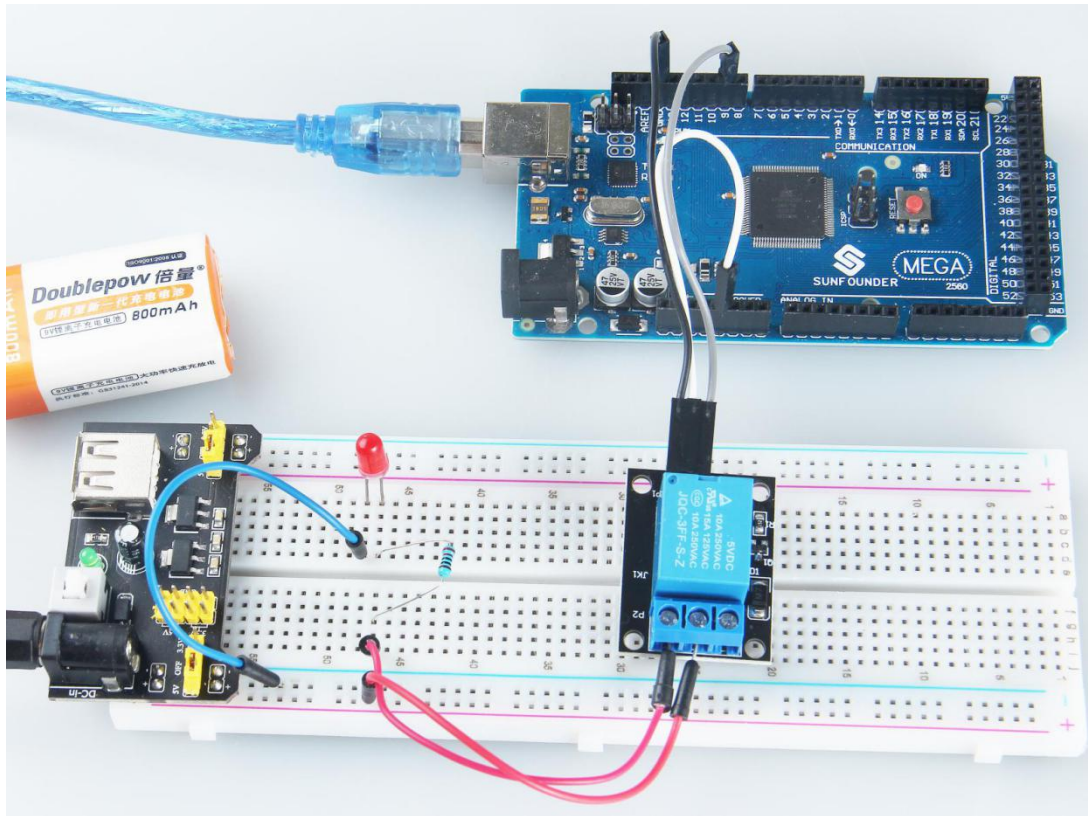
2.31.5 Code

Note:

- You can open the file `2.21_relayModule.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.21_relayModule` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

Once the codes are uploaded to the Mega2560 board, you can see that the Relay Module controls the closing and breaking of the external circuit, which will change its working state a second. For detailed code explanation, refer to *1.2 Digital Write*.

2.31.6 Phenomenon Picture

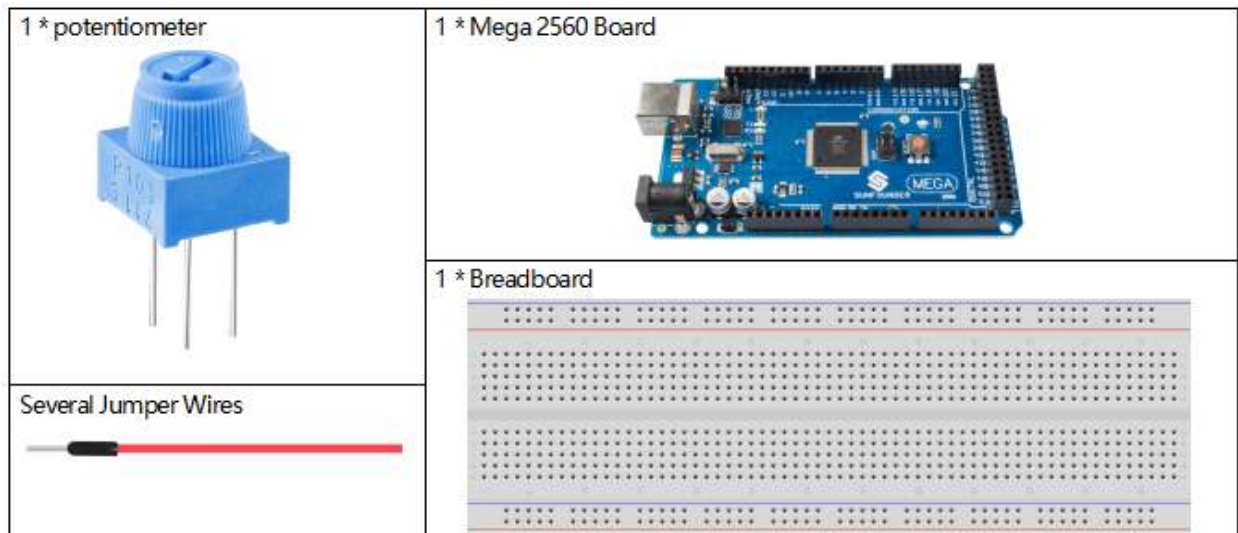


2.32 2.22 Potentiometer

2.32.1 Overview

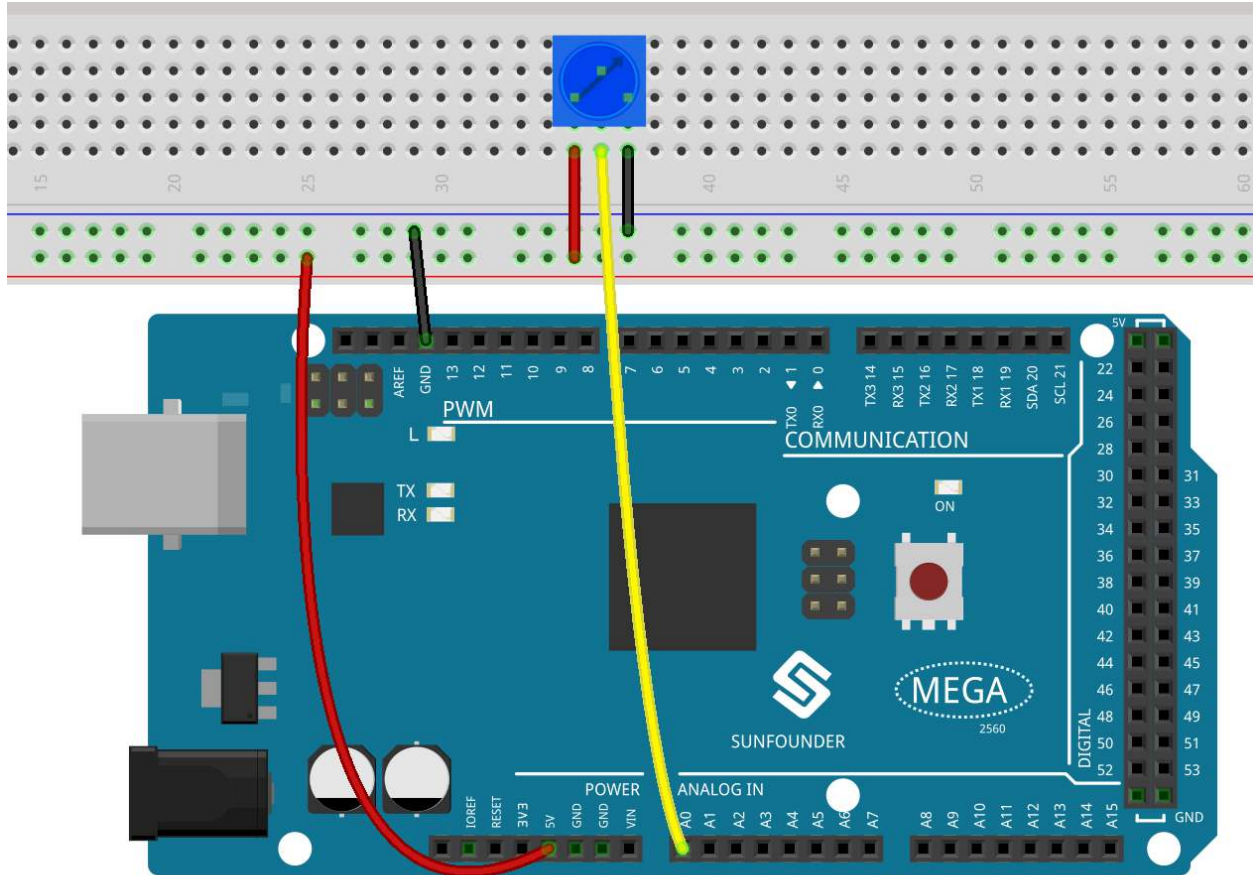
In this lesson, you will learn about Potentiometer. Potentiometer is a resistor component with 3 terminals and its resistance value can be adjusted according to some regular variation.

2.32.2 Components Required



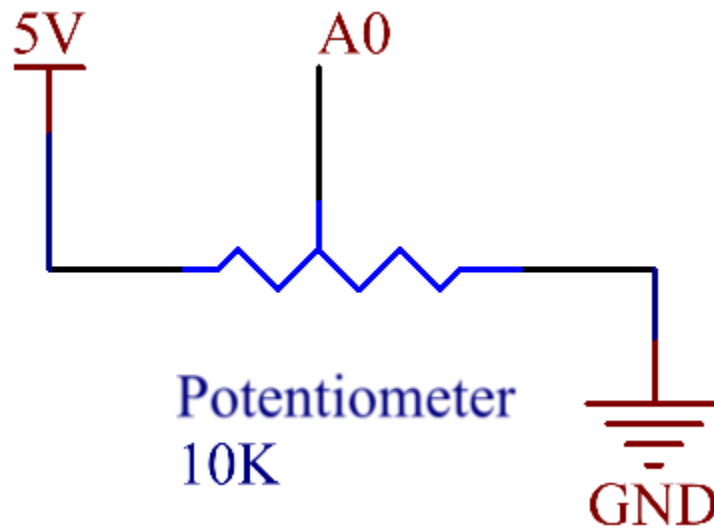
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Potentiometer*

2.32.3 Fritzing Circuit



In this example, we use the analog pin (A0) to read the value of the potentiometer. By rotating the axis of the potentiometer, you can change the distribution of resistance among these three pins, changing the voltage on the middle pin. When the resistance between the middle and a outside pin connected to 5V is close to zero (and the resistance between the middle and the other outside pin is close to 10k), the voltage at the middle pin is close to 5 V. The reverse operation (the resistance between the middle and a outside pin connected to 5V is close to 10k) will make the voltage at the middle pin be close to 0V.

2.32.4 Schematic Diagram



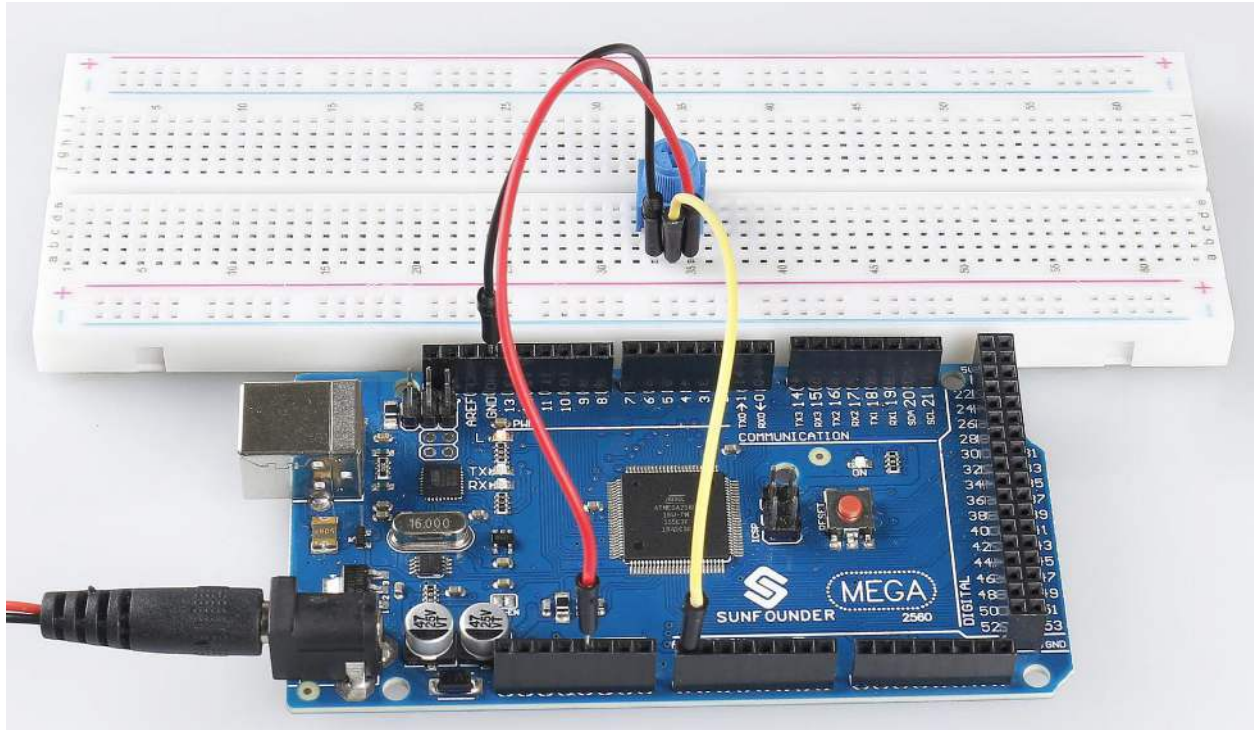
2.32.5 Code

Note:

- You can open the file `2.22_potentiometer.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.22_potentiometer` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After uploading the codes to the Mega2560 board, you can open the serial monitor to see the reading value of the pin. When rotating the axis of the potentiometer, the serial port monitor will print the value 0~1023. For the detailed explanation of code, turn to check [1.5 Analog Read](#).

2.32.6 Phenomenon Picture

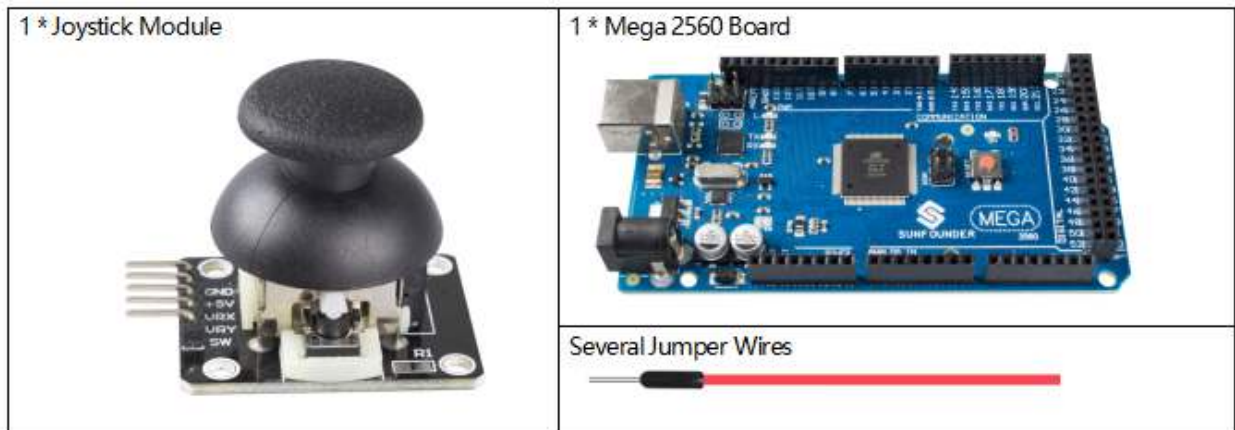


2.33 2.23 Joystick Module

2.33.1 Overview

In this lesson, you will learn something about Joystick. The basic idea of a joystick is to translate the movement of a stick into electronic information that a computer can process. It can be applied to work as the controller of devices, such as robot.

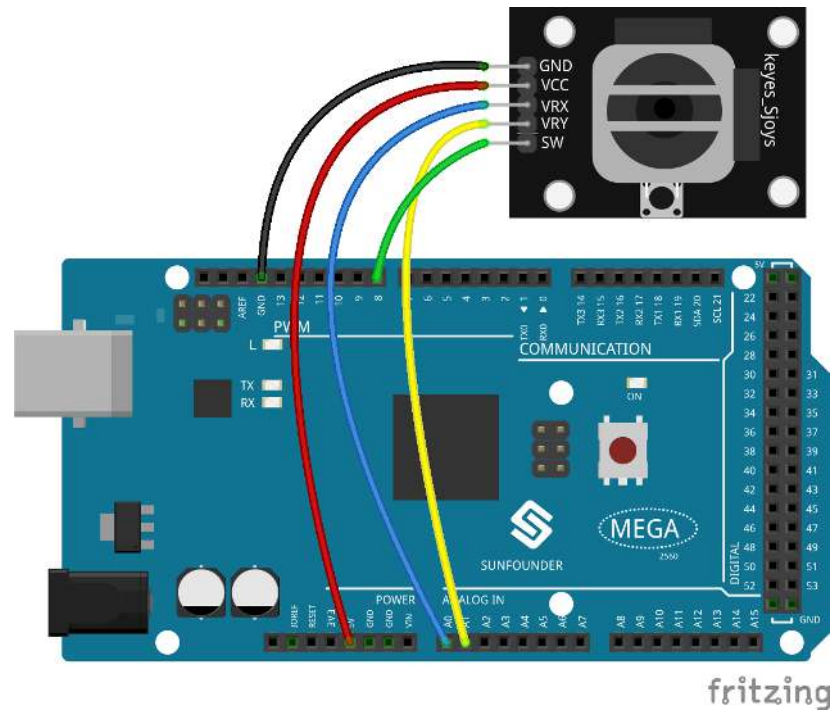
2.33.2 Components Required



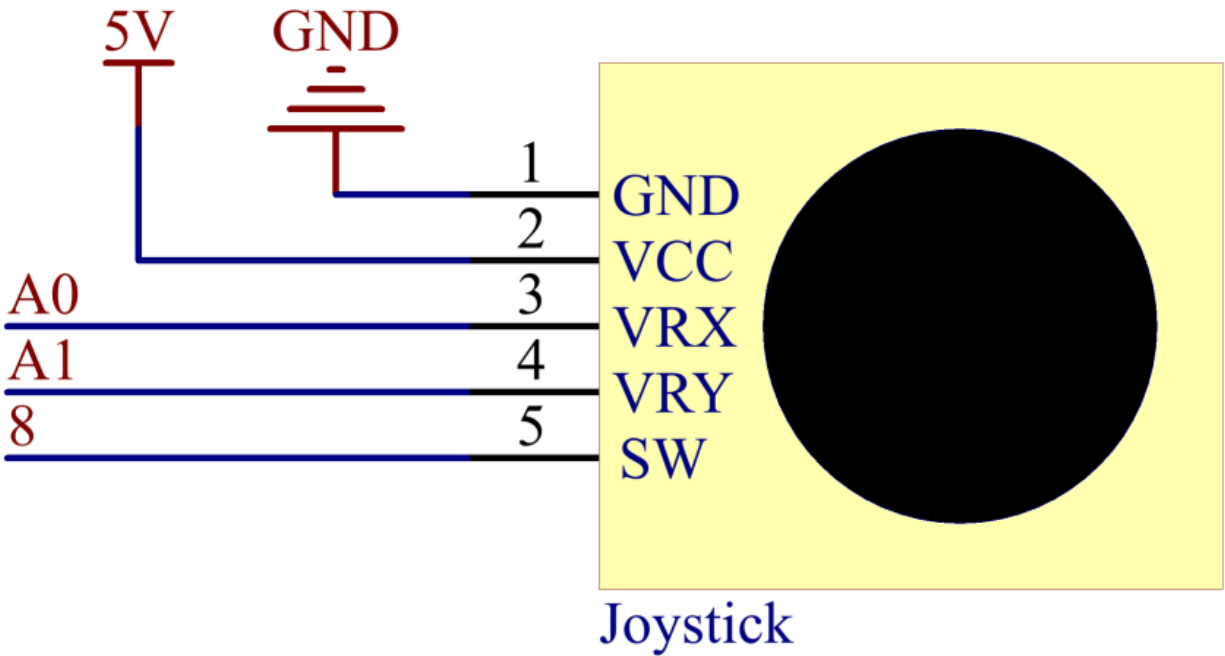
- *SunFounder Mega Board*
- *Jumper Wires*
- *Joystick Module*

2.33.3 Fritzing Circuit

In this example, we get the GND of the Joystick extended to connect with GND, VCC with 5V, VRX with pin A0. After that, we make VRY connect with pin A1, SW connect with pin 8.



2.33.4 Schematic Diagram



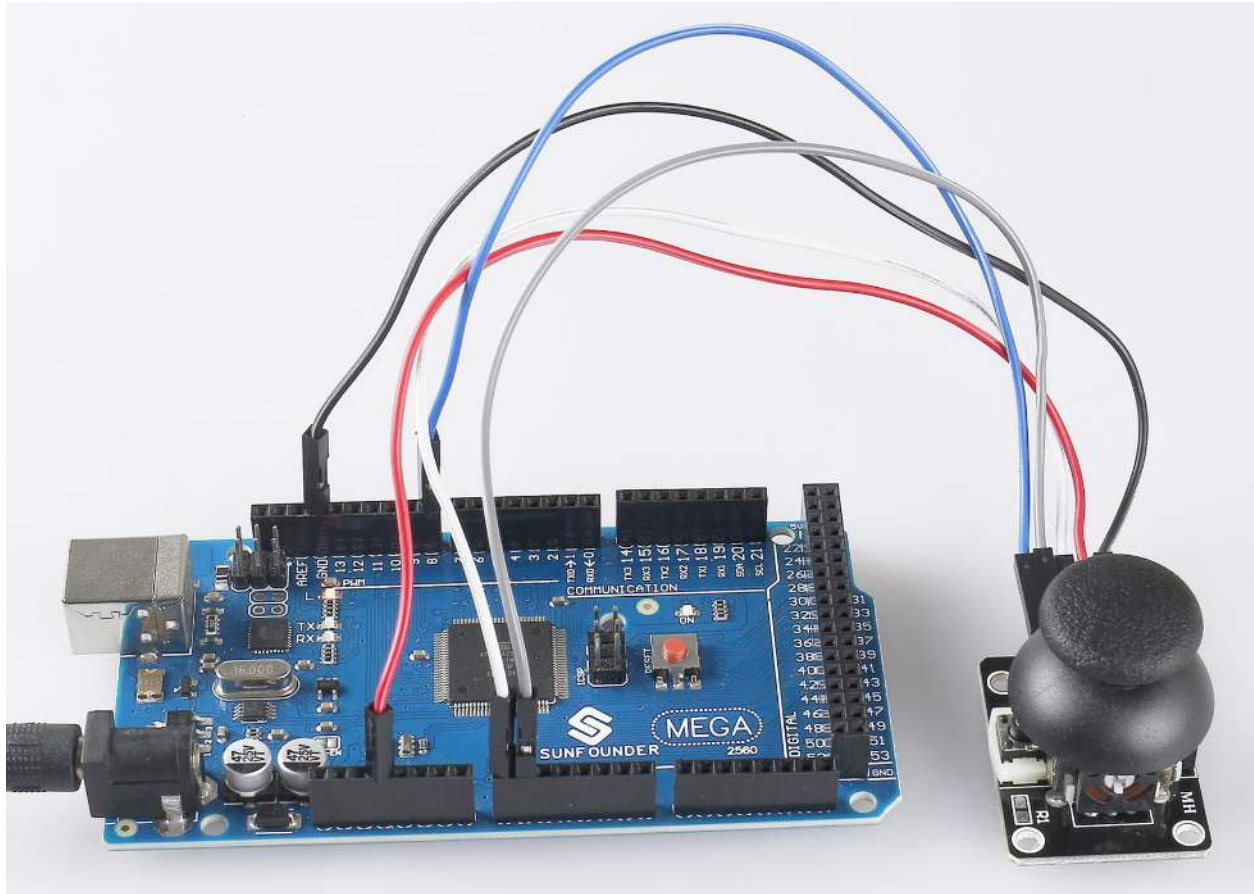
2.33.5 Code

Note:

- You can open the file `2.23_joystick.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.23_joystick` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

Uploaded the codes to the Mega2560 board, you can open the serial monitor to see readings on the X-axis and Y-axis of Joystick, as well as the button status of Z-axis. The values of the X-axis and Y-axis are the analog values, which vary within the range 0~1023. The Z-axis shows numerical value and the state is either 1 or 0. Refer to [1.5 Analog Read](#) and [1.4 Digital Read](#) to check the code explanation.

2.33.6 Phenomenon Picture

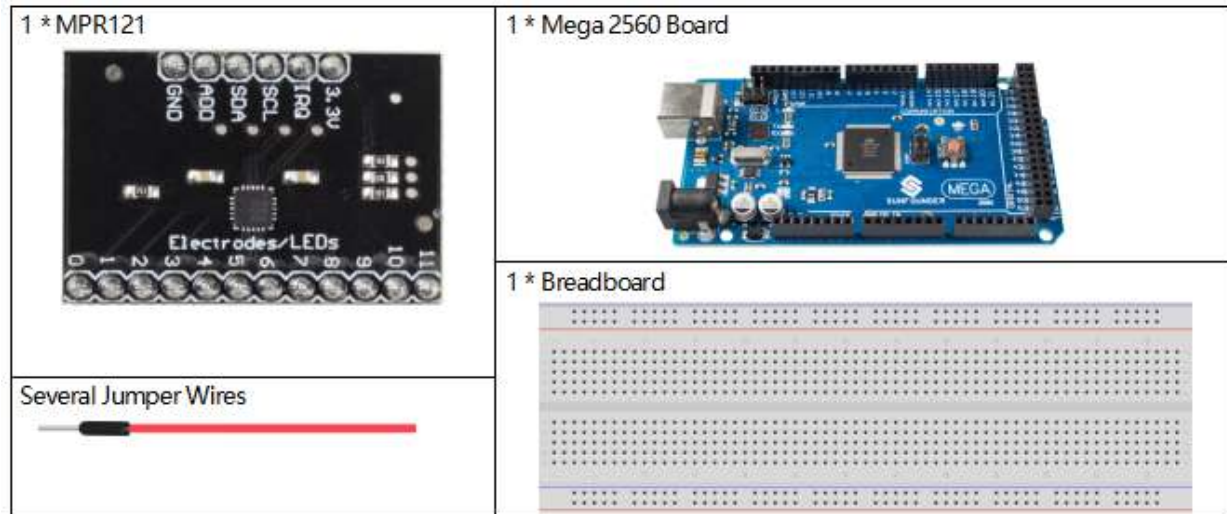


2.34 2.24 MPR121 Module

2.34.1 Overview

In this lesson, you will learn how to use MPR121. It's a good option when you want to add a lot of touch switches to your project. The electrode of MPR121 can be extended with a conductor. If you connect a wire to a banana, you can turn the banana into a touch switch, thus realizing projects such as fruit piano.

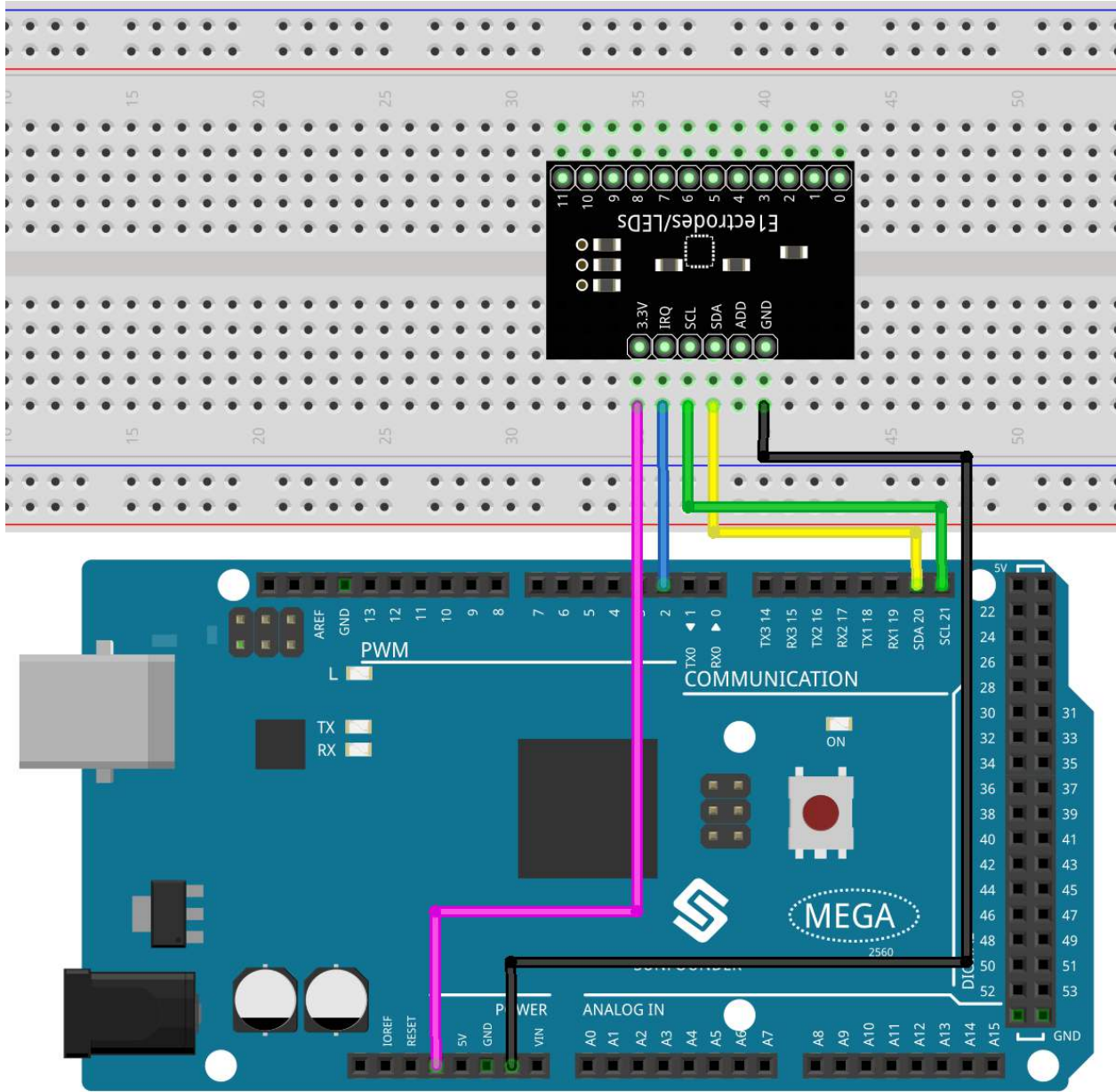
2.34.2 Components Required



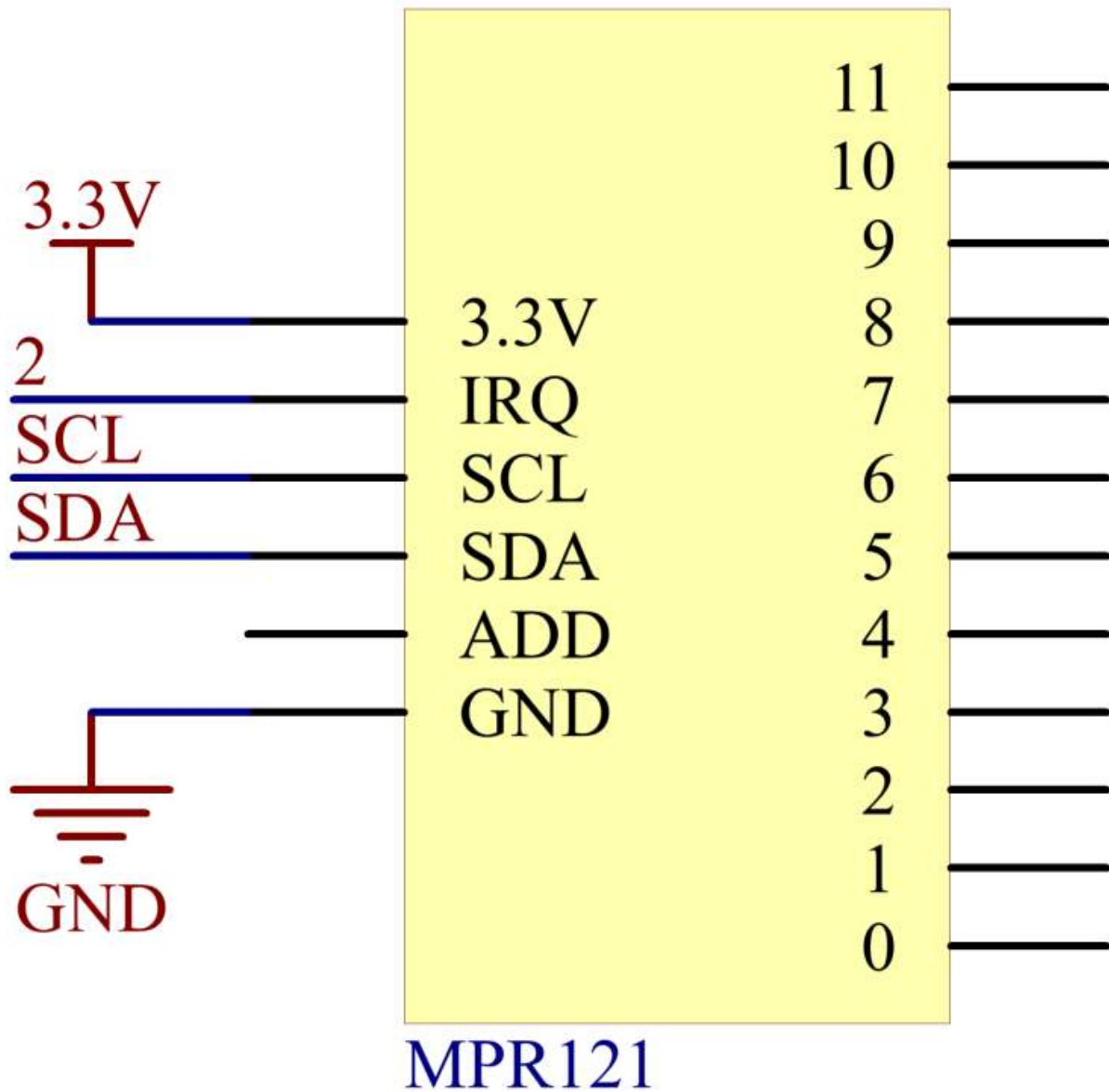
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *MPR121*

2.34.3 Fritzing Circuit

In this example, we insert MPR121 into the breadboard. Get the GND of MPR121 connected to GND, 3.3V to 3V3, IRQ to the digital pin 2, SCL to the pin SCL(21), and SDA to the pin SDA(20). There are 12 electrodes for touch sensing. Note: MPR121 is powered by 3.3V, not 5V.



2.34.4 Schematic Diagram



2.34.5 Code

Note:

- You can open the file `2.24_mpr121.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.24_mpr121` directly.
- Or copy this code into Arduino IDE 1/2.
- Then *Upload the Code* to the board.
- Please make sure you have added the library called `MPR121_JM`, detailed tutorials refer to *Add Libraries*.

After uploading the codes to the Mega2560 board, the touch state of pins of MPR121 1and0will be recorded in a 12 - bit boolean type of array that will be printed on the serial monitor.

2.34.6 Code Analysis

The function of the module is included in the library MPR121_JM.h.

```
#include <MPR121_JM.h>
```

2.34.7 Library Functions

```
MPR121(int irqpin, uint8_t touThresh, uint8_t relThresh)
```

Creates a new instance of the MPR121.

- `irqpin`: the interrupt request pin.
- `touThresh`: Touch thresholdthe electrode is recognized as a threshold of the Touch state.
- `relThresh`: Release thresholdthe electrode is recognized as a threshold of the Release state.

The range of the electrode data value is 0~255. For typical touch application, the threshold value can be in range 0x05~0x30 for example. The smaller the value, the more sensitive it is. The setting of the threshold is depended on the actual application. Typically the touch threshold is a little bigger than the release threshold to touch debounce and hysteresis.

```
void mpr121_setup()
```

Setup MPR121 module.

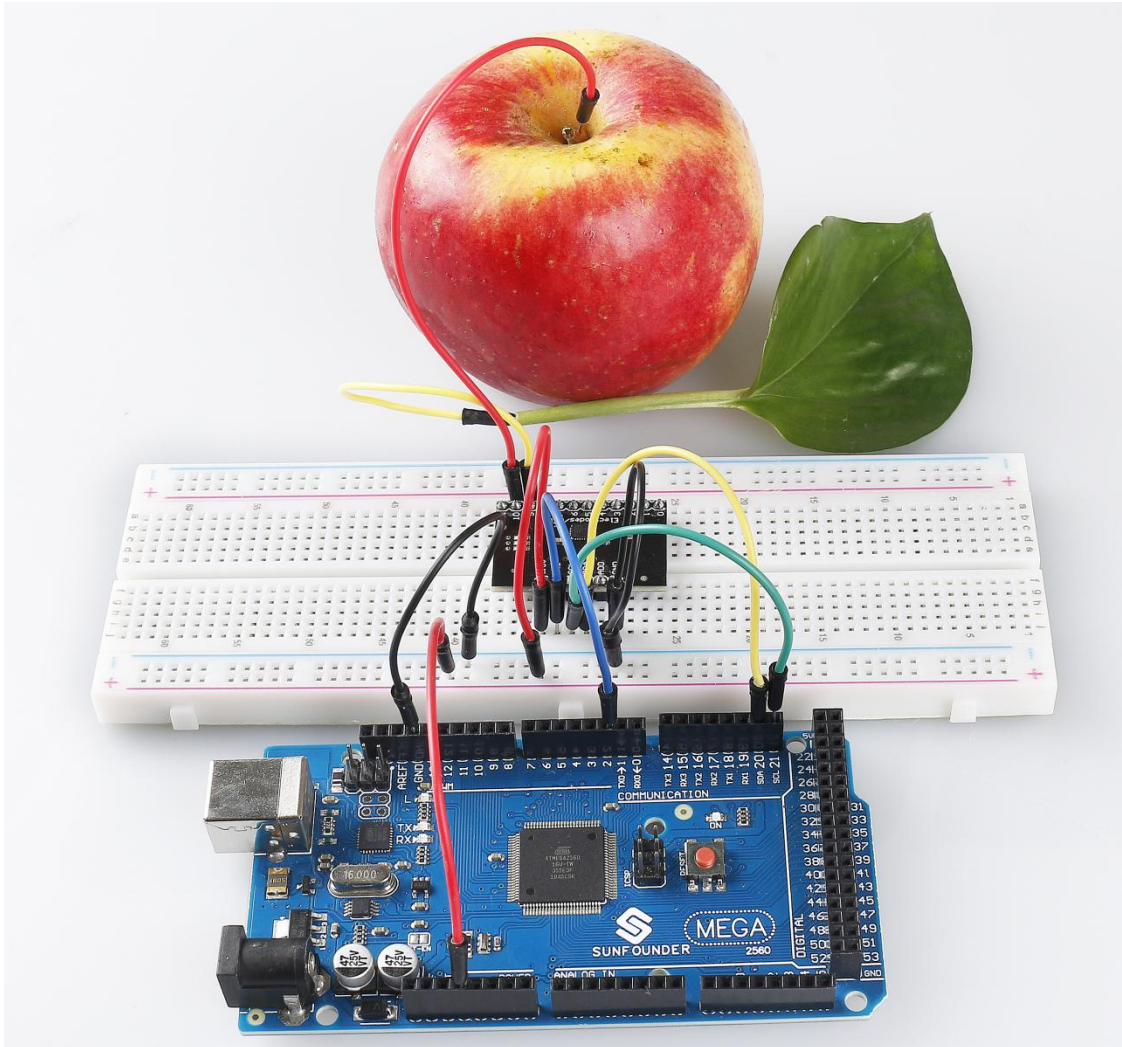
```
bool checkInterrupt()
```

Make interrupt judgment, when the electrode state changes, the function returns a Boolean value 0.

```
uint16_t readTouchInputs()
```

The touch state of the electrode produces a Boolean value. The function accumulates the Boolean values generated by all the electrodes in sequence into a 12-bit binary number as the return value. If the first and eleventh electrodes are touched, 100000000010 is returned.

2.34.8 Phenomenon Picture

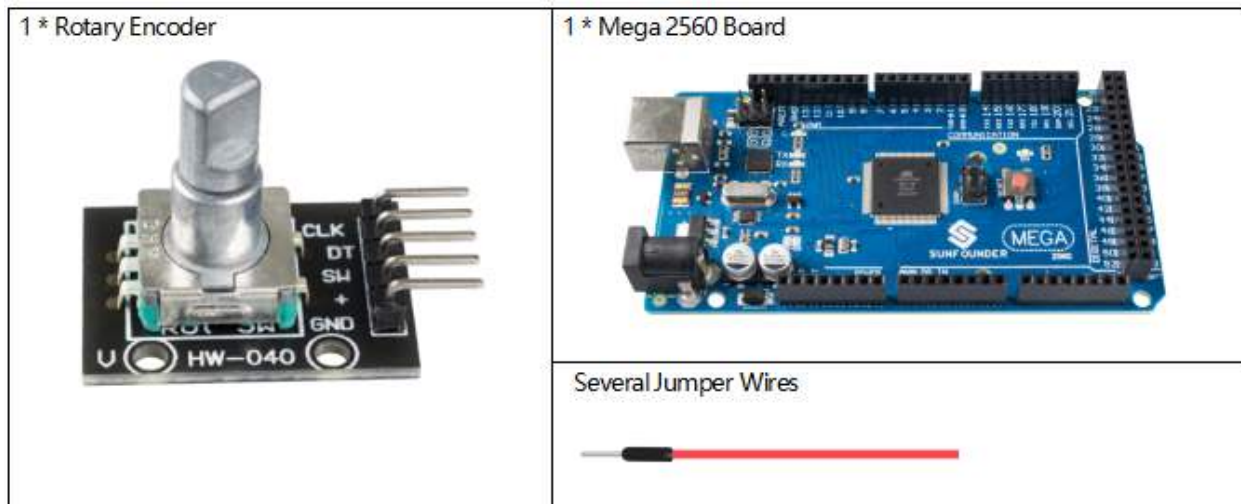


2.35 2.25 Rotary Encoder Module

2.35.1 Overview

In this lesson, you will learn about Rotary Encoder. A rotary encoder is an electronic switch with a set of regular pulses in strictly timing sequence. When used with IC, it can achieve increment, decrement, page turning and other operations such as mouse scrolling, menu selection, and so on.

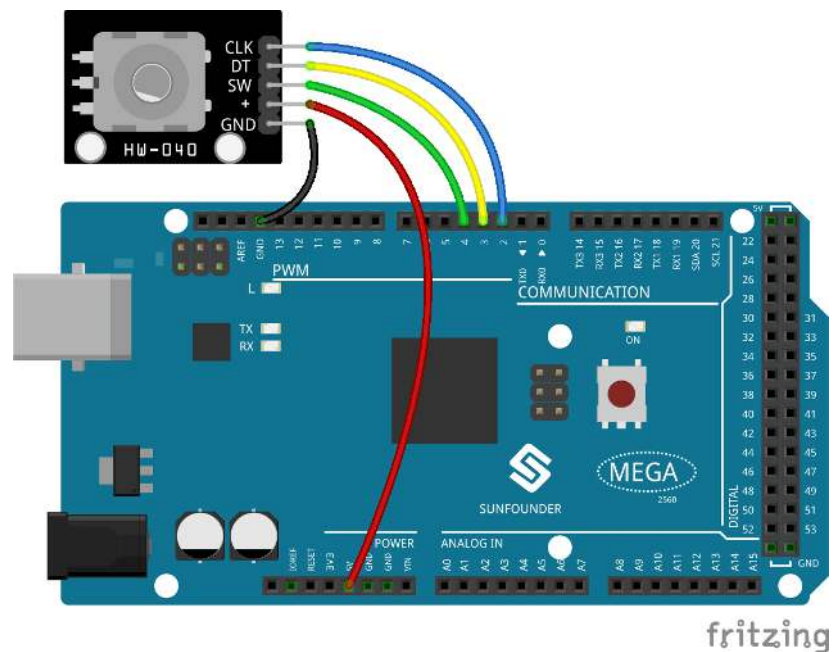
2.35.2 Components Required



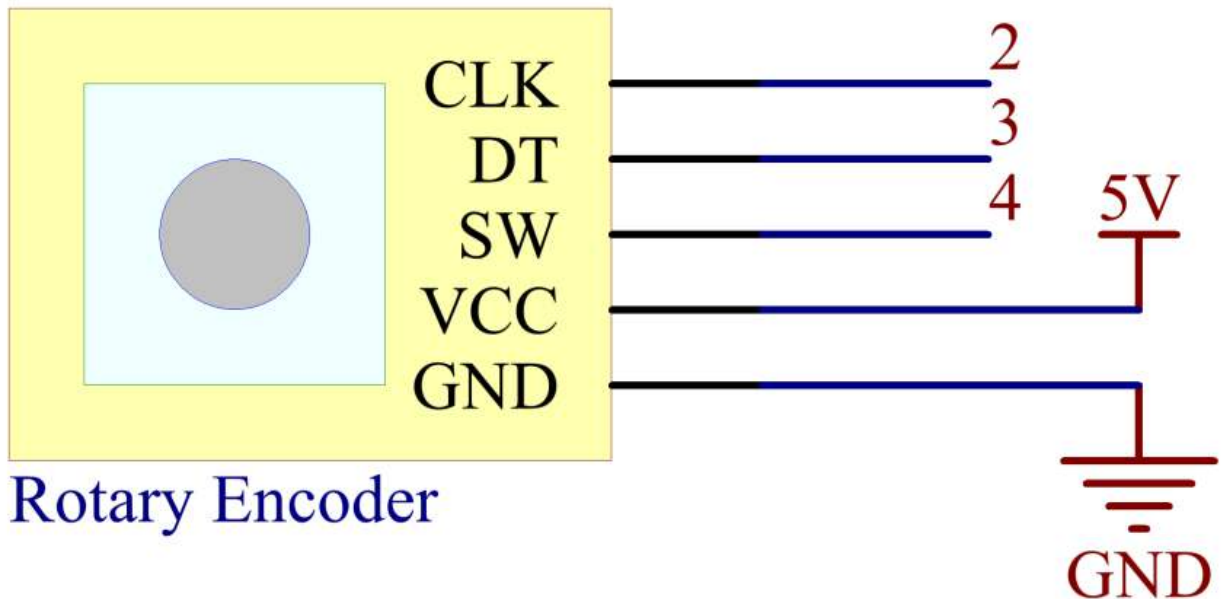
- *SunFounder Mega Board*
- *Jumper Wires*
- *Rotary Encoder Module*

2.35.3 Fritzing Circuit

In this example, we can connect the Rotary Encoder pin directly to the Mega 2560 Board pin, connect the GND of the Rotary Encoder to GND, + to 5V, SW to digital pin 4, DT to digital pin 3, and CLK to digital pin 2.



2.35.4 Schematic Diagram



2.35.5 Code

Note:

- You can open the file `2.25_rotaryEncoder.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.25_rotaryEncoder` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

You will see the angular displacement of the rotary encoder printed on Serial Monitor. When you turn the rotary encoder clockwise, the angular displacement is increased; when turn it counterclockwise, the displacement is decreased. If you press the switch on the rotary encoder, the readings will return to zero.

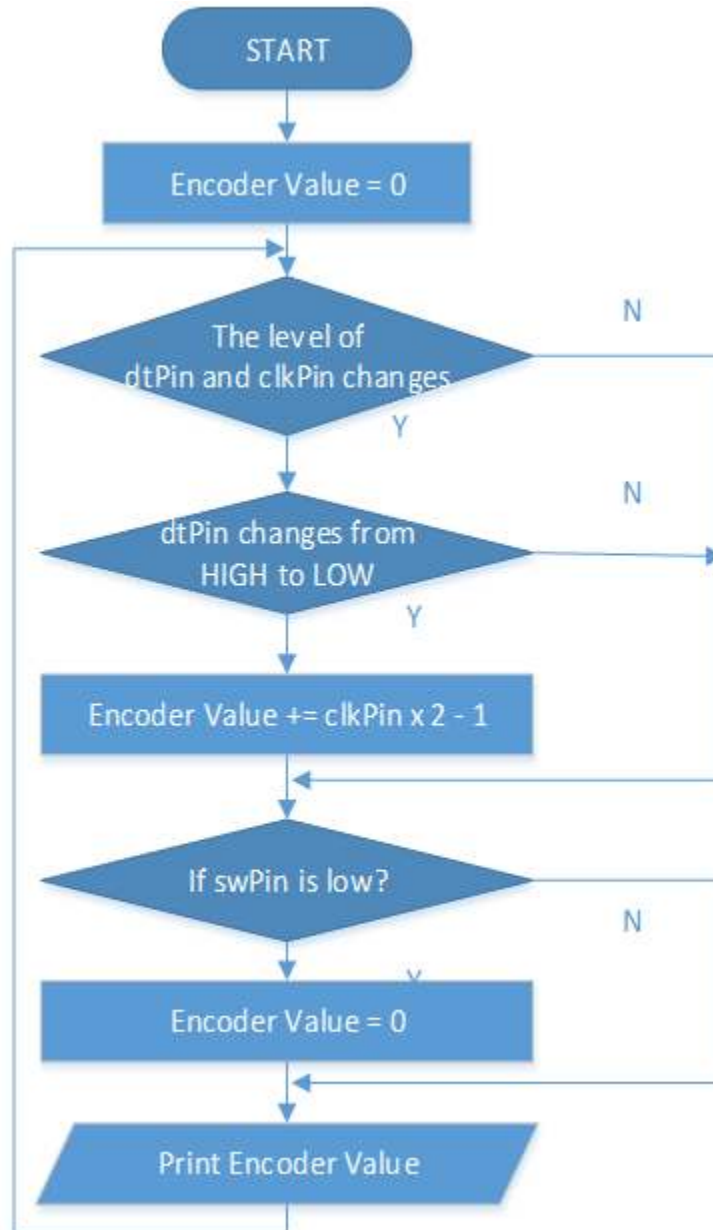
2.35.6 Code Analysis

When Rotary Encoder is used, the following situations of pin level will occur.

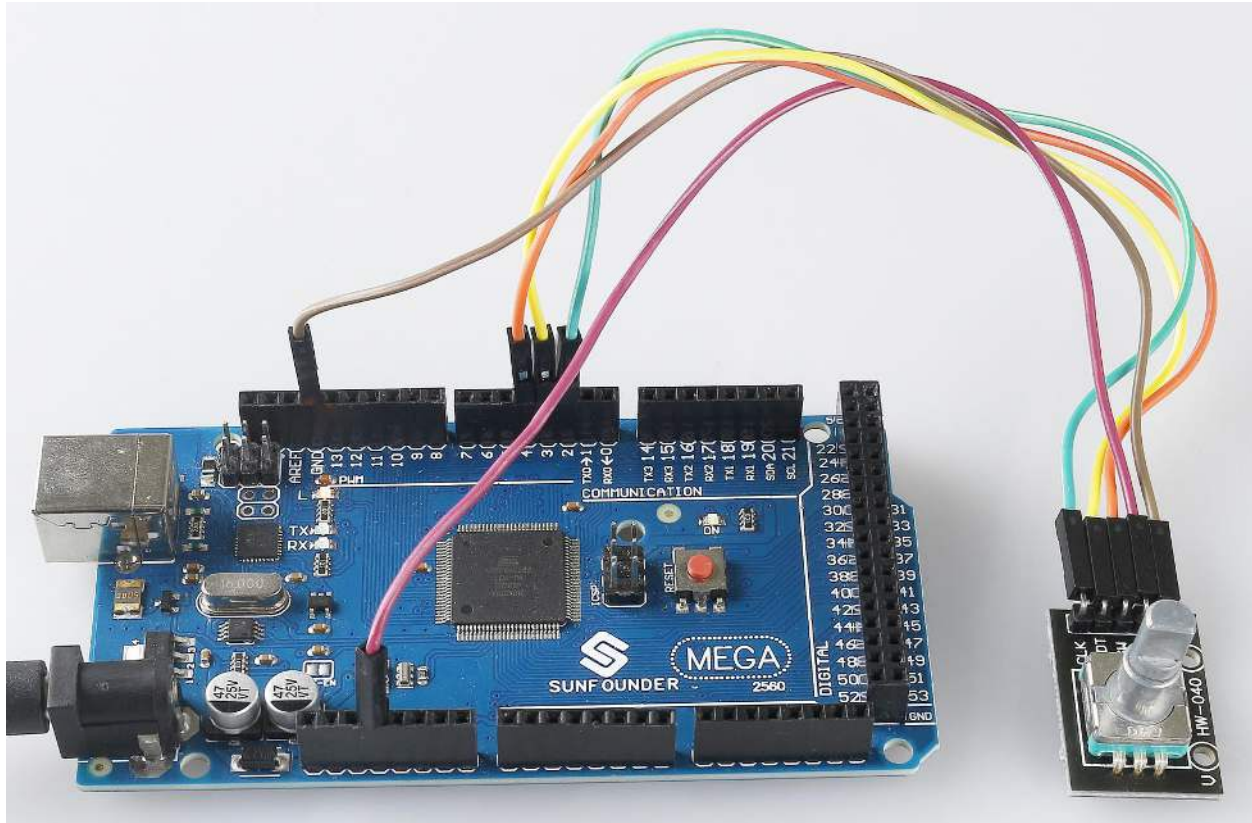
1. When rotating the shaft, dtPin will go from high level to low level.
2. clkPin will remain high level when the shaft rotates clockwise, otherwise it goes low level.
3. When the shaft is pressed, swPin will have low level.

From this, the program flow is shown on the right.

For detailed analysis of potential state change judgment, please refer to [1.10 State Change Detection](#).



2.35.7 Phenomenon Picture

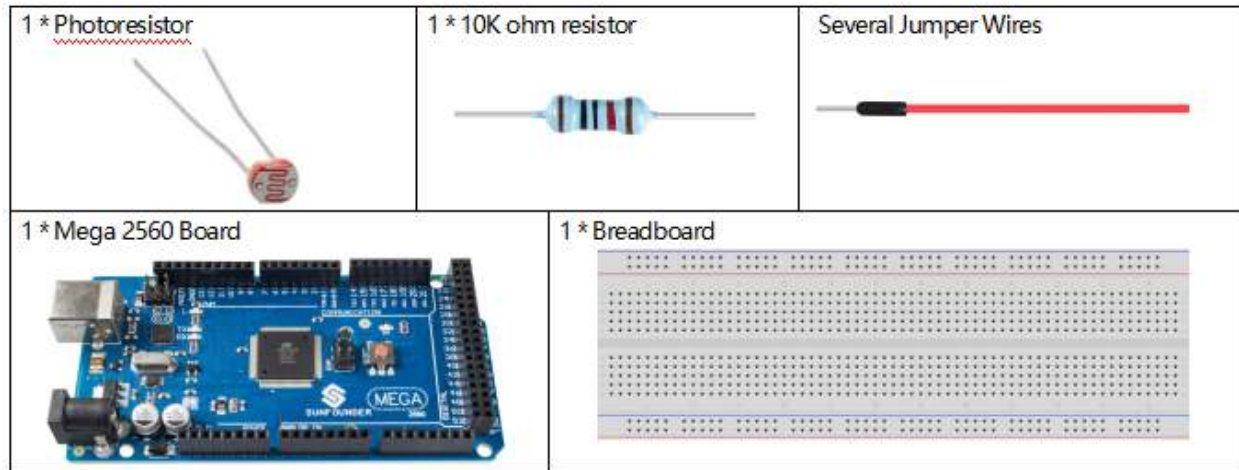


2.36 2.26 Photoresistor

2.36.1 Overview

In this lesson, you will learn about Photoresistor. Photoresistor is applied in many electronic goods, such as the camera meter, clock radio, alarm device (as beam detector), small night lights, outdoor clock, solar street lamps and etc. Photoresistor is placed in a street lamp to control when the light is turned on. Ambient light falling on the photoresistor causes street lamps to turn on or off.

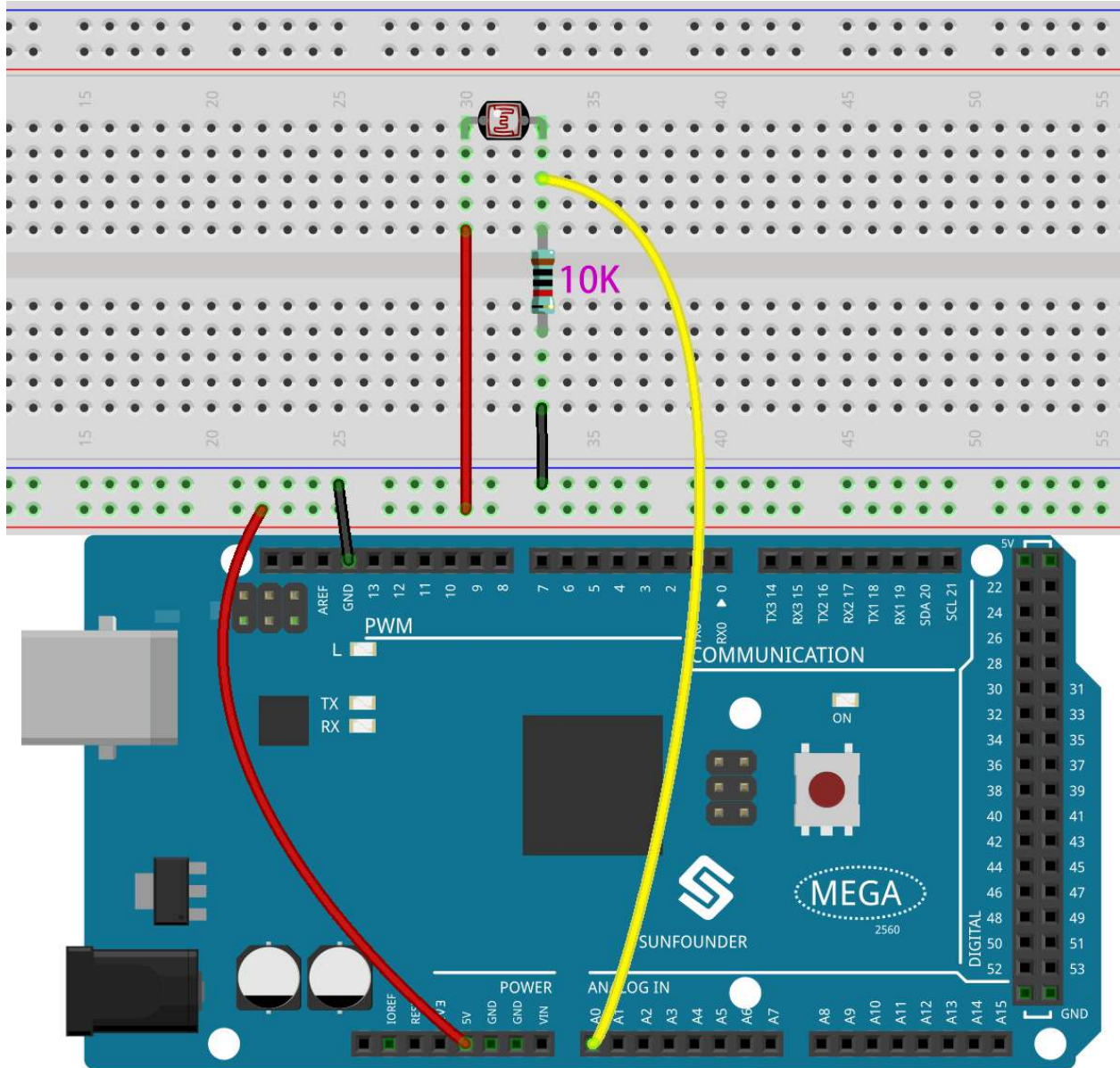
2.36.2 Components Required



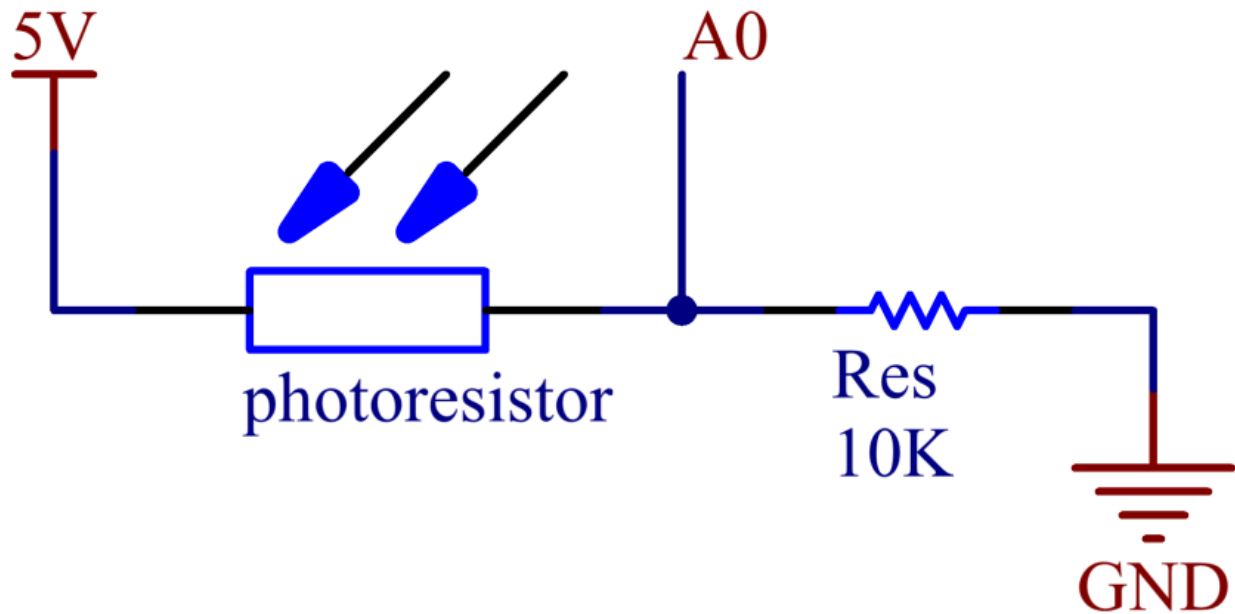
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Resistor*
- *Photoresistor*

2.36.3 Fritzing Circuit

In this example, we use analog pin (A0) to read the value of photoresistor. One pin of photoresistor is connected to 5V, the other is wired up to A0. Besides, a 10k resistor is needed before the other pin is connected to GND.



2.36.4 Schematic Diagram



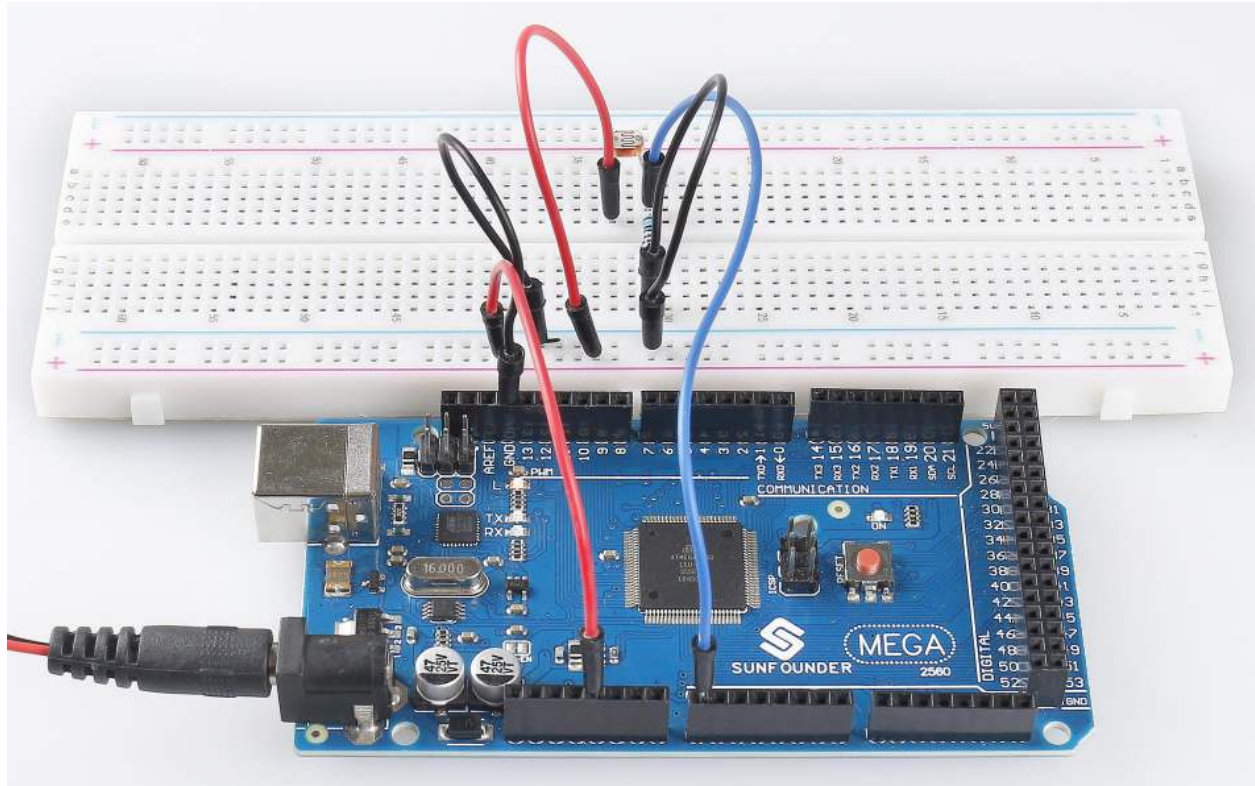
2.36.5 Code

Note:

- You can open the file `2.26_photoresistor.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.26_photoresistor` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After uploading the codes to the Mega2560 board, you can open the serial monitor to see the read value of the pin. When the ambient light becomes stronger, the reading will increase correspondingly, and the pin reading range is 0~1023. However, according to the environmental conditions and the characteristics of the photoresistor, the actual reading range may be smaller than the theoretical range. For a detailed explanation of the code, refer to [1.5 Analog Read](#).

2.36.6 Phenomenon Picture

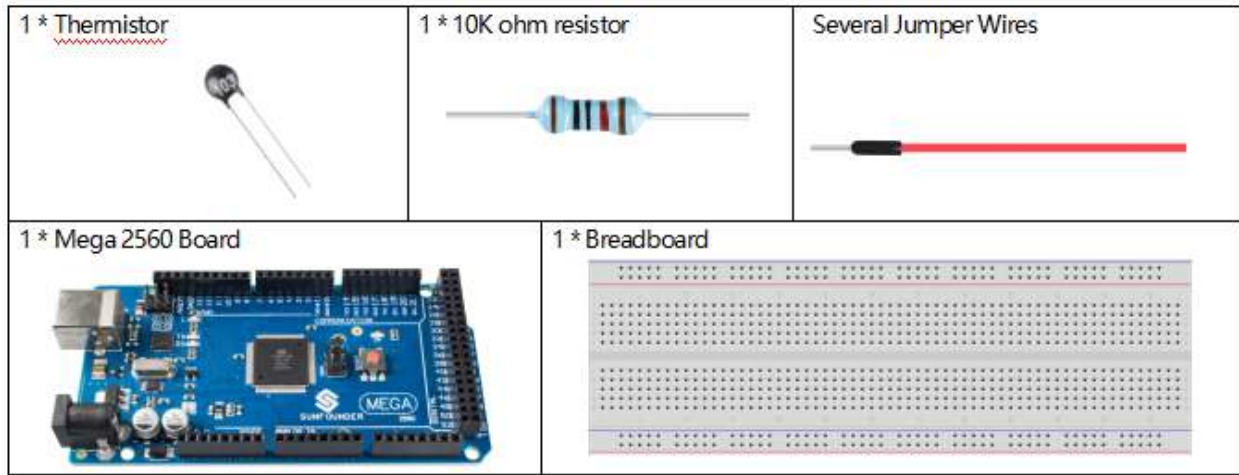


2.37 2.27 Thermistor

2.37.1 Overview

In this lesson, you will learn how to use thermistor. Thermistor can be used as electronic circuit components for temperature compensation of instrument circuits. In the current meter, flowmeter, gas analyzer, and other devices. It can also be used for overheating protection, contactless relay, constant temperature, automatic gain control, motor start, time delay, color TV automatic degaussing, fire alarm and temperature compensation.

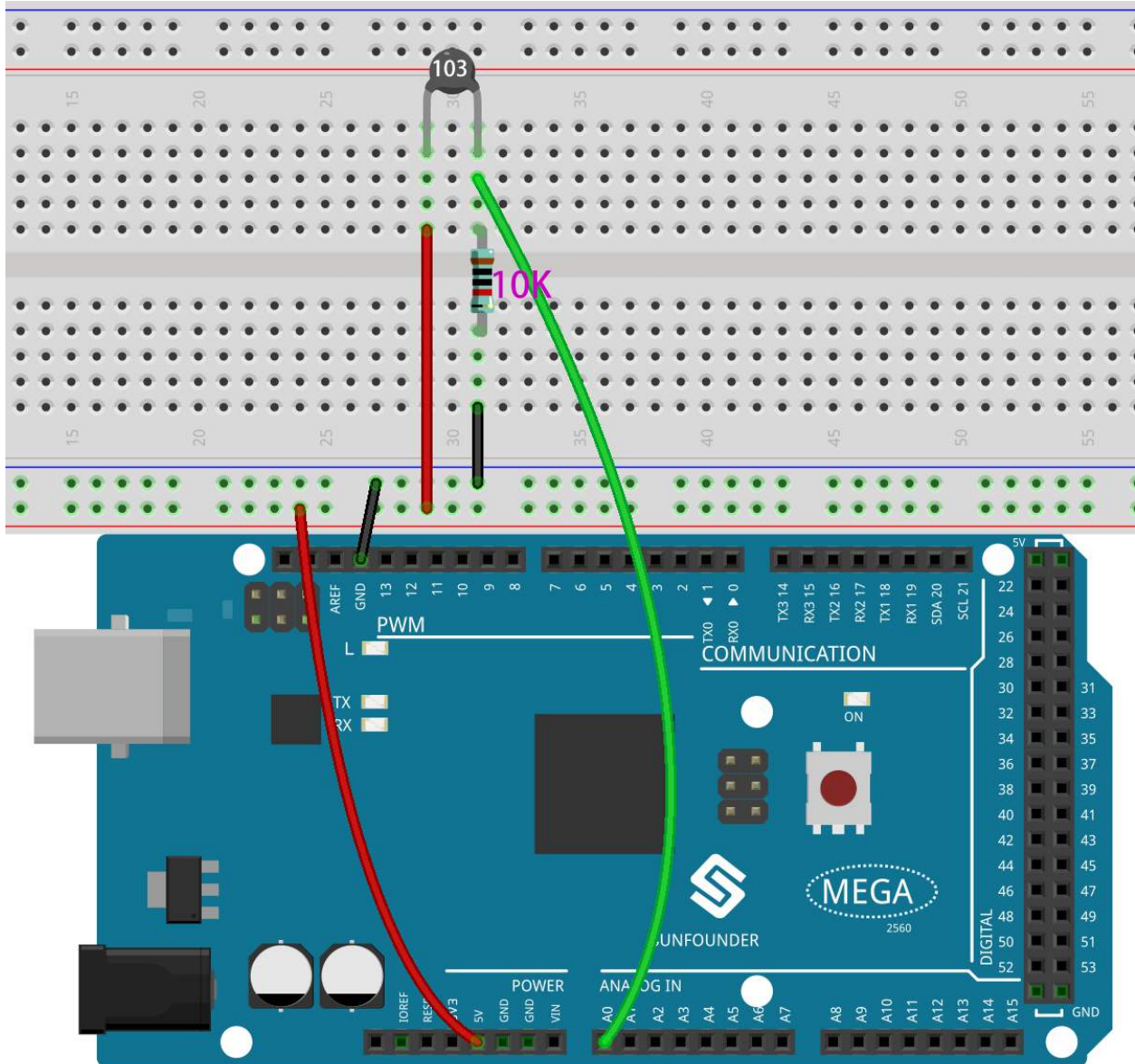
2.37.2 Components Required



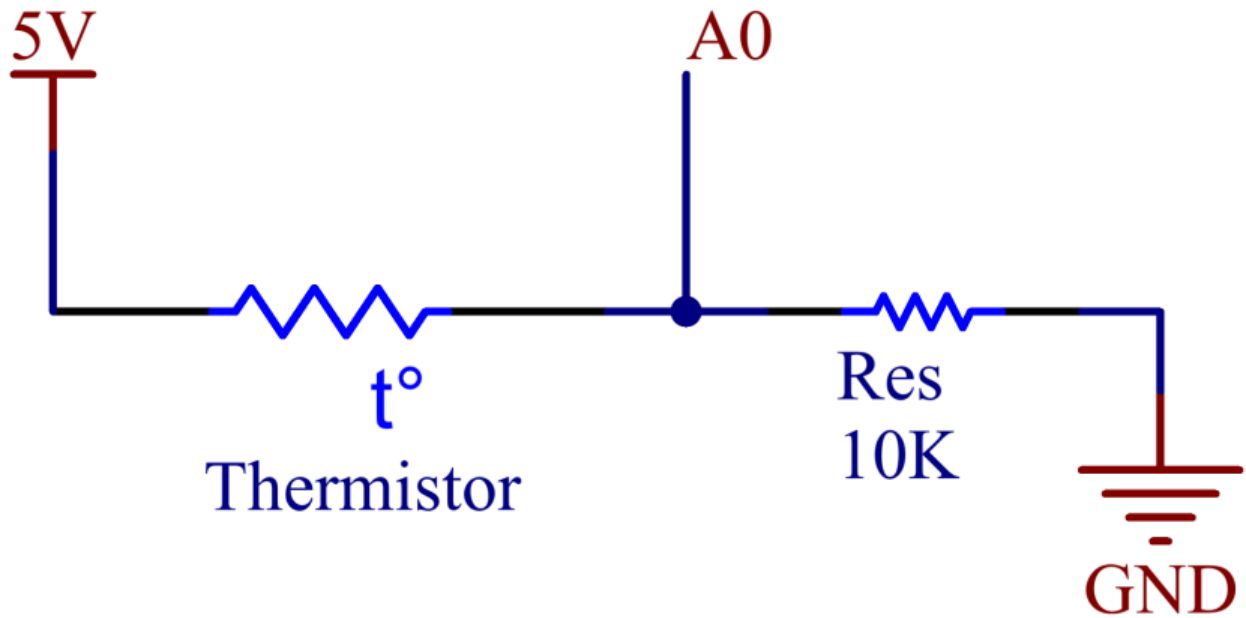
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Resistor*
- *Thermistor*

2.37.3 Fritzing Circuit

In this example, we use the analog pin A0 to get the value of Thermistor. One pin of thermistor is connected to 5V, and the other is wired up to A0. At the same time, a 10k resistor is connected with the other pin before connecting to GND.



2.37.4 Schematic Diagram



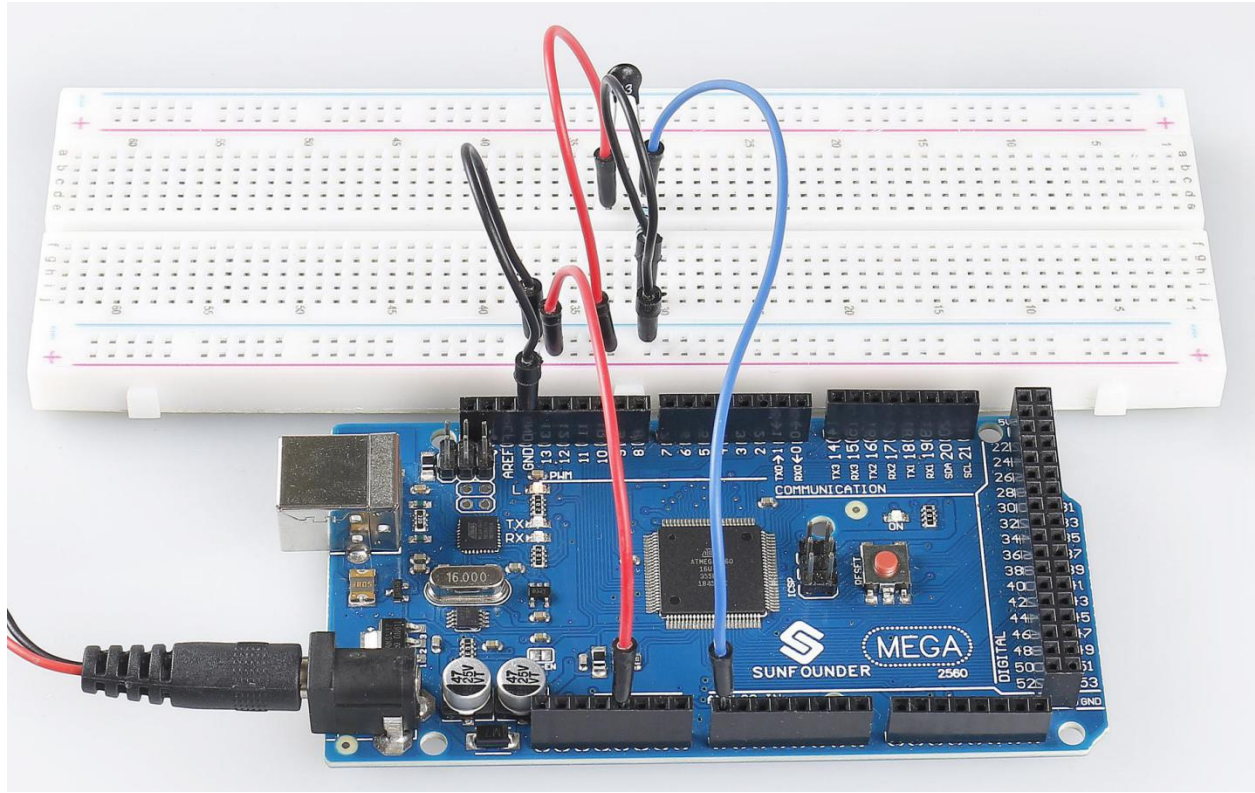
2.37.5 Code

Note:

- You can open the file `2.27_thermistor.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.27_thermistor` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After uploading the code to the Mega2560 board, you can open the serial monitor to check the current temperature. The Kelvin temperature is calculated according to the formula $TK=1/(\ln(RT/RN)/B+1/TN)$.

2.37.6 Phenomenon Picture

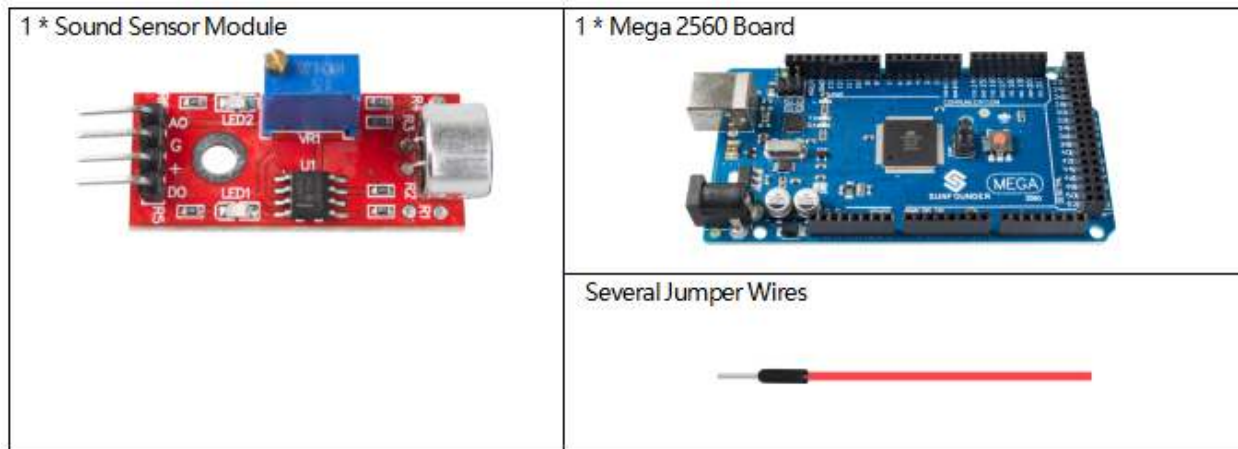


2.38 2.28 Sound Sensor Module

2.38.1 Overview

In this lesson, you will learn how to use a sound sensor module. The sound sensor module provides an easy way to detect sound and is generally used for detecting sound intensity.

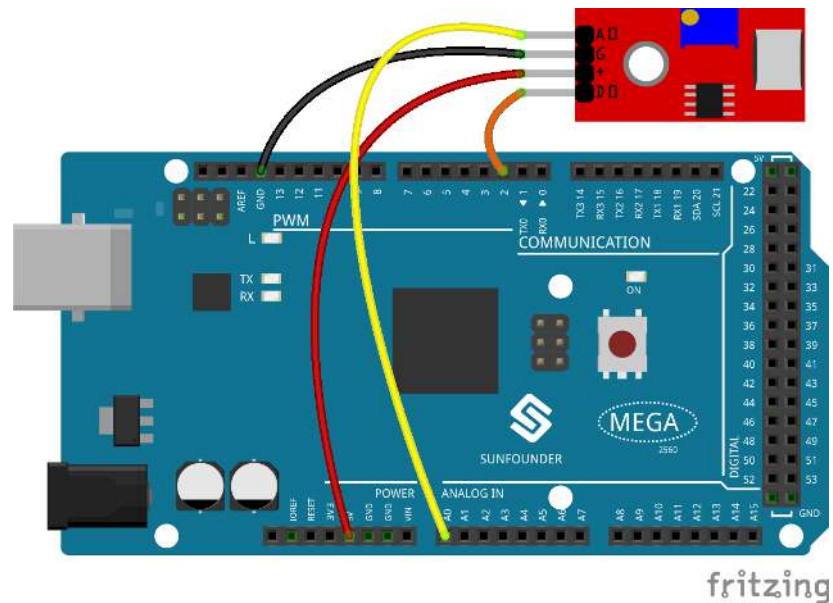
2.38.2 Components Required



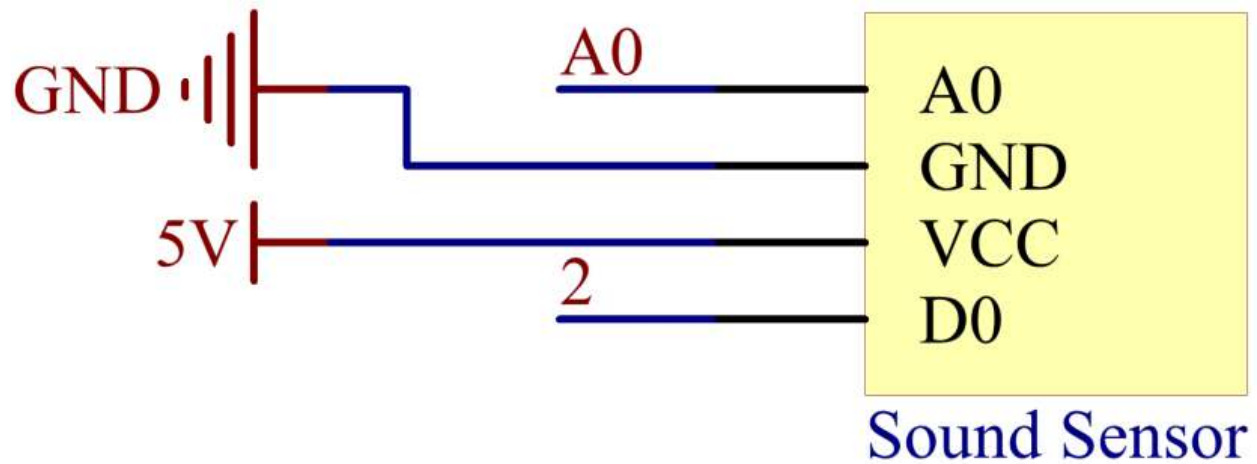
- *SunFounder Mega Board*
- *Jumper Wires*
- *Sound Sensor Module*

2.38.3 Fritzing Circuit

In this example, we can directly connect the pin of Sound Sensor Module to the pin of Mega 2560 Board, connect the pinG of Sound Sensor Module to GND, the pin +to 5V, AO to analog pin A0, and DO to digital pin 2.



2.38.4 Schematic Diagram



2.38.5 Code

Note:

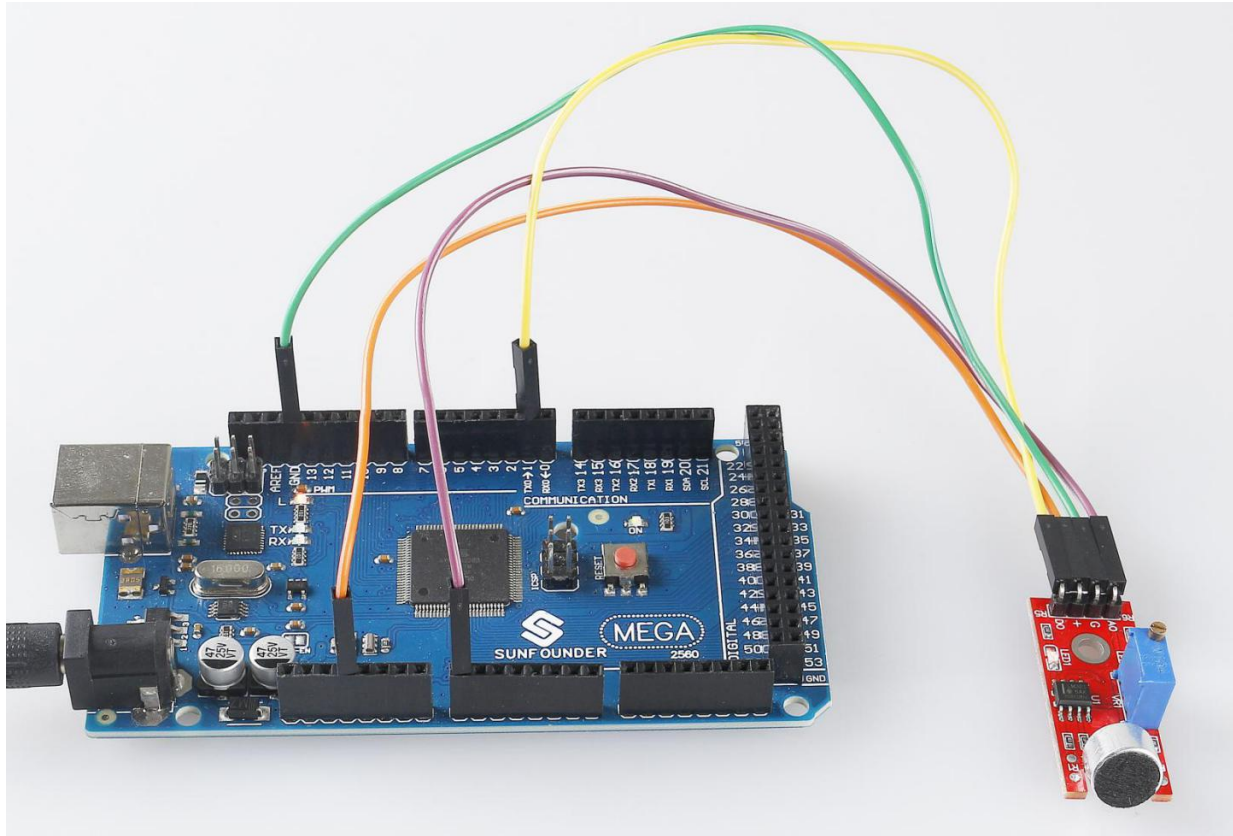
- You can open the file `2.28_soundSensorModule.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.28_soundSensorModule` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After uploading the code to the Mega2560 board, you can open the serial monitor to see the read value of the pin. When the ambient sound gets louder, the digital reading is 1 (adjust the potentiometer of the module to modify the threshold to trigger the high level), and the reading value of the analog pin will change significantly; when the environment is quiet, the digital reading is 0 and the analog reading changes smoothly.

The range of analog reading is 0~1023, but influenced by the environmental condition and the characteristics of sound sensor, the actual reading range may be smaller than the theoretical one. If an oscilloscope is used, the changing of analog reading of the sound sensor will be more obvious.

About the detail code explanation, refer to [1.5 Analog Read](#) and [1.4 Digital Read](#).

2.38.6 Phenomenon Picture

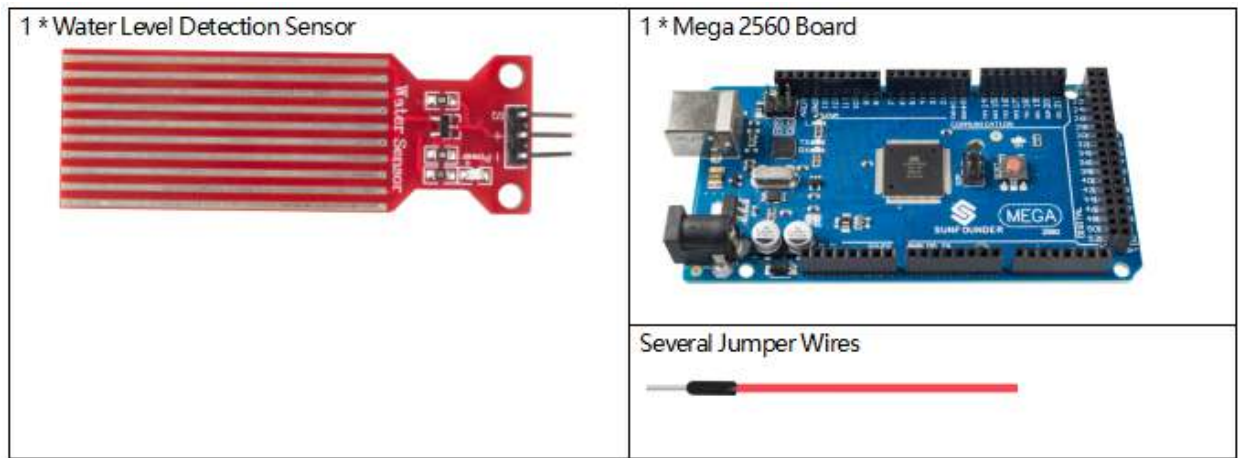


2.39 2.29 Water Sensor Module

2.39.1 Overview

In this lesson, you will learn how to use a water sensor module. A water sensor module is designed for water detection, which can be widely used in sensing the rainfall, water level, even the liquid leakage.

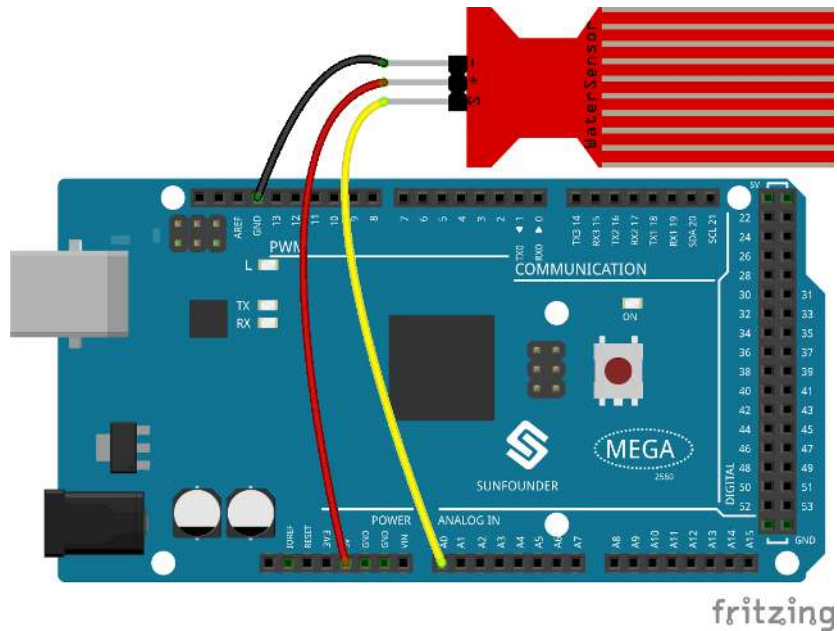
2.39.2 Components Required



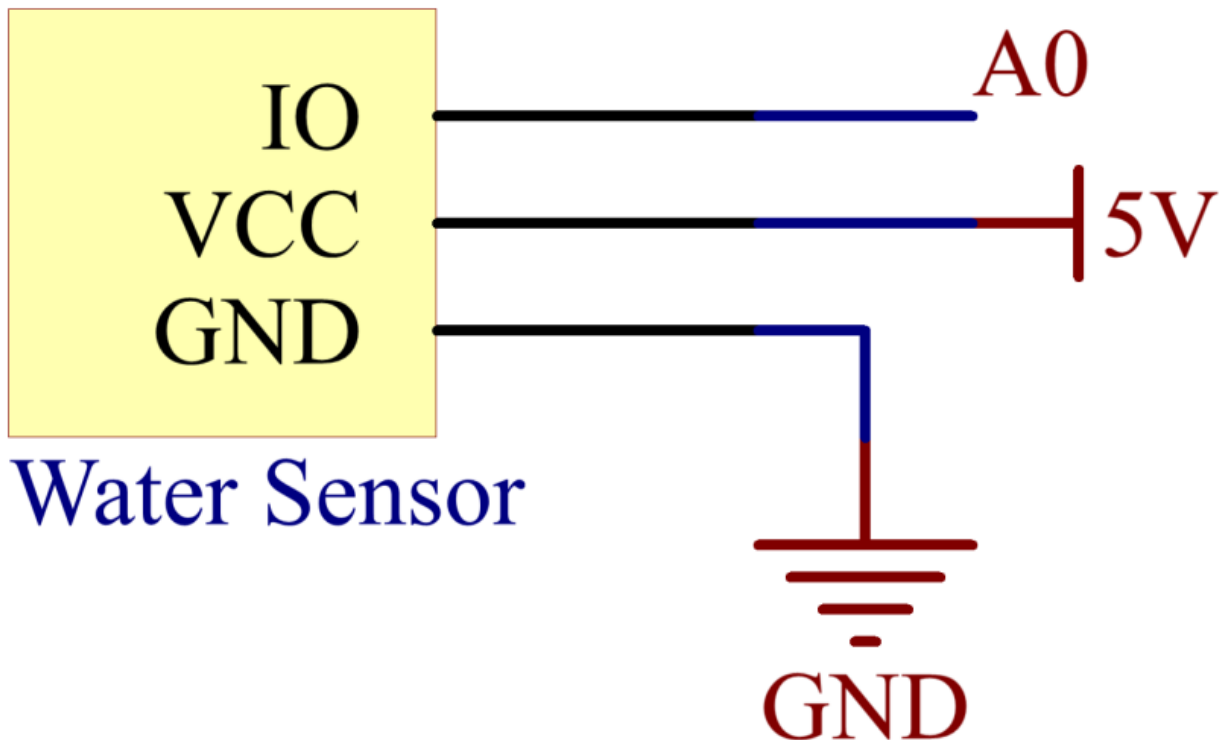
- *SunFounder Mega Board*
- *Jumper Wires*
- *Water Level Sensor Module*

2.39.3 Fritzing Circuit

In this example, we directly connect the pins of Water Sensor Module to pins of Mega 2560 Board. We use analog A0 to get the value of Water Sensor Module, and get the pinSoF Water Sensor Module to A0, -to GND, +to 5V.



2.39.4 Schematic Diagram



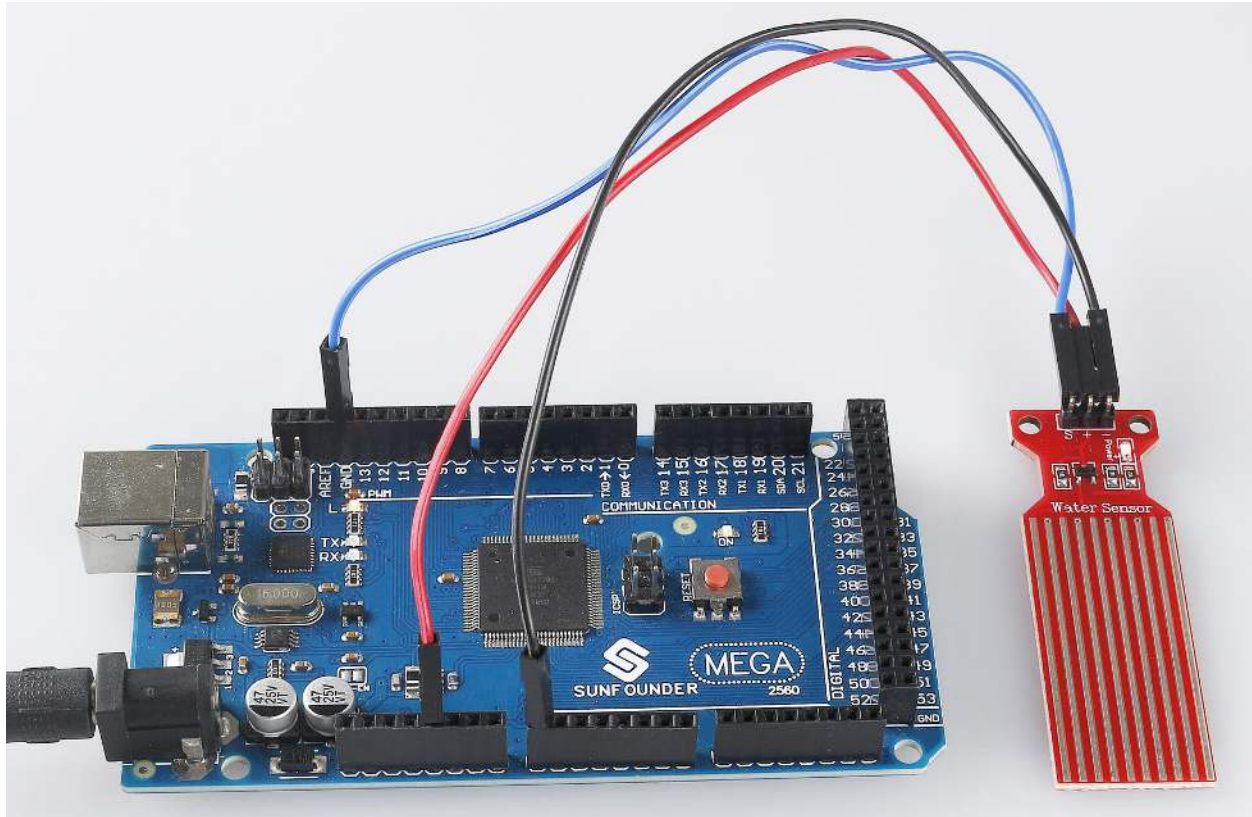
2.39.5 Code

Note:

- You can open the file `2.29_waterSensorModule.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.29_waterSensorModule` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After uploading the code to the Mega2560 board, you can open the serial monitor to see the read value of the pin. As the water level rises, the readings increase. Readings vary within the range 0~1023, but influenced by the environmental condition and the characteristics of water level sensor, the actual reading range may be smaller than the theoretical range. Refer to Part 1-1.5 Analog Read to check the detail code explanation.

2.39.6 Phenomenon Picture






2.40 2.30 IR Obstacle Avoidance Sensor

2.40.1 Overview

In this lesson, you will learn how to use IR Obstacle Avoidance Sensor. This module is commonly installed on the car and robot to judge the existence of the obstacles ahead. Also it is widely used in hand held device, water faucet and so on.

2.40.2 Components Required

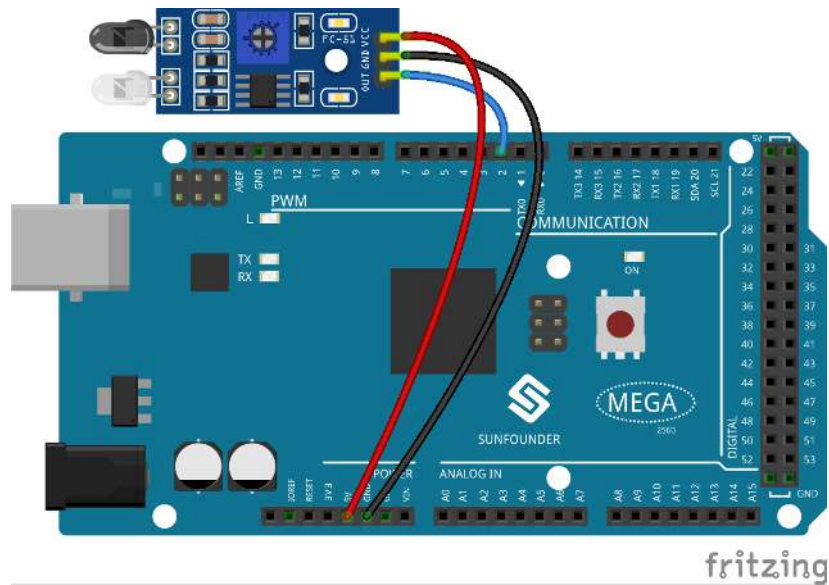
<p>1 * IR Obstacle Avoidance Sensor Module</p> 	<p>1 * Mega 2560 Board</p> 
<p>Several Jumper Wires</p> 	

- *SunFounder Mega Board*

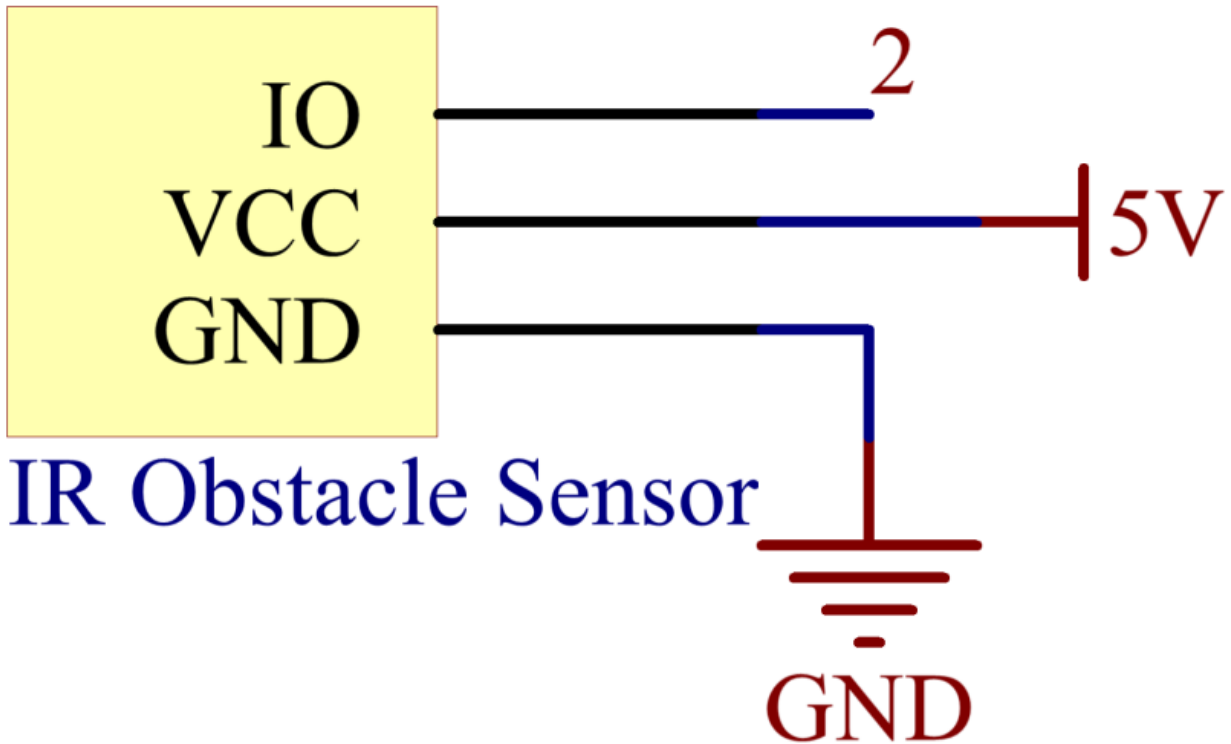
- *Jumper Wires*
- *Obstacle Avoidance Module*

2.40.3 Fritzing Circuit

We can directly connect the pins of IR Obstacle Sensor Module with the pins of Mega 2560 Board. The digital pin 2 is used to read the signal of IR Obstacle Avoidance Sensor Module. We get the VCC of IR Sensor Module connected to 5V, GND to GND, OUT to digital pin 2.



2.40.4 Schematic Diagram



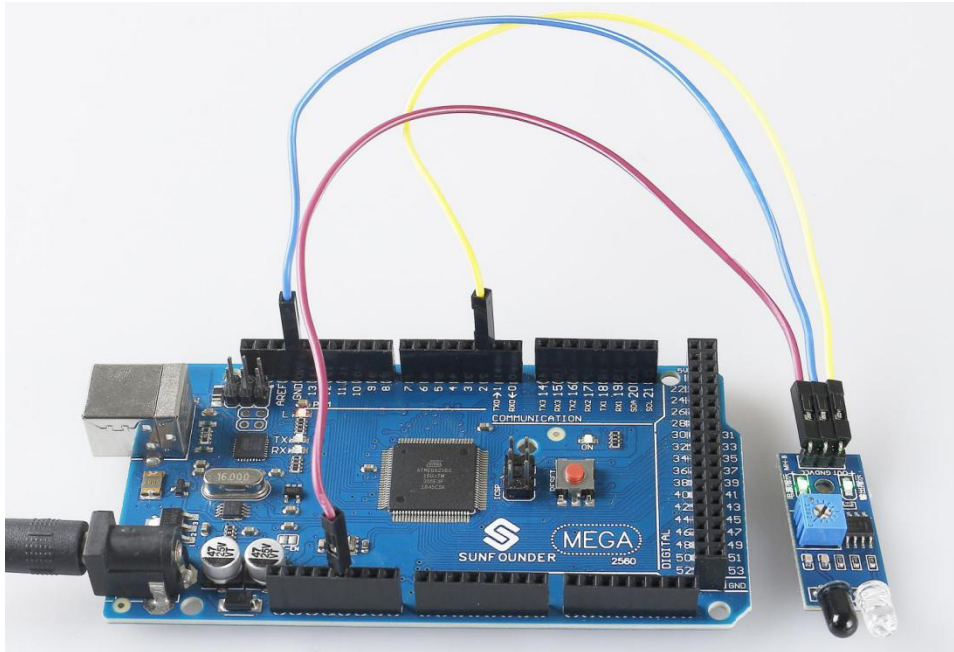
2.40.5 Code

Note:

- You can open the file `2.30_irSensorModule.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.30_irSensorModule` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Or click **Open Code** to open it in [Web Editor](#).
 - Then *Upload the Code* to the board.
-

Uploaded the codes to the Mega2560 board, you can see the readings of pins on the serial monitor. When IR Obstacle Avoidance Sensor Module detects there is something covering ahead, there appears 0 on the serial monitor; otherwise, 1 is displayed. Refer to: [ref:ar_digital_read](#) to check the detail code explanation.

2.40.6 Phenomenon Picture






2.41 2.31 PIR Module

2.41.1 Overview

In this lesson, you will learn how to use PIR Module. The PIR sensor detects infrared heat radiation or the presence of organisms that emit infrared heat radiation. This module is widely used in daily life for our intruder alarm and visiting prompt.

2.41.2 Components Required

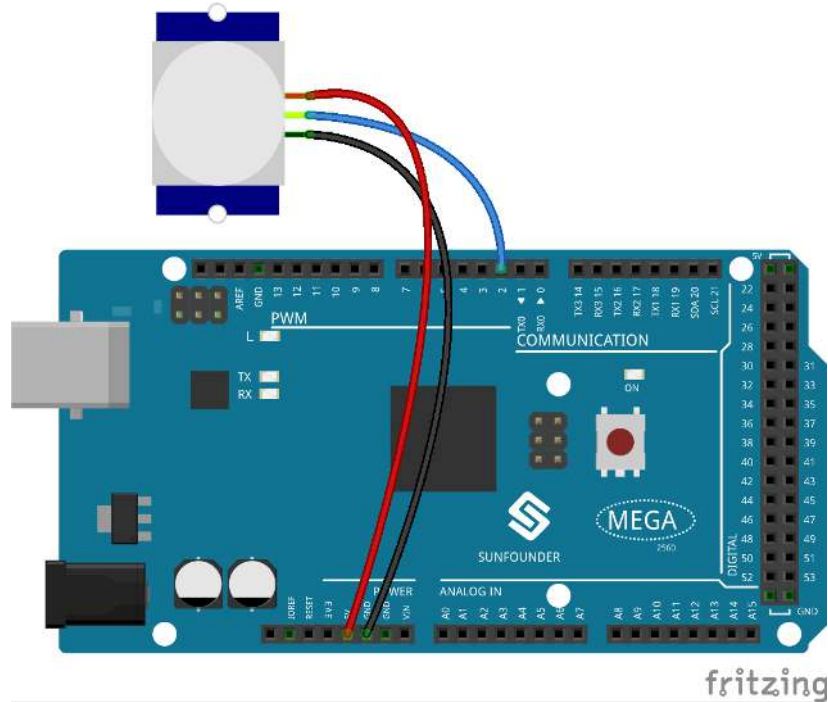
<p>1 * PIR Module</p> 	<p>1 * Mega 2560 Board</p>  <p>Several Jumper Wires</p> 
---	--

- *SunFounder Mega Board*
- *Jumper Wires*

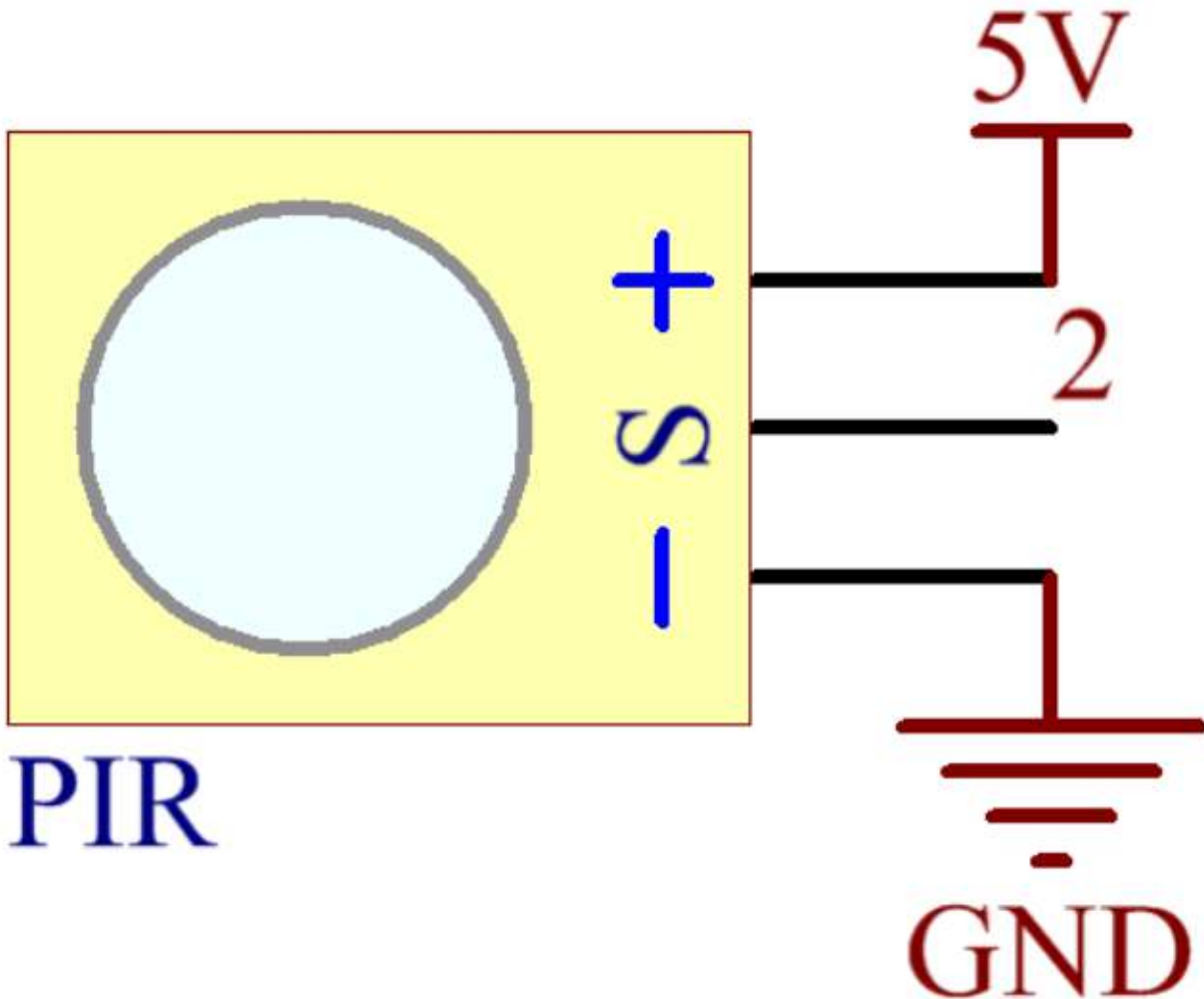
- *PIR Motion Sensor Module*

2.41.3 Fritzing Circuit

In this example, we can connect the pins of Sound Sensor Module to the pins of Mega 2560 Board directly, and we use digital pin 2 to read the signal of PIR Module. Connect the VCC of PIR Module to 5V, GND to GND, and OUT to digital pin NOTE: you can remove the PIR cover to see the pin mark.



Schematic Diagram



2.41.4 Code

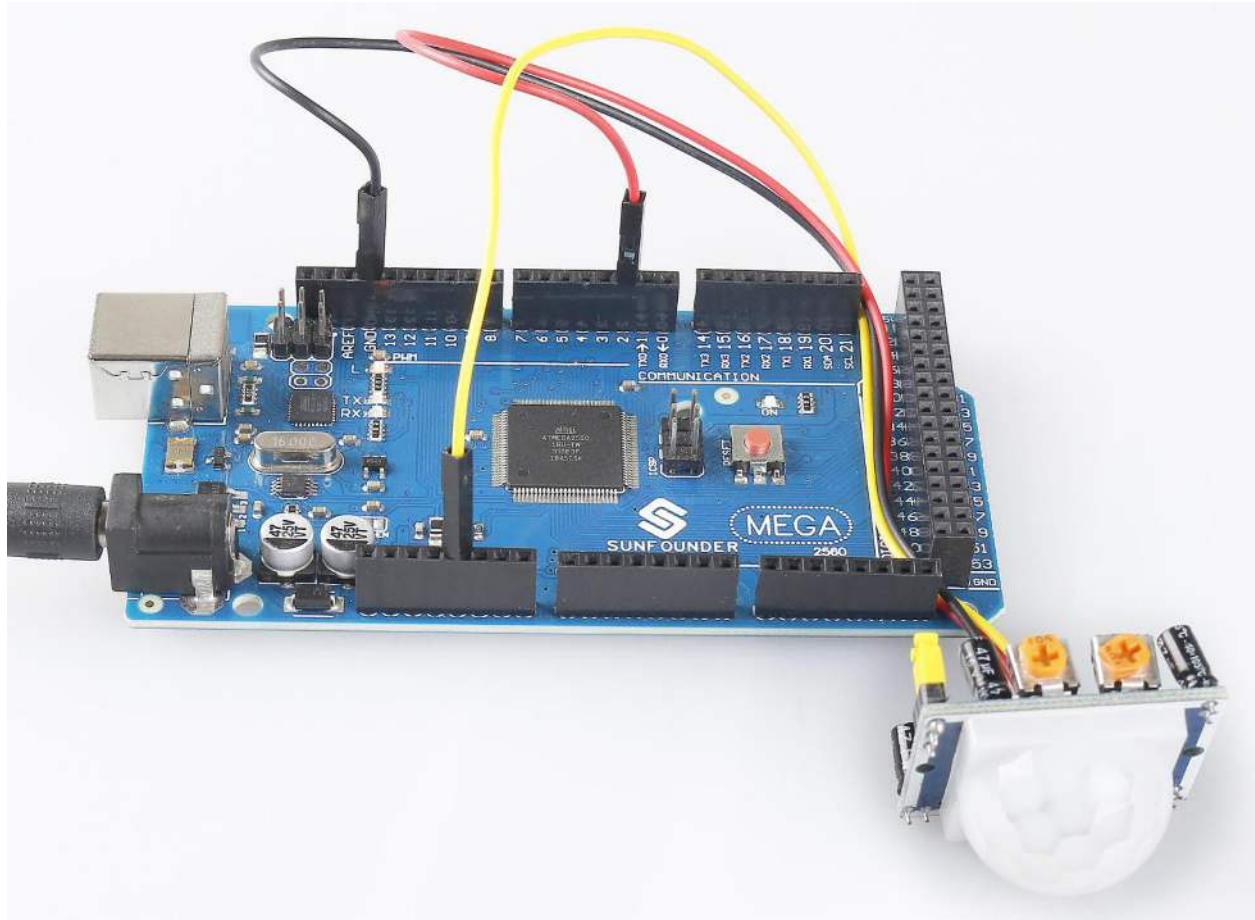
Note:

- You can open the file `2.31_pirModule.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.31_pirModule` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After the codes are uploaded to the Mega2560 board, you can open the serial monitor to see the reading value of the pin. When PIR Module detects activity nearby, the serial monitor will display 1; otherwise, it will display 0. Check [1.4 Digital Read](#) detail code explanation.

There are two potentiometers on the PIR module: one is to adjust **sensitivity** and the other is to adjust the **detection distance**. In order to make the PIR module work better, you need to try to adjust these two potentiometers.

2.41.5 Phenomenon Picture

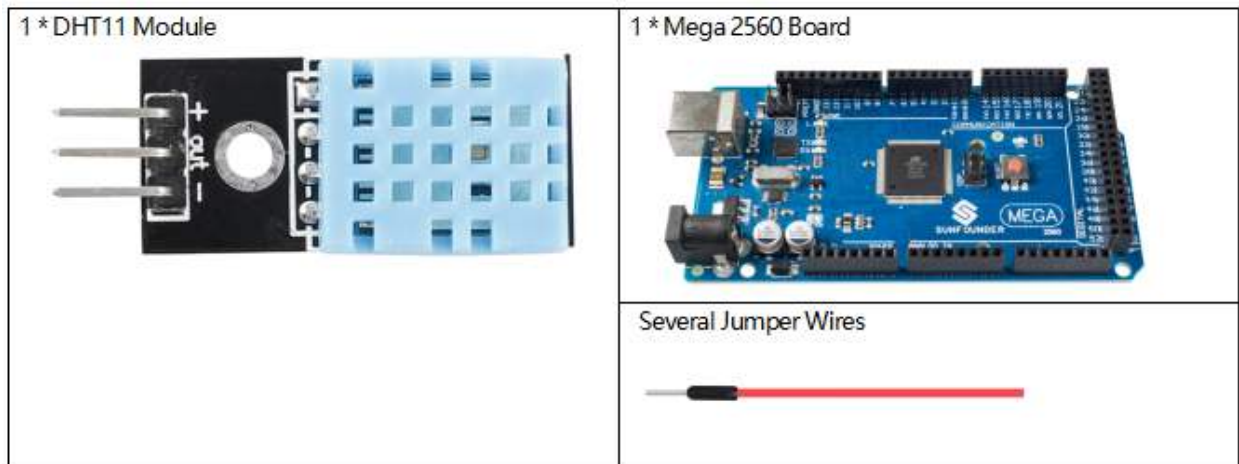


2.42 2.32 DHT11 Module

2.42.1 Overview

In this lesson, you will learn how to use DHT11 Module. The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins are needed).

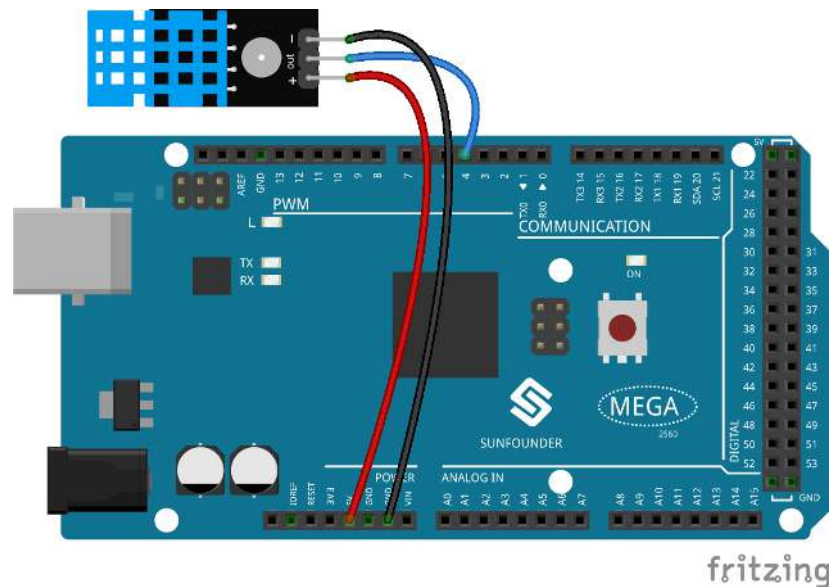
2.42.2 Components Required



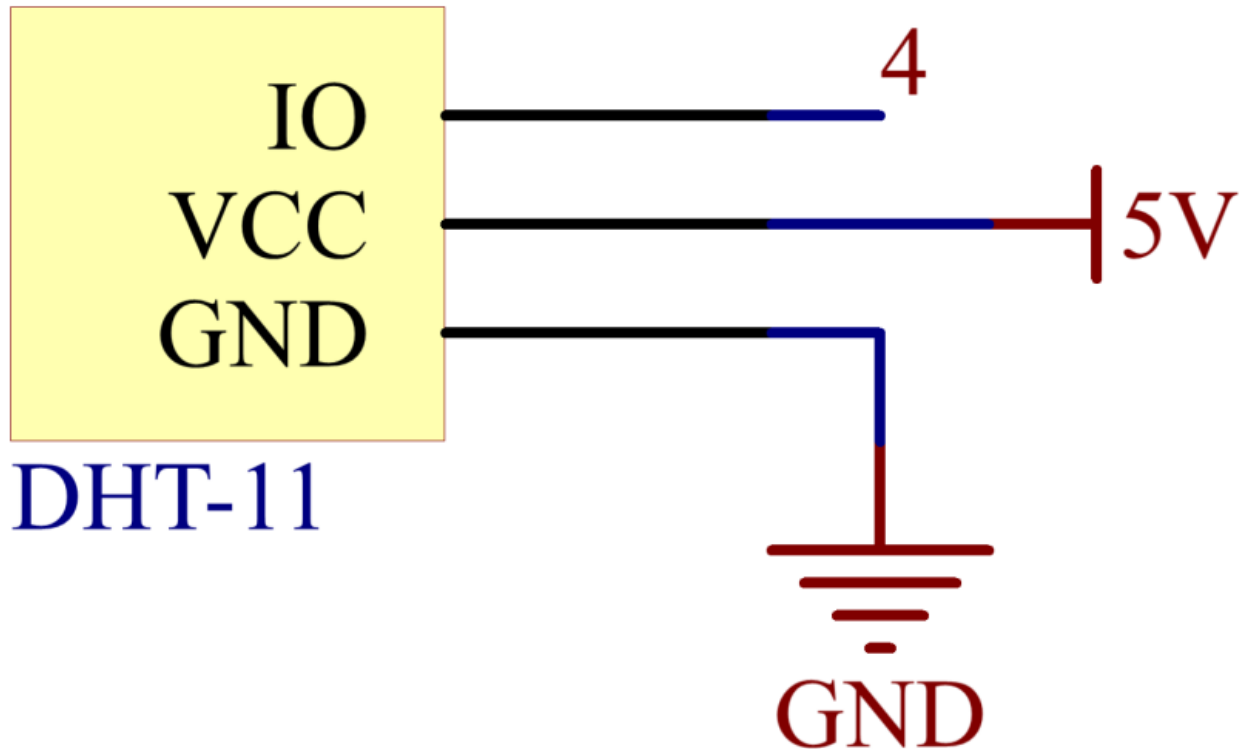
- *SunFounder Mega Board*
- *Jumper Wires*
- *Humiture Sensor Module*

2.42.3 Fritzing Circuit

In this example, we can directly connect the pins of DHT11 Module to the pins of Mega 2560 Board, and we use pin 4 to read the signal of DHT11 Module. Connect the pin+ of DHT11 Module to 5V, the pin- to GND, and the pin OUT to pin 4.



2.42.4 Wiring Diagram



2.42.5 Code

Note:

- You can open the file `2.32_dhtModule.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.32_dhtModule` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Then *Upload the Code* to the board.
 - Please make sure you have added the library called `dht`, detailed tutorials refer to *Add Libraries*.
-

After the codes are uploaded to the Mega2560 board, the serial monitor will continue to output the current temperature and humidity values of the environment.

2.42.6 Code Analysis

The function of the module is included in the library dht.h.

```
#include <dht.h>
```

Library Functions:

```
dht
```

Creates a new instance of the dht class that represents a particular DHT-11 module attached to your Arduino board.

```
int read11(uint8_t pin)
```

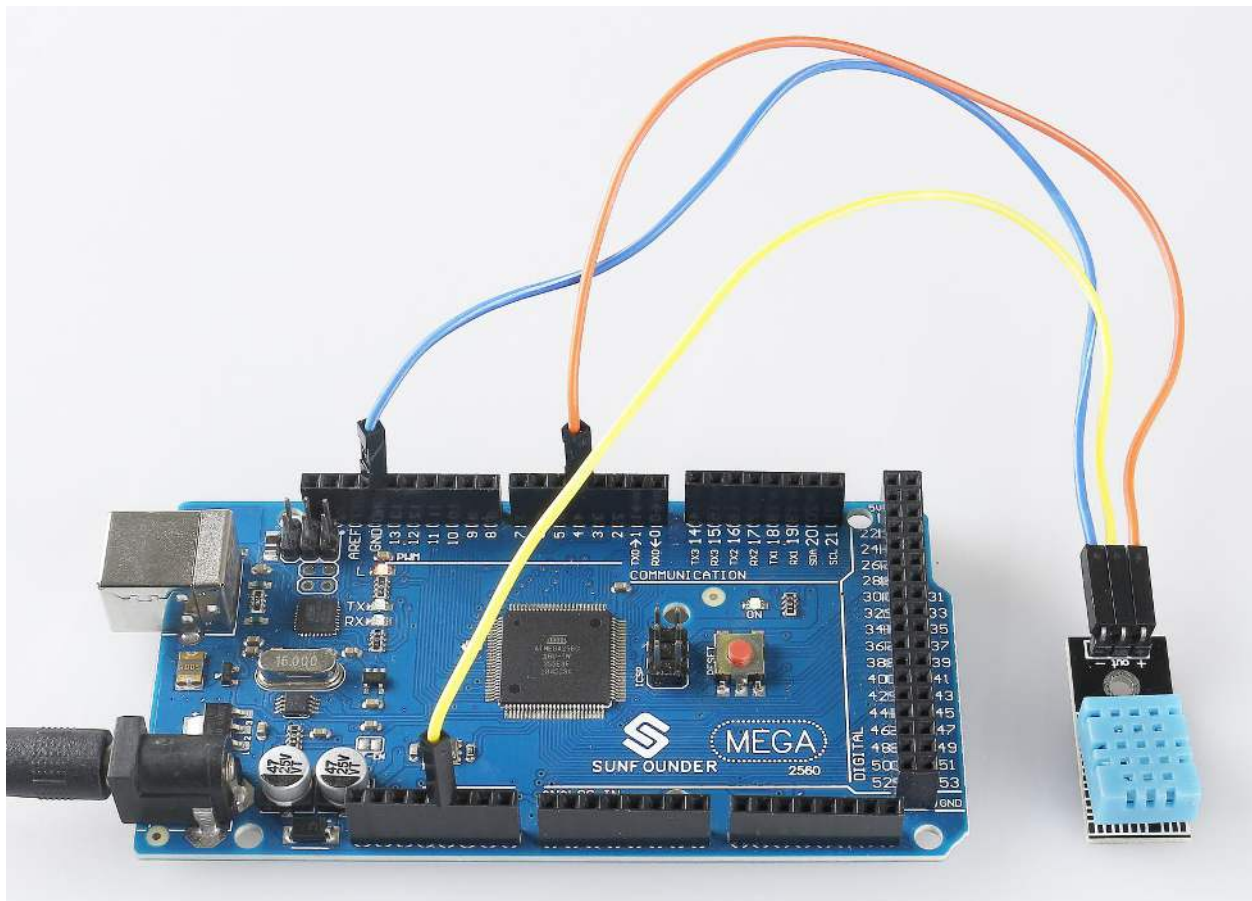
This function will return CHECK values.

- DHTLIB_OK means that DHT-11 is in good condition;
- DHTLIB_ERROR_CHECKSUM represents that the value may be abnormal;
- DHTLIB_ERROR_TIMEOUT indicates that there is timeout.

The function will store the detected humidity and temperature into the variables with the same name in dht class.

The function should be called and used directly in the main program. (e.g. Serial.println(DHT.temperature,1);)

2.42.7 Phenomenon Picture

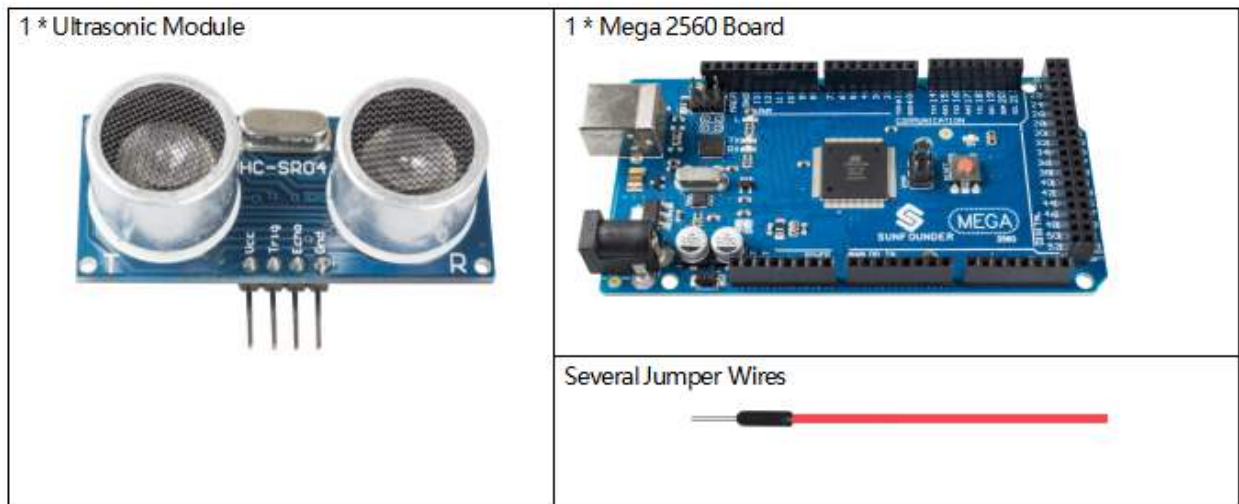


2.43 2.33 Ultrasonic Module

2.43.1 Overview

In this lesson, you will learn how to use Ultrasonic module.

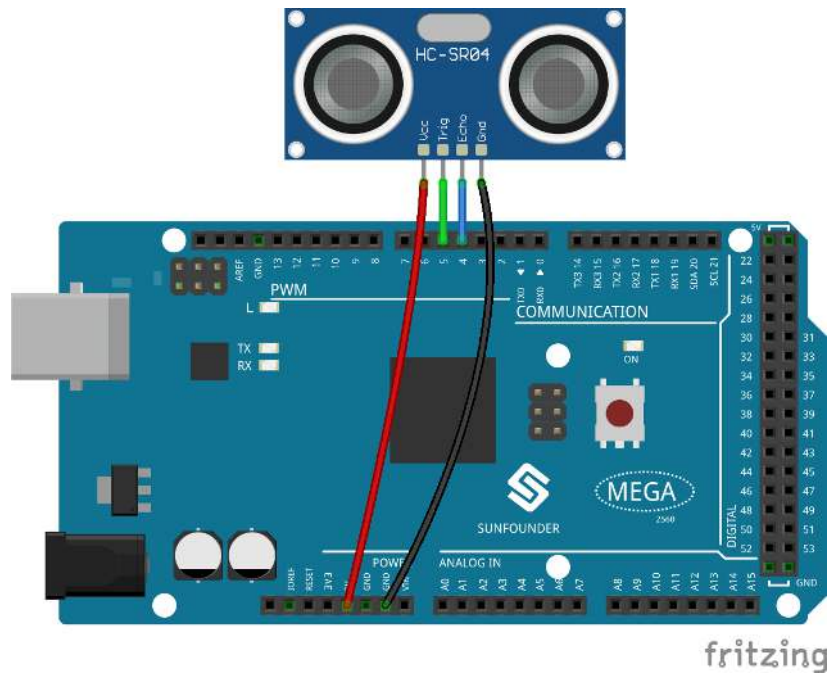
2.43.2 Components Required



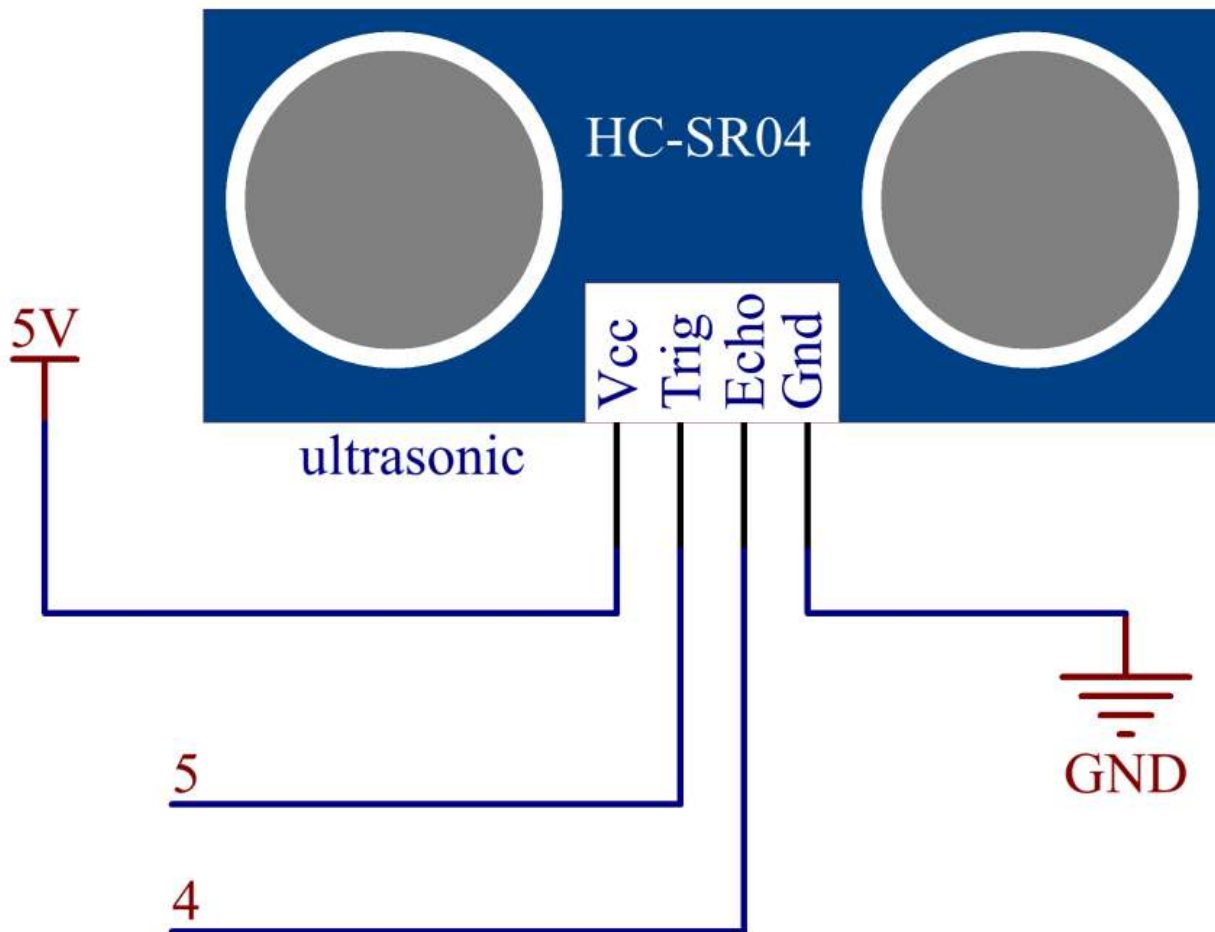
- *SunFounder Mega Board*
- *Jumper Wires*
- *Ultrasonic Module*

2.43.3 Fritzing Circuit

In this example, we directly connect the pins of Ultrasonic Module with the pins of Mega 2560 Board. And then we get VCC of the Ultrasonic Module connected to 5V, GND to GND, Trig to the digital pin 5, Echo to the digital pin 4.



2.43.4 Schematic Diagram



2.43.5 Code

Note:

- You can open the file `2.33_ultrasonicModule.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.33_ultrasonicModule` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Or click **Open Code** to open it in [Web Editor](#).
 - Then *Upload the Code* to the board.
-

After uploading the codes to the Mega2560 board, the serial monitor will display the distance of obstacles ahead that the ultrasonic sensor has detected.

2.43.6 Code Analysis

About the application of ultrasonic sensor, we can directly check the subfunction.

```
float readSensorData() { // ... }
```

PING is triggered by a HIGH pulse of 2 or more microseconds. (Give a short LOW pulse beforehand to ensure a clean HIGH pulse.)

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```

The echo pin is used to read signal from PING, a HIGH pulse whose duration is the time (in microseconds) from the sending of the ping to the reception of echo of the object.

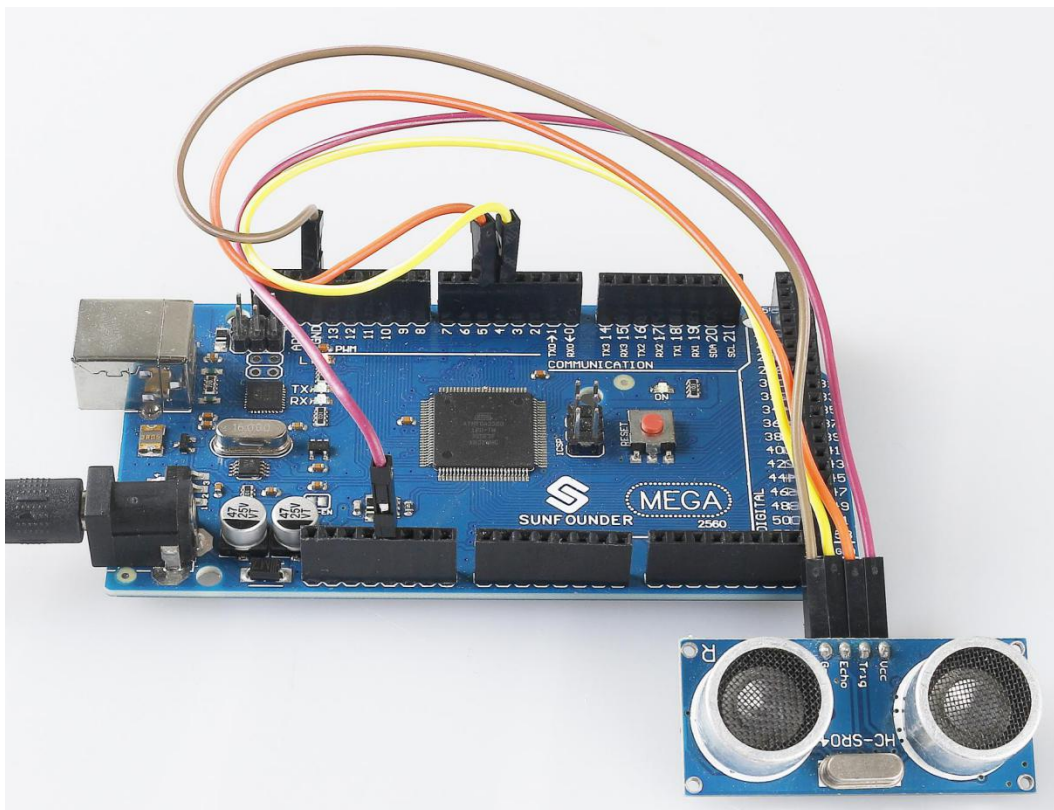
```
microsecond=pulseIn(echoPin, HIGH);
```

The speed of sound is 340 m/s or 29 microseconds per centimeter.

This gives the distance travelled by the ping, outbound and return, so we divide by 2 to get the distance of the obstacle.

```
float distance = microsecond / 29.00 / 2;
```

2.43.7 Phenomenon Picture

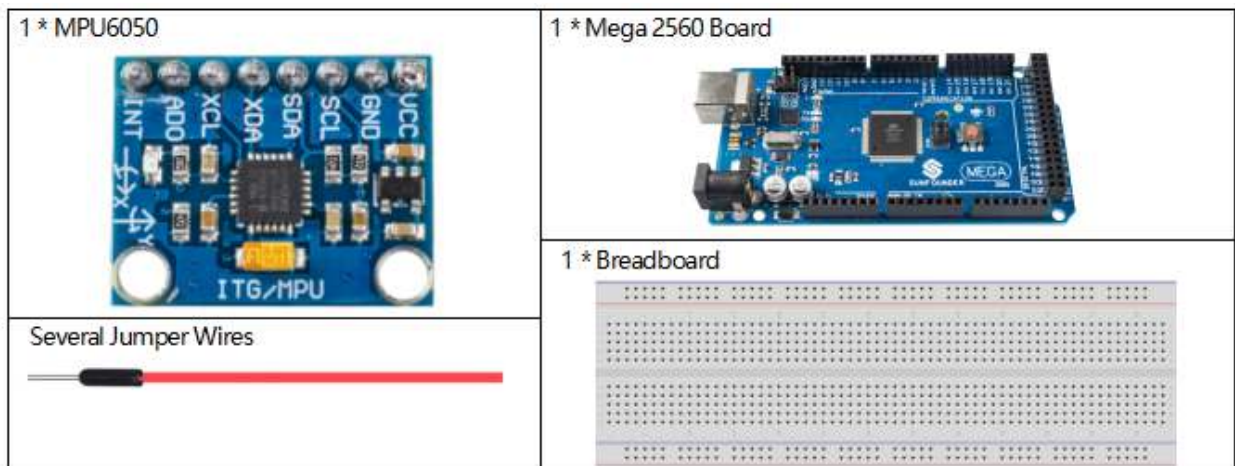


2.44 2.34 MPU6050 Module

2.44.1 Overview

In this lesson, you will learn how to use MPU6050. MPU-6050 is a 6-axis (combined 3-axis Gyroscope, 3-axis Accelerometer) motion tracking devices. It is often used for augmented reality and electronic image stabilization (EIS: Electronic Image Stabilization), optical image stabilization (OIS: Optical Image Stabilization), “Zero touch” gesture user interface.

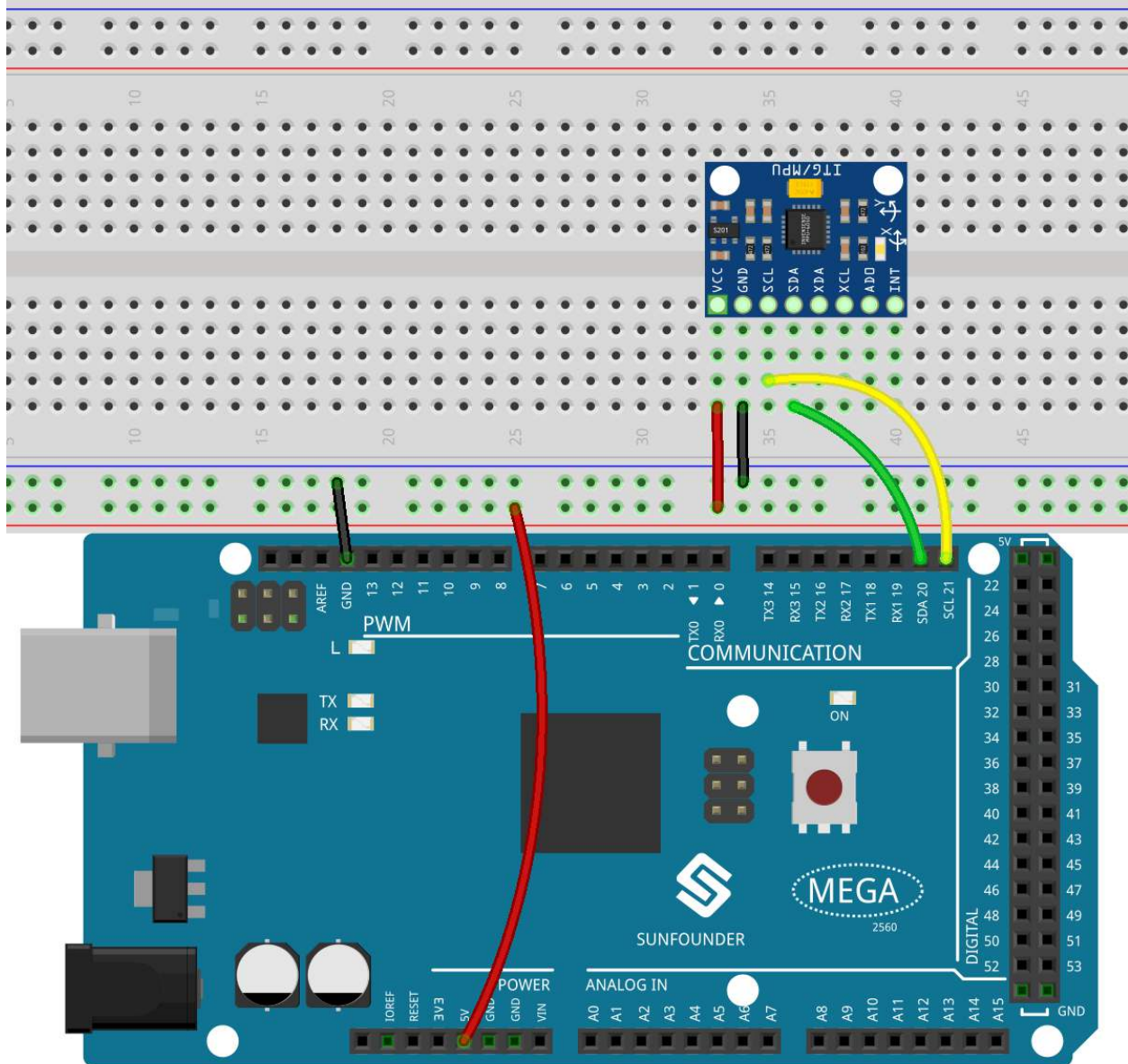
2.44.2 Components Required



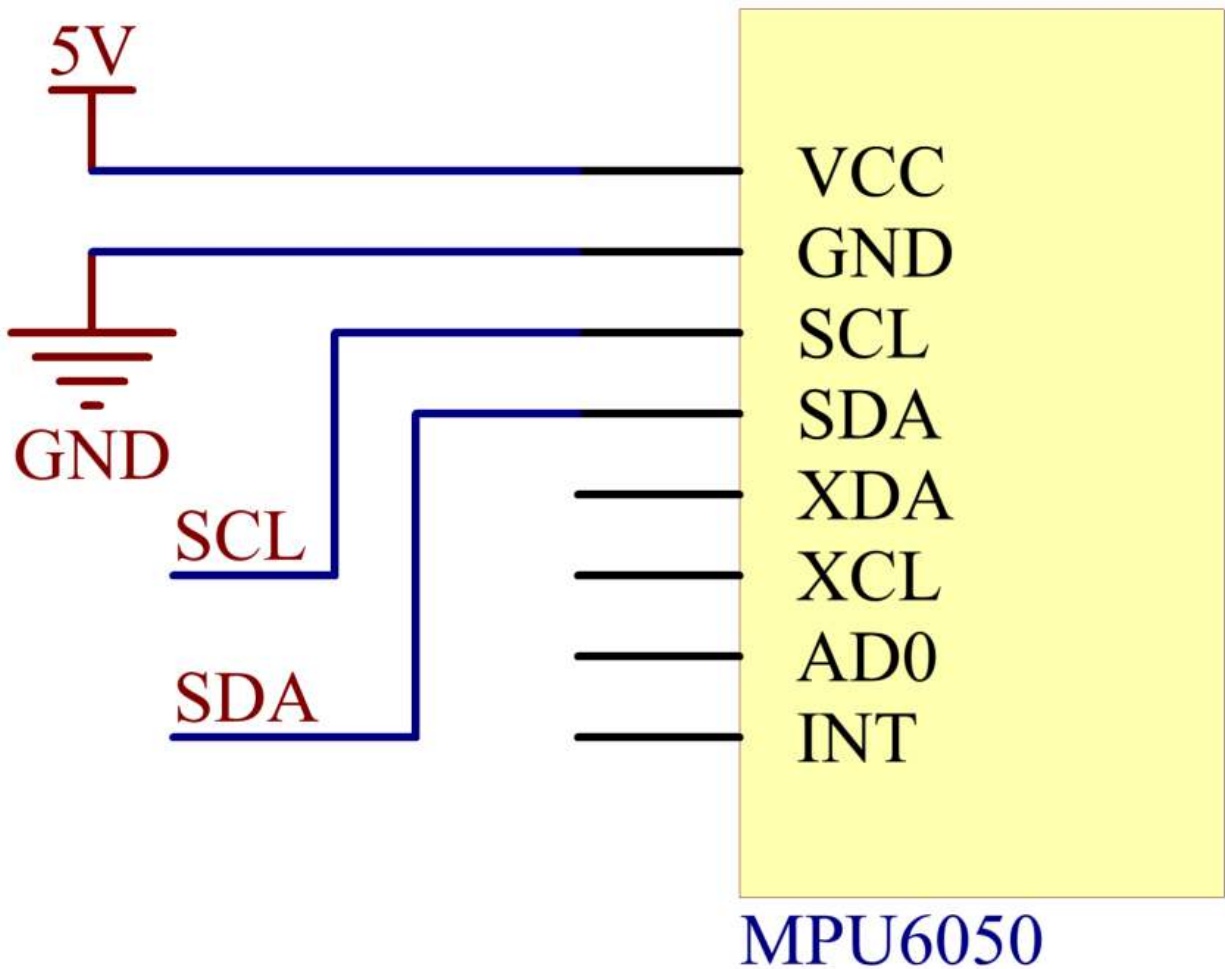
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *MPU6050 Module*

2.44.3 Fritzing Circuit

In this example, we drive MPU6050 with IIC. We inset MPU6050 into the breadboard; get the VCC connected to 5V, GND to GND, SCL to pin SCL 21, and SDA to the pin SDA 20.



2.44.4 Schematic Diagram



2.44.5 Code

Note:

- You can open the file `2.34_MPU6050.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.34_MPU6050` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

After uploading the codes to the Mega2560 board, you can open the serial monitor to see the gravity acceleration and angular velocity of MPU6050 in each direction.

2.44.6 Code Analysis

In the stationary desktop state, the Z-axis acceleration is 1 gravity unit, and the X and Y axes are 0.

Before your use, you need to calibrate the module, the methods are as follows:

1. MPU6050 modules are placed horizontally on the desktop and then you can fix them with clamps or adhesive tape.
2. Run the sample codes to get the RAW DATA of MPU6050 when it is static.
3. Add compensation according to the readings when MPU6050 is static.

Take MPU6050 we use as an example, and the results of compensation are as follows:

```
Serial.print(AcX / 65536 * ACCELE_RANGE - 0.02);
Serial.print(AcY / 65536 * ACCELE_RANGE + 0);
Serial.print(AcZ/65536 * ACCELE_RANGE + 0.02);
Serial.print(GyX / 65536 * GYROSC_RANGE + 1.70);
Serial.print(GyY/65536 * GYROSC_RANGE - 1.70);
Serial.print(GyZ/65536*GYROSC_RANGE + 0.25);
```

```
AcX = -0.00g | AcY = -0.00g | AcZ = 0.99g
GyX = 0.14d/s | GyY = -0.18d/s | GyZ = -0.20d/s

AcX = 0.00g | AcY = 0.00g | AcZ = 1.00g
GyX = -0.02d/s | GyY = -0.20d/s | GyZ = -0.06d/s

AcX = -0.00g | AcY = -0.00g | AcZ = 1.01g
GyX = -0.06d/s | GyY = -0.06d/s | GyZ = 0.05d/s

AcX = -0.01g | AcY = 0.00g | AcZ = 1.00g
GyX = 0.04d/s | GyY = -0.04d/s | GyZ = -0.16d/s

AcX = -0.01g | AcY = 0.00g | AcZ = 1.01g
GyX = -0.15d/s | GyY = -0.20d/s | GyZ = -0.06d/s
```

Before the calibration

```
AcX = 0.02g | AcY = -0.00g | AcZ = 0.98g
GyX = -1.72d/s | GyY = 1.67d/s | GyZ = -0.31d/s

AcX = 0.02g | AcY = 0.00g | AcZ = 0.98g
GyX = -1.70d/s | GyY = 1.63d/s | GyZ = -0.29d/s

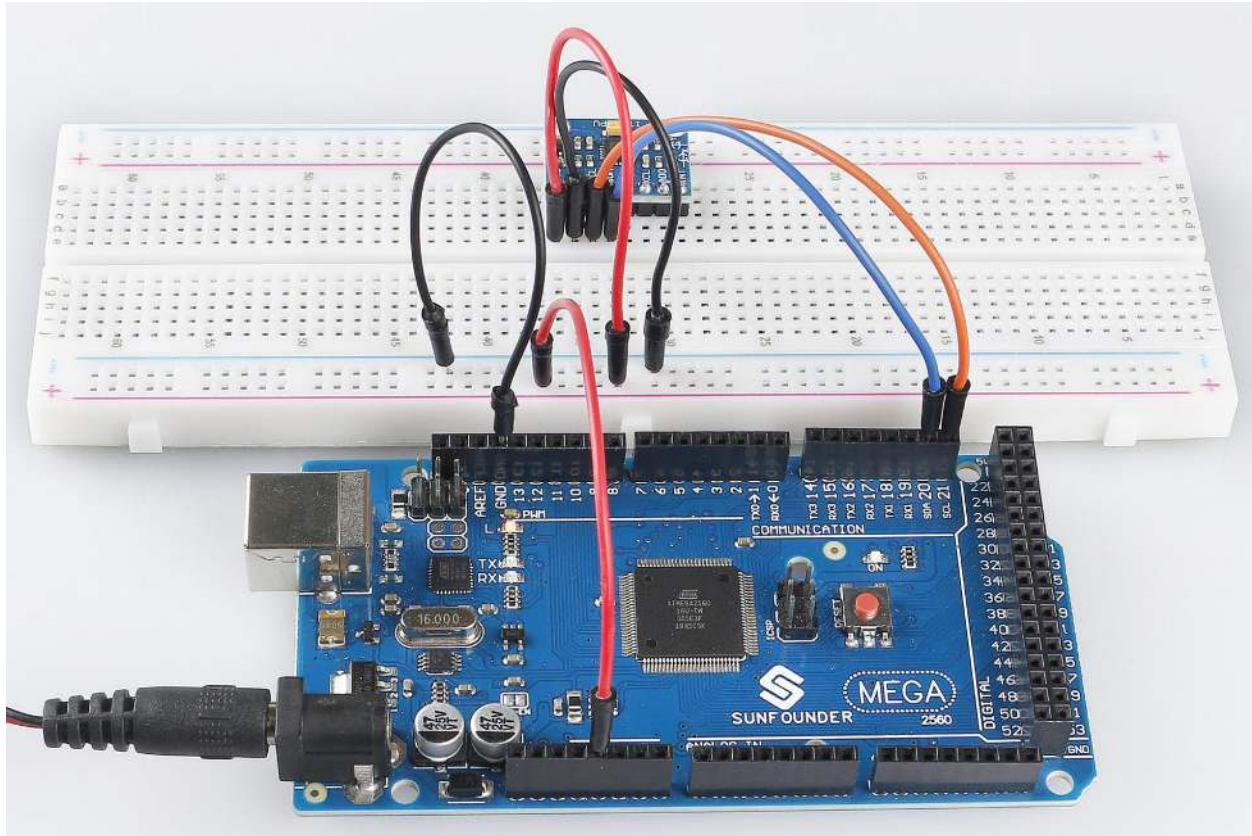
AcX = 0.02g | AcY = -0.00g | AcZ = 0.97g
GyX = -1.64d/s | GyY = 1.60d/s | GyZ = -0.26d/s

AcX = 0.02g | AcY = -0.00g | AcZ = 0.98g
GyX = -1.82d/s | GyY = 1.52d/s | GyZ = -0.20d/s

AcX = 0.02g | AcY = -0.00g | AcZ = 0.98g
GyX = -1.75d/s | GyY = 1.75d/s | GyZ = -0.39d/s
```

After the calibration

2.44.7 Phenomenon Picture

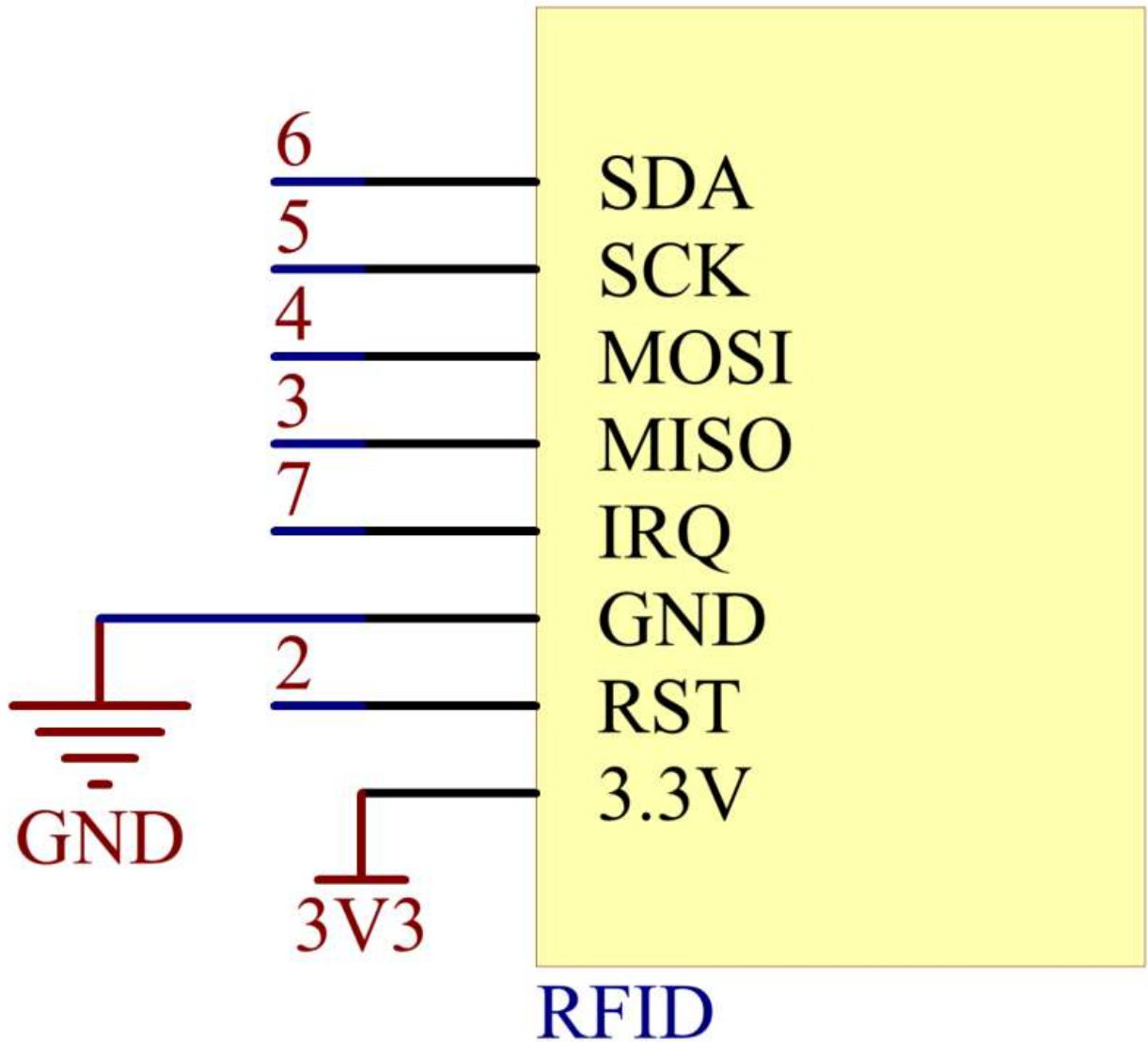


2.45 2.35 RFID-RC522 Module

2.45.1 Overview

In this lesson, you will learn how to use RFID Module. RFID is the abbreviation of Radio Frequency Identification. Its working principle is to carry on the contactless data communication between the reader and the label to achieve the goal of identifying the target. The application of RFID is very extensive, currently the typical applications are animal chips, immobilizer, access control, parking control, production chain automation, material management and so on.

2.45.4 Schematic Diagram



2.45.5 Code

Note:

- You can open the file `2.35_RFID.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\2.35_RFID` directly.
- Or copy this code into Arduino IDE 1/2.
- Then *Upload the Code* to the board.
- Please make sure you have added the library called `rfid1`, detailed tutorials refer to *Add Libraries*.

Uploaded the codes to the Mega2560 board, you can get your RFID card (secret key) close to the RFID Reader. The module will read the card information and then print it on the serial monitor.

2.45.6 Code Analysis

The functions of the module are included in the library rfid1.h.

```
#include <rfid1.h>
```

Library Functions

```
RFID1
```

Create a new instance of the rfid1 class that represents a particular RFID module attached to your Arduino .

```
void begin (IRQ_PIN, SCK_PIN, MOSI_PIN, MISO_PIN, SDA_PIN, RST_PIN)
```

Pin configuration.

- IRQ_PIN, SCK_PIN, MOSI_PIN, MISO_PIN: the pins used for the SPI communication.
- SDA_PIN: Synchronous data adapter.
- RST_PIN: The pins used for reset.

```
void init ()
```

Initialize the RFID.

```
uchar request (uchar reqMode, uchar *TagType);
```

Search card and read card type, and the function will return the current read status of RFID and return MI_OK if succeeded.

- reqMode: Search methods. PICC_REQIDL is defined that 0x26 command bits (Search the cards that does not in the sleep mode in the antenna area).
- *TagType: It is used to store card type, and its value can be 4byte (e.g. 0x0400).

```
char * readCardType (uchar *TagType)
```

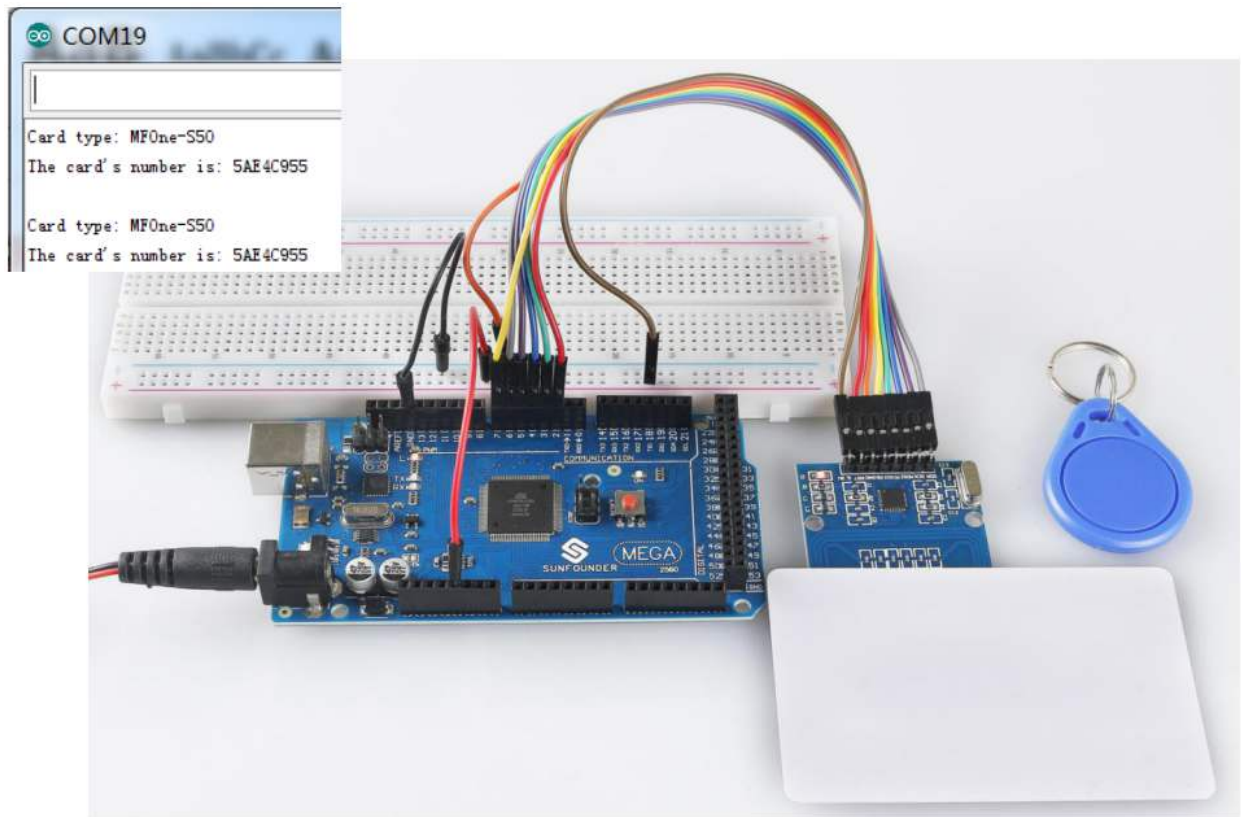
This function decodes the four-digit hexadecimal number of *tagType into the specific card type and returns a string. If passed 0x0400, "MFOne-S50" will be returned.

```
uchar anticoll (uchar *serNum);
```

Prevent conflict, and read the card serial number. The function will return the current reading status of RFID. It returns MI_OK if succeeded.

- *serNum: It is used to store the card serial number, and return the 4 bytes card serial number. The 5th byte is recheck byte(e.g. e.g. my magnetic card ID is 5AE4C955).

2.45.7 Phenomenon Picture



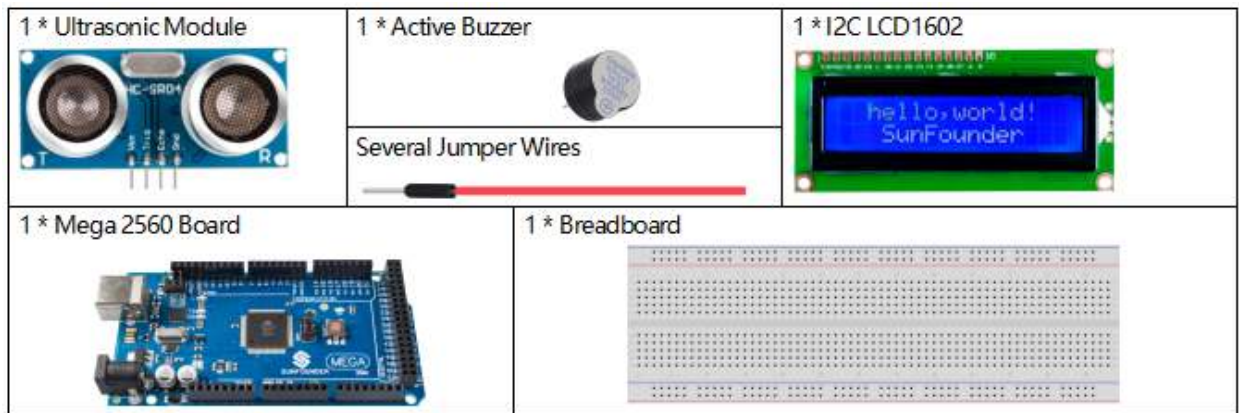
3.Funny Projects

2.46 3.1 Reversing Aid

2.46.1 Overview

With the development of science and technology, a lot of high-tech products have been installed in cars, among which the reversing assist system is one of them. Here we use ultrasonic sensors, LCD, LED and buzzer to make a simple ultrasonic reversing assist system.

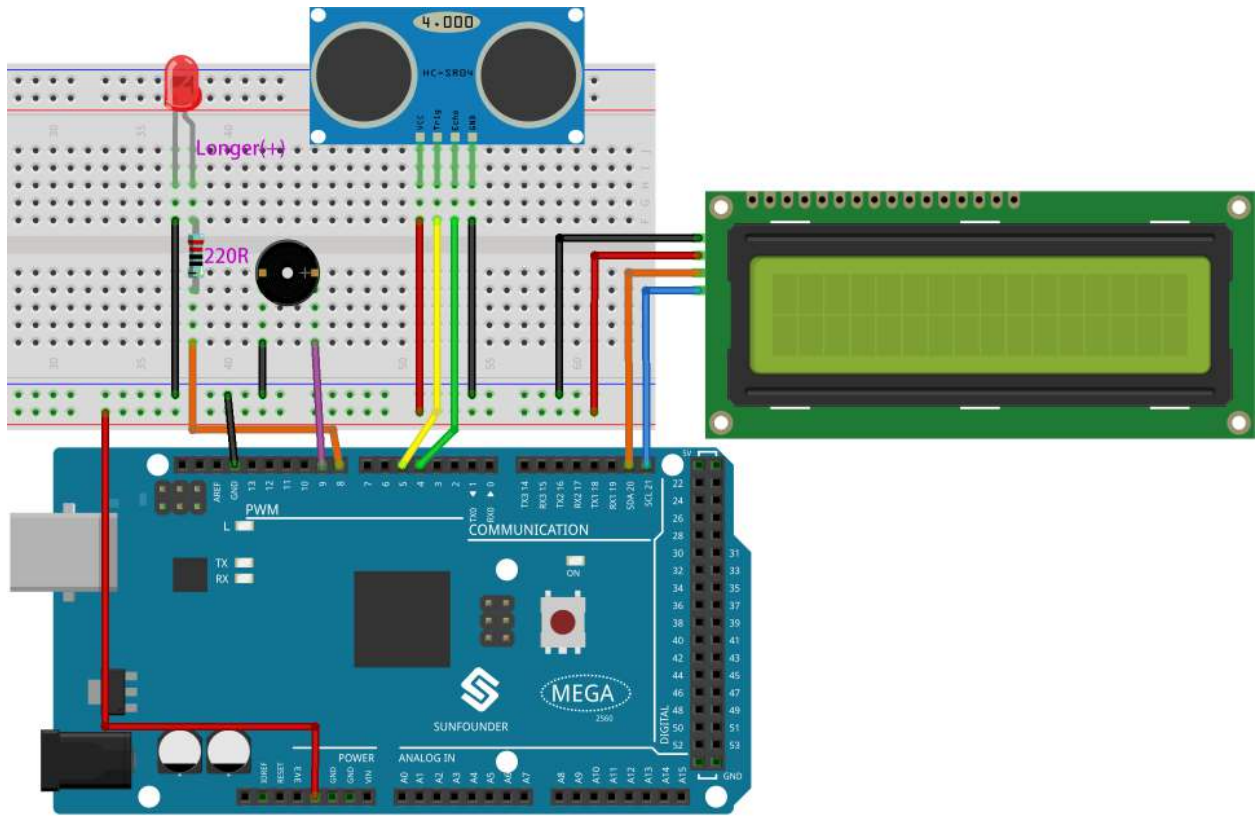
2.46.2 Components Required



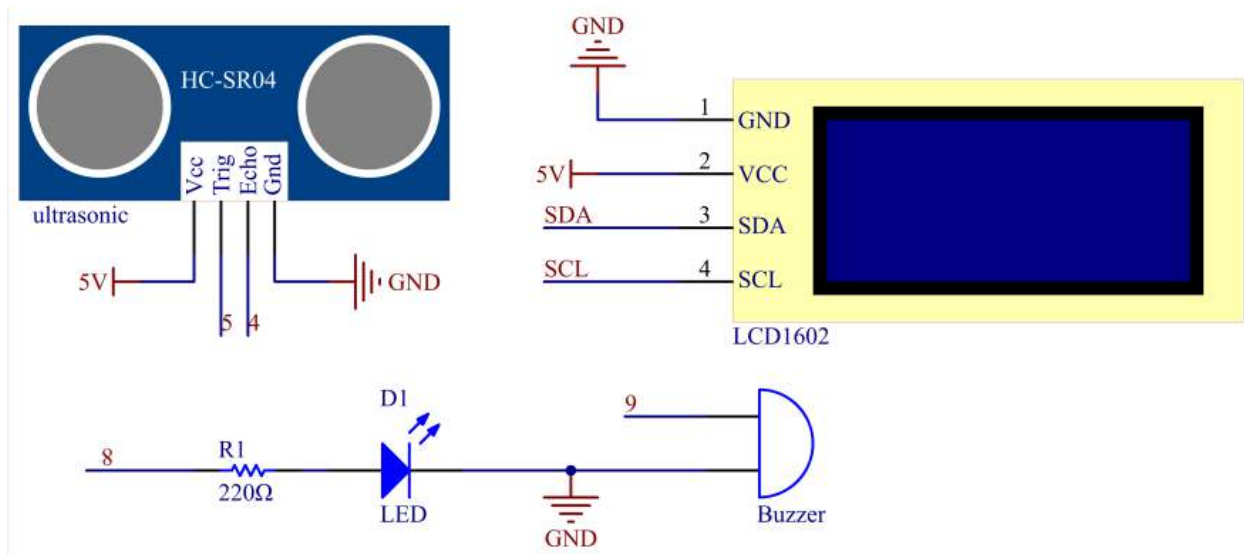
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Buzzer*
- *I2C LCD1602*
- *Ultrasonic Module*

2.46.3 Fritzing Circuit

In this example, the wiring is shown below.



2.46.4 Schematic Diagram



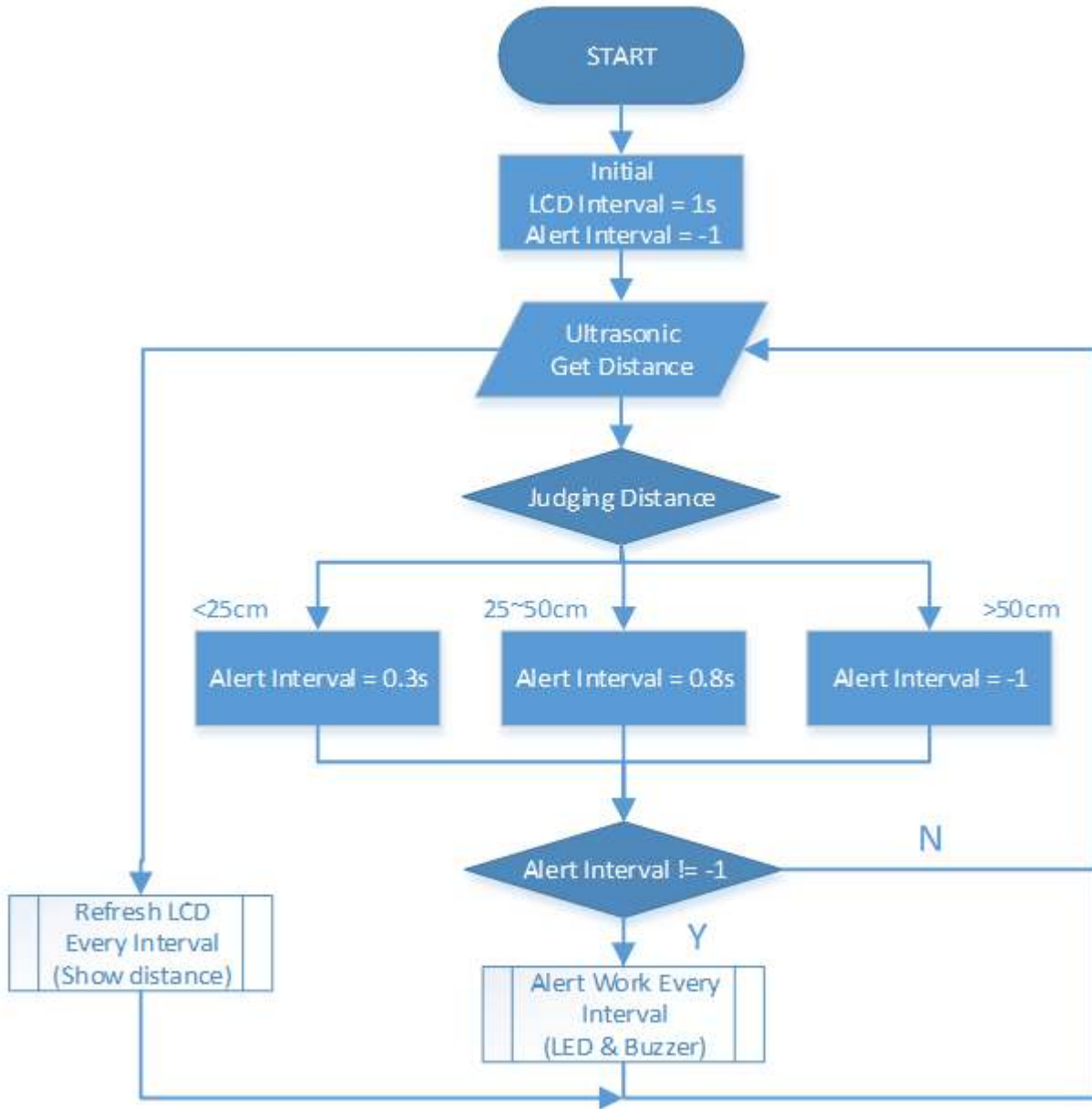
2.46.5 Code

Note:

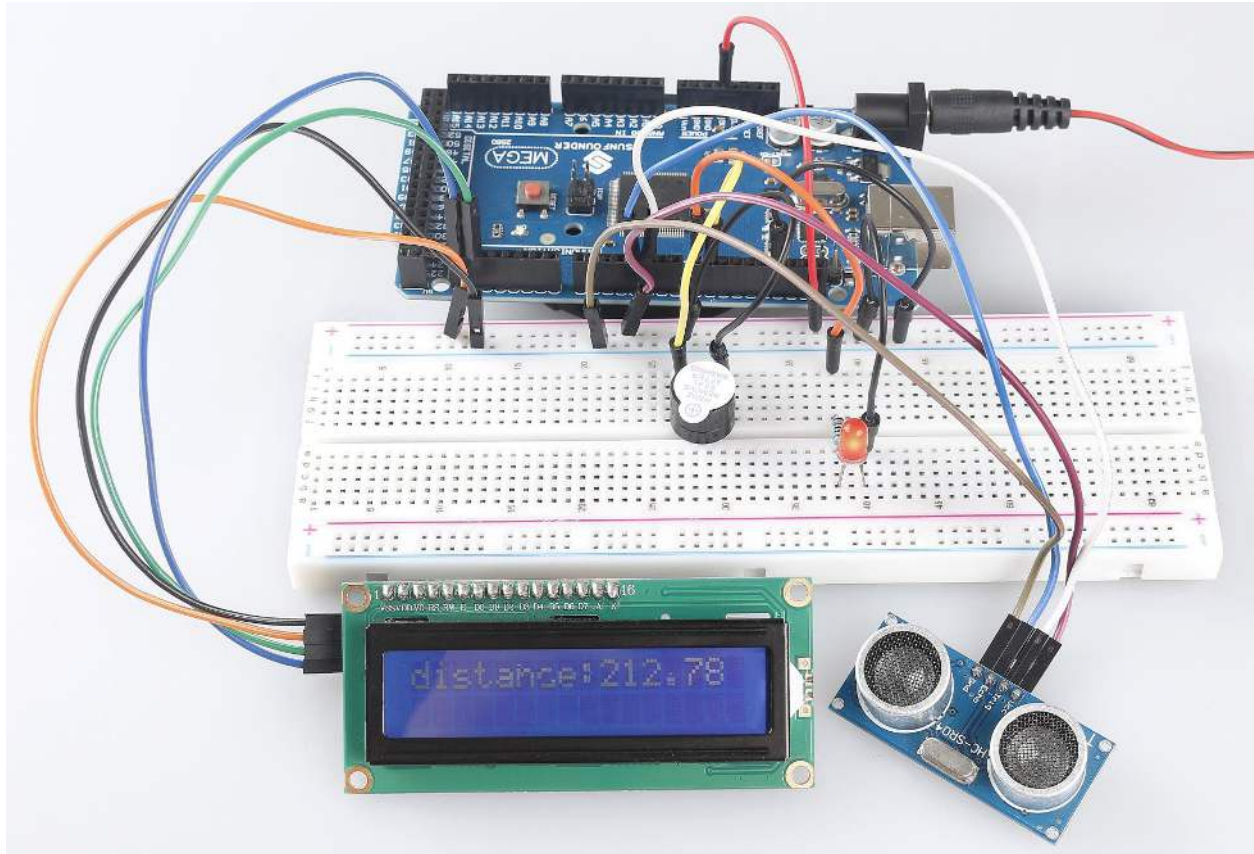
- You can open the file `3.1_reversingAid.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\3.1_reversingAid` directly.
 - Or copy this code into Arduino IDE 1/2.
 - Then *Upload the Code* to the board.
 - Please make sure you have added the `LiquidCrystal_I2C` library, detailed tutorials refer to *Add Libraries*.
-

2.46.6 Example Explanation

In this project, we need to avoid the interference between the LCD screen and the alarm system as much as possible (for example, the LED flicker time is too long and the LCD refresh is delayed), so please avoid using the `delay()` statement and use two separate intervals to control the working frequency of the LCD and alarm system respectively. Its workflow is shown in the flow chart. For analysis of Interval function, refer to *1.11 Interval*.



2.46.7 Phenomenon Picture

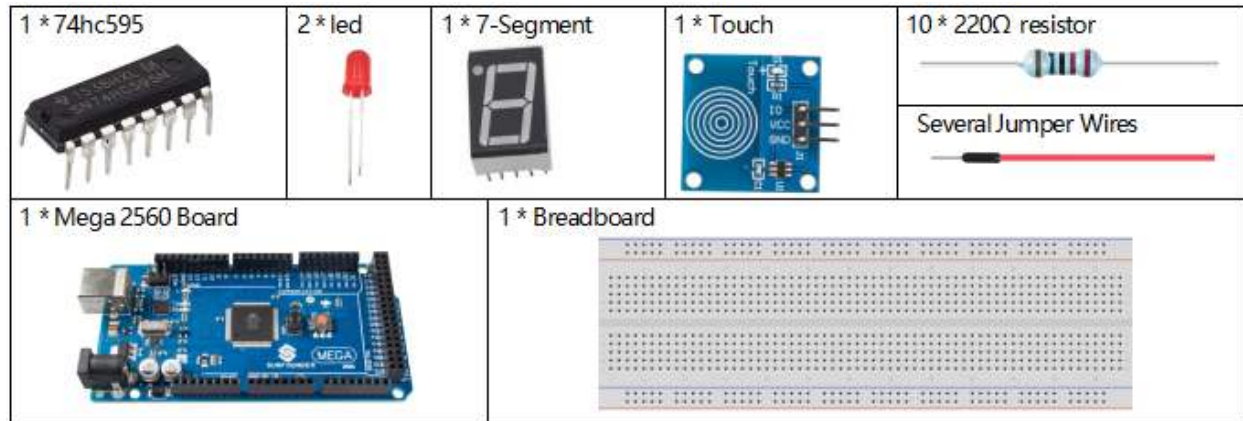


2.47 3.2 Pedestrian Crossing Button

2.47.1 Overview

When pedestrians cross the street, they just need to touch the button on the lamppost of the roadside signal lamp, and the green light above the traffic lane will turn into red then pedestrians can pass safely. Thus, the hard situation of citizens crossing the street is comprehensively resolved. At the same time, when there is no pedestrian to press, the light above the lane that is set for vehicles to pass will always be green, thus greatly improving the use efficiency of the road and traffic capacity.

2.47.2 Components Required

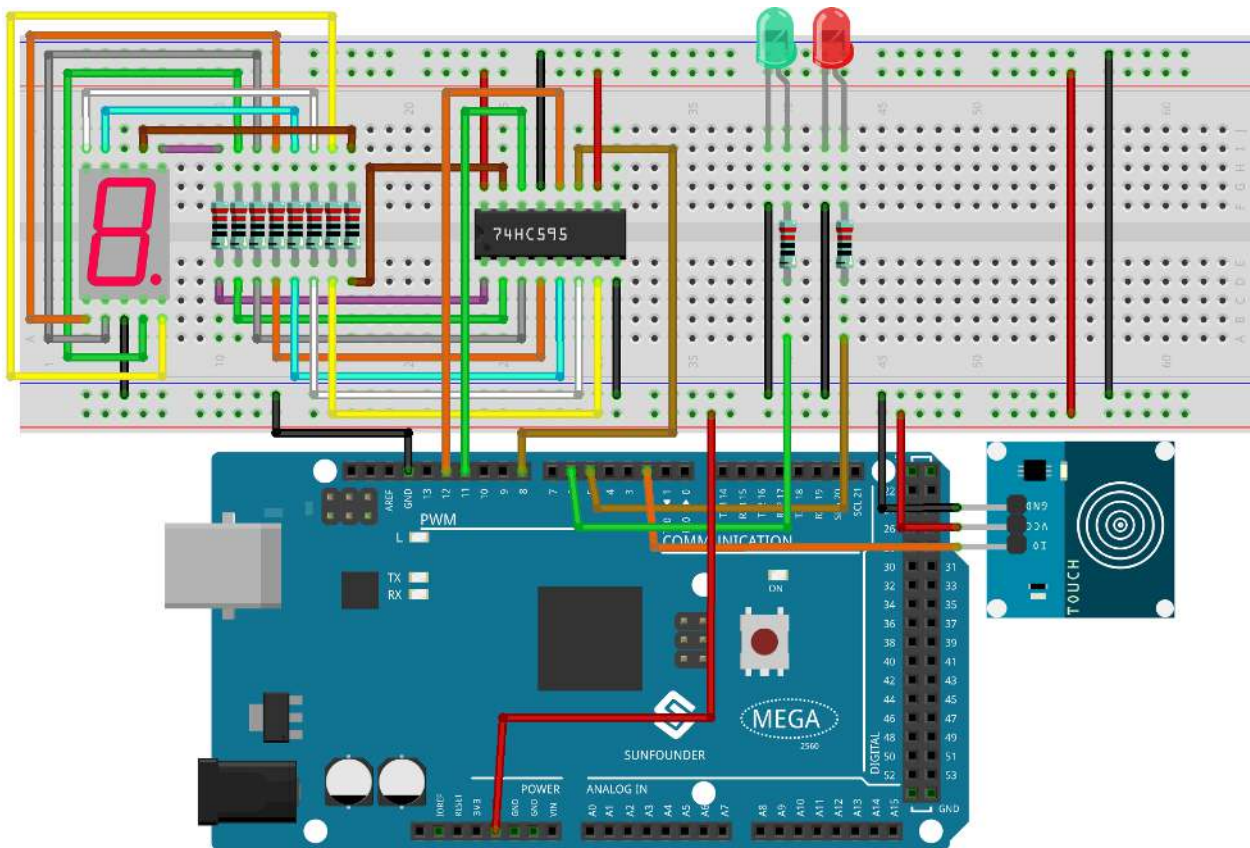


- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Resistor*
- *LED*
- *74HC595*
- *7-segment Display*
- *Touch Switch Module*

2.47.3 Fritzing Circuit

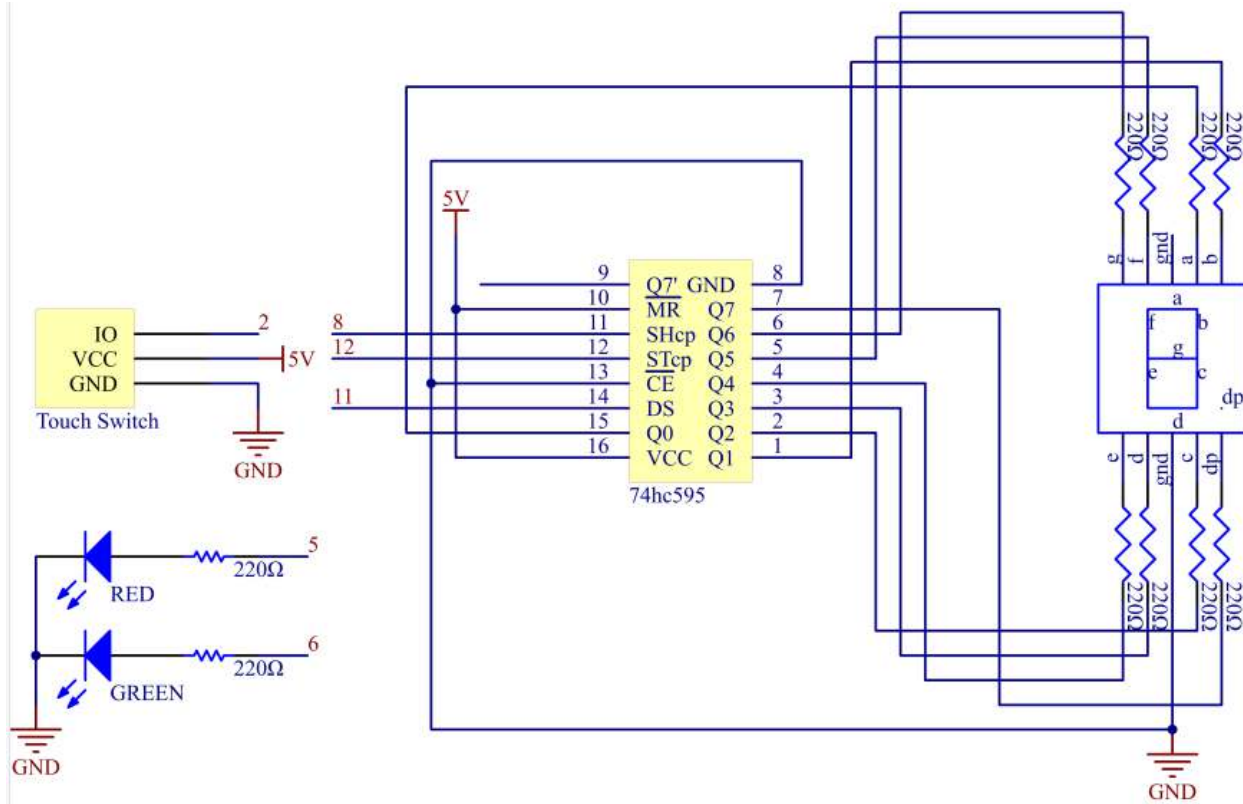
In this example, 74hc595, 7-Segment, LED, touch sensor are to be connected according to the chart.

Mega 2560 Board	74hc595	7-Segment	Mega 2560 Board	Led	Touch
	Q1	b	GND	Short pin	GND
	Q2	c	5V		VCC
	Q3	d	5	Long pin (red)	
	Q4	e	6	Long pin (green)	
	Q5	f	2		IO
	Q6	g			
	Q7	dp			
GND	GND				
5V	MR				
8	SHcp				
12	STcp				
GND	CE				
11	DS				
	Q0	a			
5V	VCC				



fritzing

2.47.4 Schematic Diagram



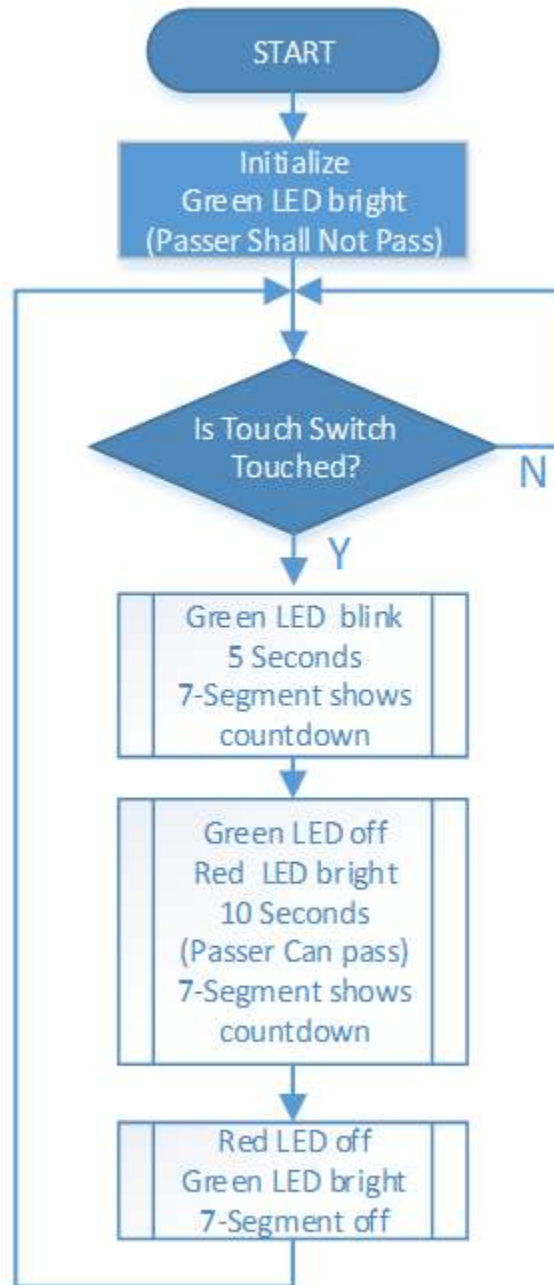
2.47.5 Code

Note:

- You can open the file `3.2_pedestrianCrossingButton.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\3.2_pedestrianCrossingButton` directly.
- Or copy this code into Arduino IDE 1/2.
- Or click **Open Code** to open it in [Web Editor](#).
- Then *Upload the Code* to the board.

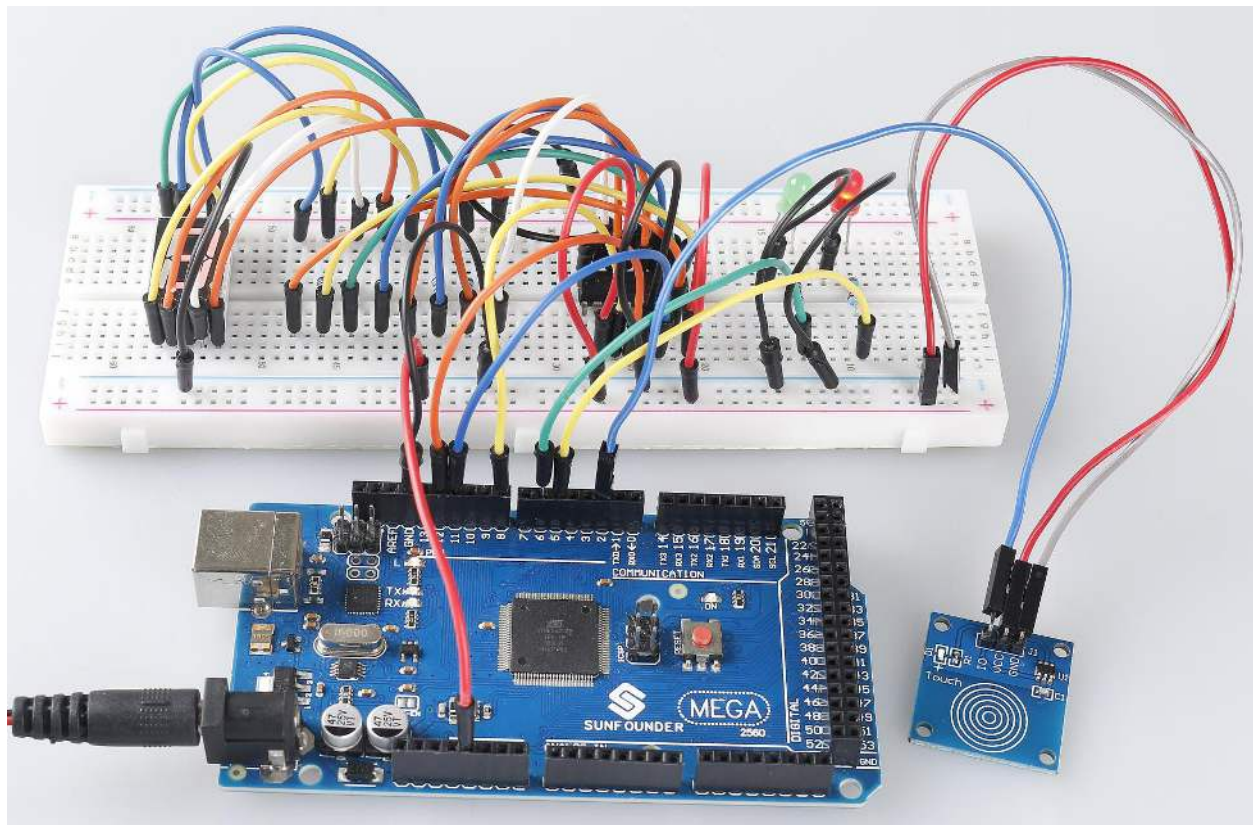
2.47.6 Example Explanation

The workflow of the project is as shown in the flow chart. The function of number display of 7-Segment is realized by writing 8 bit data into 74HC595. When there is a need of displaying 0, the pins abcdef of the segment display will be connected to the high level. The pins, g and dp need to be connected to low level to write `0x3f(B00111111)` in the codes. The complete codes for number display of 7-Segment are as follows.



Numbers	Common Cathode (DP)GFEDCBA	Hex Code
0	00111111	0x3f
1	00001110	0x06
2	01011011	0x5b
3	01001111	0x4f
4	01100110	0x66
5	01101101	0x6d
6	01111101	0x7d
7	00001111	0x07
8	01111111	0x7f
9	01101111	0x6f

2.47.7 Phenomenon Picture














2.48 3.3 Overheat Monitor

2.48.1 Overview

You may want to make an overheat monitoring device that applies to various situations. When the temperature of room is above 30°C in summer, the electric fan or the air conditioner will be turned on automatically. If the refrigerator stops to refrigerate, there will emit alarm. When the CPU gets overheated, the water-cooling system turns on. Next, we will use thermistor, relay, button, rotary encoder and LCD to make an intelligent temperature monitoring device whose threshold is adjustable. You can make it suitable for the scene you want by inserting different peripherals into the relay and using a rotary encoder to adjust the high temperature threshold.

2.48.2 Components Required

1 * Rotary Encoder 	1 * button 	1 * Relay Module 	1 * thermistor 	1 * led 
1 * 220Ω resistor 	1 * LCD1602 			
2 * 10KΩ resistor 				
Several Jumper Wires 				
1 * Mega 2560 Board 	1 * Breadboard 			

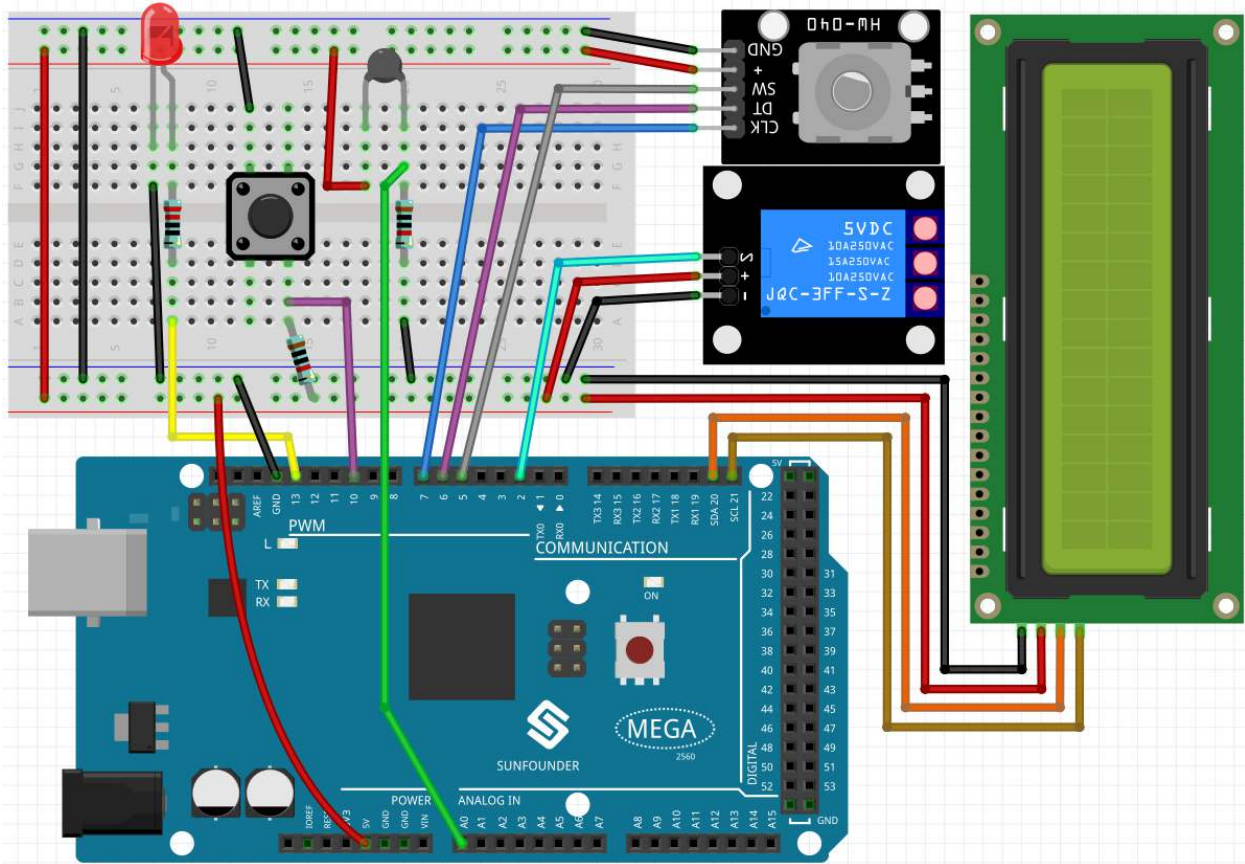
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Resistor*
- *LED*
- *Button*

- *I2C LCD1602*
- *Rotary Encoder Module*

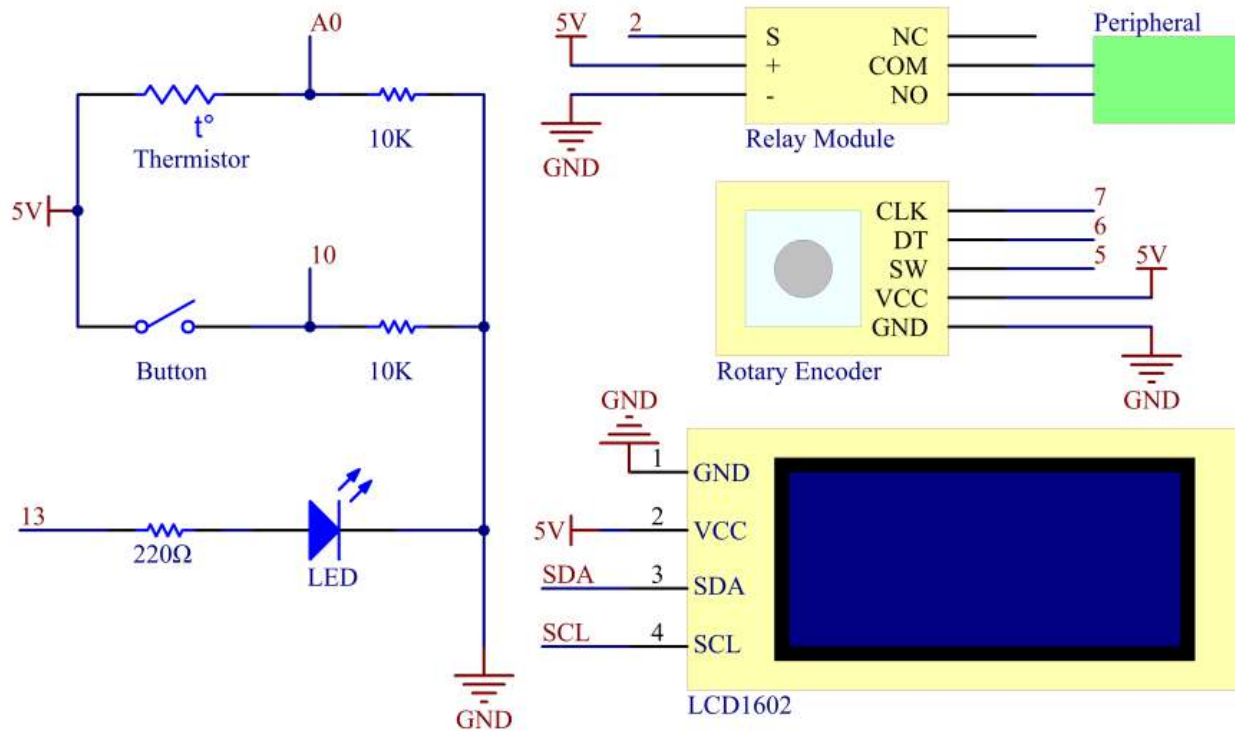
2.48.3 Fritzing Circuit

In this example, the component modules are connected as shown in the table.

Mega 2560 Board	LCD1602	Rotary Encoder	Relay Module	LED	thermistor	button
GND	GND	GND	(-)	(-)	(-)	(-)
5V	VCC	VCC	(+)			
SDA	SDA					
SCL	SCL					
7		CLK				
6		DT				
5		SW				
2			S			
13				(+)		
A0					(+)	
10						(+)



2.48.4 Schematic Diagram



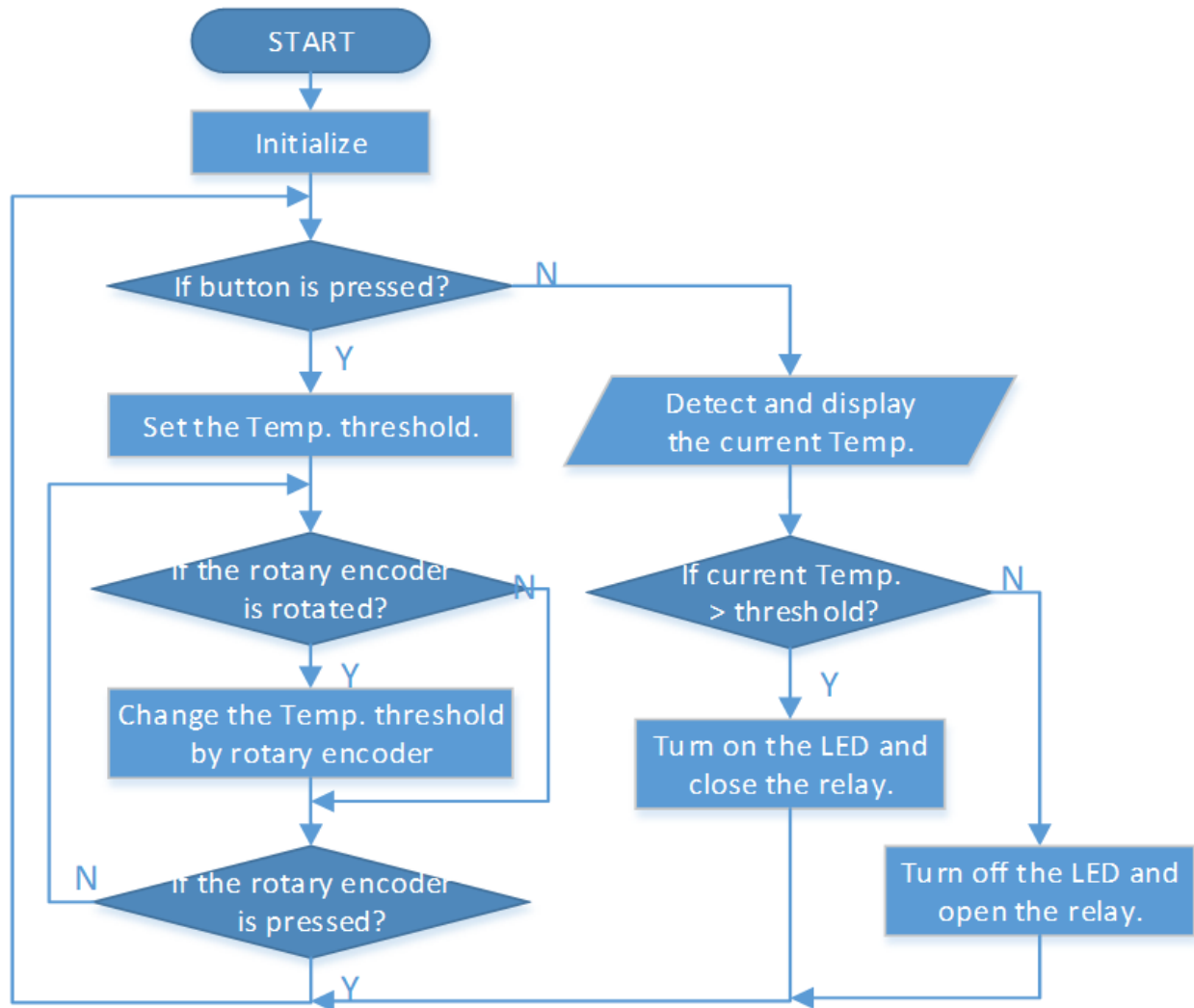
2.48.5 Code

Note:

- You can open the file `3.3_overheatMonitor.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\3.3_overheatMonitor` directly.
- Or copy this code into Arduino IDE 1/2.
- Then *Upload the Code* to the board.
- Please make sure you have added the `LiquidCrystal_I2C` library, detailed tutorials refer to *Add Libraries*.

2.48.6 Example Explanation

The flow diagram of the project is as follows:



By using EEPROM.h library, the high temperature threshold is saved in EEPROM to avoid the value reset after the restart of MCU.

Library Functions

```
void write(address, value)
```

Write a byte to the EEPROM.

```
void Read(address)
```

Reads a byte from the EEPROM. Locations that have never been written to have the value of 255.

```
void update(address, value)
```

Write a byte to the EEPROM. The value is written only if differs from the one already saved at the same address.

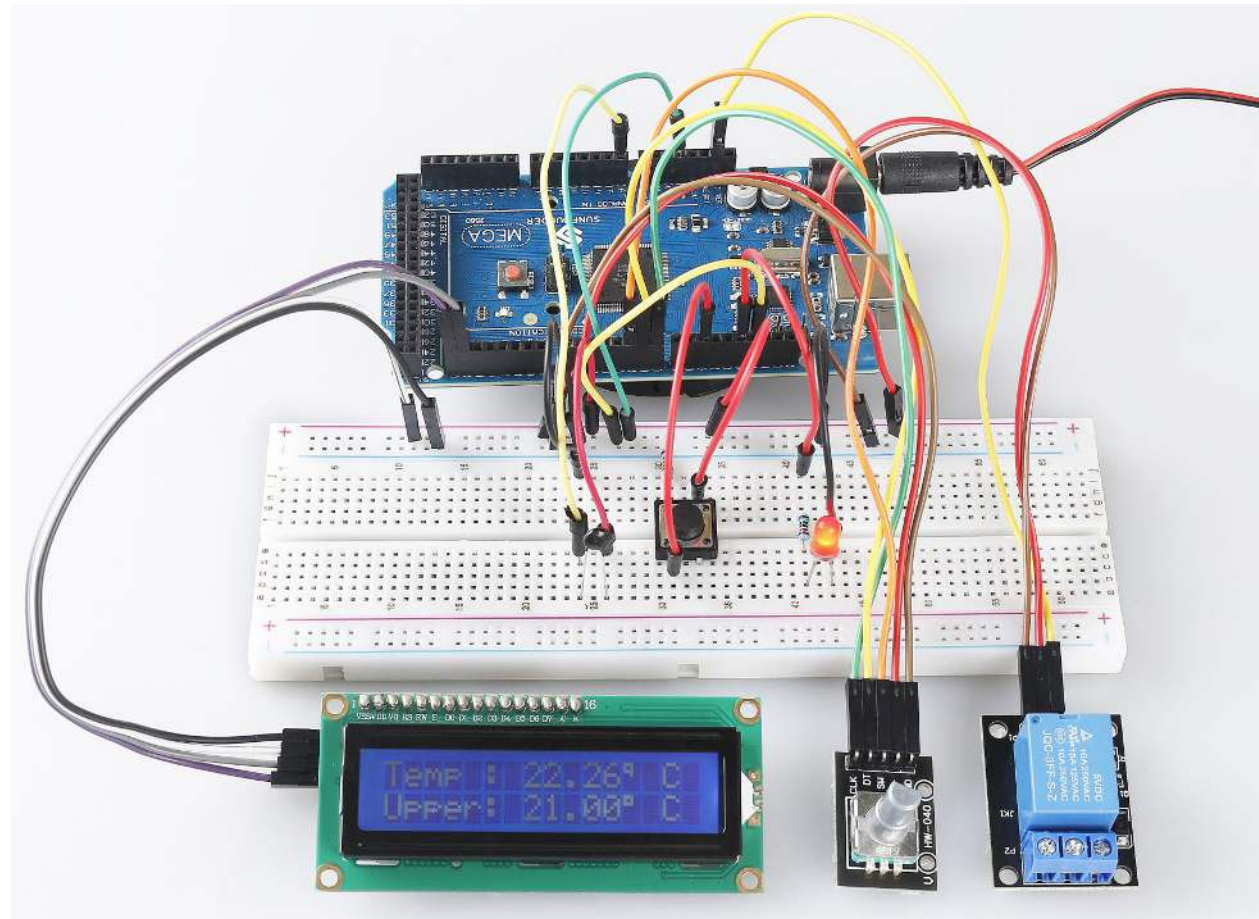
```
void put(address, value)
```

Write any data type or object to the EEPROM.

```
void get (address)
```

Read any data type or object from the EEPROM.

2.48.7 Phenomenon Picture





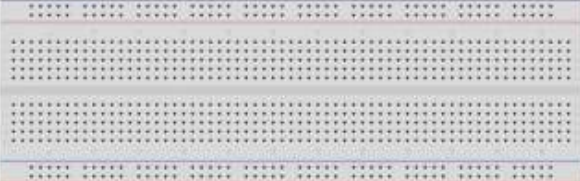


2.49 3.4 Guess Number

2.49.1 Overview

Guessing Numbers is a fun party game where you and your friends take turns inputting a number (0~99). The range will be smaller with the inputting of the number till a player answers the riddle correctly. Then the player is defeated and punished. For example, if the lucky number is 51 which the players cannot see, and the player 1 inputs 50, the prompt of number range changes to 50~99; if the player 2 inputs 70, the range of number can be 50~70; if the player 3 inputs 51, he or she is the unlucky one. Here, we use IR Remote Controller to input numbers and use LCD to output outcomes.

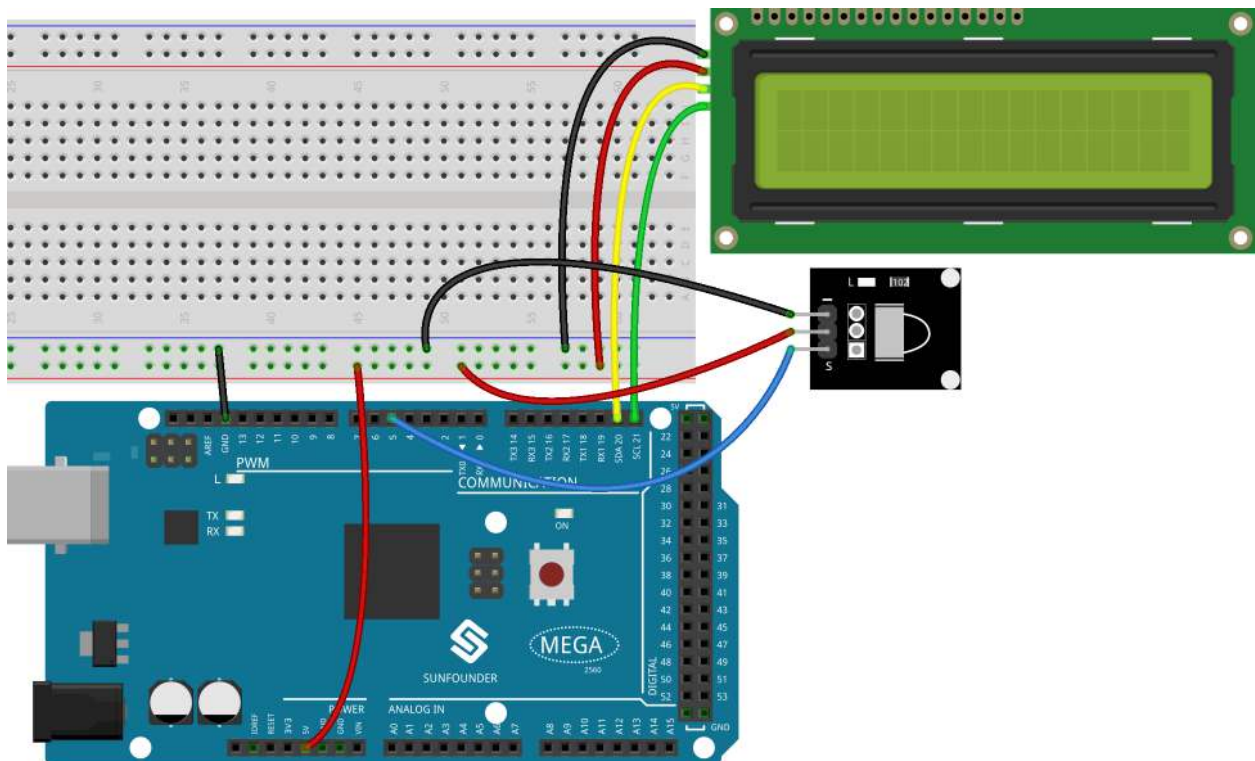
2.49.2 Components Required

<p>1 * LCD1602</p> 	<p>1 * IR Remote Controller</p> 	<p>Several Jumper Wires</p> 
<p>1 * Mega 2560 Board</p> 	<p>1 * Breadboard</p> 	

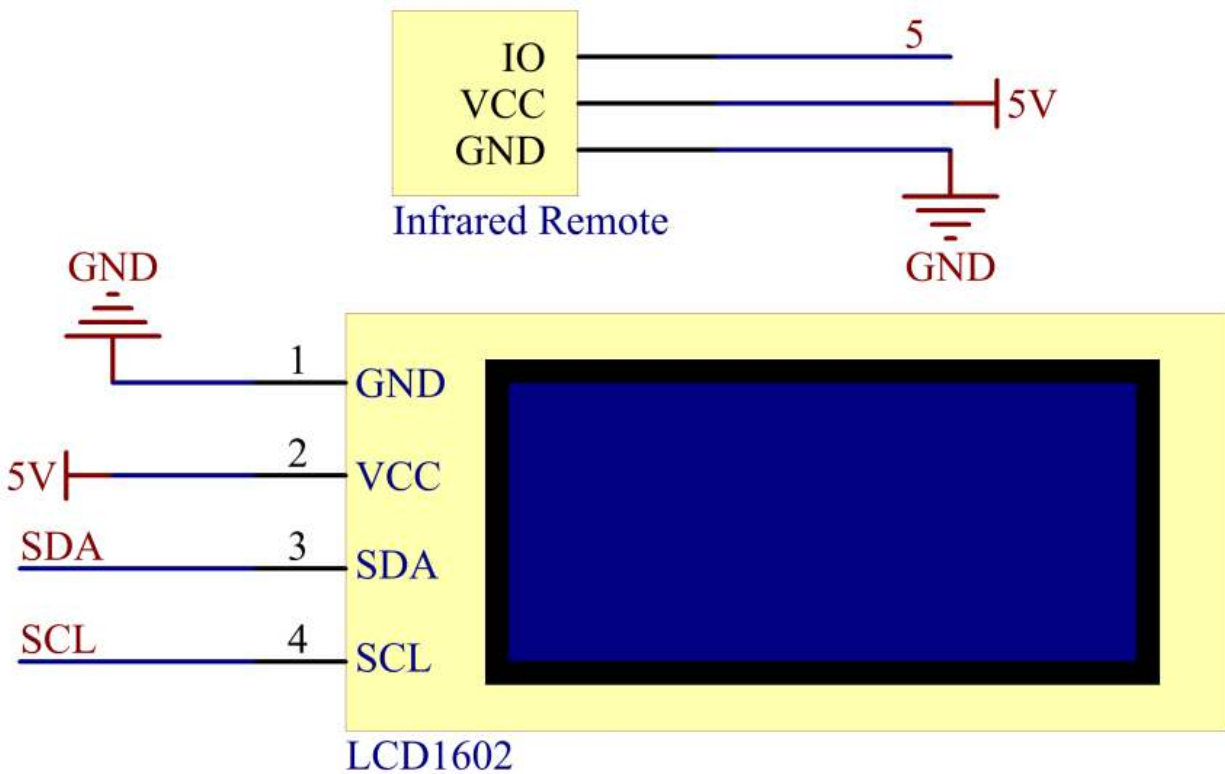
- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *I2C LCD1602*
- *IR Receiver Module*

2.49.3 Fritzing Circuit

In this example, the wiring of LCD1602 and infrared receiving module is as follows.



2.49.4 Schematic Diagram

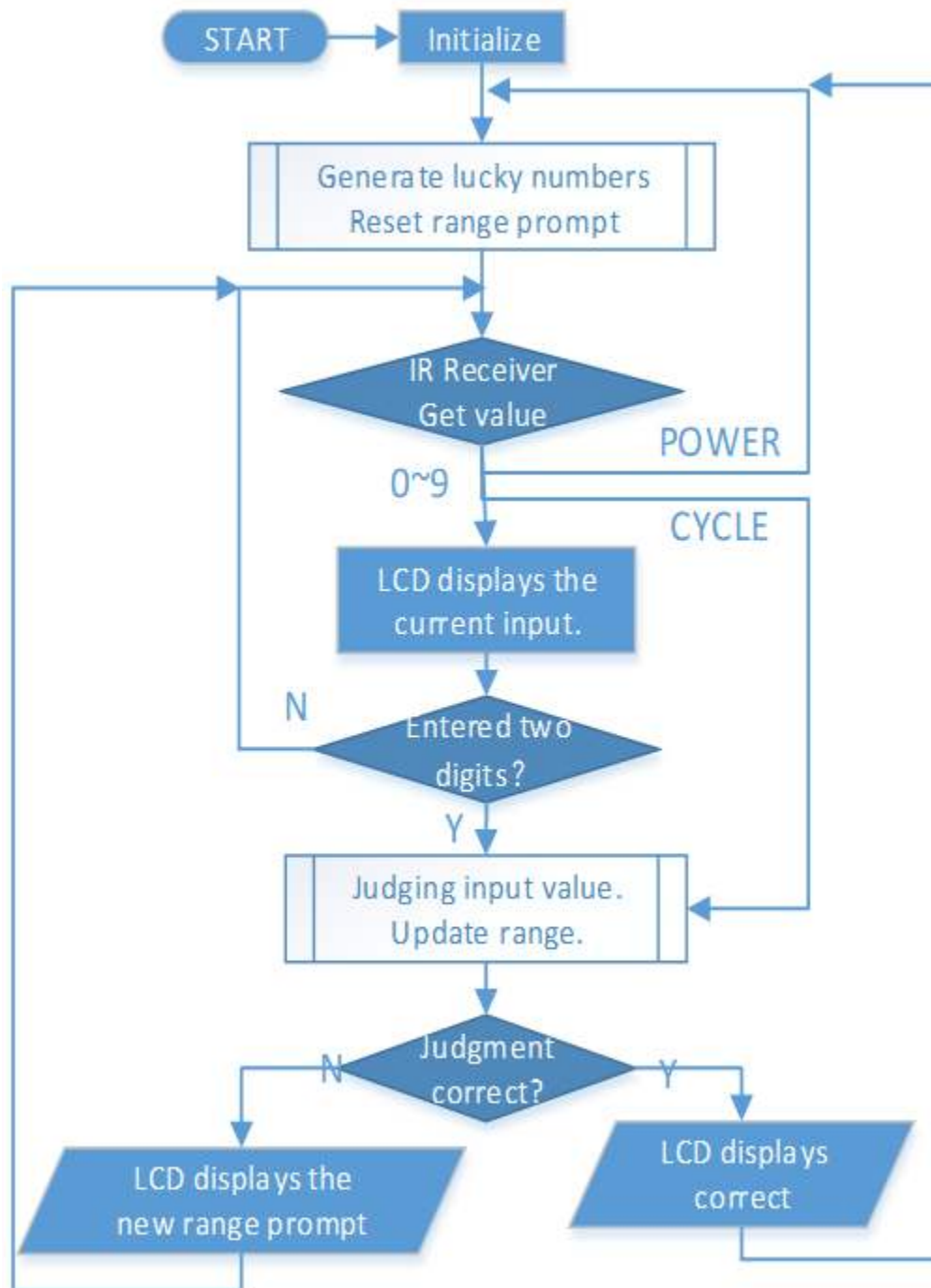


2.49.5 Code

Note:

- You can open the file `3.4_guessNumber.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\3.4_guessNumber` directly.
- Or copy this code into Arduino IDE 1/2.
- Then *Upload the Code* to the board.
- Please make sure you have added the `IRremote` and `LiquidCrystal_I2C` libraries, detailed tutorials refer to *Add Libraries*.

2.49.6 Example Explanation



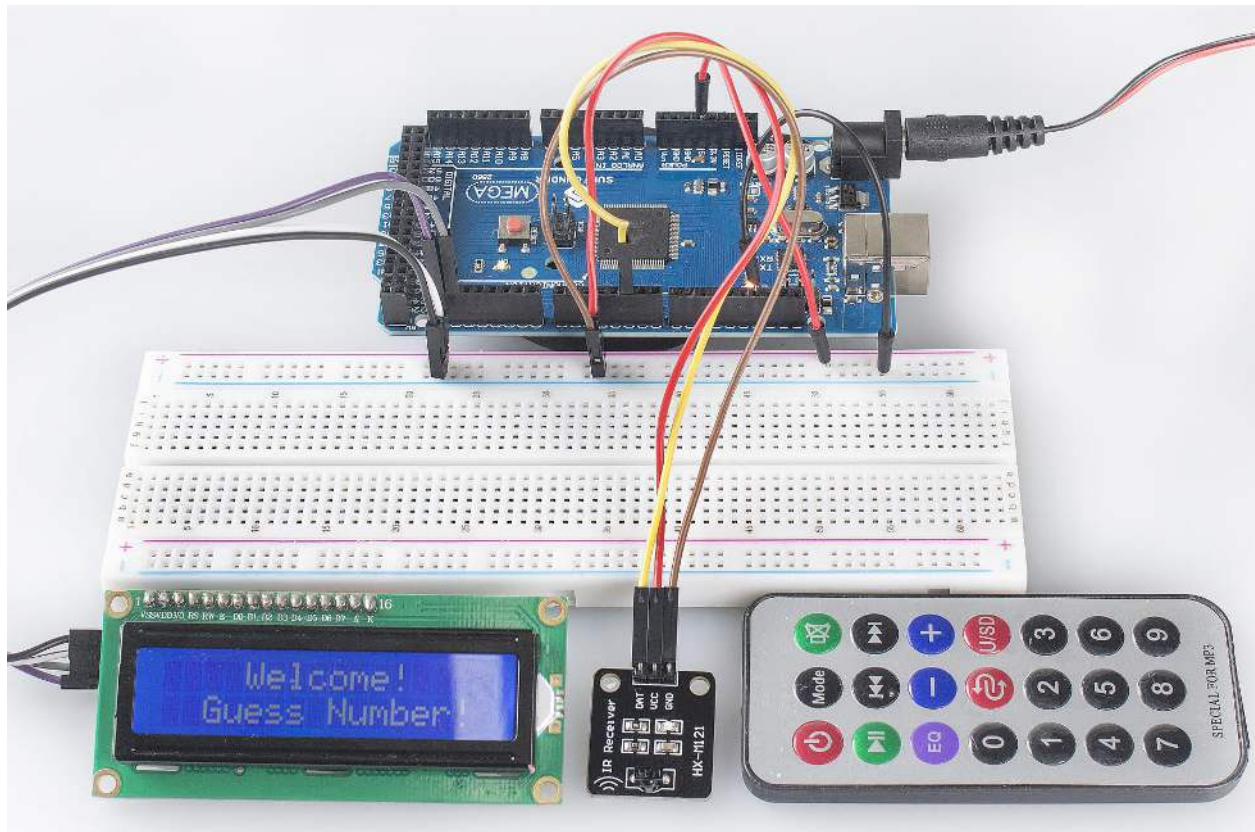
In order to make the number guessing game become vivid and funny, we need to achieve the following functions:

1. The lucky number will be displayed when we start and reset the game, and the number range prompt is reset to 0 ~ 99.
2. LCD will display the number being input and the number range prompt.
3. After inputting two digits, there appears result judgment automatically.

4. If you input a single digit, you can press the CYCLE key (the key at the center of the Controller) to start the result judgment.
5. If the answer is not guessed, the new number range prompt will be displayed (if the lucky number is 51 and you enter 50, the number range prompt will change to 50~99).
6. The game is automatically reset after the lucky number is guessed, so that the player can play a new round.
7. The game can be reset by directly pressing the POWER button (the button in the upper left corner).

In conclusion, the work flow of the project is shown in the flow chart.

2.49.7 Phenomenon Picture









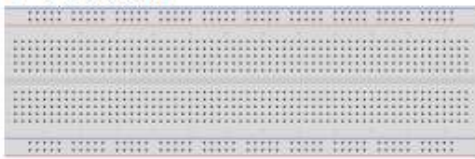



2.50 3.5 Access Control System

2.50.1 Overview

The access control system is the system that is used to control the entrance channel, which is developed on the basis of the traditional door lock. The traditional mechanical door lock is only a simple mechanical device, and no matter how reasonable the structural design is, how strong the material is, people can always open it through various means. The key to the entrance and exit (like an office building, a hotel room) is cumbersome. If the key is missed or replaced, the lock is to be replaced with the key. In order to solve these problems, the electronic magnetic card lock and the electronic coded lock are present, which has raised the management level of the access channel to a certain extent, and then the channel management enters into the electronic age.

2.50.2 Components Required

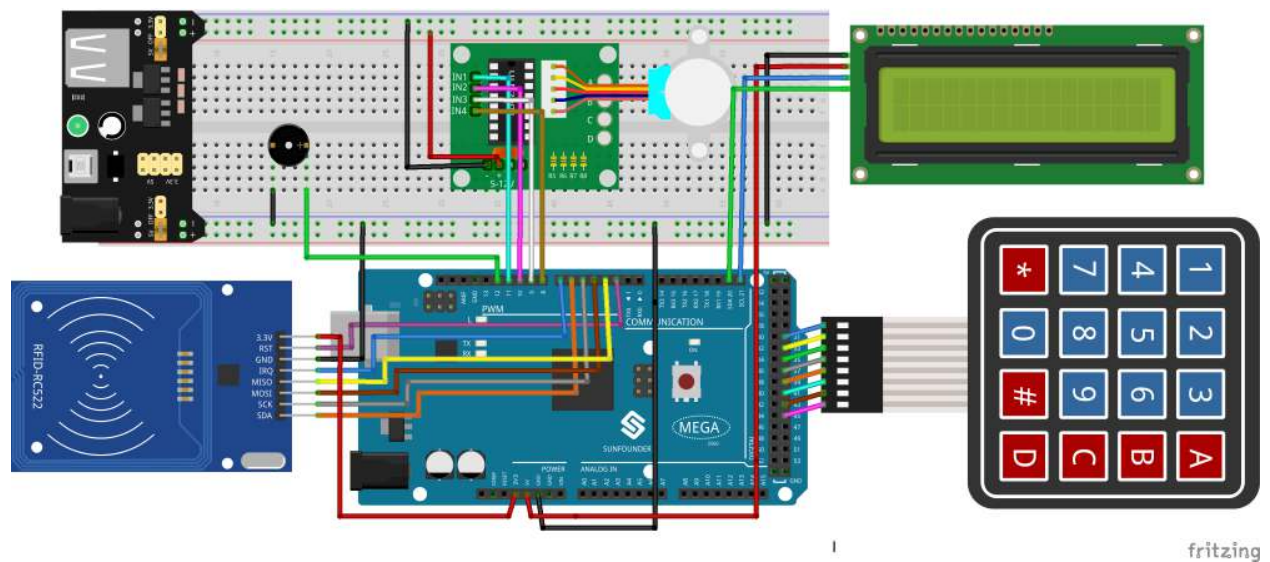
<p>1 * RFID</p> 	<p>1 * Keypad</p> 	<p>1 * Power Supply Module</p> 	
<p>1 * I2C LCD1602</p> 	<p>1 * ULN2003</p> 	<p>1 * Buzzer</p> 	<p>1 * Stepper Motor</p> 
<p>1 * Mega 2560 Board</p> 	<p>1 * Breadboard</p> 	<p>Several Jumper Wires</p> 	

- *SunFounder Mega Board*
- *Breadboard*
- *Jumper Wires*
- *Buzzer*
- *I2C LCD1602*
- *Stepper Motor*
- *Keypad*
- *Power Supply Module*

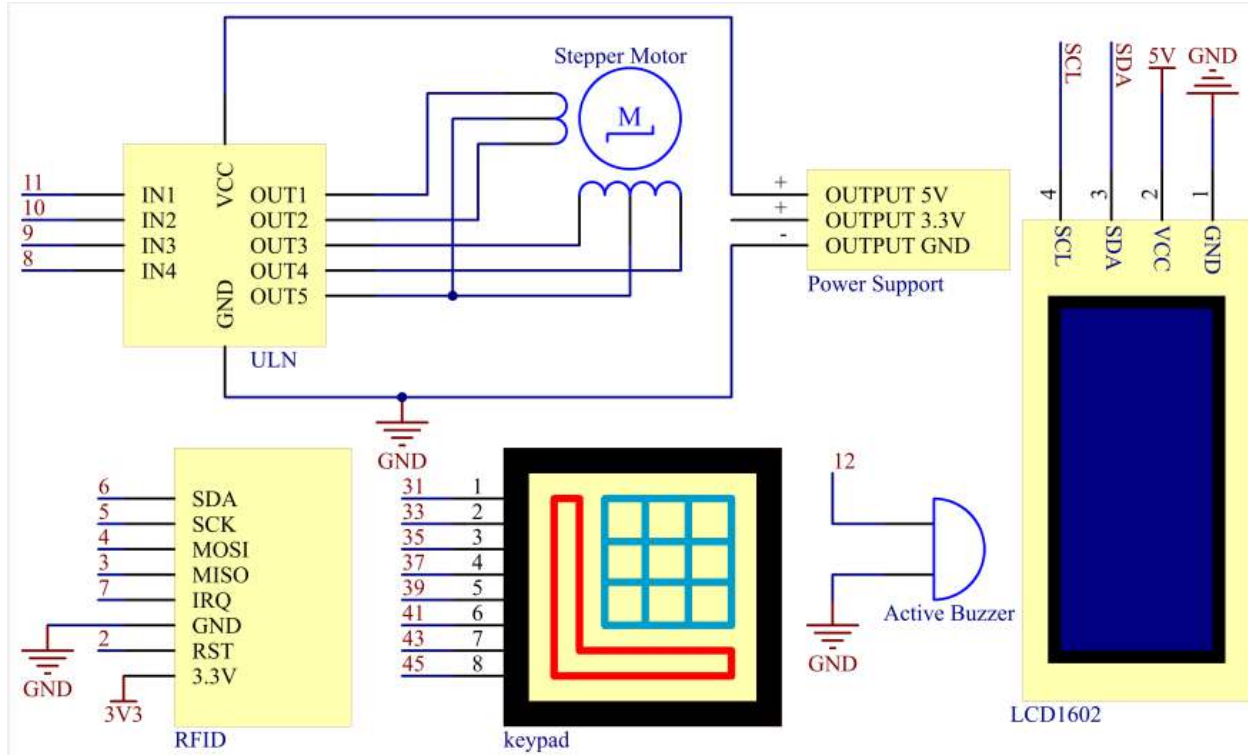
2.50.3 Fritzing Circuit

In this example, Power Supply Module is used to power the breadboard. We get the GND of Mega 2560 Board connected to the cathode rail of the breadboard, GND of ULN2003 to the cathode rail of the breadboard, VCC to 5V OUTPUT of Power Supply and the Stepper Motor to OUT1-OUT5 of ULN.

Mega 2560 Board	LCD1602	RFID	Buzzer	Mega 2560 Board	ULN	Keypad
GND	GND	GND	(-)	GND	GND	
5V	VCC			5V	VCC	
SDA	SDA			11	IN1	
SCL	SCL			10	IN2	
3.3V		3.3V		9	IN3	
6		SDA		8	IN4	
5		SCK		31		1
4		MOSI		33		2
3		MISO		35		3
7		IRQ		37		4
2		RST		39		5
12			(+)	41		6
				43		7
				45		8



2.50.4 Schematic Diagram



2.50.5 Code

Note:

- You can open the file `3.5_accessControlSystem.ino` under the path of `sunfounder_vincent_kit_for_arduino\code\3.5_accessControlSystem` directly.
- Or copy this code into Arduino IDE 1/2.
- Then *Upload the Code* to the board.
- Please make sure you have added the `RFID1`, `Keypad` and `LiquidCrystal_I2C` libraries, detailed tutorials refer to *Add Libraries*.

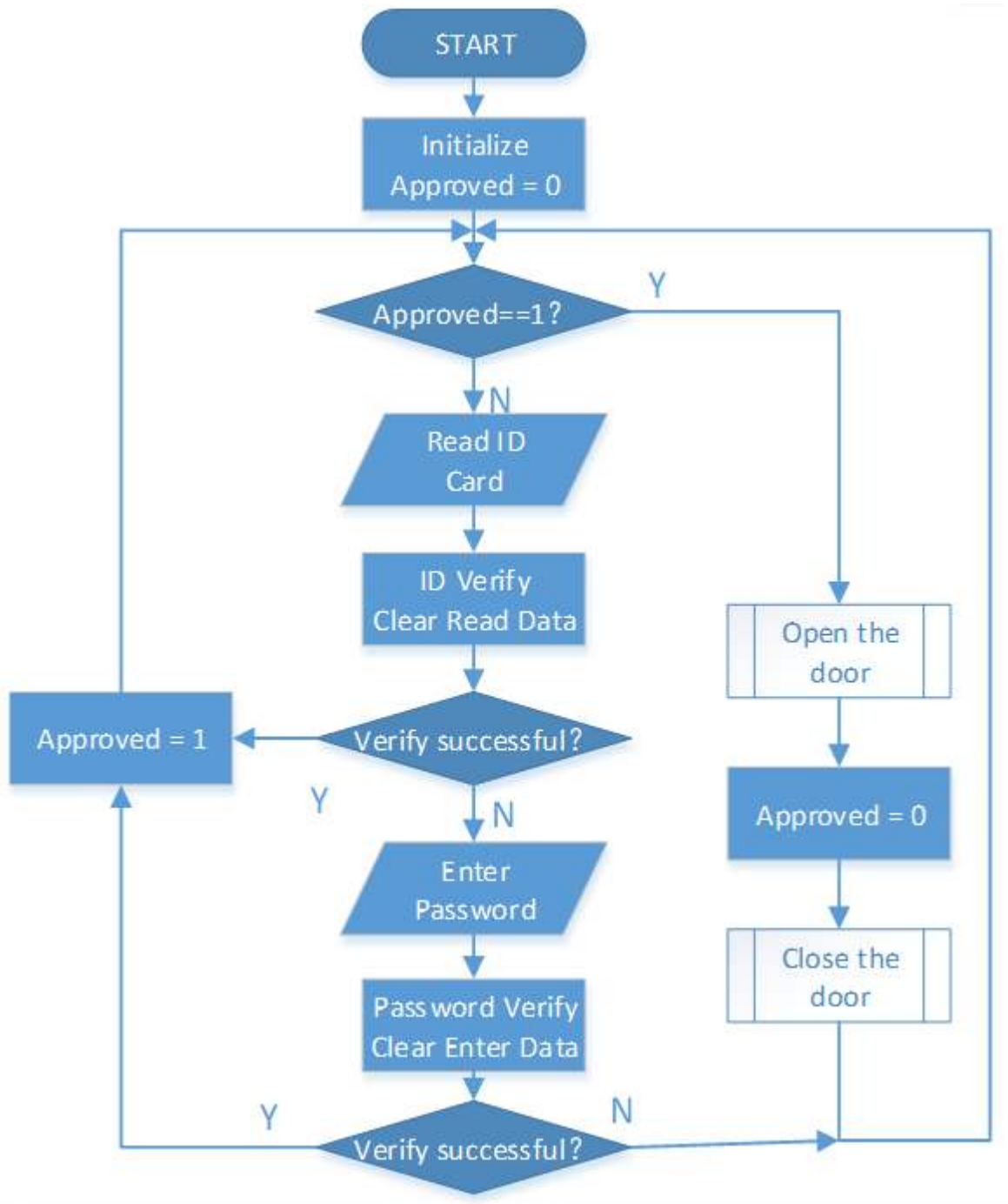
2.50.6 Example Explanation

The workflow of the access control system is shown in the flow chart.

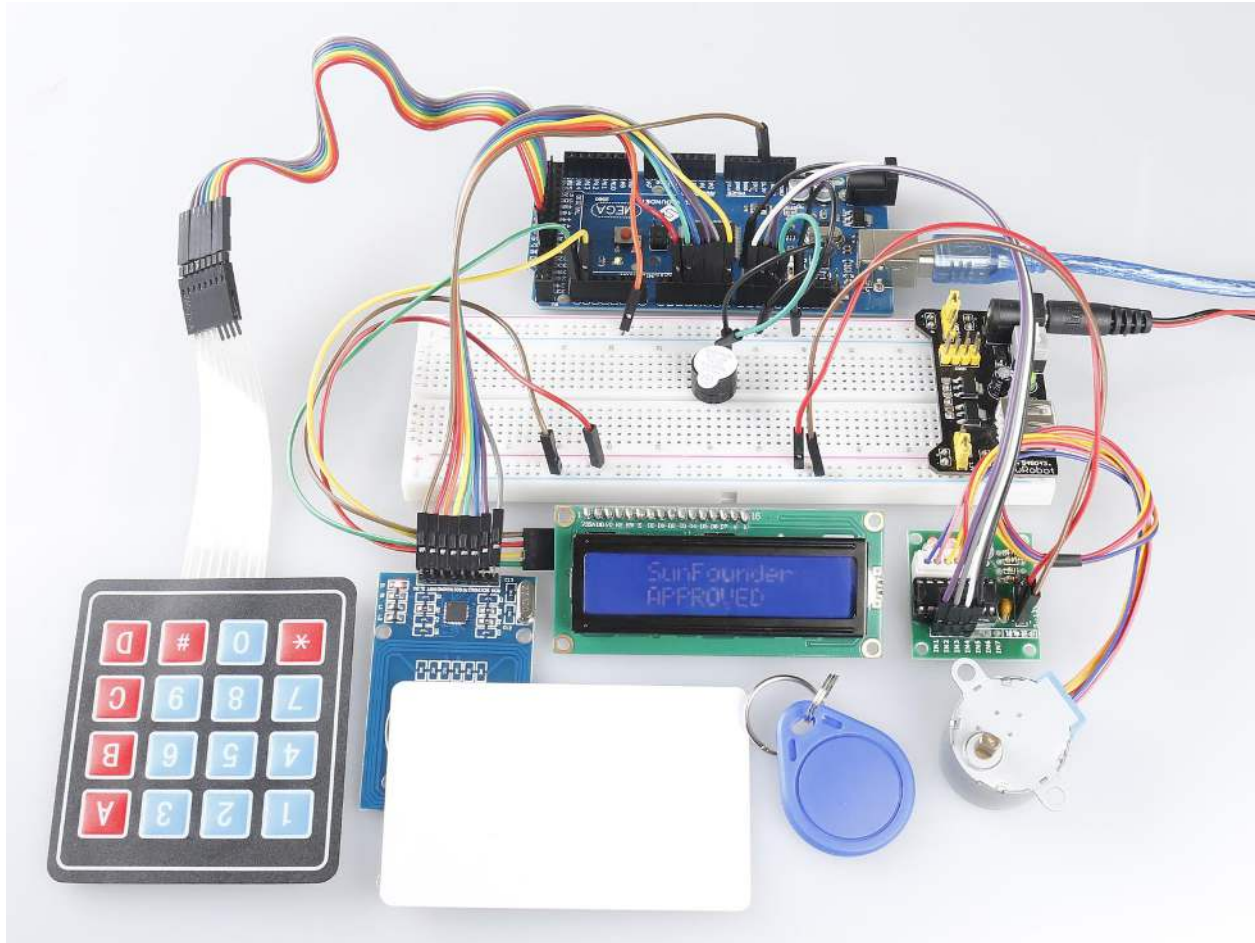
When there is no authorized ID identified (the variable “Approved” equals to 0), the device will perform functions of ID identification and password identification.

If the authorized ID is identified (“Approved” equals to 1), the door will open. After that, the door will be closed a few seconds later and the identified status will be reset (“Approved” equals to 0).

In addition to the core access control function, the project also uses LCD and active buzzer to complete the work of the user interaction system of the access control system.



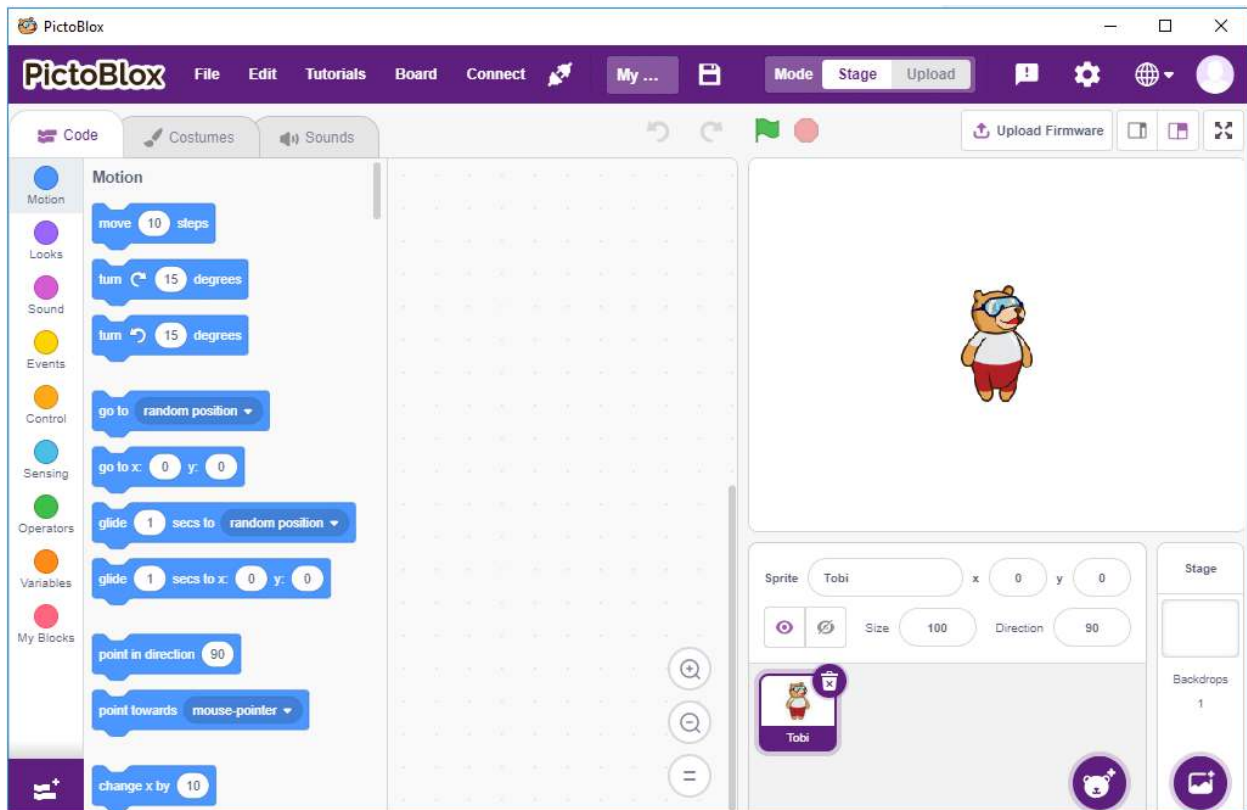
2.50.7 Phenomenon Picture



PLAY WITH SCRATCH

Besides programming on the Arduino IDE with Uno/Mega2560 boards, we can also use them for graphical programming.

Here we recommend programming with Scratch, but the official Scratch is currently only compatible with Raspberry Pi, so we have partnered with a company, STEMPedia, who has developed a Scratch 3 based graphical programming software for Arduino boards (Uno, Mega2560 and Nano) - [PictoBlox](#).



It keeps the basic functions of Scratch 3, but also adds control boards, such as Arduino Uno, Mega, Nano, ESP32, Microbit and STEMPedia homemade main boards, which can use external sensors, robots to control the sprites on the stage, with strong hardware interaction capabilities.

In addition, it has AI and machine learning, even if you do not have much programming foundation, you can learn and use these popular and high-tech.

Just drag and drop the Scratch coding blocks and make cool games, animations, interactive projects, and even control robots the way you want!

Now let's start the journey of discovery!

1. Get Started

3.1 1.1 Install PictoBlox

Click this link <https://thetempedia.com/product/pictoblox/download-pictoblox/> choose the appropriate Operating System (Windows, macOS, Linux) and follow the steps to install.

STEMpedia

QUARKY **NEW** LEARN EDUCATORS PRODUCTS PROJECTS SHOP SIGN IN

Download PictoBlox

CHOOSE YOUR DEVICE

Click here to choose difference systems

Windows macOS Linux Android

Windows Installer (.exe)

STEP 1: Download the Pictoblox Installer (.exe) for Windows 7 and above ([Release Notes](#)).

WINDOWS INSTALLER 64-BIT V4.1.0

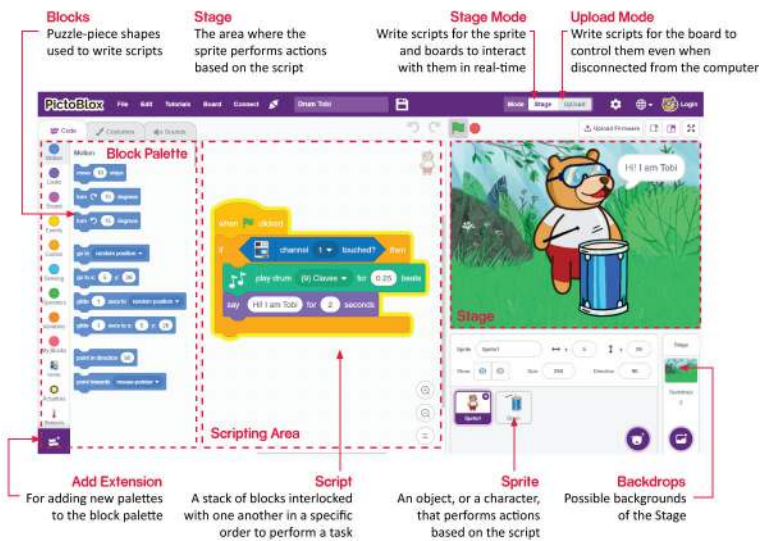
WINDOWS INSTALLER 32-BIT V4.1.0

STEP 2: Run the .exe file.

Some of the device gives the following popup. You don't have to worry, this software is harmless. Click on **More info** and then click on **Run anyway**.

STEP 3: Rest of the installation is straight forward, you can follow the popup and check on the option appropriate for your need.

3.2 1.2 Interface Introduction



Sprites

A sprite is an object, or a character, that performs different actions in a project. It understands and obeys the commands given to it. Each sprite has specific costumes and sounds that you can also customize.

Stage

The stage is the area where the sprite performs actions in backdrops according to your program.

Backdrops

Backdrops are used to decorate the stage. You can choose a backdrop from PictoBlox, draw one yourself or upload an image from your computer.

Script Area

A script is a program or a code in PictoBlox/Scratch lingo. It is a set of “blocks” arranged in a specific order to perform a task or a series of tasks. You can write multiple scripts, all of which can run simultaneously. You can only write scripts in the script area in the center of the screen.

Blocks

Blocks are like pieces of a puzzle that are used to write programs by simply stacking them together in the script area. Using blocks to write code can make programming easier and reduce the probability of errors.

Block Palette

The block palettes are located in the left area and are named by their functions, such as motion, sound and control. Each palette has different blocks, for example, the blocks in the Motion palette will control the movement of the sprites, and the blocks in the Control palette will control the work of the script based on specific conditions.

There are other kinds of block palettes that can be loaded from the **Add Extension** button located at the bottom left.

Modes

Unlike Scratch, PictoBlox has two modes:

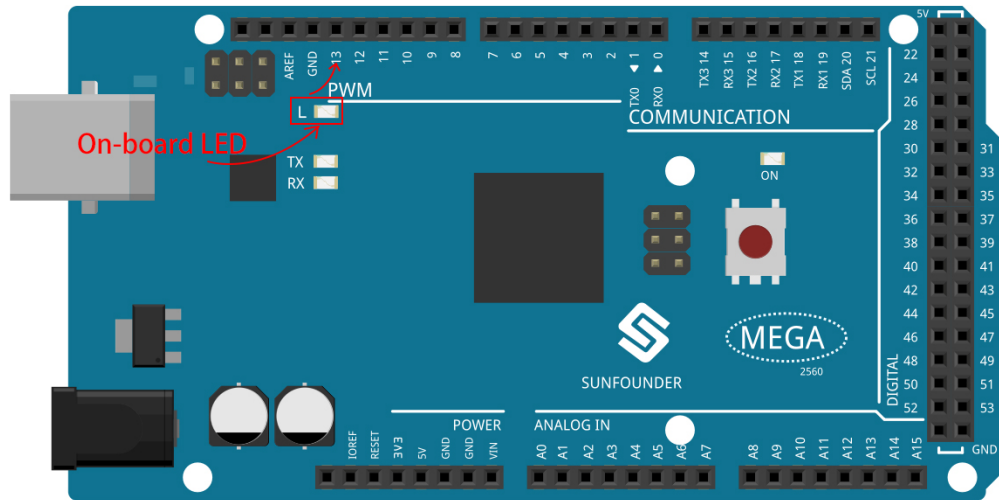
- *Stage Mode:* In this mode, you can write scripts for the sprite and boards to interact with sprites in real-time. If you disconnect the board with Pictoblox, you cannot interact anymore.
- *Upload Mode:* This mode allows you to write scripts and upload it to the board so that you can use even when it is not connected to your computer, for example, you need to upload a script for making moving robots.

For more information, please refer to: <https://thestempedia.com/tutorials/getting-started-pictoblox>

3.3 1.3 Quick Guide on PictoBlox

Now let's learn how to use PictoBlox in two modes.

Also, there is an On-board LED connected to Pin 13 on the Arduino Uno/Mega2560, we will learn to make this LED blink in 2 different modes.

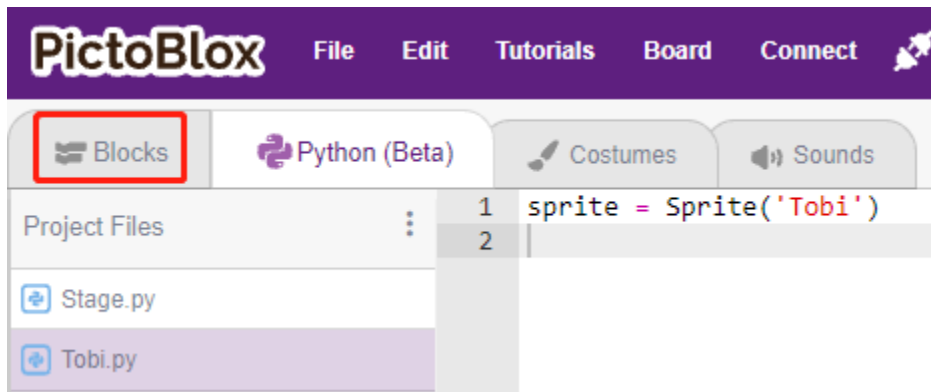


3.3.1 Stage Mode

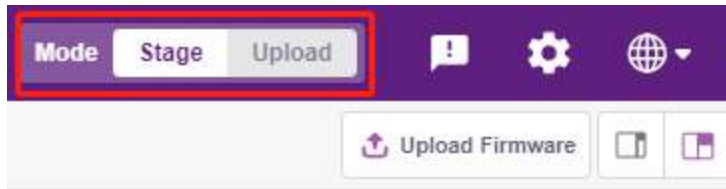
1. Connect to Arduino Board

Connect your Arduino board to the computer with a USB cable, usually the computer will automatically recognize your board and finally assign a COM port.

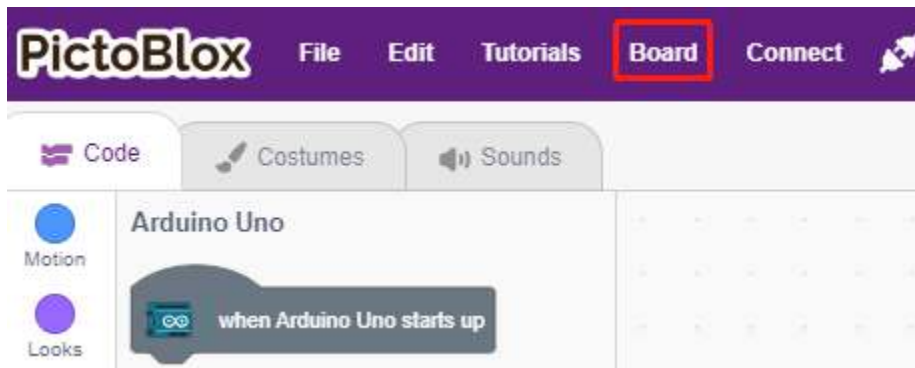
Open PictoBlox, the Python programming interface will open by default. And we need to switch to the Blocks interface.



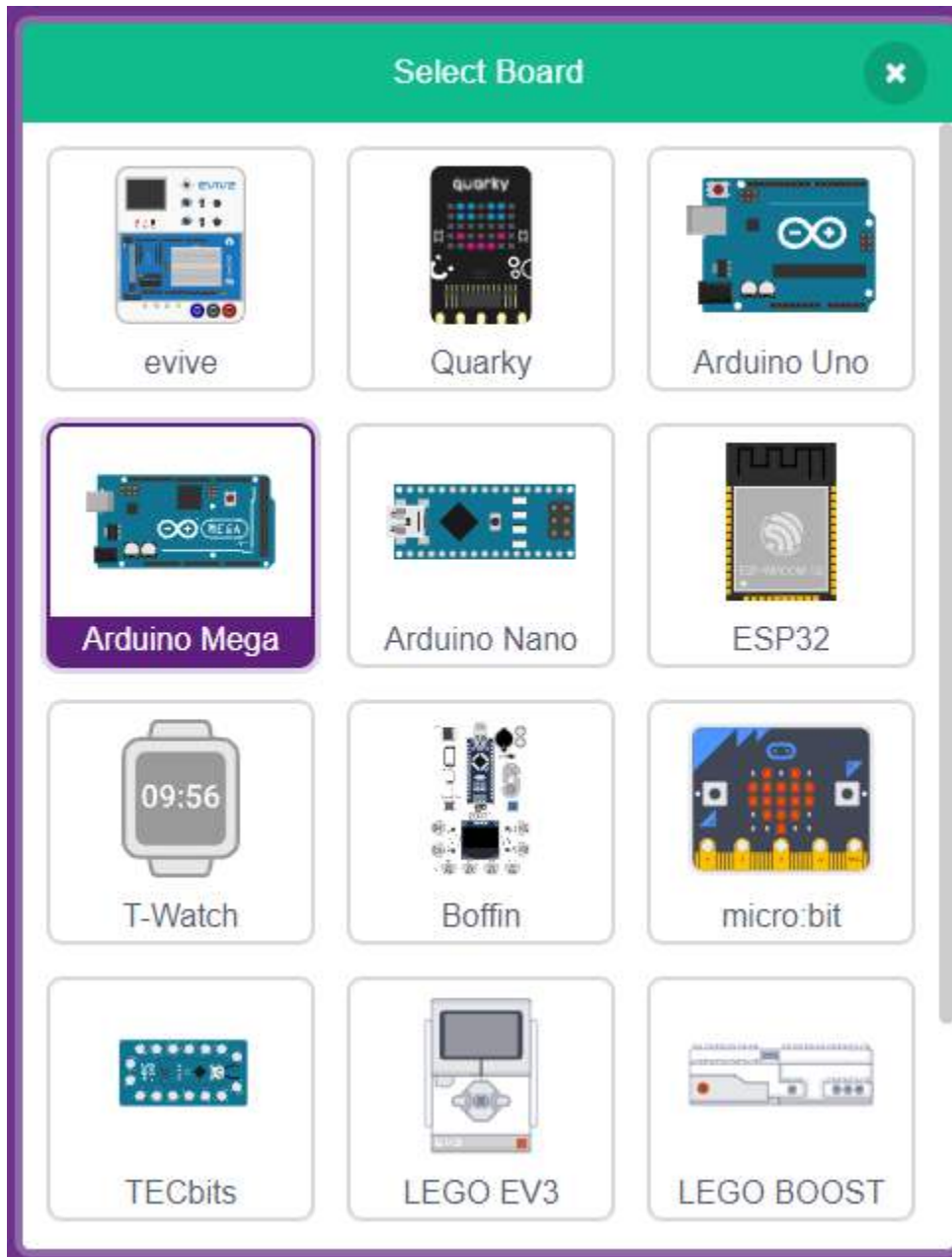
Then you will see the top right corner for mode switching. The default is Stage mode, where Tobi is standing on the stage.



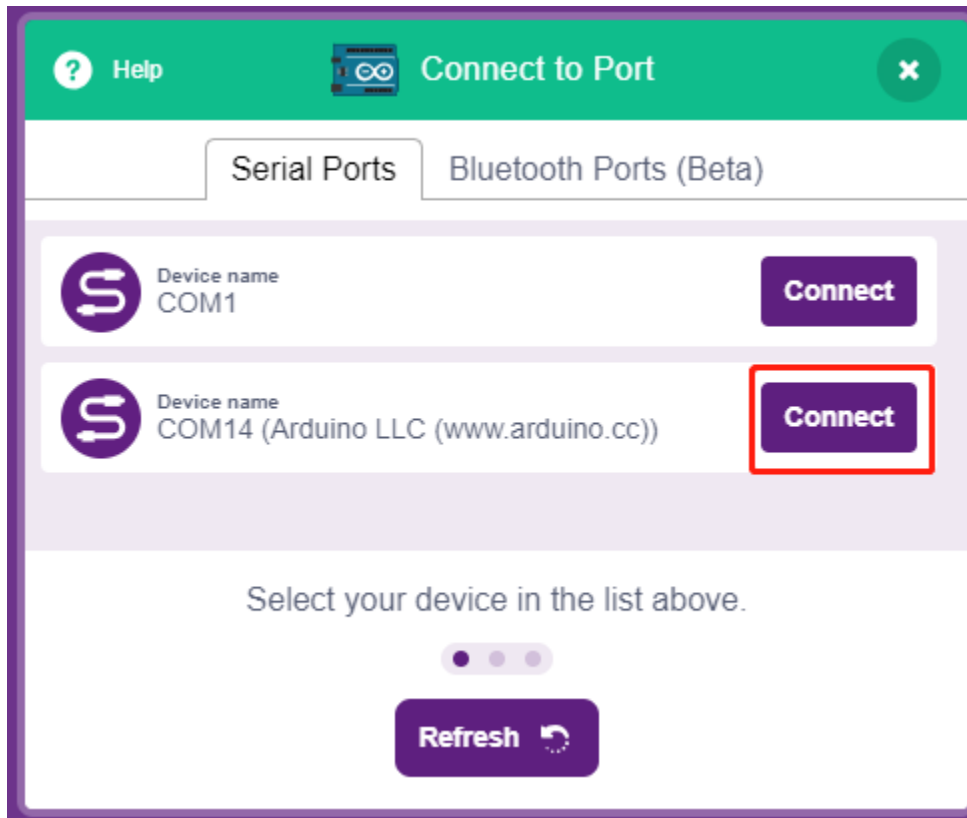
Click **Board** in the upper right navigation bar to select the board.



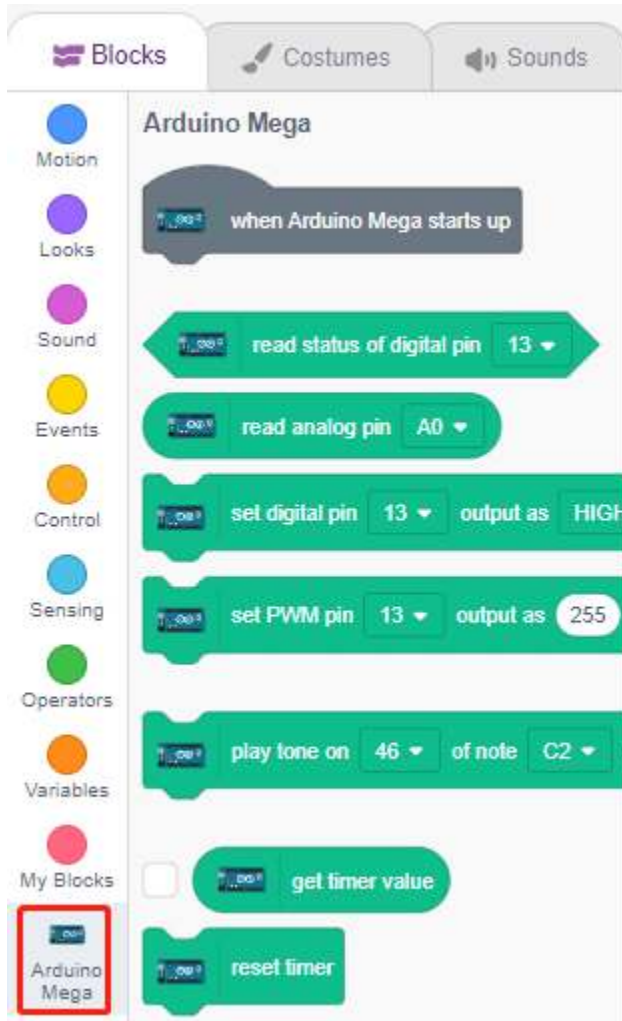
For example, choose **Arduino Mega**.



A connection window will then pop up for you to select the port to connect to, and return to the home page when the connection is complete. If you break the connection during use, you can also click **Connect** to reconnect.



At the same time, Arduino Mega related palettes, such as Arduino Mega, Actuators, etc., will appear in the **Block Palette**.

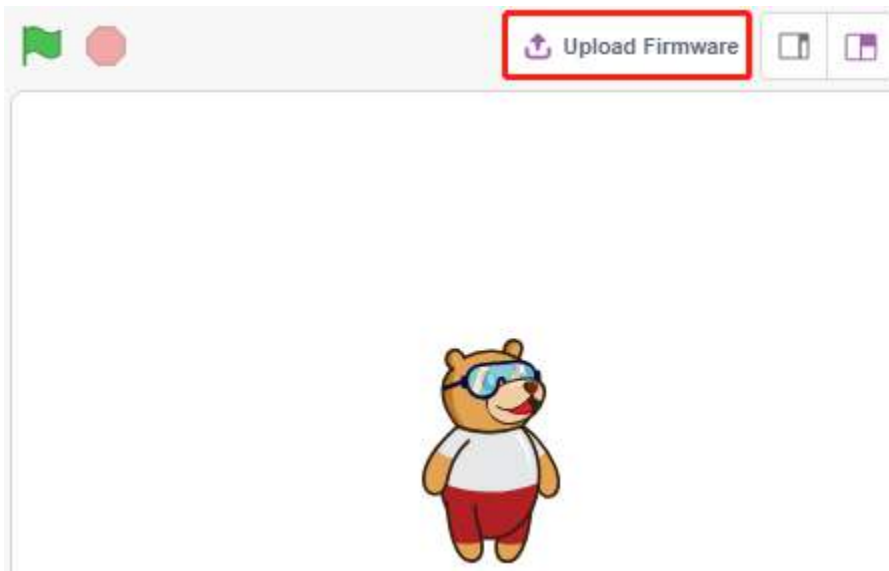


2. Upload Firmware

Since we're going to work in the Stage mode, we must upload the firmware to the board. It will ensure real-time communication between the board and the computer. Uploading the firmware it is a one-time process. To do so, click on the Upload Firmware button.

After waiting for a while, the upload success message will appear.

Note: If you are using this Arduino board in PictoBlox for the first time, or if this Arduino was previously uploaded with the Arduino IDE. Then you need to tap **Upload Firmware** before you can use it.

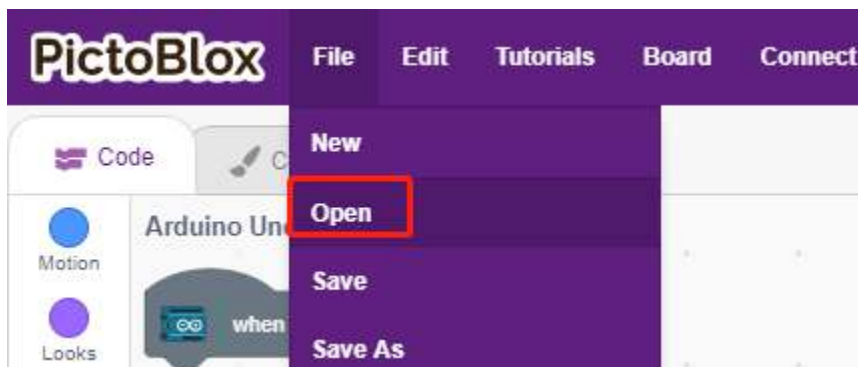


3. Programming

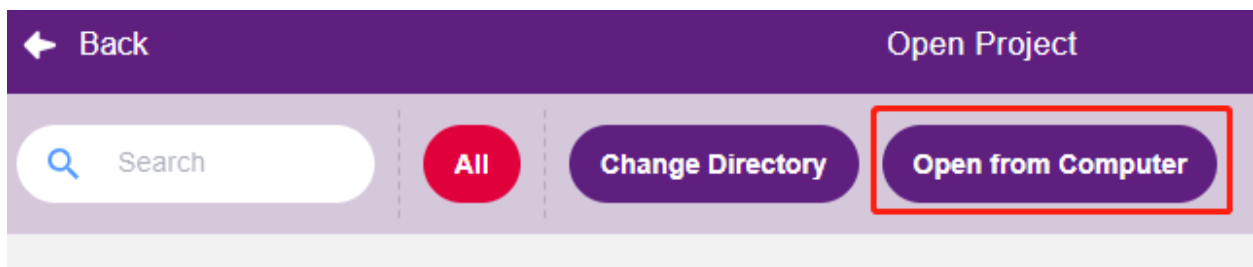
- Open and run the script directly

Of course, you can open the scripts directly to run them, but please download them from [github](#) first.

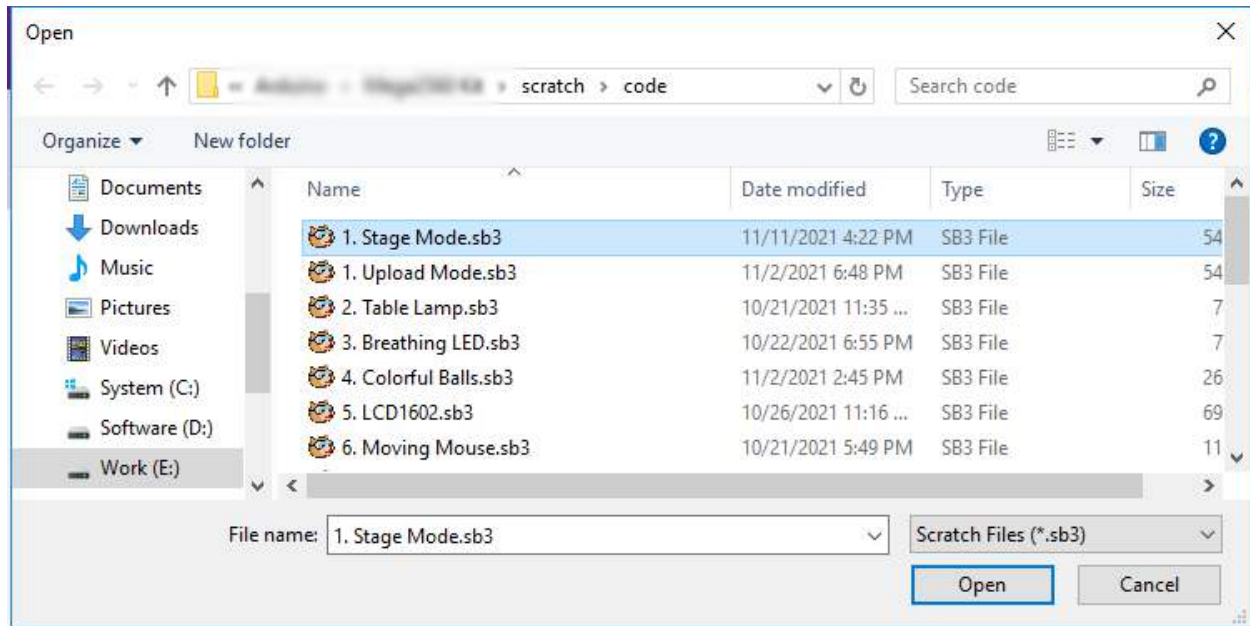
You can click on **File** in the top right corner and then choose **Open**.



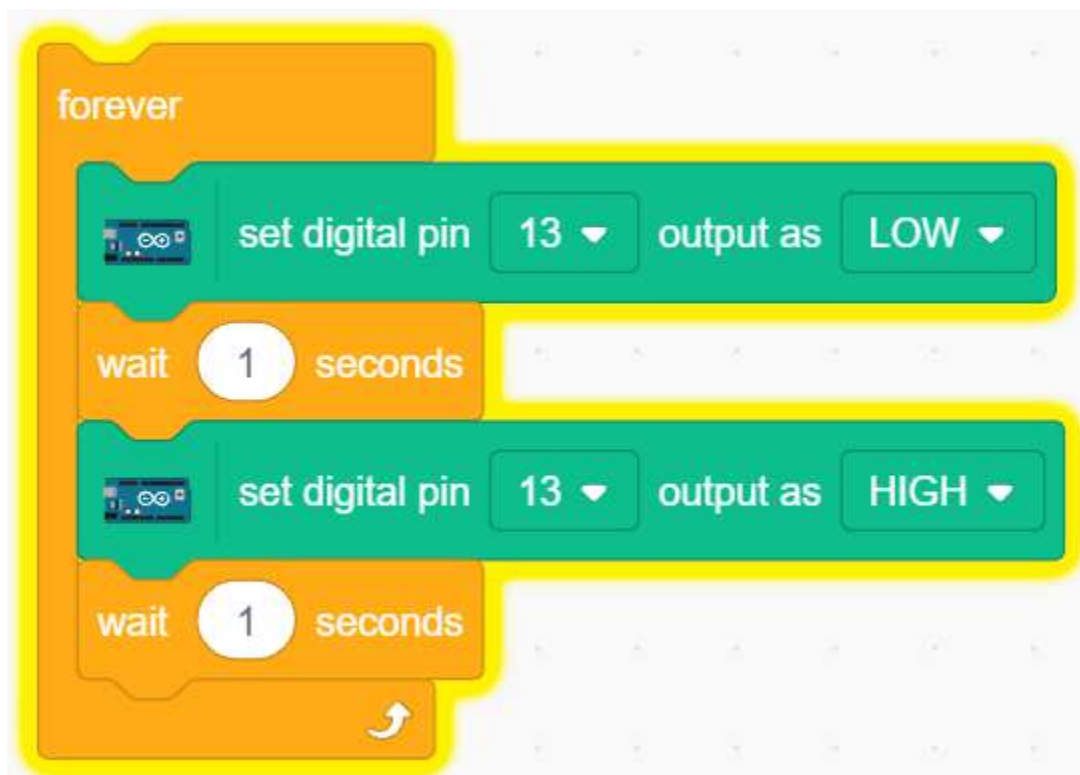
Choose **Open from Computer**.



Then go to the path of `sunfounder_vincent_kit_for_arduino\scratch\code`, and open **1. Stage Mode.sb3**. Please ensure that you have downloaded the required code from [github](#).



Click directly on the script to run it, some projects are click on the green flag or click on the sprite.



- Program step by step

You can also write the script step by step by following these steps.

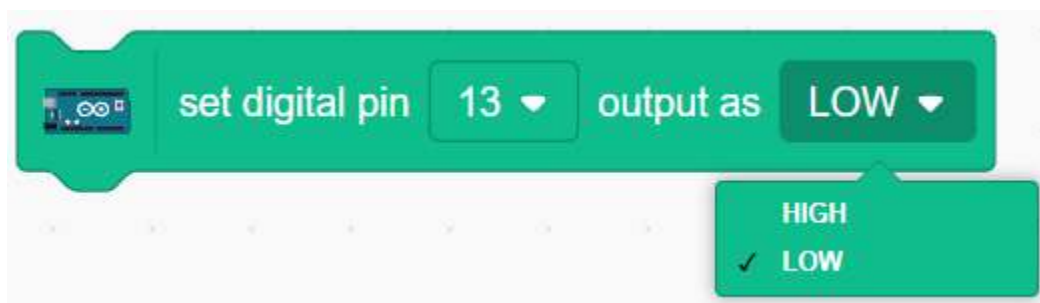
Click on the **Arduino Mega** palette.



The LED on the Arduino board is controlled by the digital pin 13 (only 2 states, HIGH or LOW), so drag the [set digital pin out as] block to the script area.

Since the default state of the LED is lit, now set pin 13 to LOW and click on this block and you will see the LED go off.

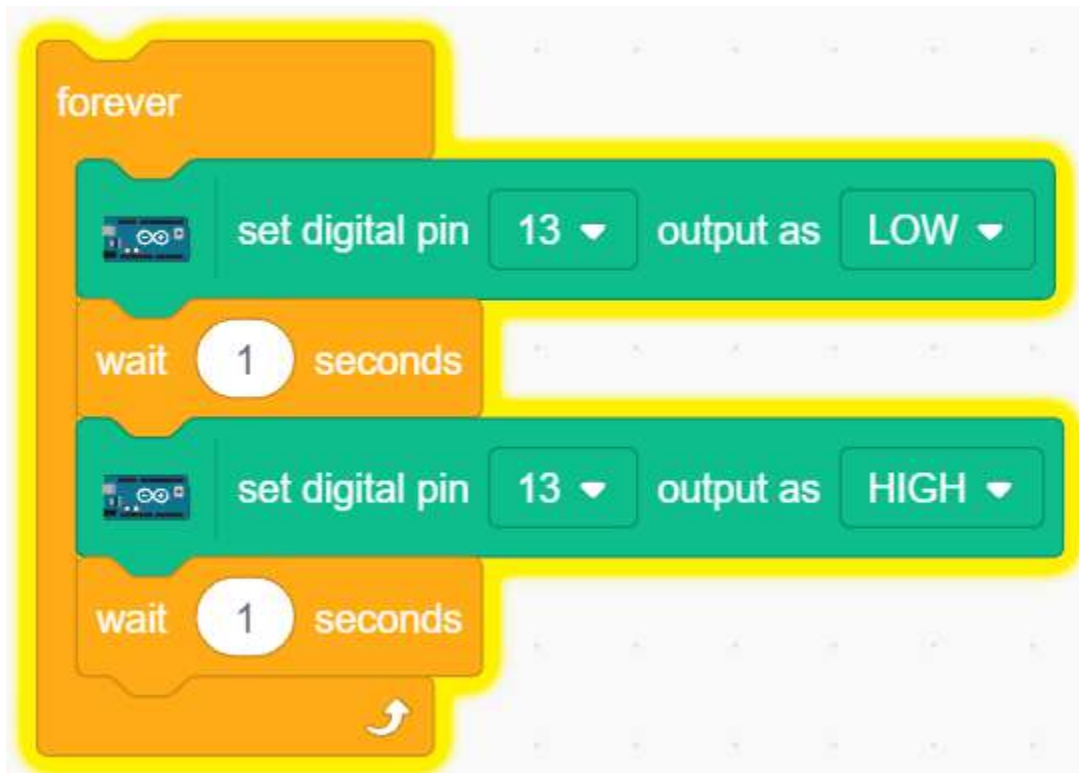
- [set digital pin out as]: Set the digital pins (2~13) to (HIGH/LOW) level.



In order to see the effect of continuous blinking LED, you need to use the [Wait 1 seconds] and [forever] blocks in the **Control** palette. Click on these blocks after writing, there is a yellow halo means it is running.

- [Wait 1 seconds]: from the **Control** palette, used to set the time interval between 2 blocks.

- [forever]: from the **Control** palette, allows the script to keep running unless manually paused.

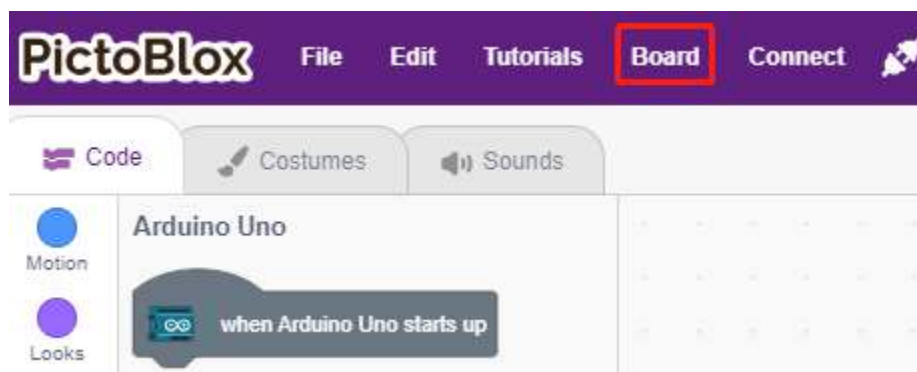


3.3.2 Upload Mode

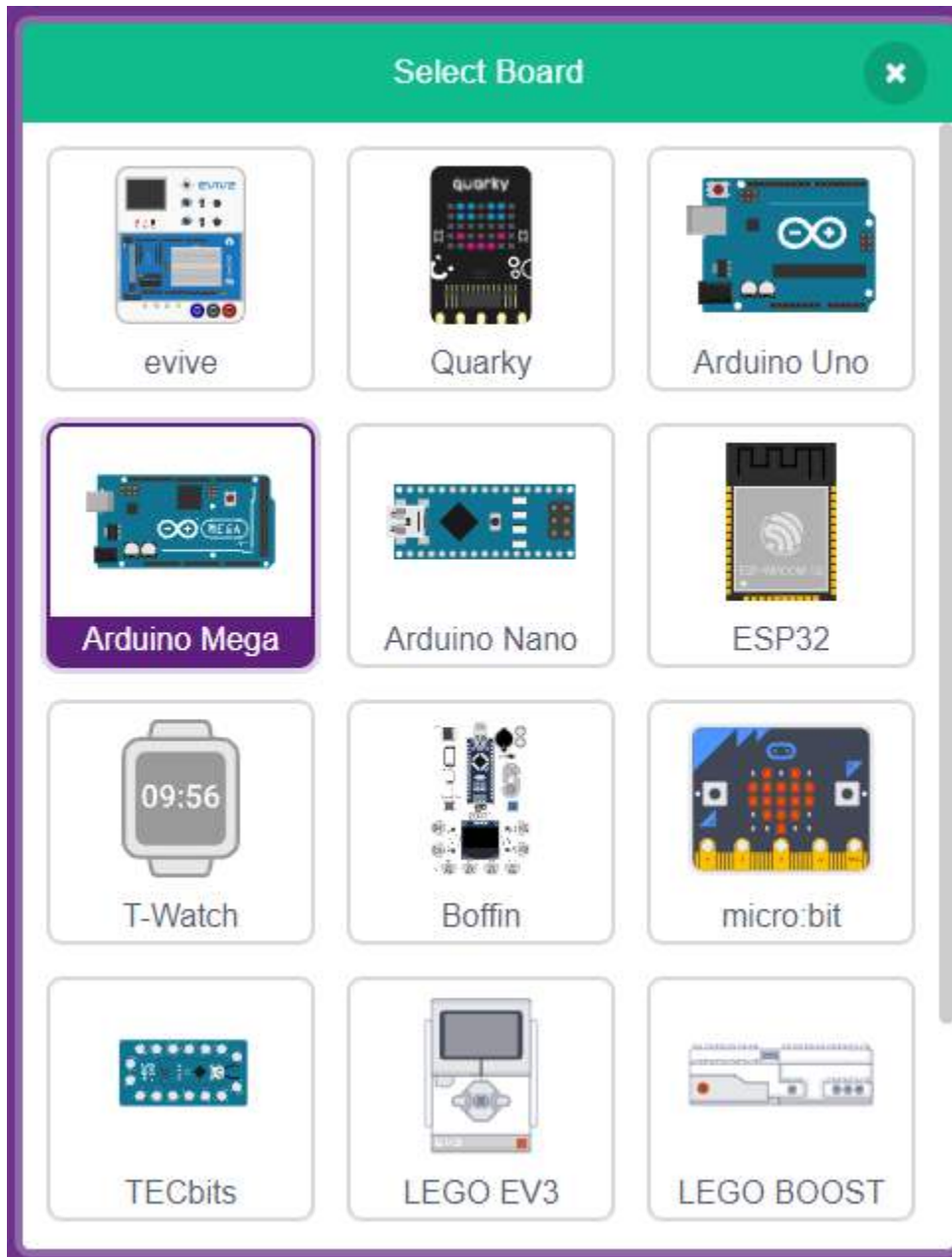
1. Connect to Arduino Board

Connect your Arduino board to the computer with a USB cable, usually the computer will automatically recognize your board and finally assign a COM port.

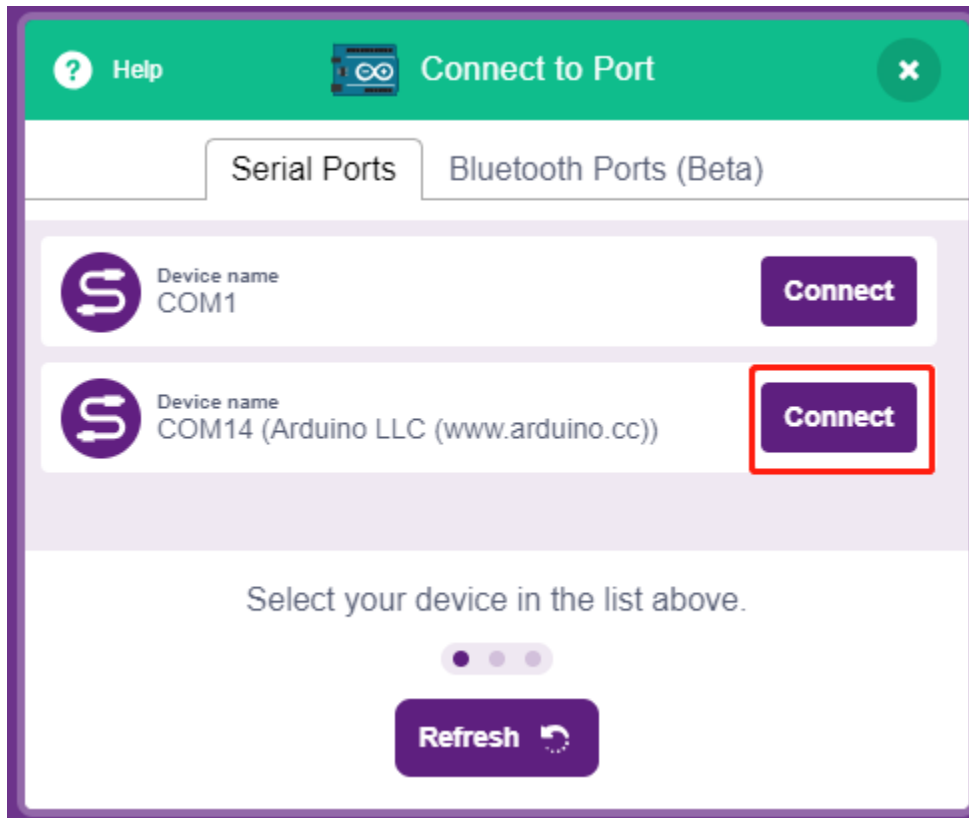
Open PictoBlox and click **Board** in the top right navigation bar to select the board.



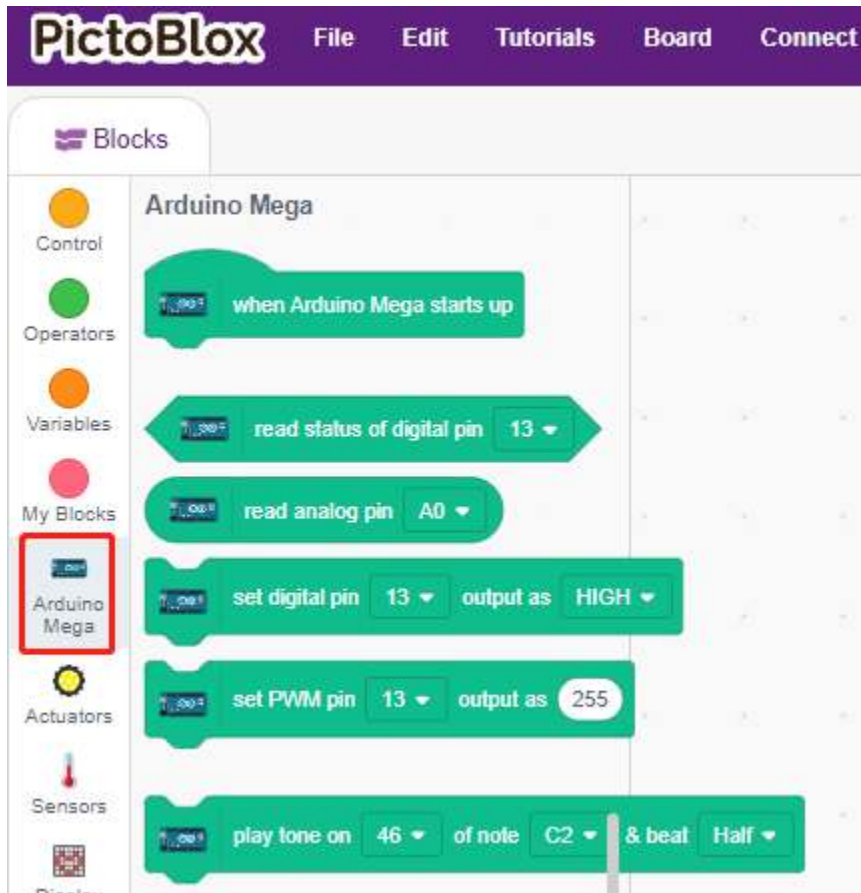
For example, choose **Arduino Mega**.



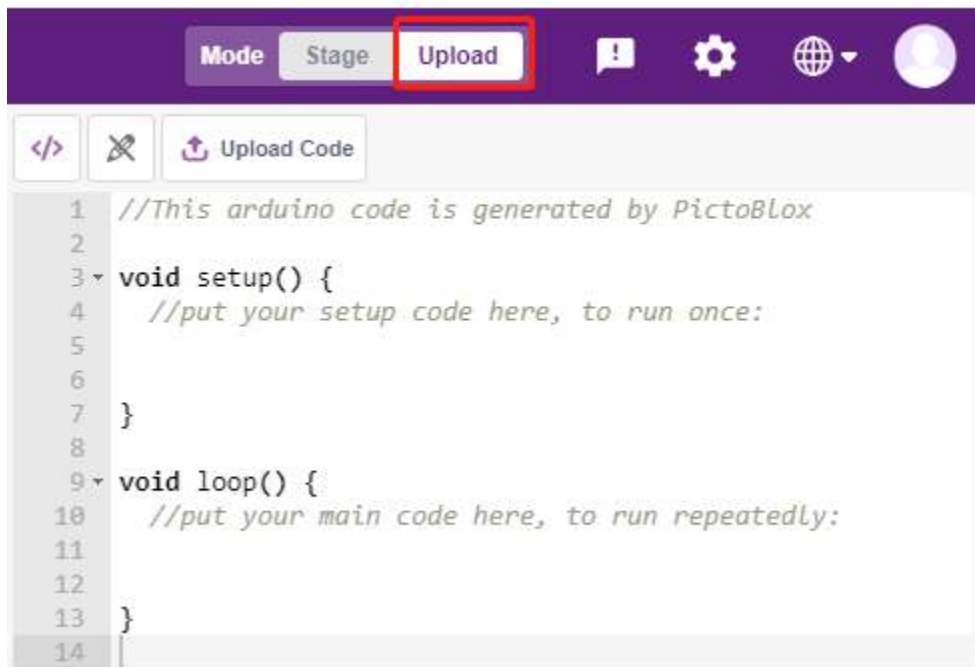
A connection window will then pop up for you to select the port to connect to, and return to the home page when the connection is complete. If you break the connection during use, you can also click **Connect** to reconnect.



At the same time, Arduino Mega related palettes, such as Arduino Mega, Actuators, etc., will appear in the **Block Palette**.



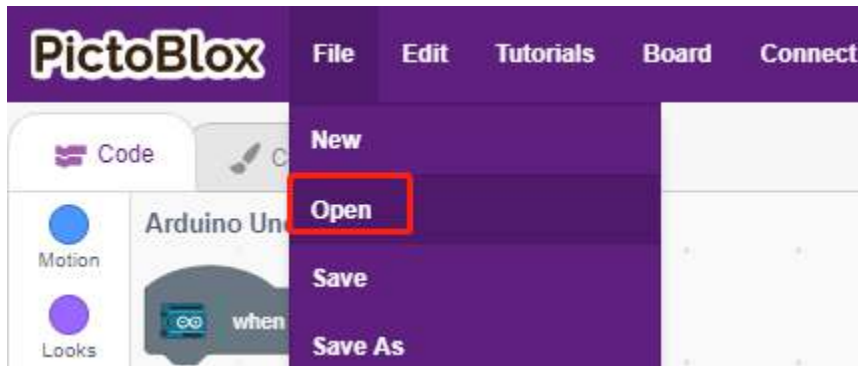
After selecting Upload mode, the stage will switch to the original Arduino code area.



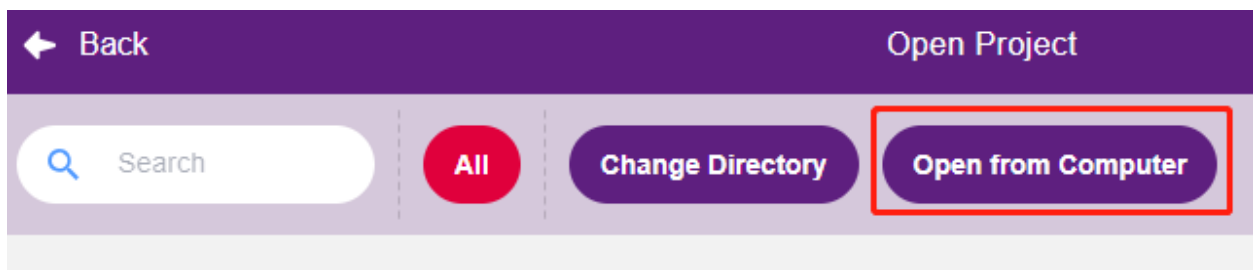
2. Programming

- Open and run the script directly

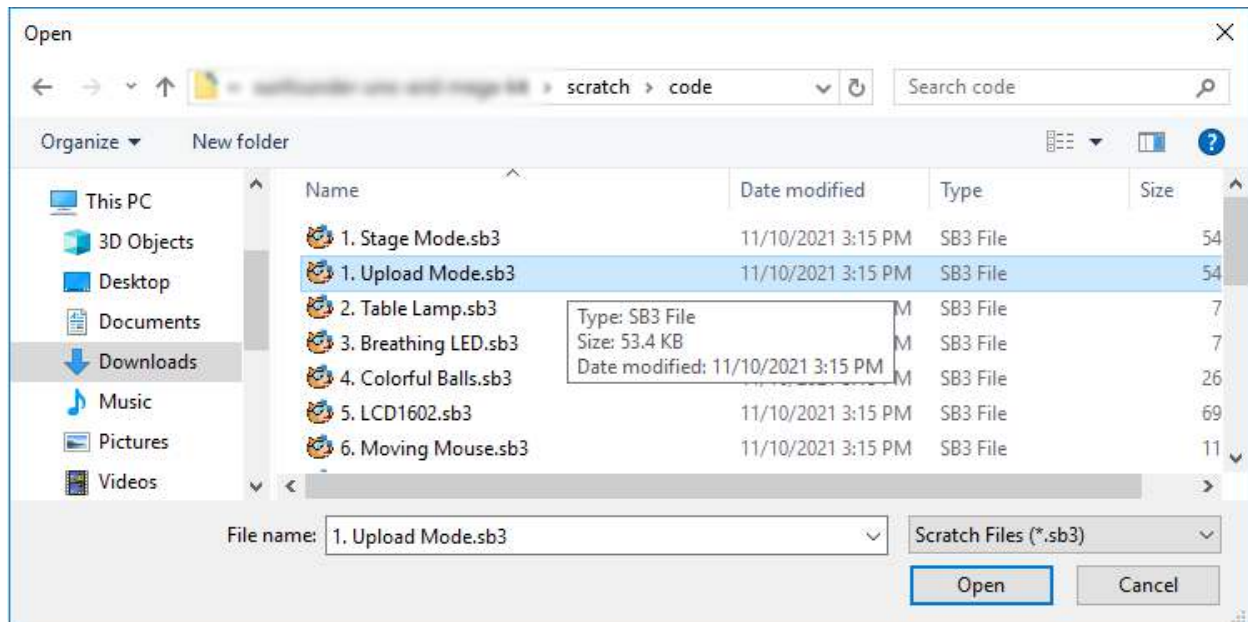
You can click on **File** in the top right corner.



Choose **Open from Computer**.



Then go to the path of `sunfounder_vincent_kit_for_arduino\scratch\code`, and open **1. Upload Mode.sb3**. Please ensure that you have downloaded the required code from github.



Finally, click the **Upload Code** button.

The screenshot shows the PictoBlox IDE interface. On the left, a script is built with the following blocks: a 'when Arduino Mega starts up' block, followed by a 'forever' loop containing 'set digital pin 13 output as LOW', 'wait 1 seconds', 'set digital pin 13 output as HIGH', and 'wait 1 seconds'. On the right, the generated C++ code is displayed, with the 'Upload Code' button highlighted in red. The code is as follows:

```

1 //This c++ code is generated by PictoBlox
2
3 void setup() {
4 //put your setup code here, to run once:
5 pinMode(13, OUTPUT);
6
7 }
8
9
10 void loop() {
11 //put your main code here, to run repeatedly:
12
13
14 digitalWrite(13, false);
15 delay(1 * 1000);
16 digitalWrite(13, true);
17 delay(1 * 1000);
18 }
19

```

- Program step by step

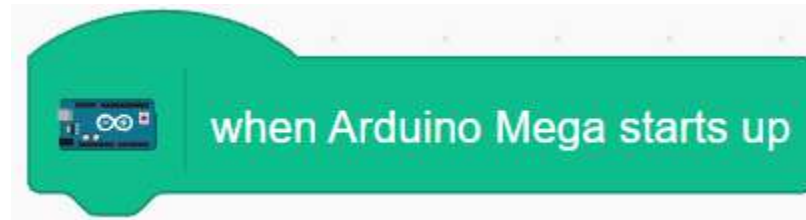
You can also write the script step by step by following these steps.

Click on the **Arduino Mega** palette.

The screenshot shows the PictoBlox 'Blocks' palette. The 'Arduino Mega' category is selected and highlighted with a red box. The palette includes the following categories and blocks:

- Control**: when Arduino Mega starts up
- Operators**: read status of digital pin 13
- Variables**: read analog pin A0
- My Blocks**: set digital pin 13 output as HIGH
- Actuators**: set PWM pin 13 output as 255
- Sensors**: play tone on 46 of note C2 & beat Half

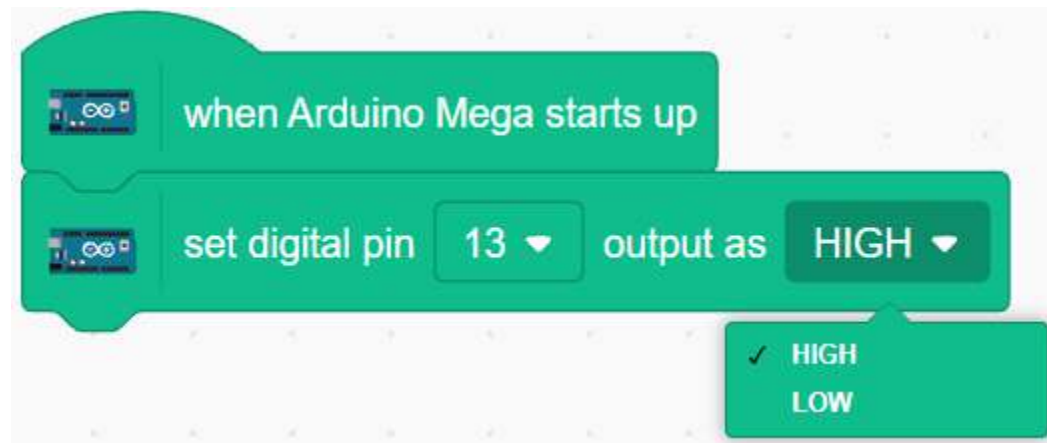
Drag [when Arduino Mega starts up] to the script area, which is required for every script.



The LED on the Arduino board is controlled by the digital pin13 (only 2 states HIGH or LOW), so drag the [set digital pin out as] block to the script area.

Since the default state of the LED is lit, now set pin 13 to LOW and click on this block and you will see the LED go off.

- [set digital pin out as]: Set the digital pin (2~13) to (HIGH/LOW) level.



At this point you will see the Arduino code appear on the right side, if you want to edit this code, then you can turn Edit mode on.

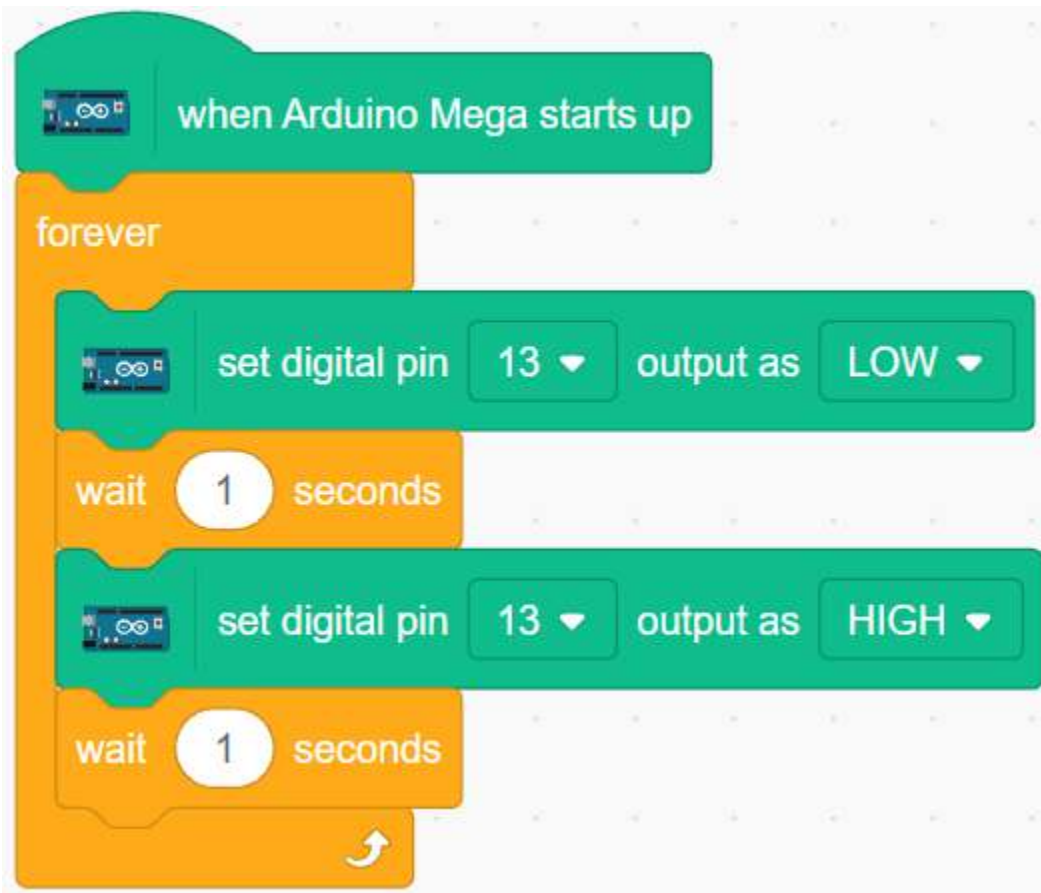
The image shows the Scratch code editor interface. At the top, there are three icons: a code editor icon, a pencil icon (highlighted with a red box), and an "Upload Code" button. Below the icons, the text "Edit Mode" is written in red. The main area contains the following Arduino code:

```
1 //Arduino code is generated by PictoBlox
2
3 void setup() {
4   //put your setup code here, to run once:
5   pinMode(13, OUTPUT);
6
7
8   digitalWrite(13, false);
9 }
10
11 void loop() {
12   //put your main code here, to run repeatedly:
13
14
15 }
16
```

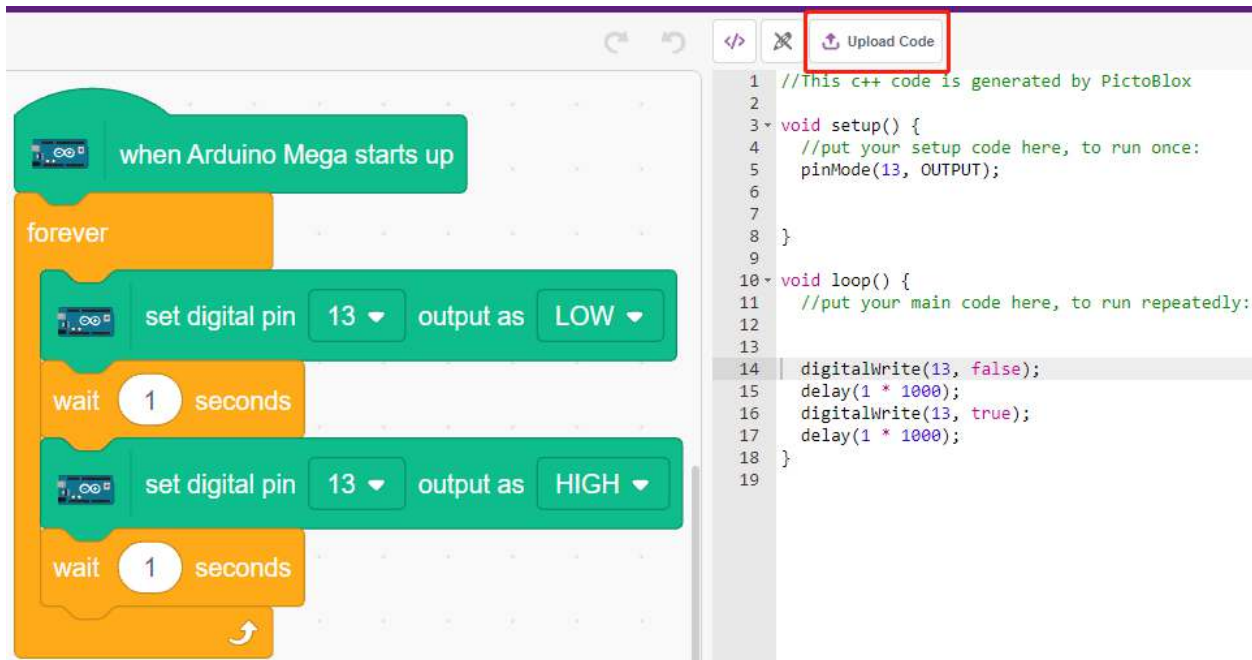
In order to see the effect of continuous blinking LED, you need to use the [Wait 1 seconds] and [forever] blocks in the

Control palette. Click on these blocks after writing, there is a yellow halo means it is running.

- [Wait 1 seconds]: from the **Control** palette, used to set the time interval between 2 blocks.
- [forever]: from the **Control** palette, allows the script to keep running unless the power is off.



Finally, click the **Upload Code** button.



2. Projects

The following projects are written in order of programming difficulty, so we recommend reading them in order.

In each project, there are very detailed steps to teach you how to build the circuit and how to program it step by step to achieve the final result.

Of course, you can also open the script directly to run it, but you need to make sure you have downloaded the relevant material from [github](#).

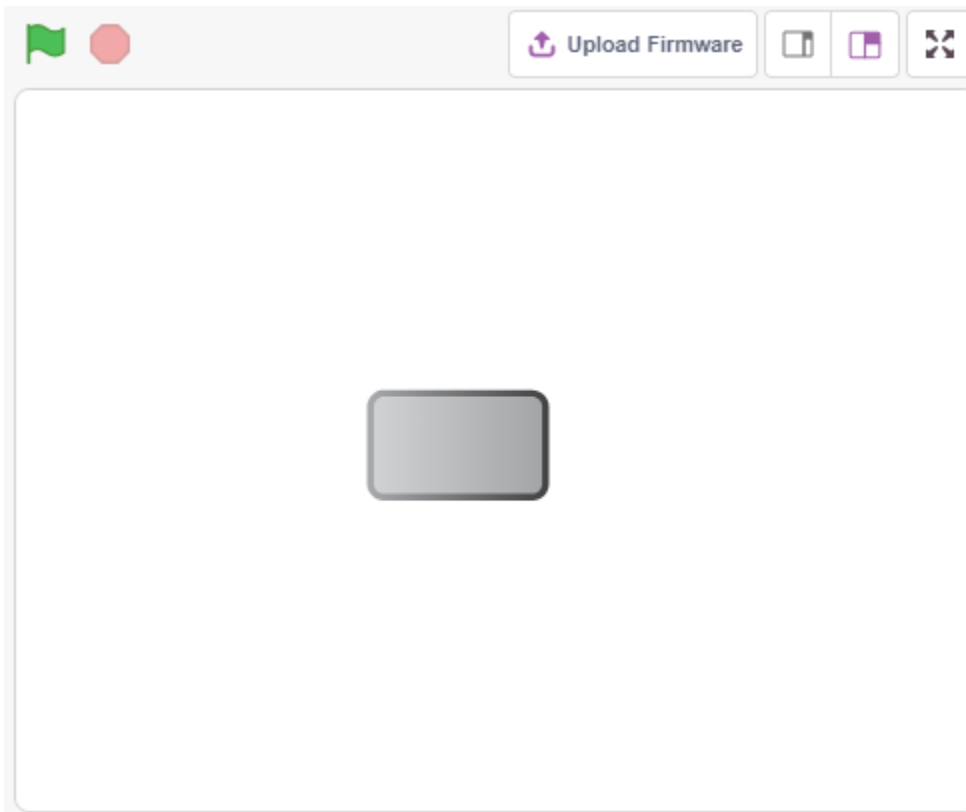
Once the download is complete, unzip it. Refer to *Stage Mode* to run individual scripts directly.

But the *2.10 Read Temperature and Humidity* is used the *Upload Mode*.

3.4 2.1 Table Lamp

Here, we connect an LED on the breadboard and have the sprite control the blinking of this LED.

When the Button sprite on the stage is clicked, the LED will blink 5 times and then stop.



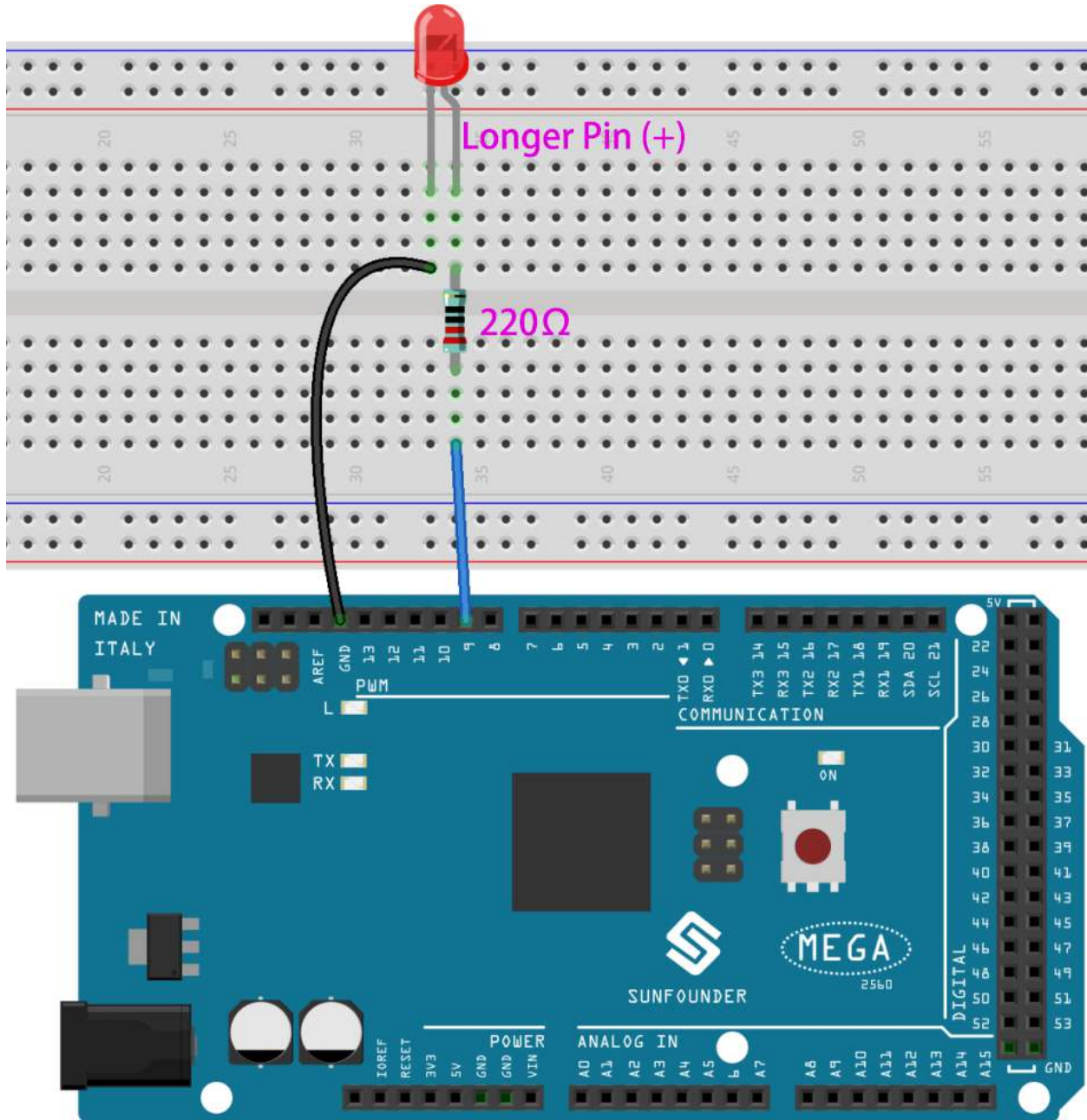
3.4.1 You Will Learn

- Breadboard, LEDs and Resistors
- Building a circuit on a breadboard
- Delete and select sprites
- Switching costumes
- Set a limited number of repeat loops

3.4.2 Build the Circuit

Follow the diagram below to build the circuit on the breadboard.

Since the anode of the LED (the longer pin) is connected to pin 9 through a 220 resistor, and the cathode of the LED is connected to GND, you can light up the LED by giving pin 9 a high level.



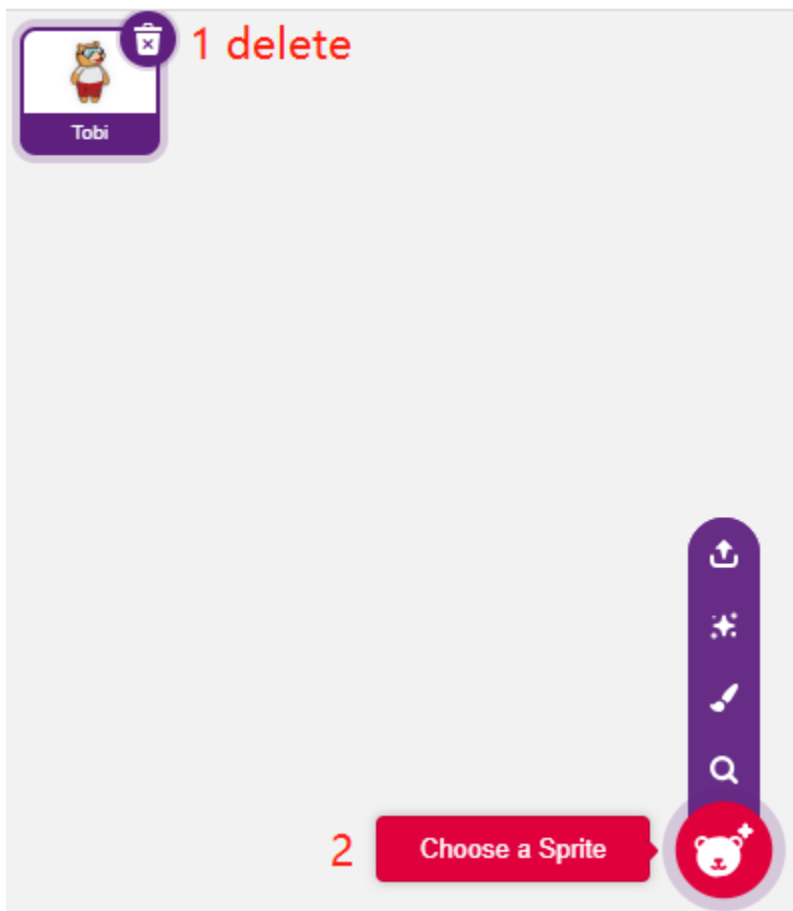
- Breadboard
- LED
- Resistor

3.4.3 Programming

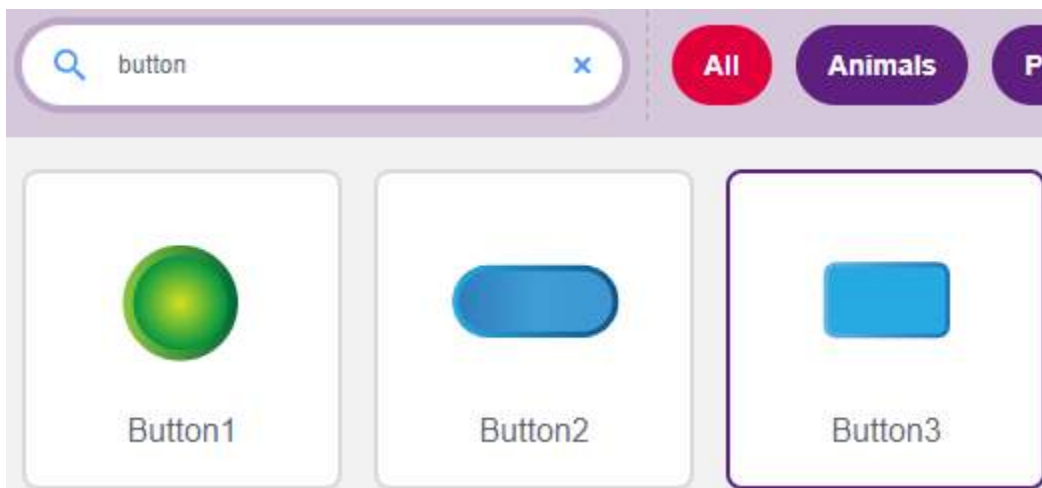
The whole programming is divided into 3 parts, the first part is to select the desired sprite, the second part is to switch the costume for the sprite to make it look clickable, and the third part is to make the LED blink.

1. Select Button3 sprite

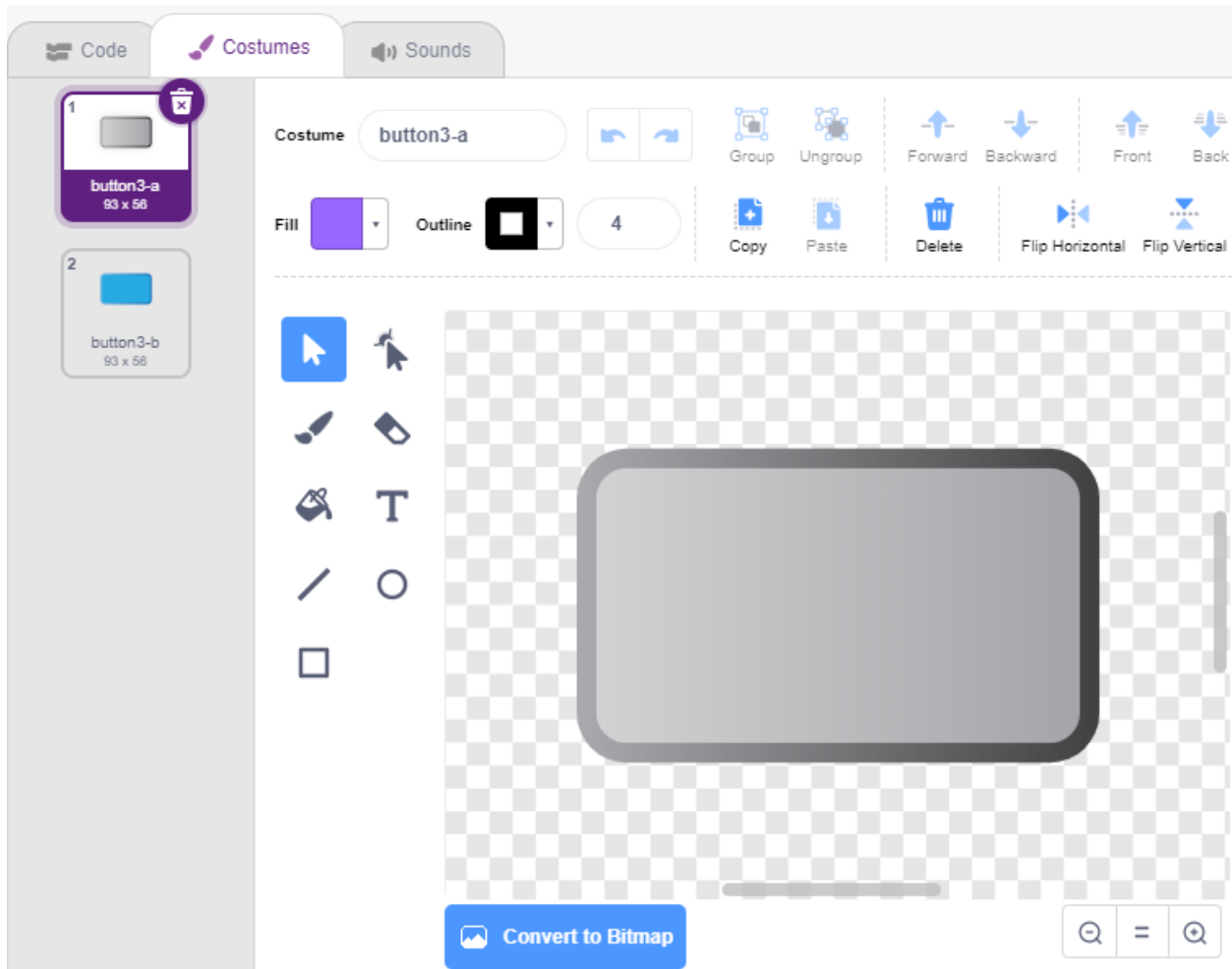
Delete the existing Tobi sprite by using the Delete button in the upper right corner, and select a sprite again.



Here, we select the **Button3** sprite.



Click on Costumes in the top right corner and you will see that the Button3 sprite has 2 costumes, we set **button3-a** to be released and **button3-b** to be pressed.



2. Switching costumes.

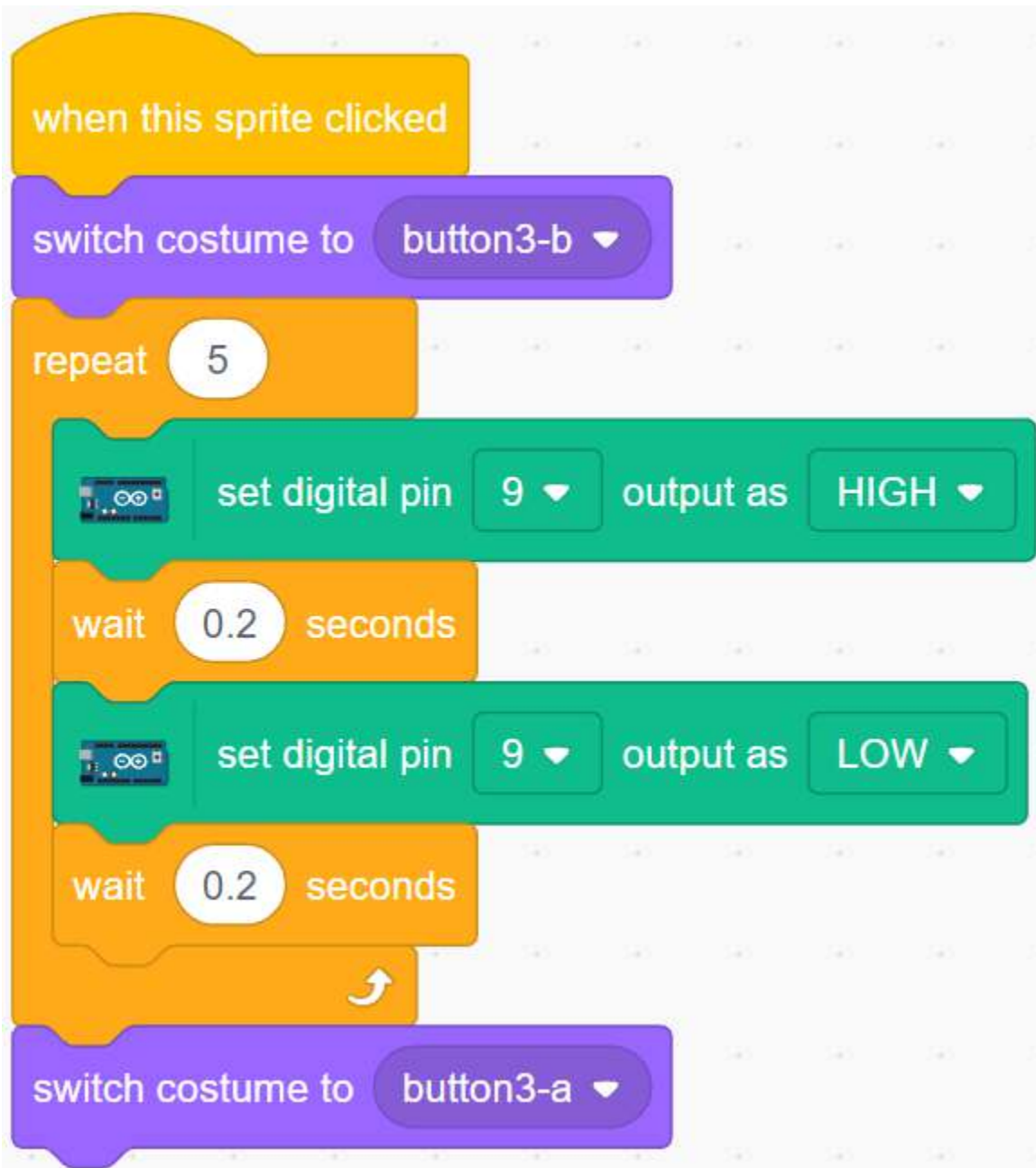
When the sprite is clicked (**Events** palette), it switches to costume for **button3-b** (**looks** palette).



3. Make the LED blink 5 times

Use the [Repeat] block to make the LED blink 5 times (High-> LOW cycle), remember to change pin 13 to pin 9, and finally switch the costume back to **button3-a**.

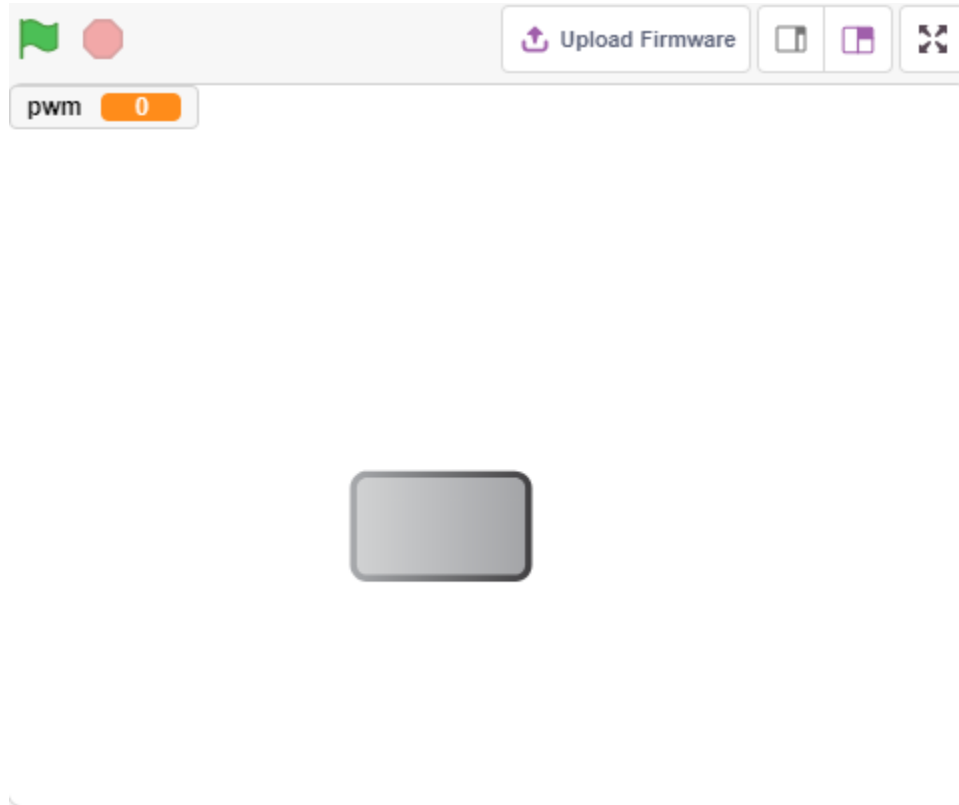
- [Repeat 10]: limited number of repeat loops, you can set the number of repeats yourself, from the **Control** palette.



3.5 2.2 Breathing LED

Now use another method to control the brightness of the LED. Unlike the previous project, here the brightness of the LED is made to slowly diminish until it disappears.

When the sprite on the stage is clicked, the brightness of the LED slowly increases and then goes out instantly.



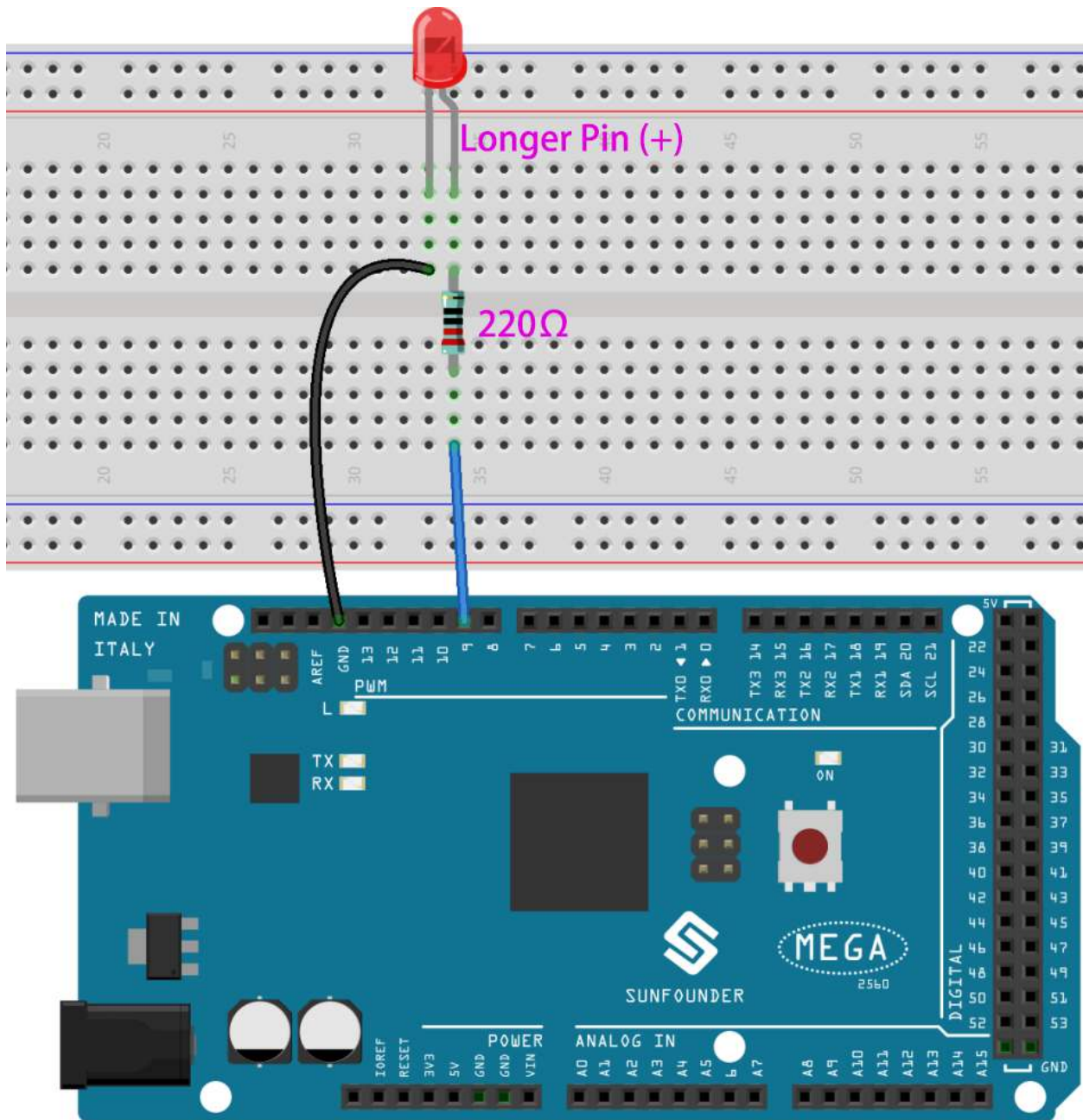
3.5.1 You Will Learn

- Set the output value of the PWM pin
- Create variables
- Change the brightness of the sprite

3.5.2 Build the Circuit

This project uses the same circuit as the previous project *2.1 Table Lamp*, but instead of using HIGH/LOW to make the LEDs light up or turn off, this project uses the [PWM - Wikipedia](#) signal to slowly light up or dim down the LED.

The PWM signal range is 0-255, on the Arduino Uno board, 3, 5, 6, 9, 10, 11 can output PWM signal; on the Mega2560, 2 - 13, 44 - 46 can output PWM signal.

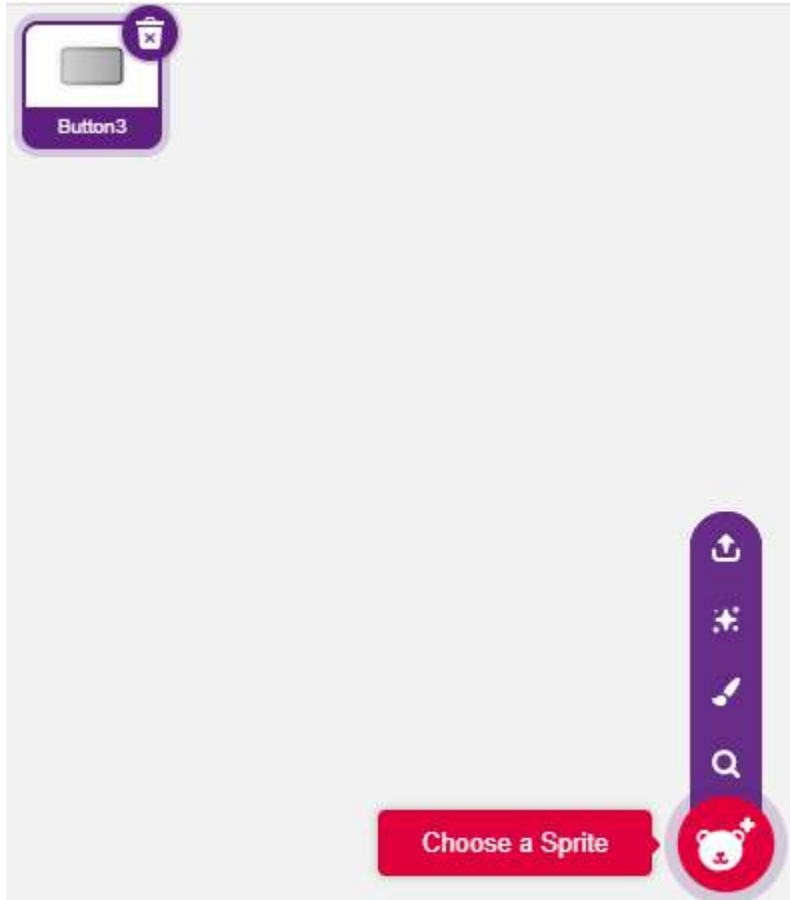


- Breadboard
- LED
- Resistor

3.5.3 Programming

1. Select a sprite

Delete the default sprite, click the **Choose a Sprite** button in the lower right corner of the sprite area, enter **button3** in the search box, and then click to add it.



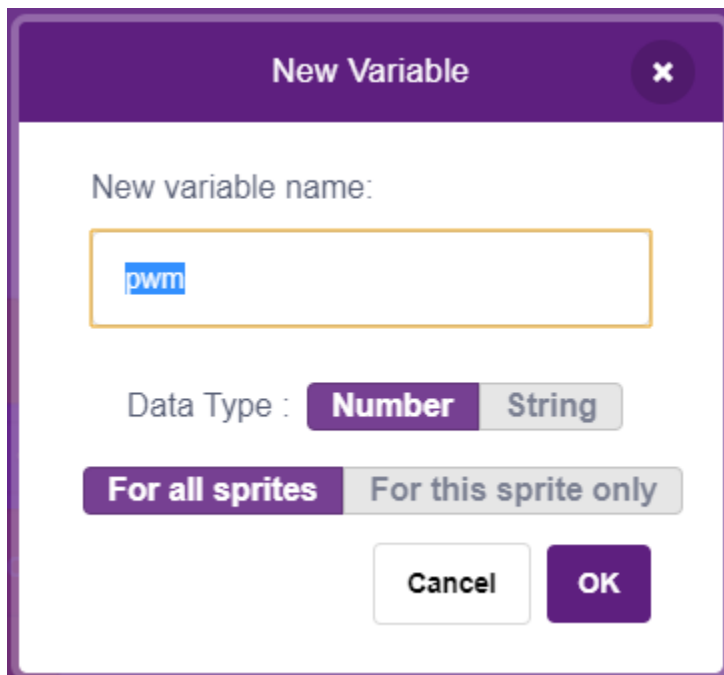
2. Creating a variable.

Create a variable called **pwm** to store the value of the pwm change.

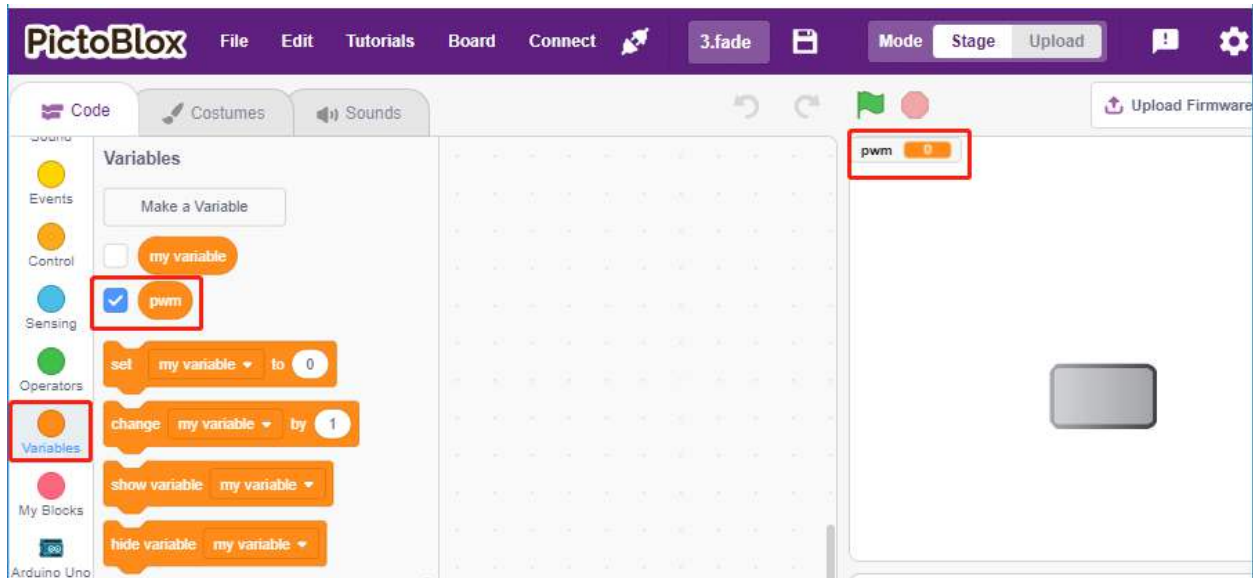
Click on the **Variables** palette and select **Make a Variable**.



Enter the name of the variable, it can be any name, but it is recommended to describe its function. The data type is number and For all sprites.



Once created, you will see **pwm** inside the **Variables** palette and in the checked state, which means this variable will appear on the stage. You can try unchecking it to see if pwm is still present on the stage.



3. Set the initial state

When the **button3** sprite is clicked, switch the costume to **button-b** (clicked state), and set the initial value of the variable **pwm** to 0.

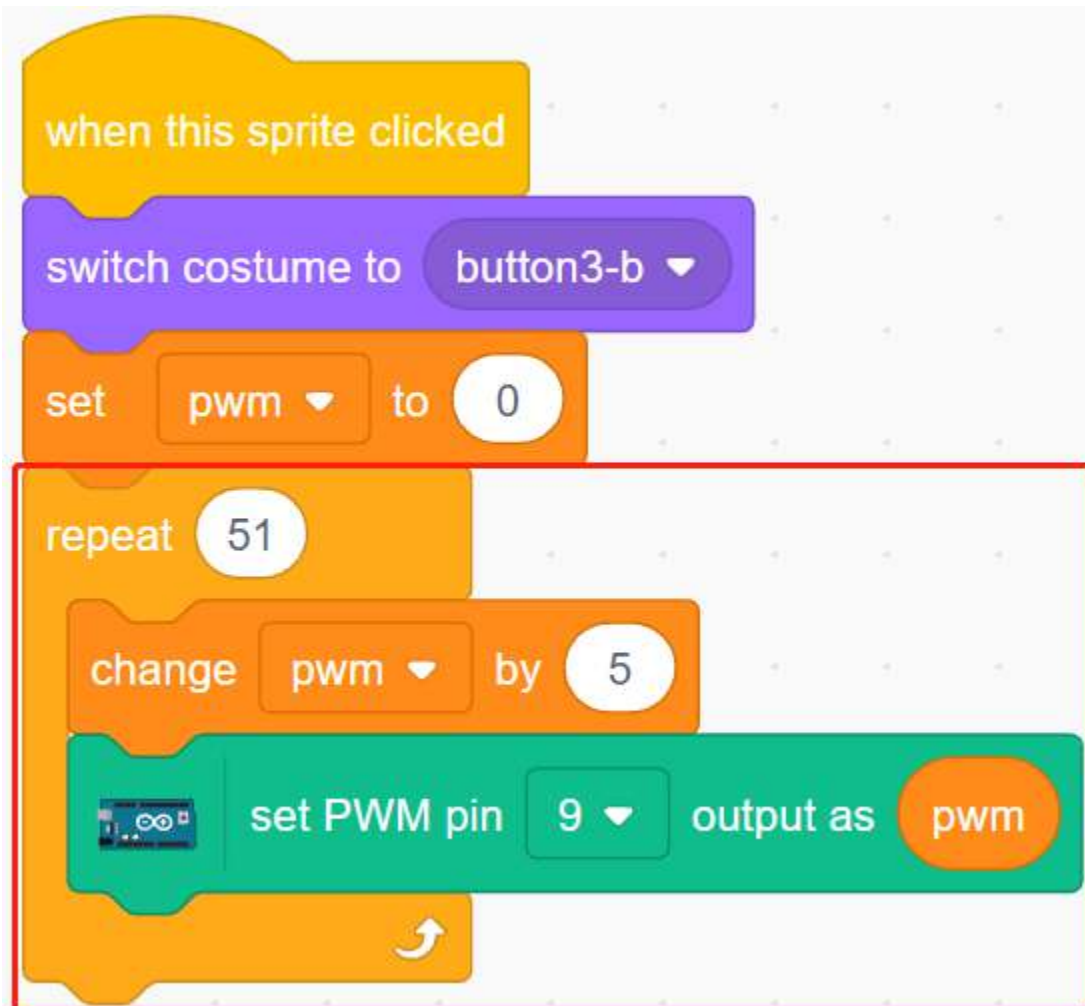
- [set pwm to 0]: from **Variables** palette, used to set the value of the variable.



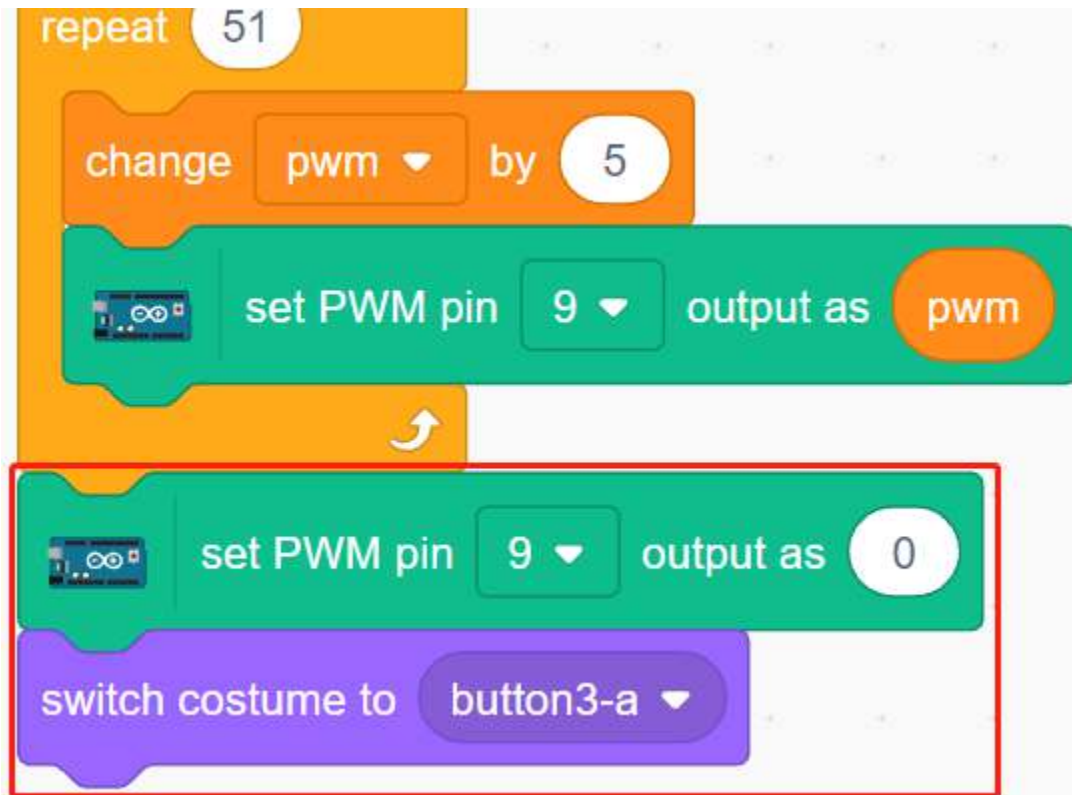
4. Make the LED brighter and brighter

Since the range of pwm is 255, so by [repeat] block, the variable **pwm** is accumulated to 255 by 5, and then put into [set PWM pin] block, so you can see the LED slowly light up.

- [change pwm by 5]: from **Variables** palette, let the variable change a specific number each time. It can be a positive or negative number, positive is increasing each time, negative is decreasing each time, for example, here the variable pwm is increased by 5 each time.
- [set PWM pin]: from the **Arduino Uno** palette, used to set the output value of the pwm pin.



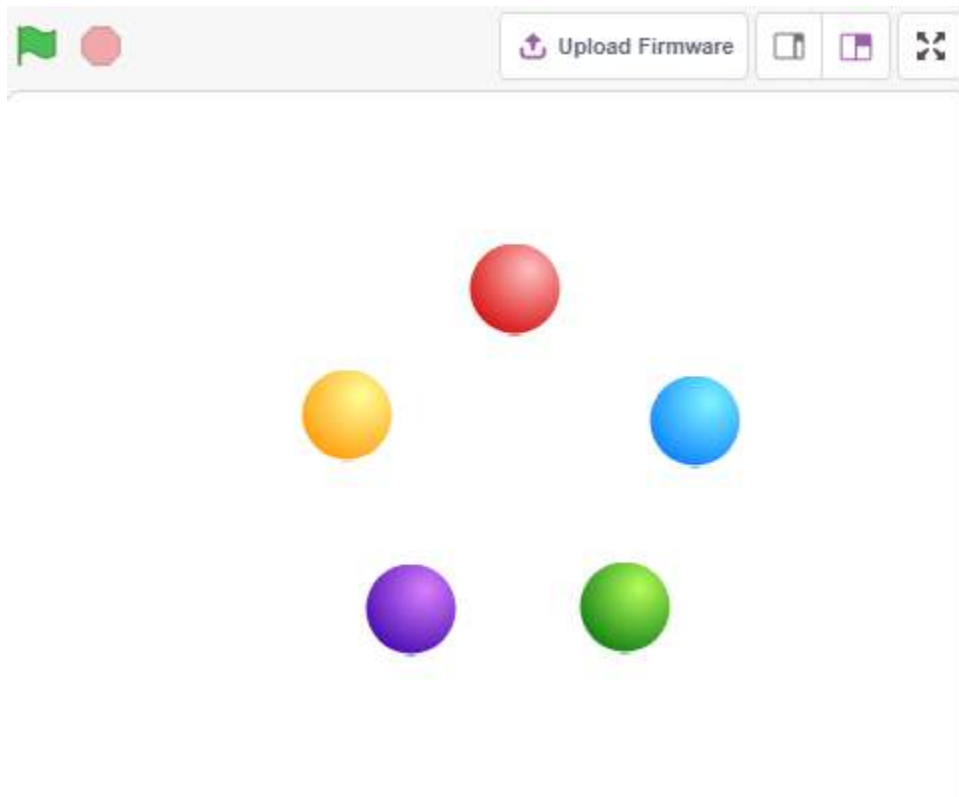
Finally, switch the costume of button3 back to **button-a** and make the PWM pin value 0, so that the LED will light up slowly and then turn off again.



3.6 2.3 Colorful Balls

In this project, we will make the RGB LEDs display different colors.

Clicking on different colored balls on the stage area will cause the RGB LED to light up in different colors.



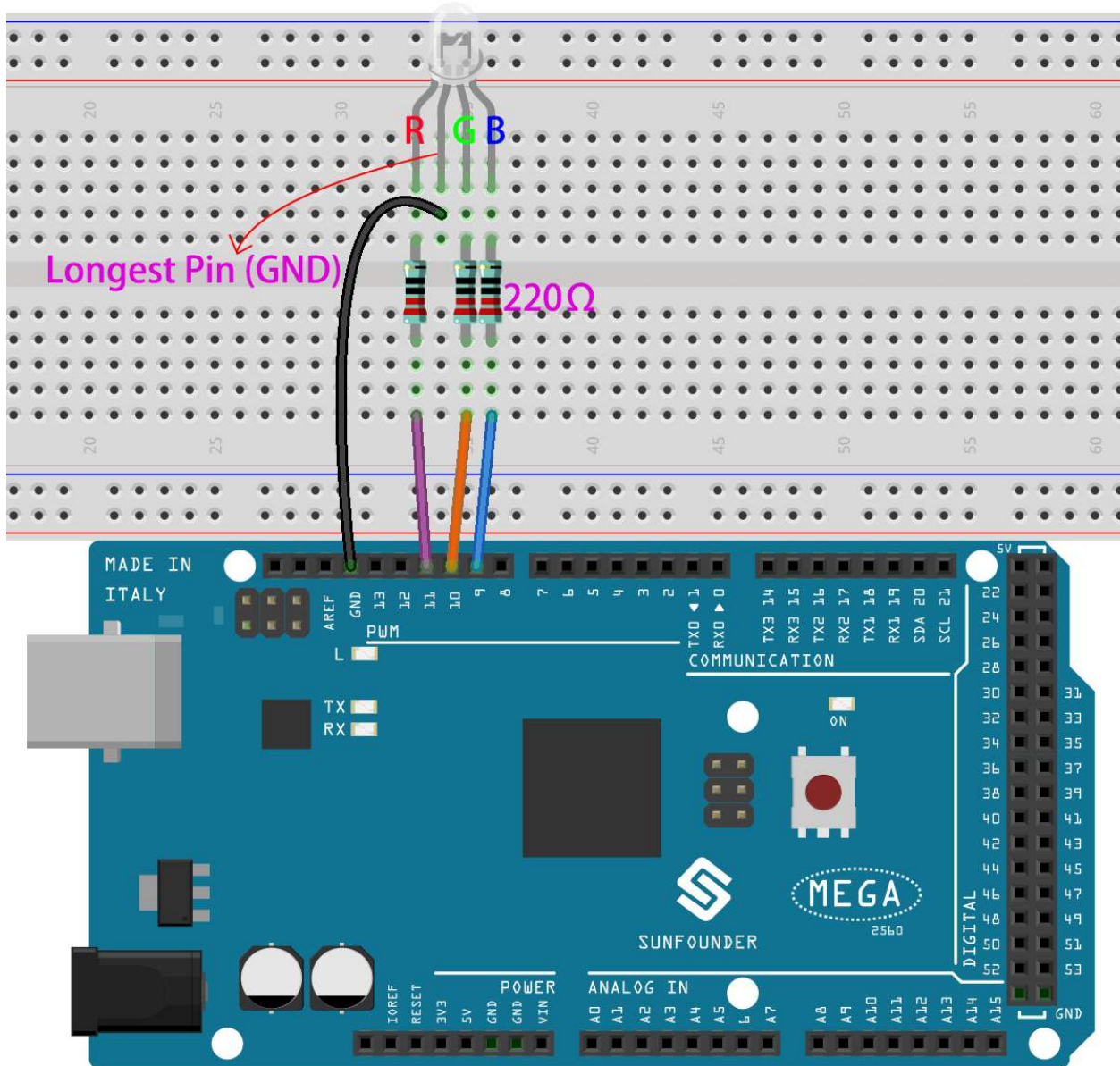
3.6.1 You Will Learn

- The principle of RGB LED
- Copy sprites and select different costumes
- Three primary colors superimposed

3.6.2 Build the Circuit

An RGB LED packages three LEDs of red, green, and blue into a transparent or semitransparent plastic shell. It can display various colors by changing the input voltage of the three pins and superimpose them, which, according to statistics, can create 16,777,216 different colors.



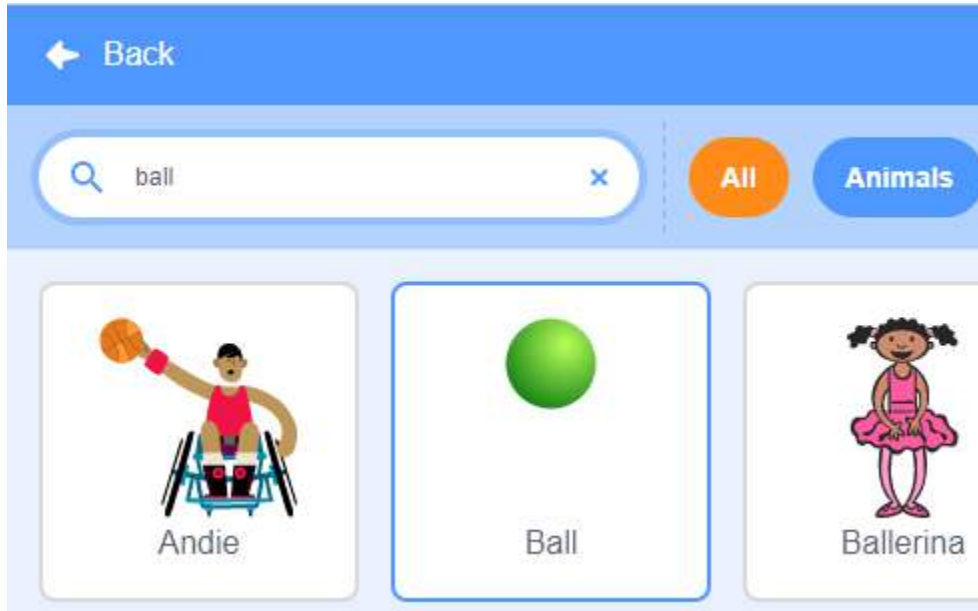


- Breadboard
- RGB LED
- Resistor

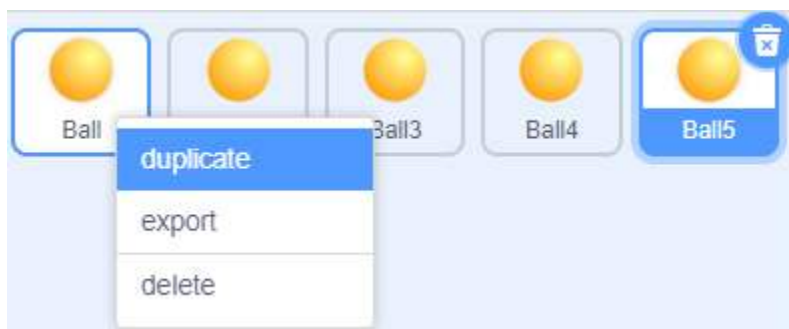
3.6.3 Programming

1. Select sprite

Delete the default sprite, then choose the **Ball** sprite.

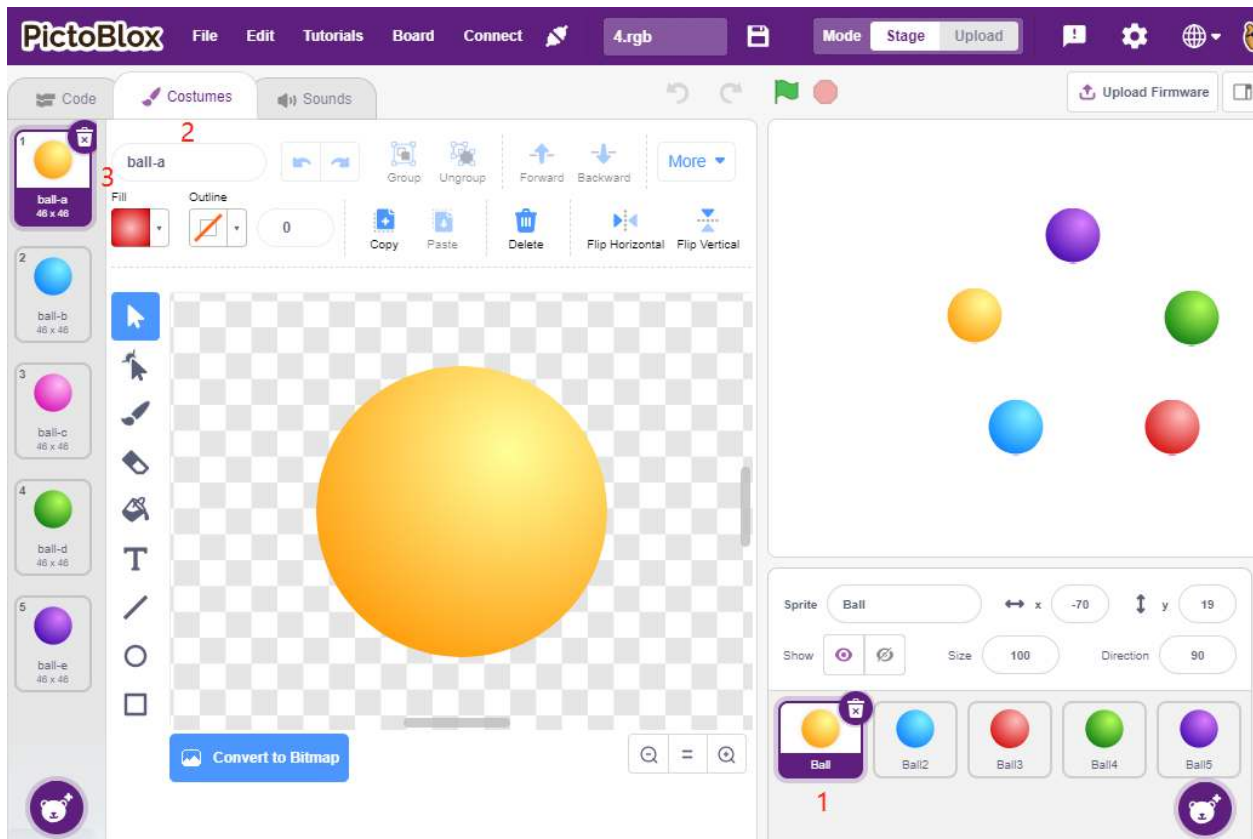


And duplicate it 5 times.



Choose different costumes for these 5 **Ball** sprites and move them to the corresponding positions.

Note: **Ball3** sprite costume color needs to be manually changed to red.

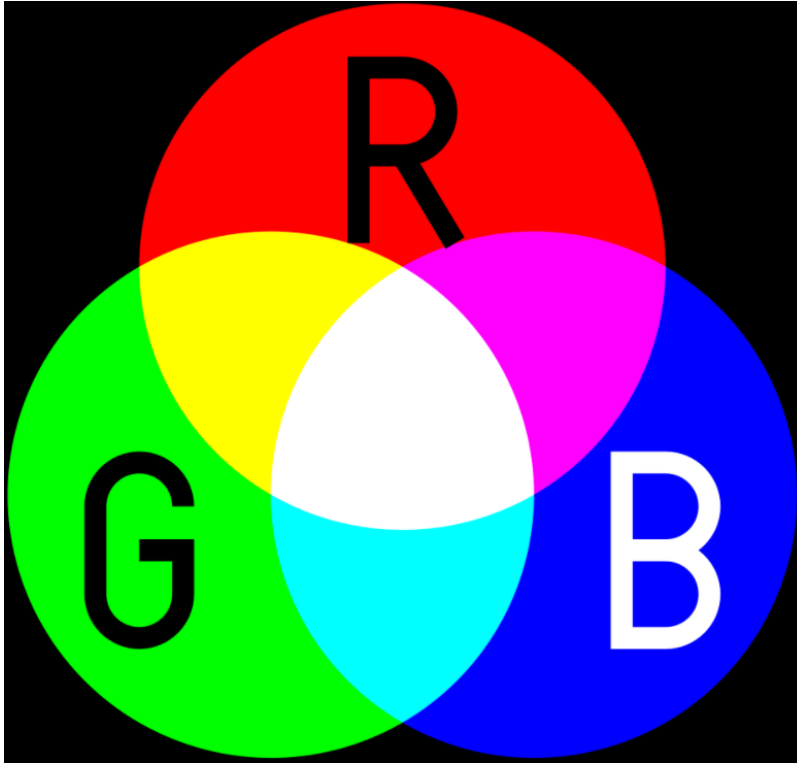


2. Make RGB LEDs light up in the appropriate color

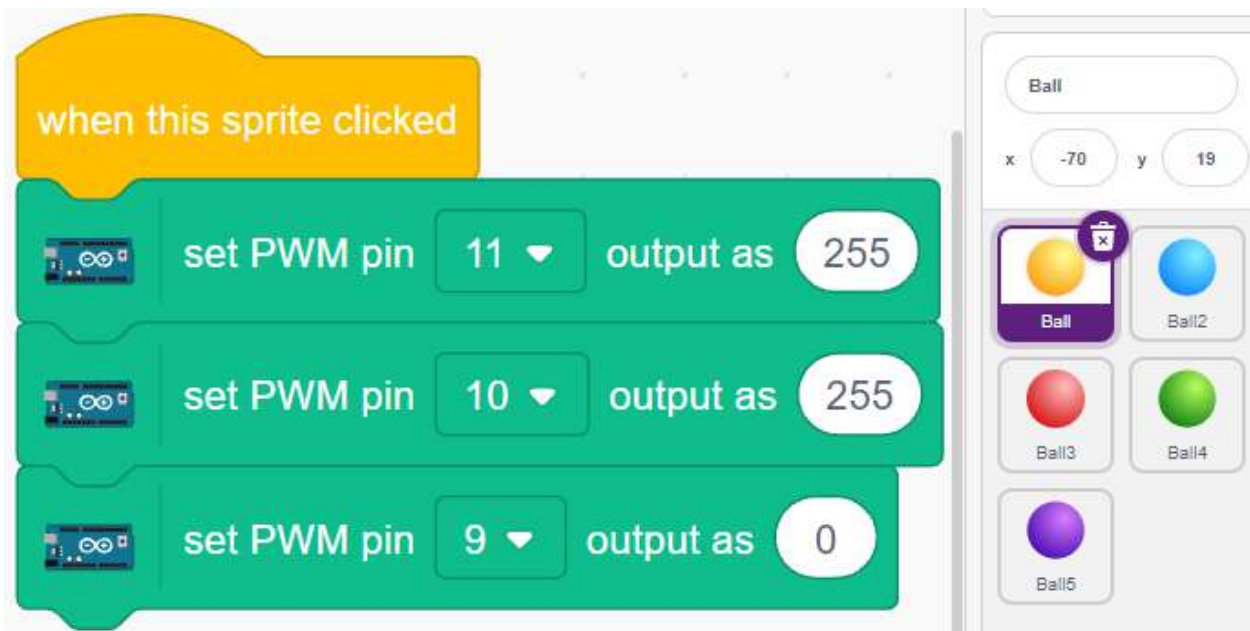
Before understanding the code, we need to understand the [RGB color model](#).

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.

Additive color mixing: adding red to green yields yellow; adding green to blue yields cyan; adding blue to red yields magenta; adding all three primary colors together yields white.



So the code to make the RGB LED light yellow is as follows.



When the Ball sprite (yellow ball) is clicked, we set pin 11 high (red LED on), pin 10 high (green LED on) and pin 9 low (blue LED off) so that the RGB LED will light yellow.

You can write codes to other sprites in the same way to make the RGB LEDs light up in the corresponding colors.

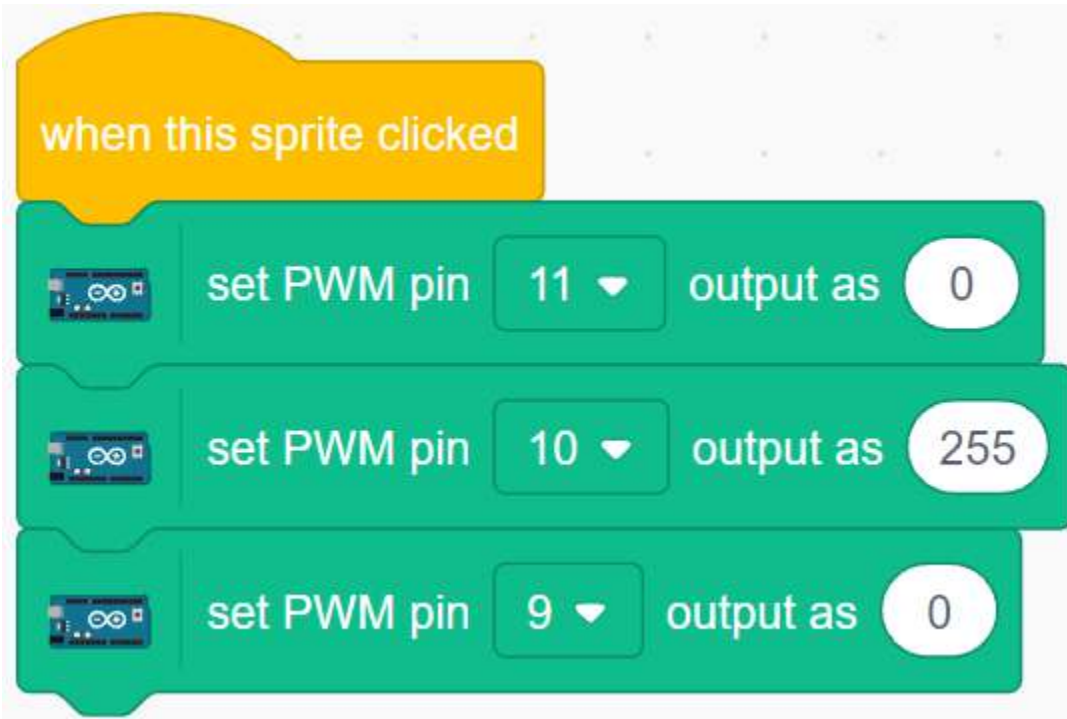
3. Ball2 sprite (light blue)



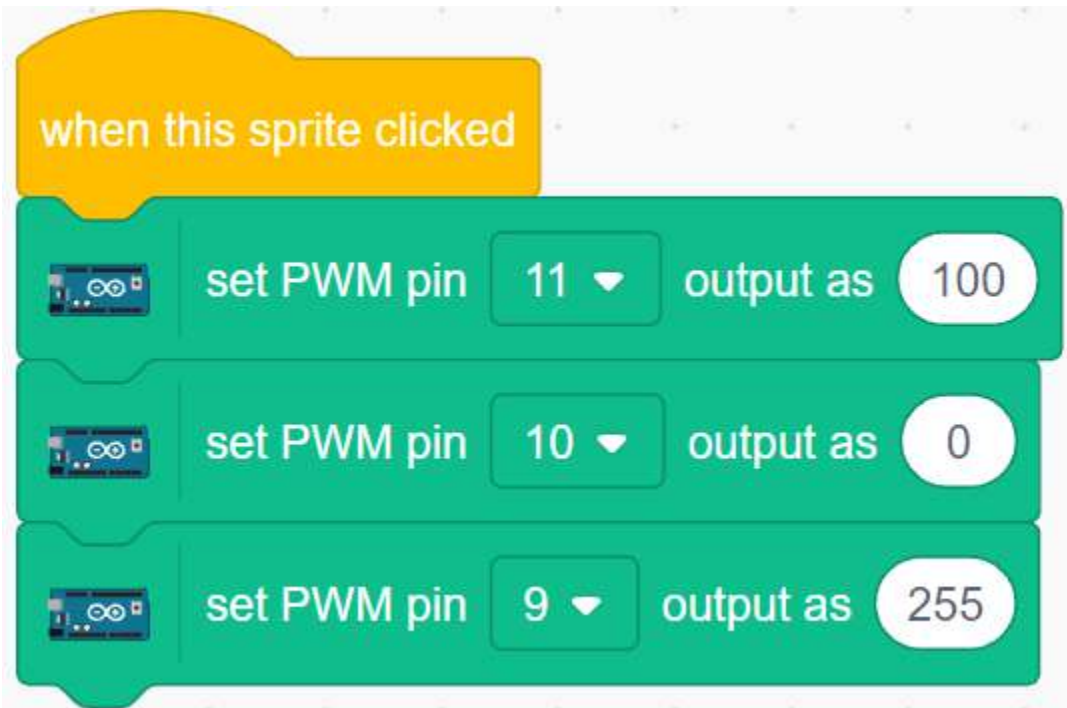
4. Ball3 sprite (red)



5. Ball4 sprite (green)



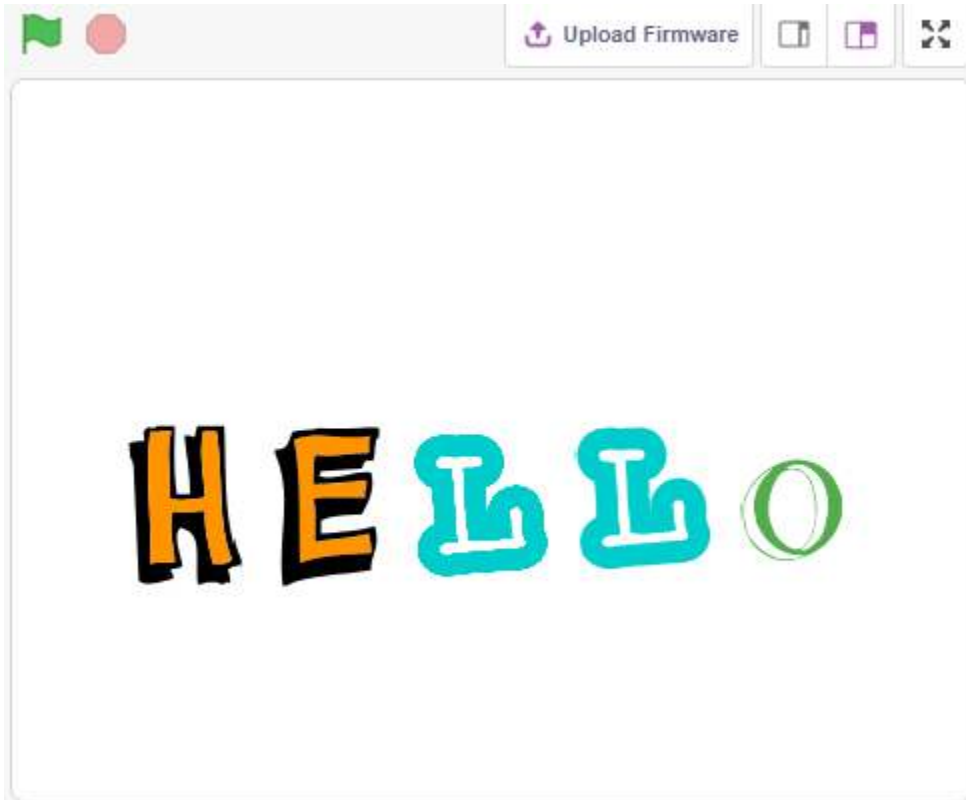
6. Ball5 sprite (purple)



3.7 2.4 LCD1602

LCD1602 can be used to display 2x16 characters, now we let it display the corresponding characters with the character sprites on the stage.

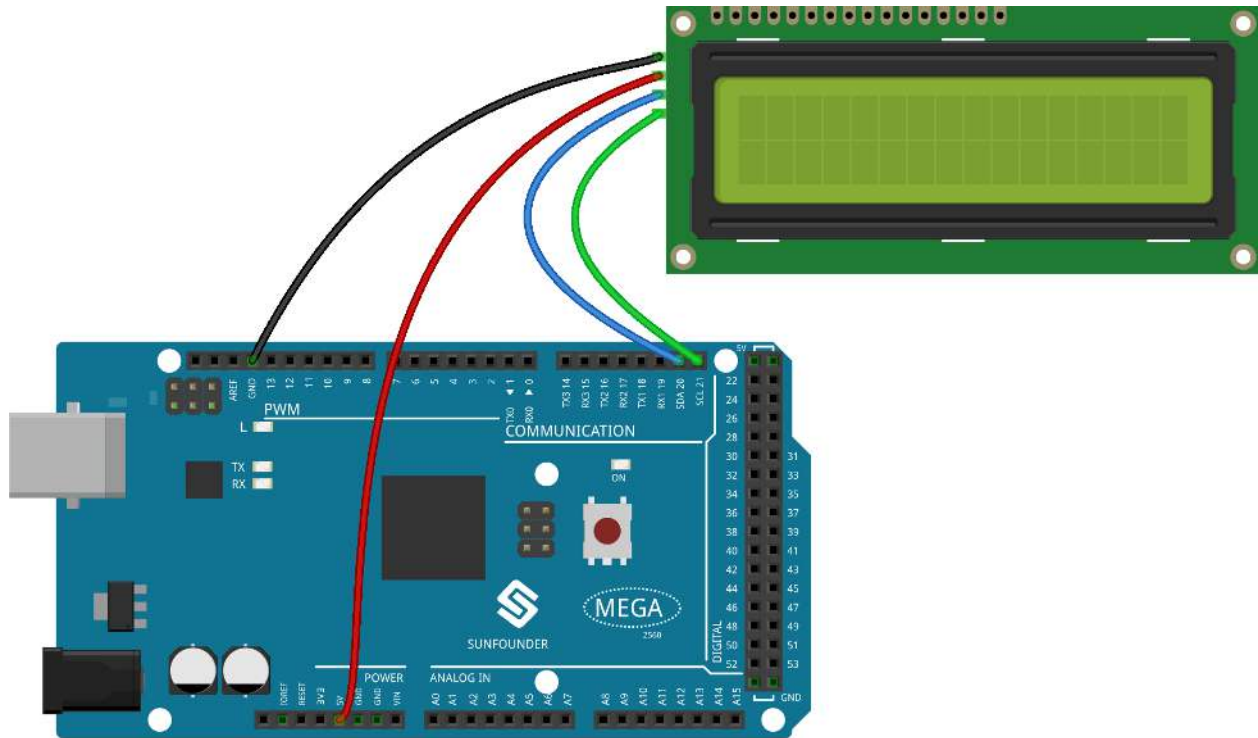
When you click the Hello on the stage one by one, they will have different animation effects and the characters will be displayed on the LCD1602 at the same time.



3.7.1 You Will Learn

- Using the LCD1602
- Select multiple different sprites
- Change sprite size, rotation angle, color and show or hide.

3.7.2 Build the Circuit

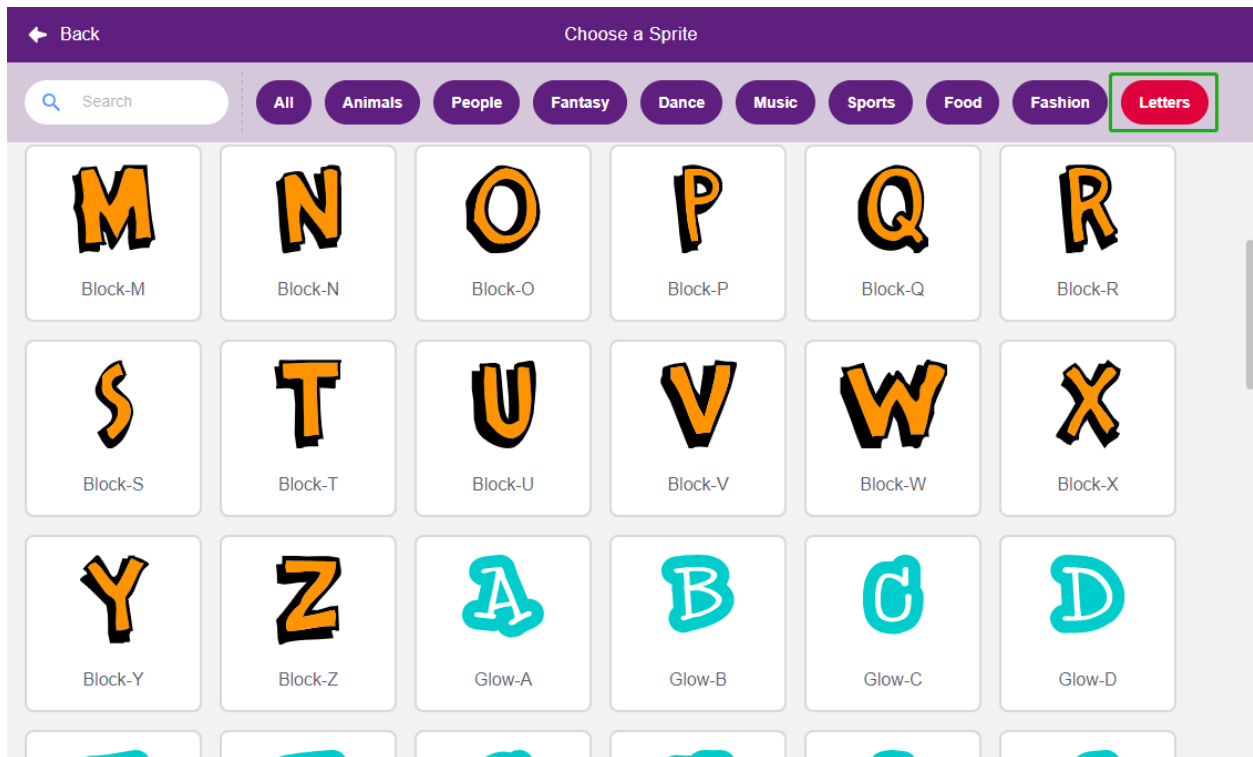


- *SunFounder Mega Board*
- *I2C LCD1602*

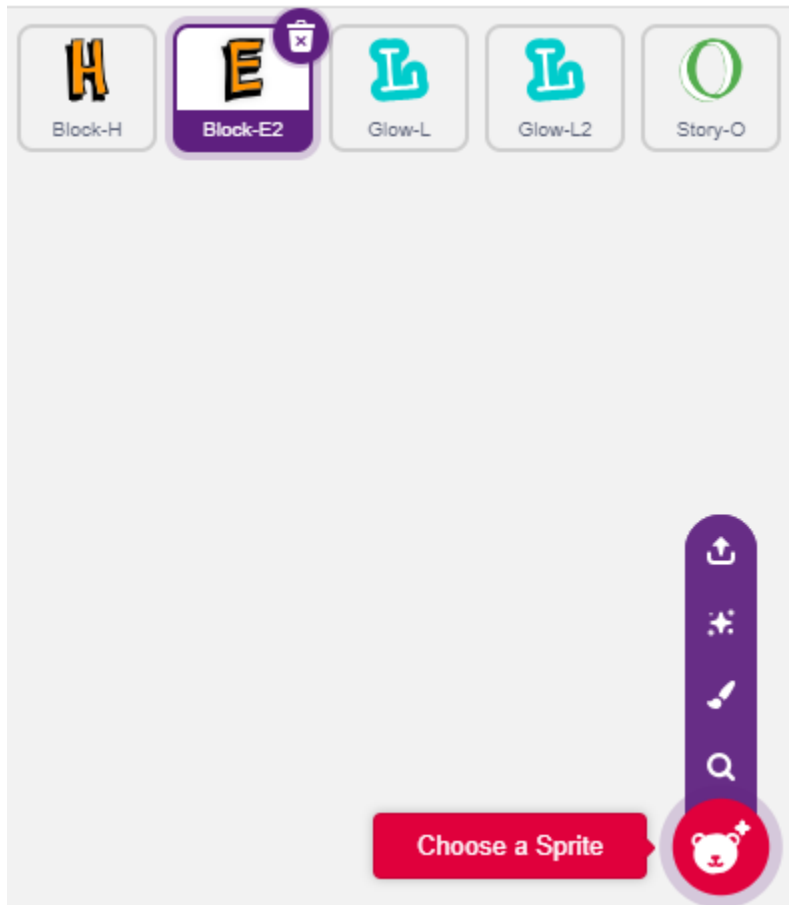
3.7.3 Programming

1. Select sprite

Delete the default sprite, click **Choose a Sprite**, then click **letters** and select the sprite you want.



For example, I chose Hello, as shown below.



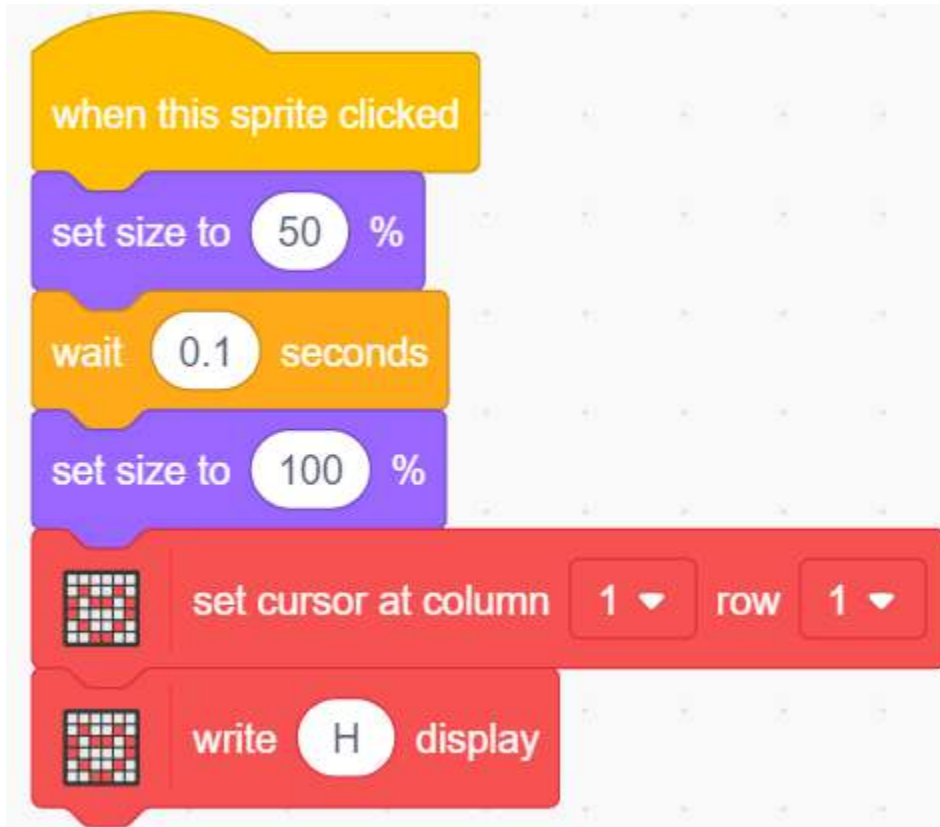
Now to set different effects for these sprites and display them on the LCD1602 while clicking.

2. H is zoom in and zoom out

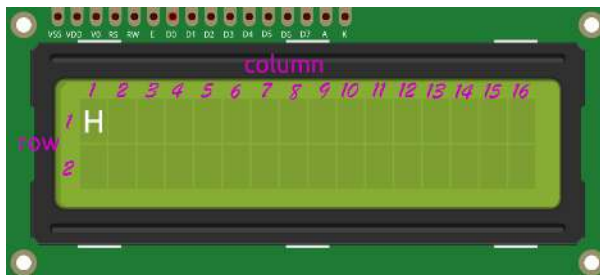
Click on the **H** sprite, and now write a script for it.

When the sprite **H** is clicked, make its size to 50%, then restore it; while displaying H on the first row and column of the LCD1602.

- [set size to]: From **Looks** palette, used to set the size of the sprite, from 0% to 100%.
- [set cursor at column row]: From **Display Modules** palette, used to set the cursor at a specific row of the LCD1602 to start displaying characters.
- [write display]: From the **Display Modules** palette, used to display characters or strings on the LCD1602.



The distribution of rows and columns on the LCD1602 is shown in the figure.

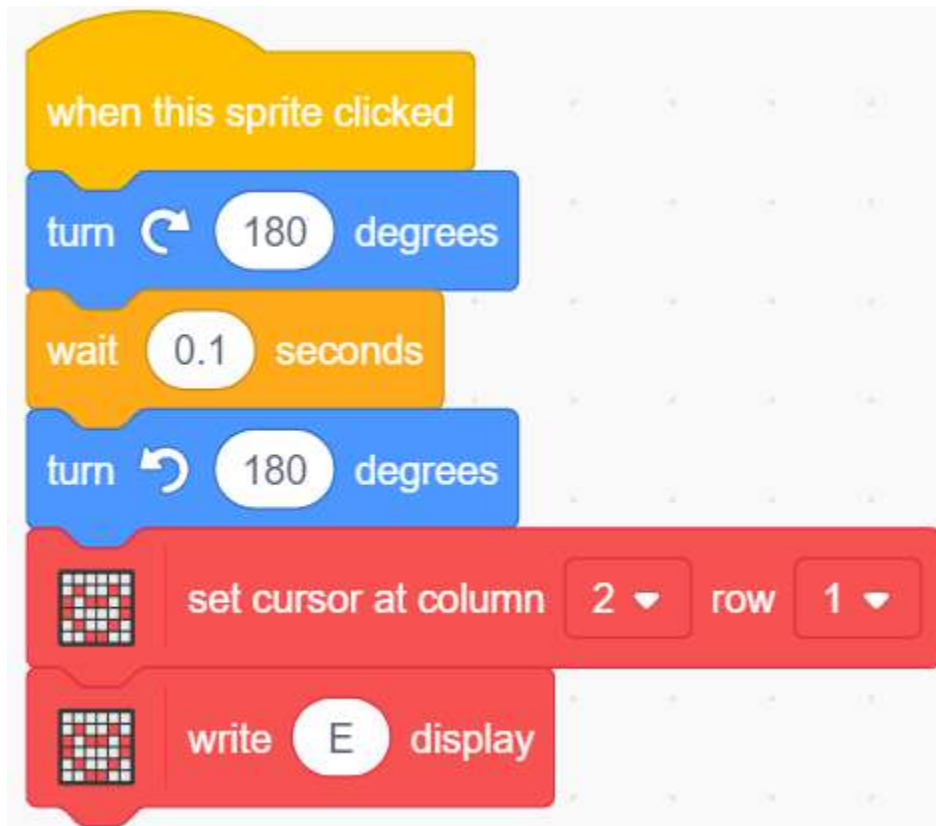


3. E is flipping left and right

Click on the **E** sprite, and now write a script for it.

When the sprite **E** is clicked, have it turn 180 degrees clockwise, then 180 degrees counterclockwise so you can see it flip left and right; and show H in the first row and column 2 of the LCD1602.

- [turn degrees]: From the **Motions** palette, used to turn the sprite clockwise or counterclockwise, the range is 0-360 degrees.

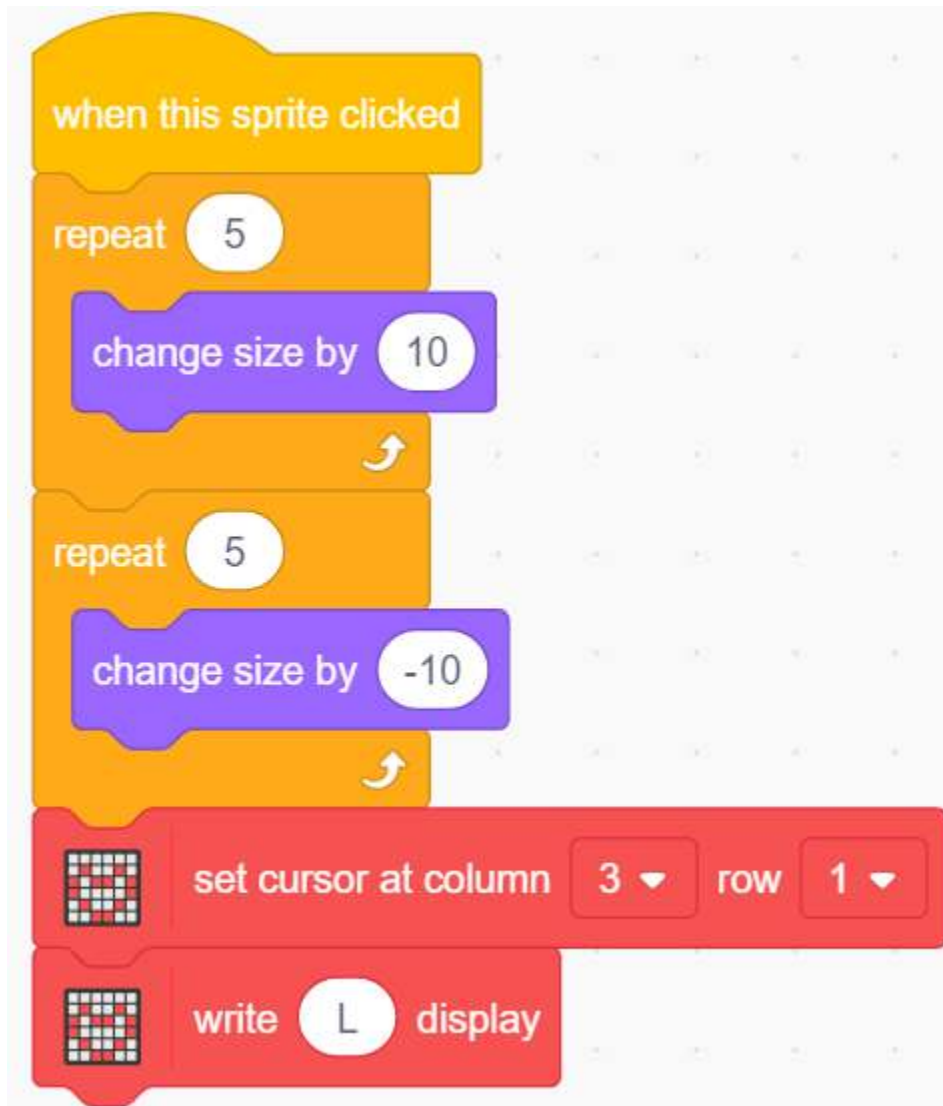


4. L is slowly shrinking and zooming in

Click on the **first L** sprite and now write a script for it.

When the sprite **L** is clicked, use the [repeat] block to increase its size by 50% (5 times, 10 each time), then shrink it back to its original size in the same way, while displaying L in the first row and column 3 of the LCD1602.

- [change size by]: From the Motions palette, used to change the size of the sprite.

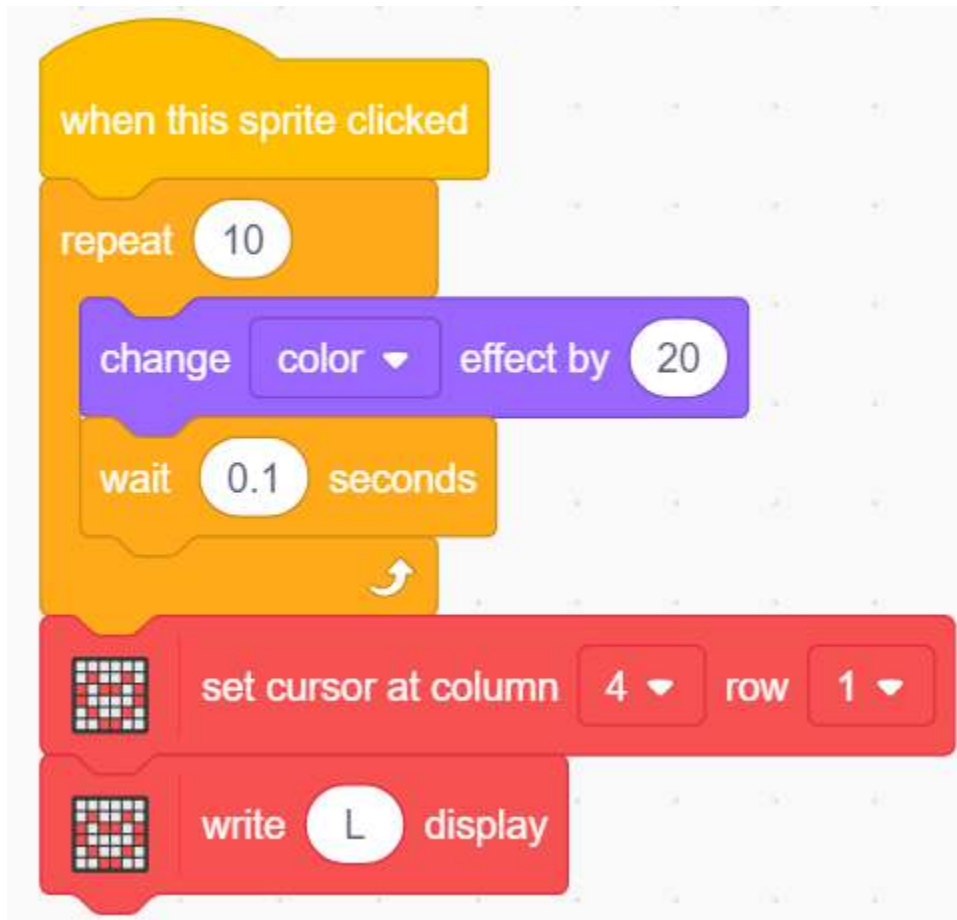


5. The second L is changing color

Click on the **second L** sprite and now write a script for it.

When the sprite **L** is clicked, use the [repeat] block to repeat 10 times at a rate of 20 increments to switch between colors and return to the original color. Also display L in the first row and column 4 of the LCD1602.

- [change color effect by]: Used to change the color Effect, one costume can take on 200 different color-schemes using the color effect, 0 and 200 are the same color.

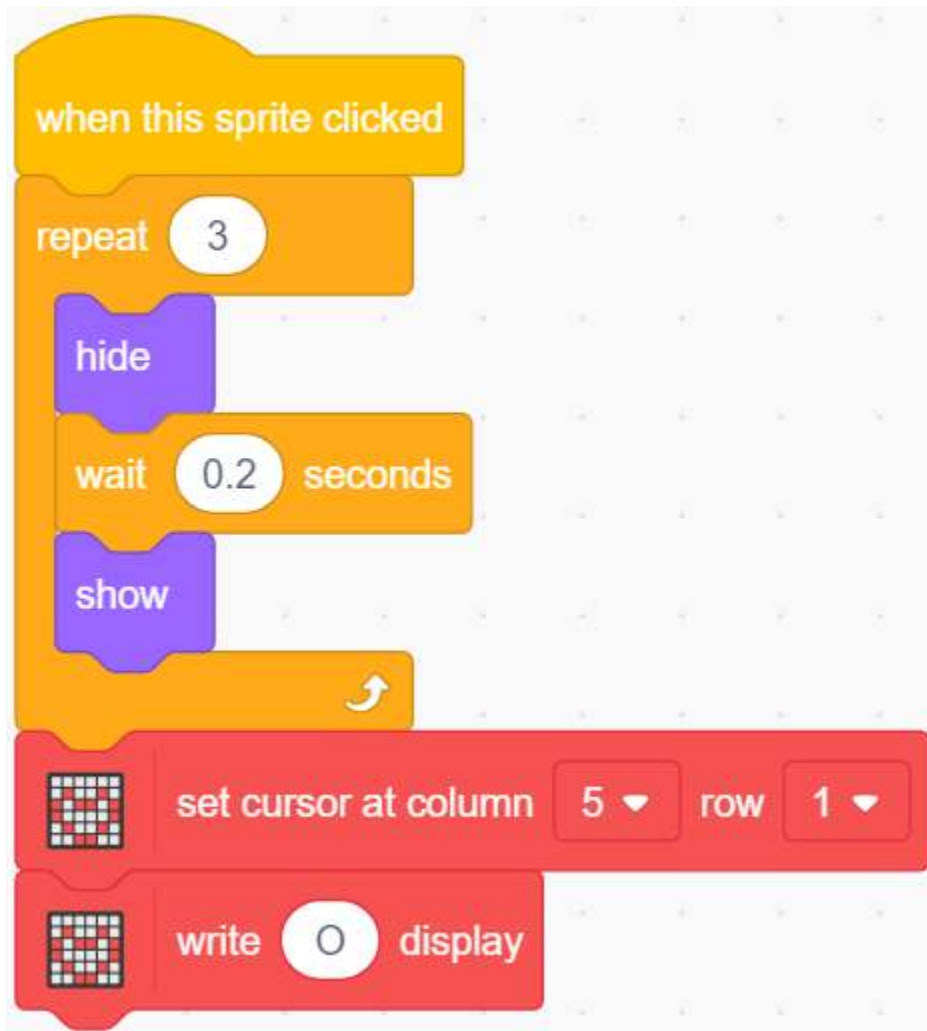


6. O is hide and show

Click on the **O** sprite and now write a script for it.

When the **O** sprite is clicked, it repeats the hide and show process 3 times, while displaying O in the first row and column 5 of the LCD1602.

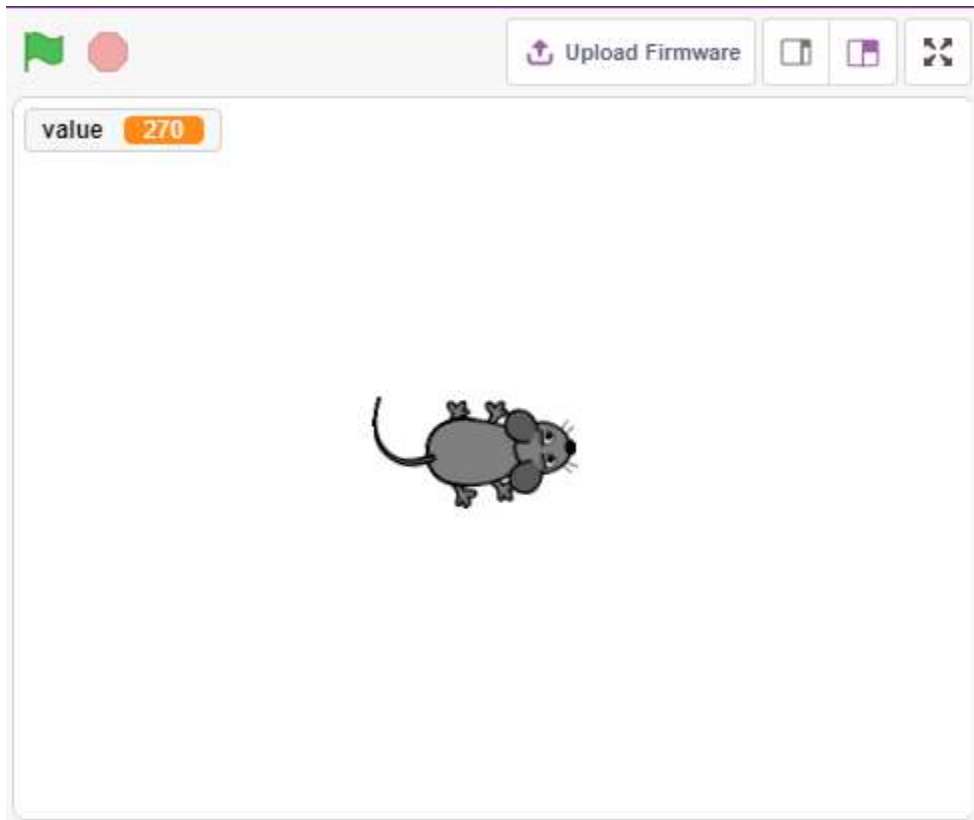
- [Hide] & [Show]: make the sprite hide and show.



3.8 2.5 Moving Mouse

Today we are going to make a mouse toy controlled by a potentiometer.

When the green flag is clicked, the mouse on the stage moves forward, and when you rotate the potentiometer, the mouse will change the direction of movement.

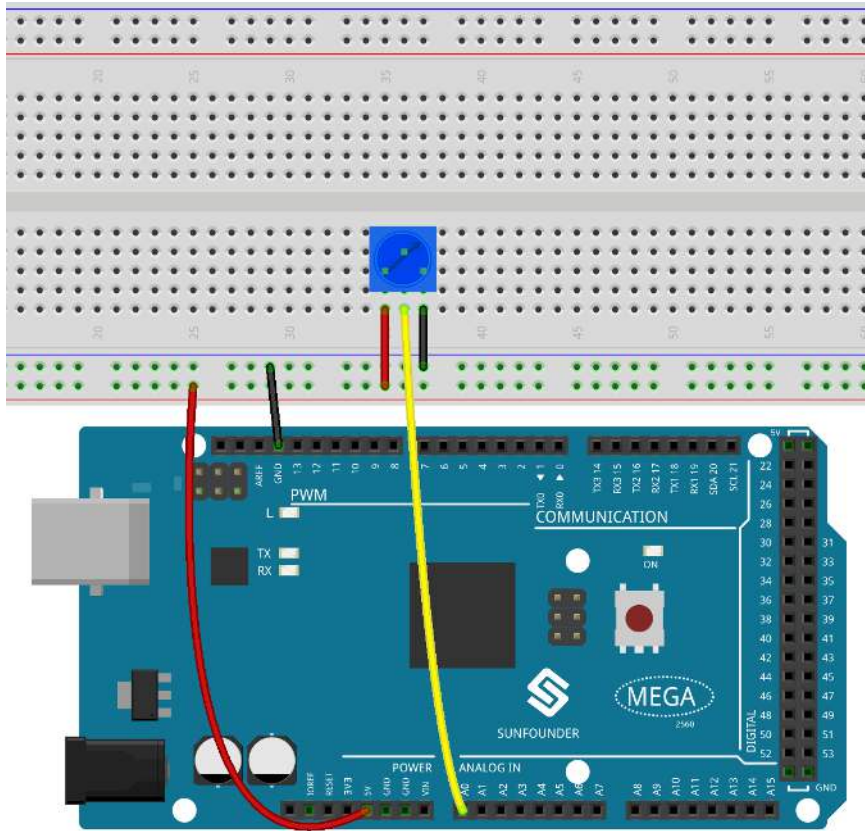


3.8.1 You Will Learn

- Potentiometer principle
- Read analog pin and ranges
- Mapping one range to another
- Moving and changing the direction of sprite

3.8.2 Build the Circuit

The potentiometer is a resistive element with 3 terminals, the 2 side pins are connected to 5V and GND, and the middle pin is connected to A0. After conversion by the ADC converter of the Arduino board, the value range is 0-1023.

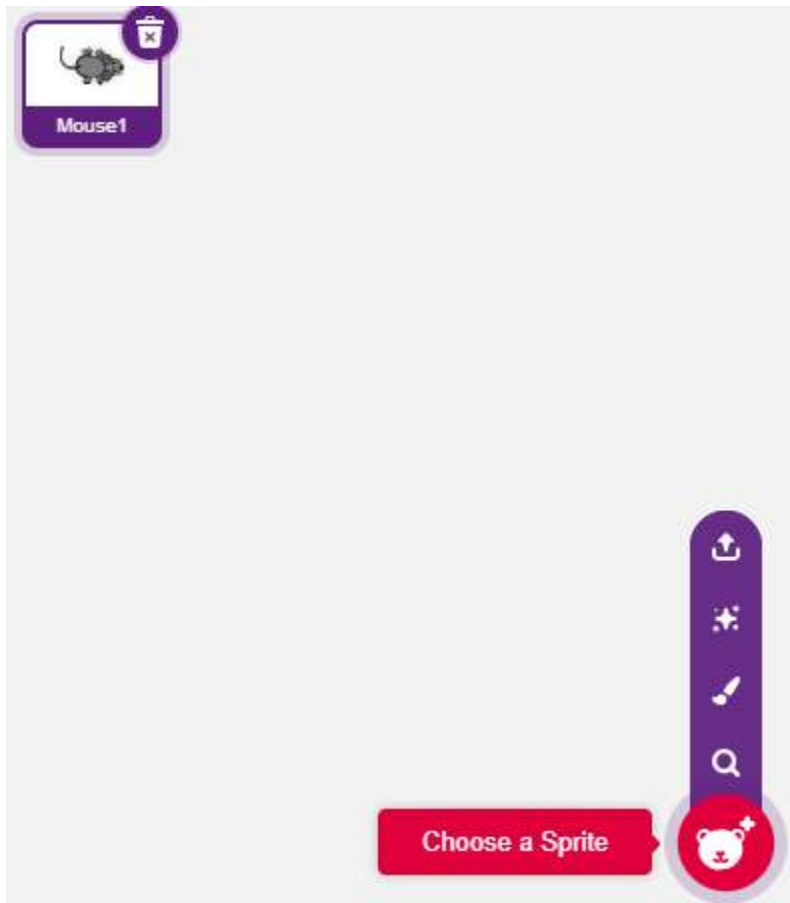


- *Breadboard*
- *Potentiometer*

3.8.3 Programming

1. Choose a sprite

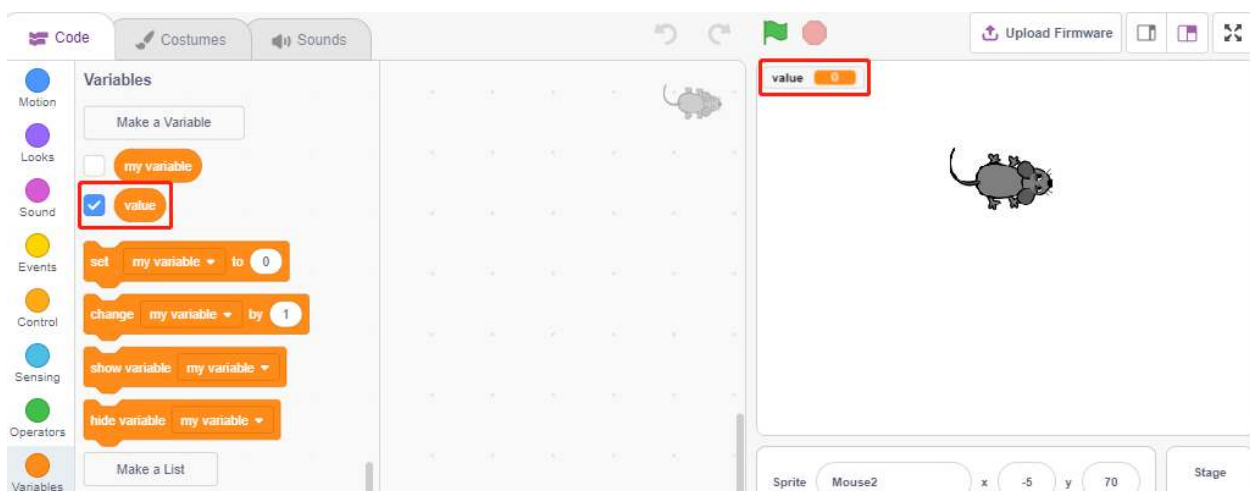
Delete the default sprite, click the **Choose a Sprite** button in the lower right corner of the sprite area, enter **mouse** in the search box, and then click to add it.



2. Creating a variable.

Create a variable called **value** to store the value of the potentiometer read.

Once created, you will see **value** appear inside the **Variables** palette and in the checked state, which means this variable will appear on the stage.



3. Read the value of A0

Store the value of A0 read into the variable **value**.

- [set my variable to 0]: Set the value of the variable.

- [read analog pin A0]: Read the value of A0~A5 in the range of 0-1023.

The screenshot shows the Scratch IDE interface. On the left, the 'Variables' menu is open, showing a variable named 'my variable' with a value of 0. The 'read analog pin A0' block is highlighted with a red box. The main workspace shows a script with the following blocks: 'set my variable to 0', 'change my variable by 1', and 'read analog pin A0'. The 'read analog pin A0' block is highlighted with a red box. The 'my variable' variable is also highlighted with a red box.

To be able to read all the way through, you need to use the [forever] block. Click on this script to run it, rotate the potentiometer in both directions, and you will see that the value range is 0-1023.

The screenshot shows a 'forever' loop block in the Scratch IDE. Inside the loop, there is a 'set value to read analog pin A0' block. The 'forever' loop block is highlighted with a yellow glow.

4. Move the sprite

Use the [move steps] block to move the sprite, run the script and you will see the sprite move from the middle to the right.

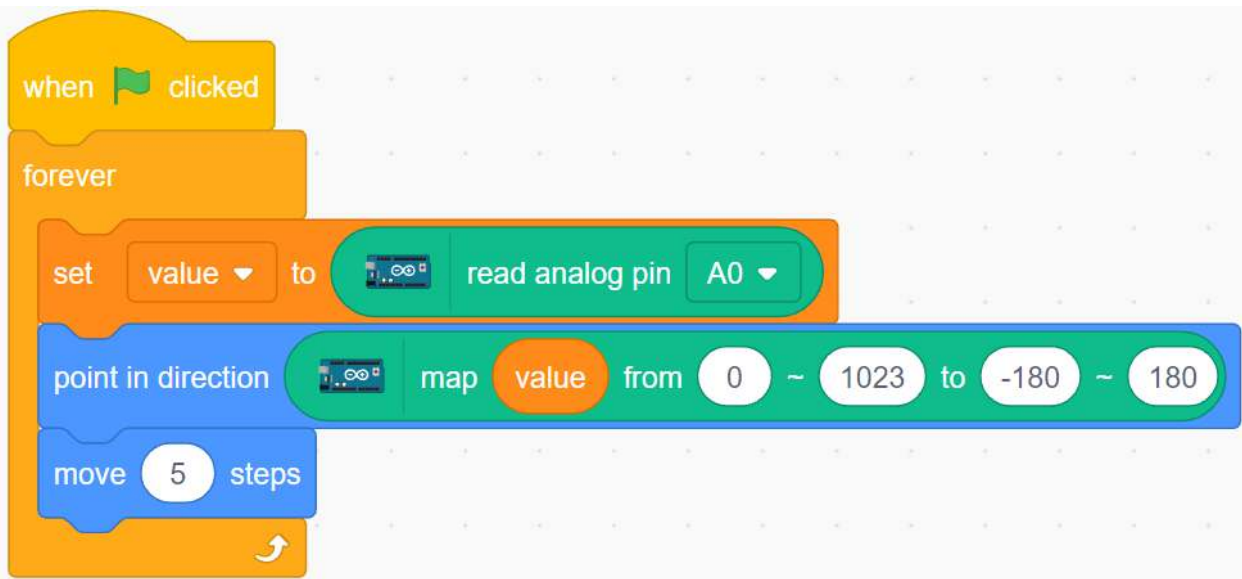


5. Changing the sprite's direction

Now change the direction of the sprite's movement by the value of A0. Since the value of A0 ranges from 0-1023, but the sprite's rotation direction is -180~180, a [map] block needs to be used.

Also add [when green flag clicked] at the beginning to start the script.

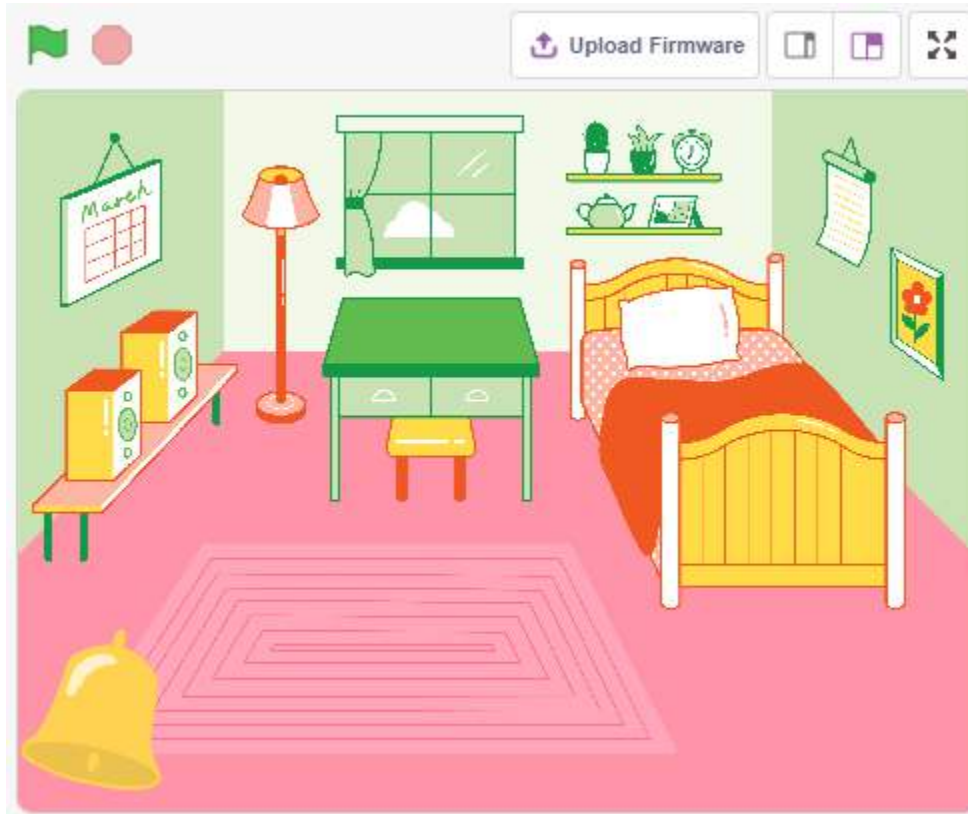
- [point in direction]: Set the steering angle of the sprite, from **Motion** palette.
- [map from to]: Map a range to another range.



3.9 2.6 Doorbell

Here, we will use the button and the bell on the stage to make a doorbell.

When the green flag is clicked, you can press the button and the bell on the stage will make a sound.

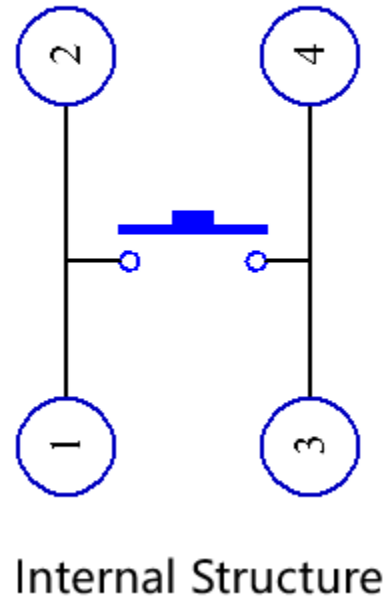
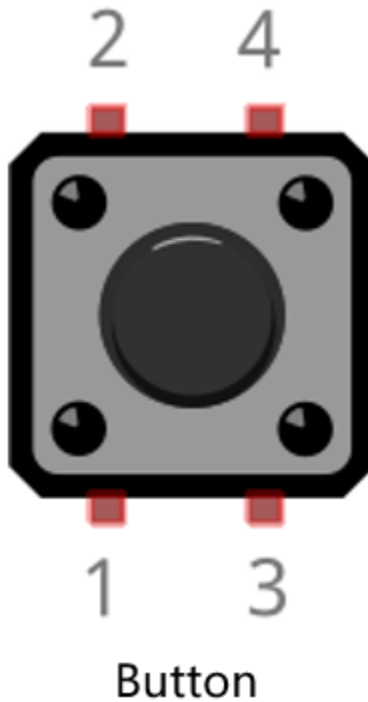


3.9.1 You Will Learn

- How the button work
- Reading digital pin and ranges
- Creating a conditional loop
- Adding a backdrop
- Playing sound

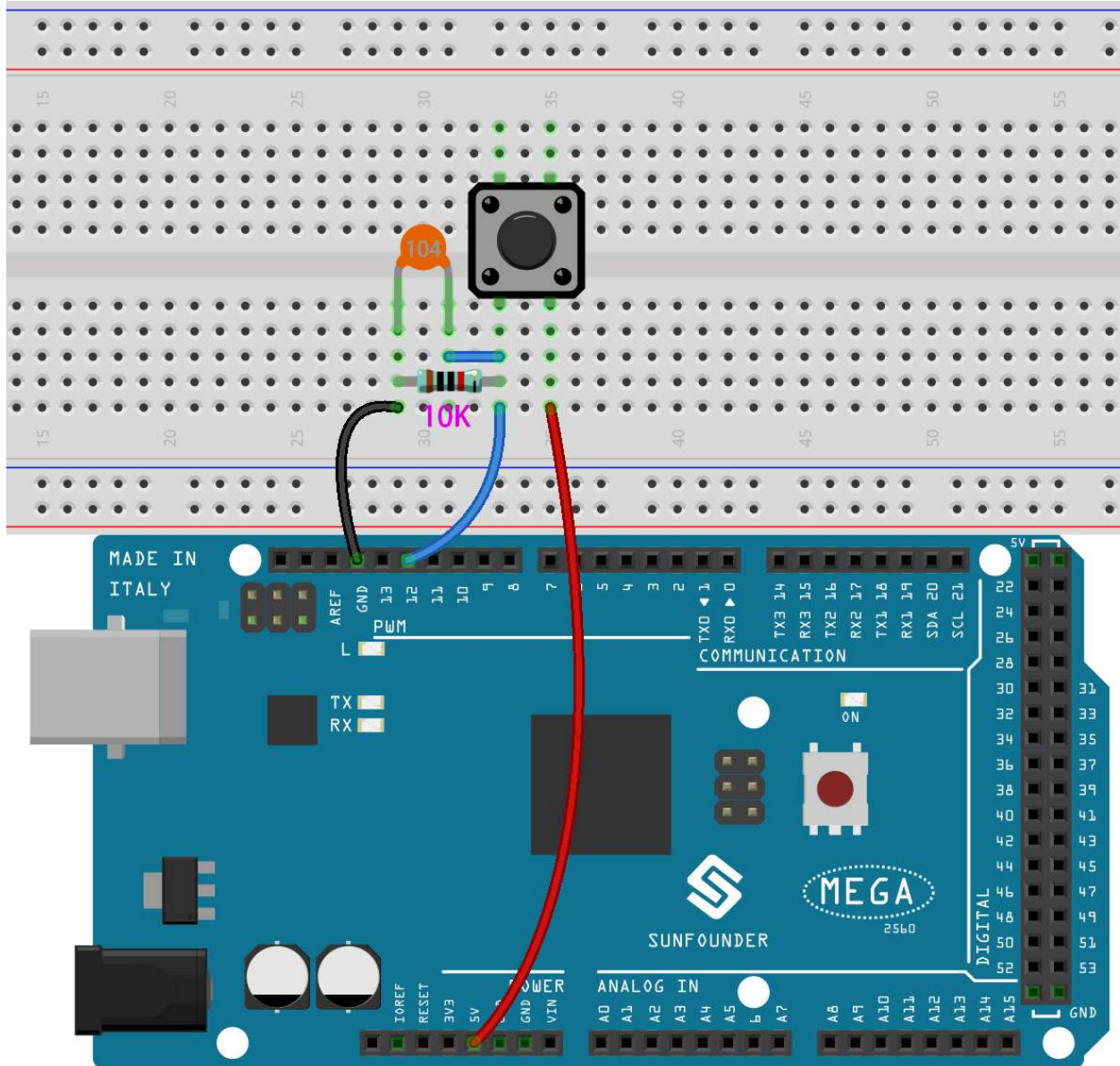
3.9.2 Build the Circuit

The button is a 4-pin device, since the pin 1 is connected to pin 2, and pin 3 to pin 4, when the button is pressed, the 4 pins are connected, thus closing the circuit.



Build the circuit according to the following diagram.

- Connect one of the pins on the left side of the button to pin 12, which is connected to a pull-down resistor and a 0.1 μ F (104) capacitor (to eliminate jitter and output a stable level when the button is working).
- Connect the other end of the resistor and capacitor to GND, and one of the pins on the right side of the button to 5V.

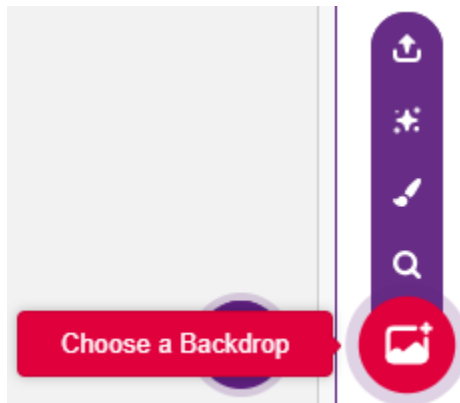


- Breadboard
- Button
- Resistor
- Capacitor

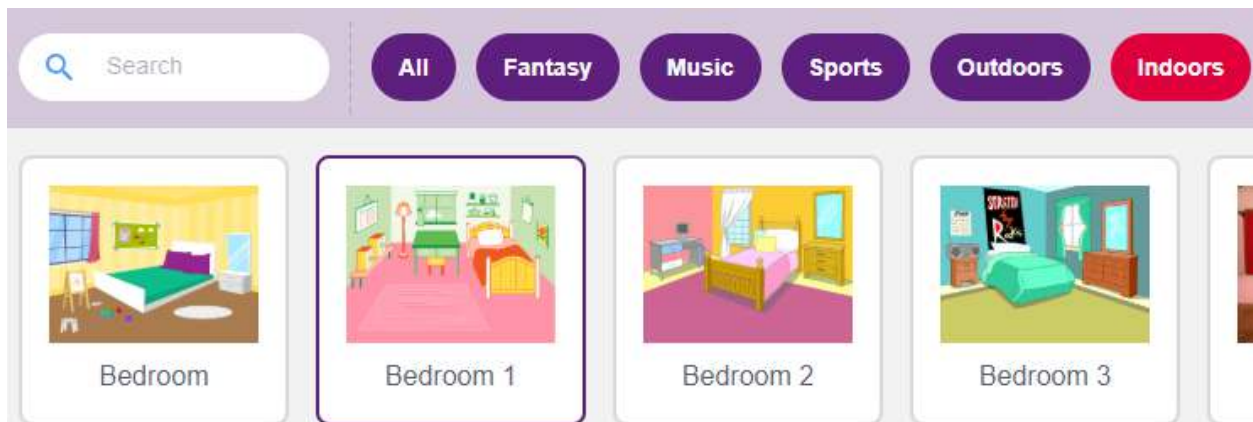
3.9.3 Programming

1. Add a Backdrop

Click the **Choose a Backdrop** button in the lower right corner.

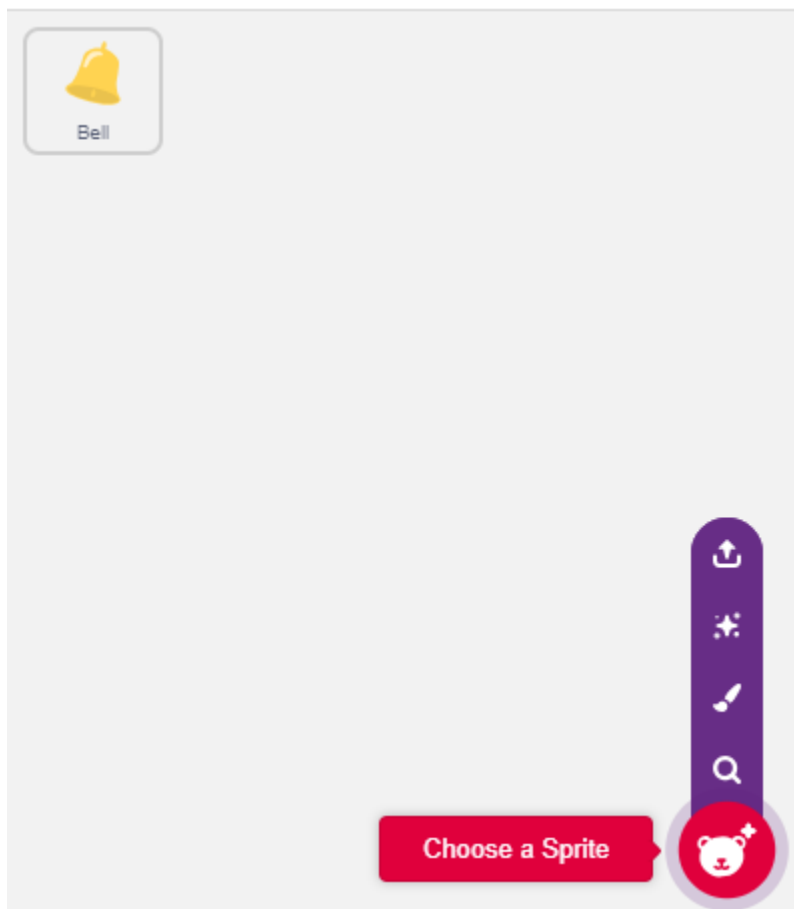


Choose **Bedroom 1**.

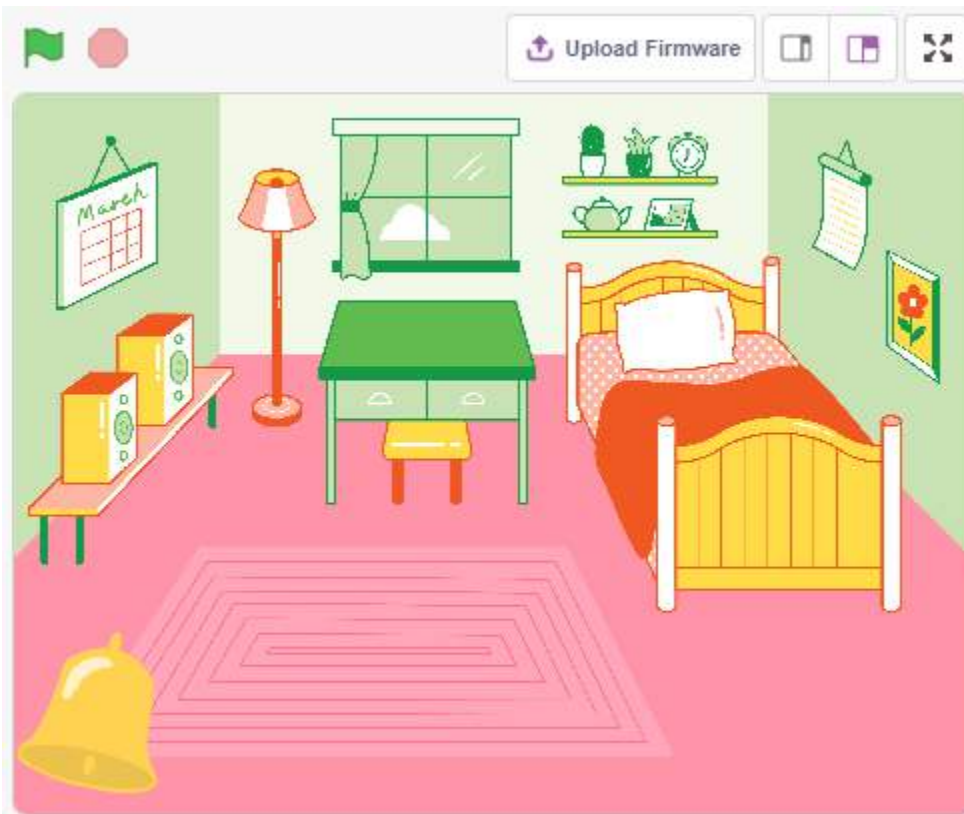


2. Select the sprite

Delete the default sprite, click the **Choose a Sprite** button in the lower right corner of the sprite area, enter **bell** in the search box, and then click to add it.



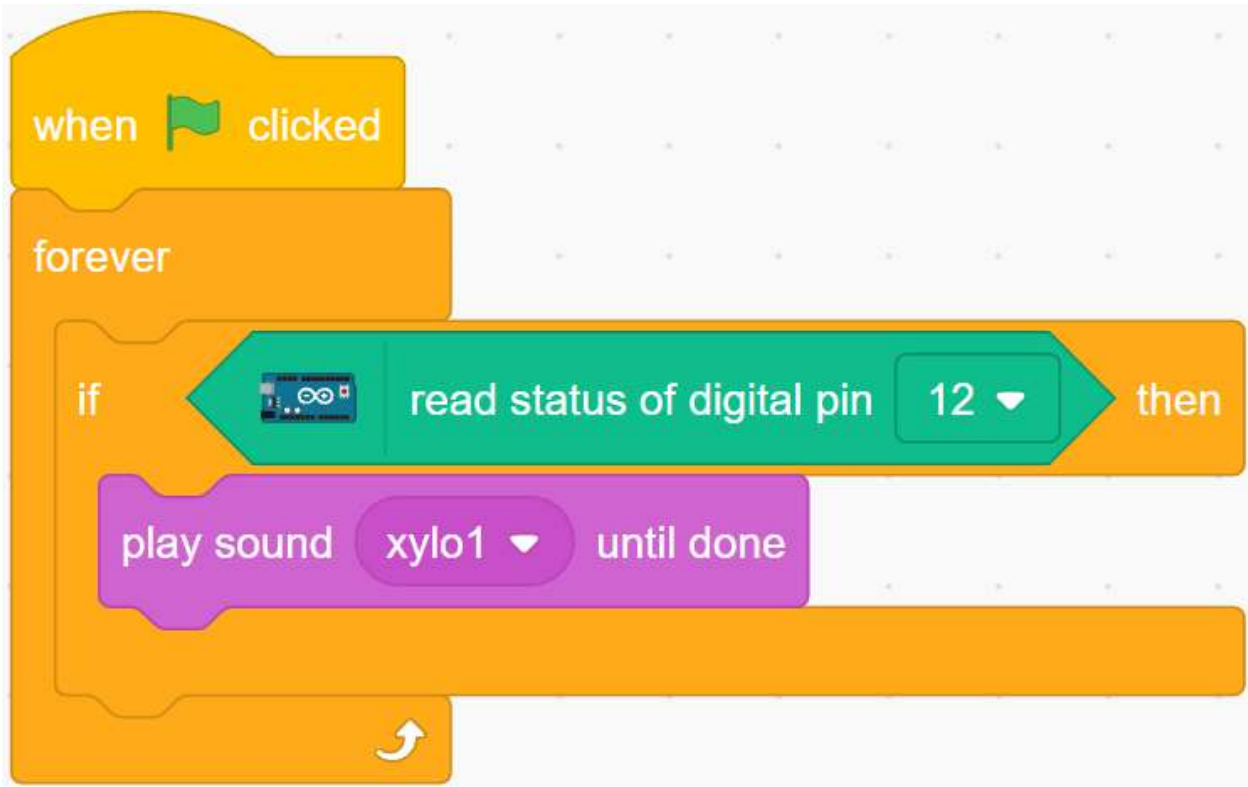
Then select the **bell** sprite on the stage and move it to the right position.



3. Press the button and the bell makes a sound

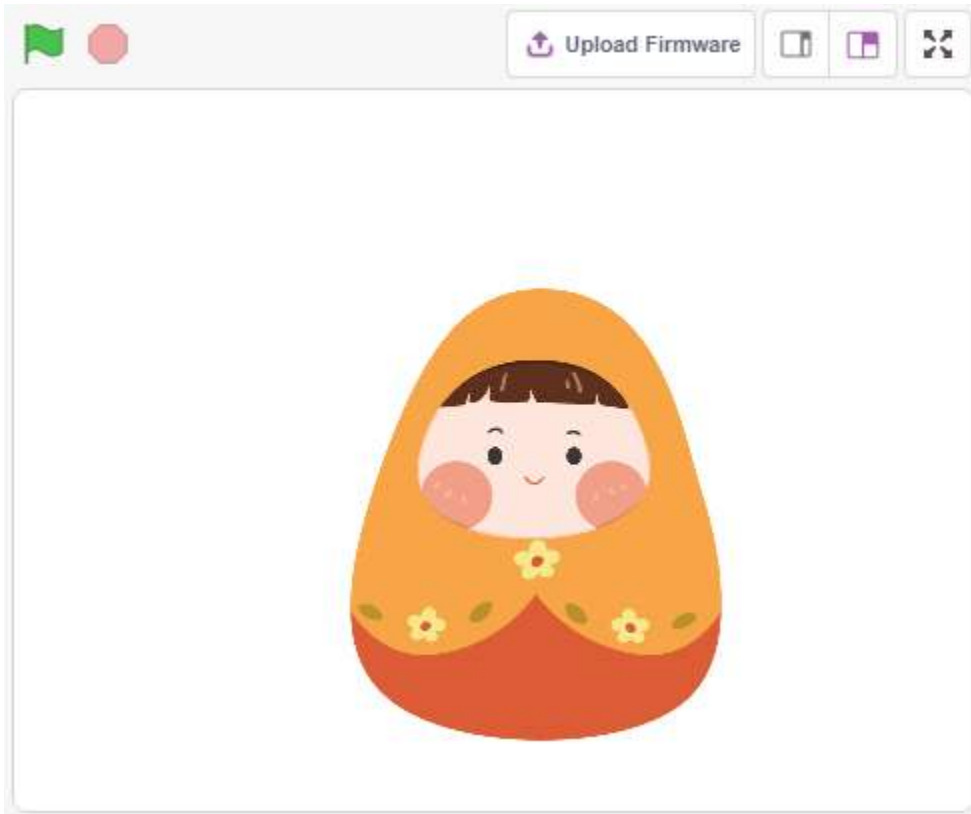
Use [if then] to make a conditional statement that when the value of the pin12 read is equal to 1 (the key is pressed), the sound **xylo1** will be played.

- [read status of digital pin]: This block is from the **Arduino Mega** palette and used to read the value of a digital pin, the result is 0 or 1.
- [if then]: This block is a control block and from **Control** palette. If its boolean condition is true, the blocks held inside it will run, and then the script involved will continue. If the condition is false, the scripts inside the block will be ignored. The condition is only checked once; if the condition turns to false while the script inside the block is running, it will keep running until it has finished.
- [play sound until done]: from the Sound palette, used to play specific sounds.



3.10 2.7 Tumbler

Now we use a tilt switch to control tumbler on the stage, so that the switch tilted, tumbler also tilted.



3.10.1 You Will Learn

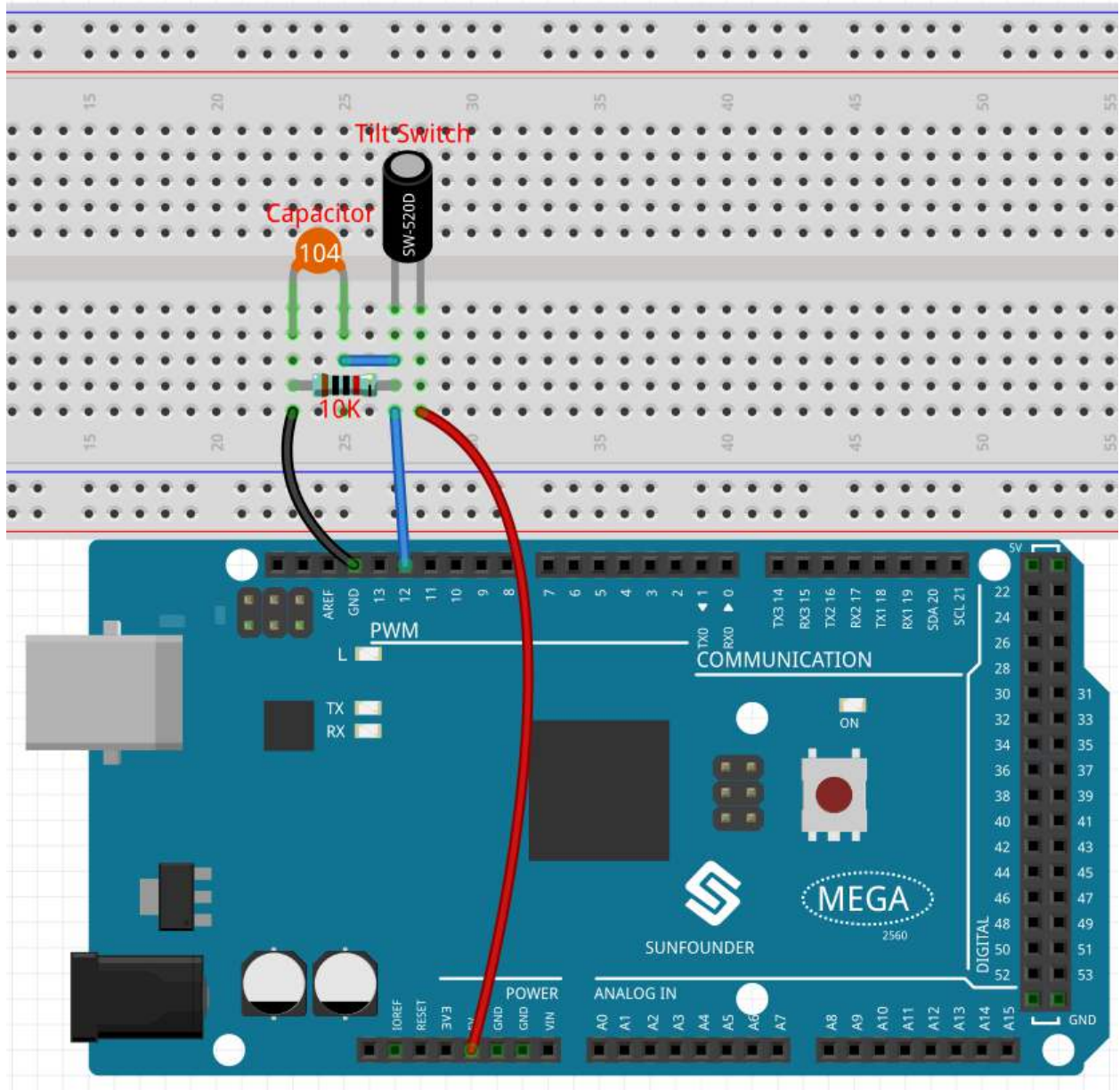
- How the tilt switch works
- [if then else] block
- Adding external sprite

3.10.2 Build the circuit

The tilt switch used here is a ball with a metal ball inside. When it is upright, the 2 pins are connected together, and when it is tilted, they are separate.

Build the circuit according to the following diagram:

- Connect one pin of the tilt switch to pin 12, which is connected to a pull-down resistor and a 0.1uF (104) capacitor (used to eliminate jitter and output a stable level when the tilt switch is operating).
- Connect the other end of the resistor and capacitor to GND and the other pin of the tilt switch to 5V.

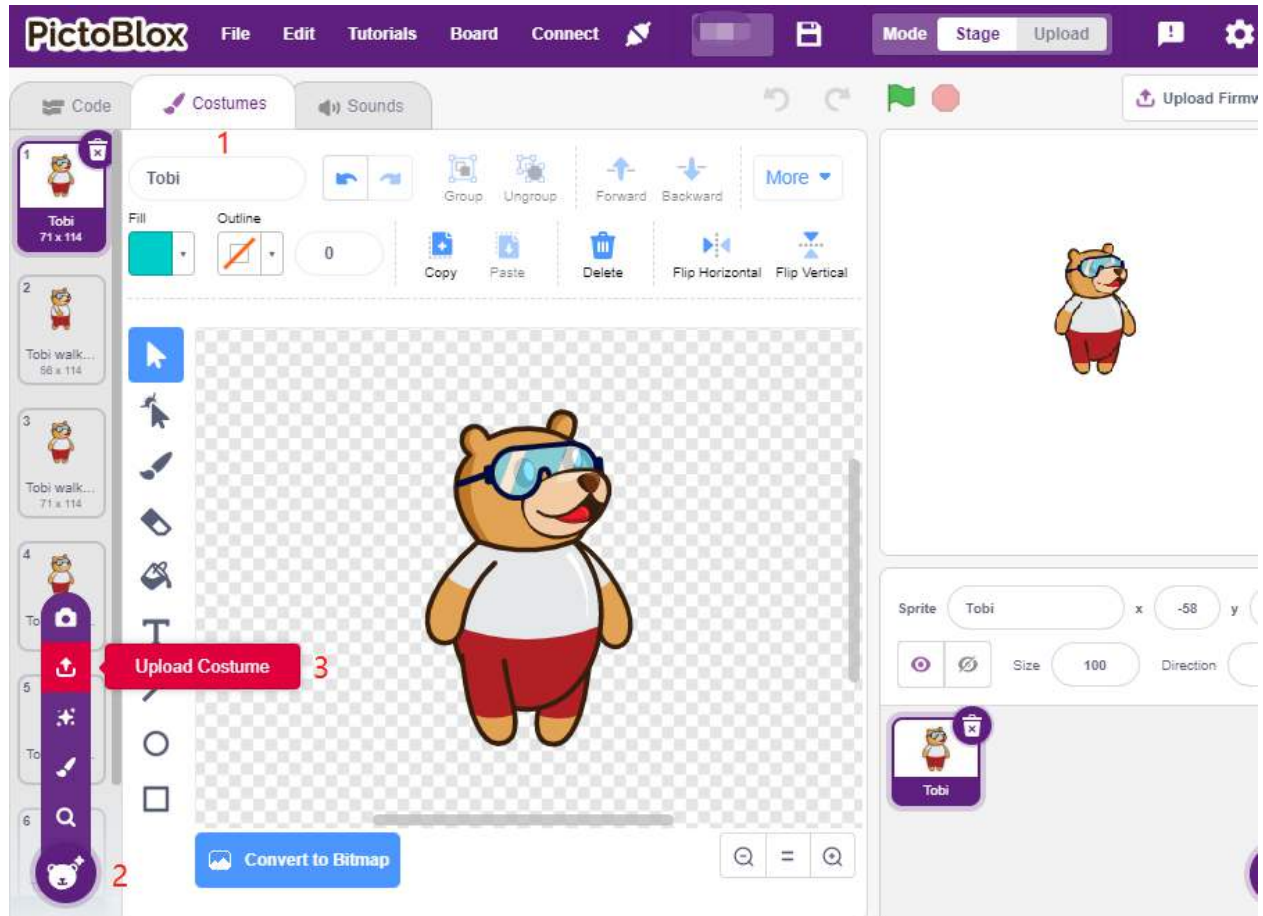


- Breadboard
- Tilt Switch
- Resistor
- Capacitor

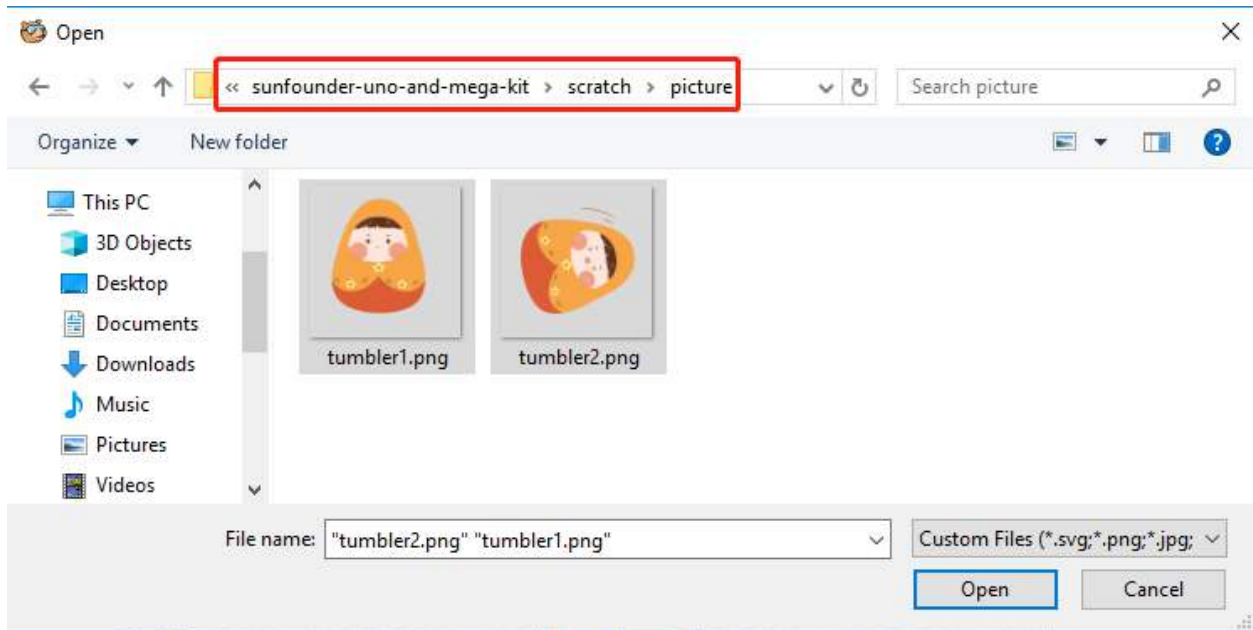
3.10.3 Programming

1. Customize the sprite

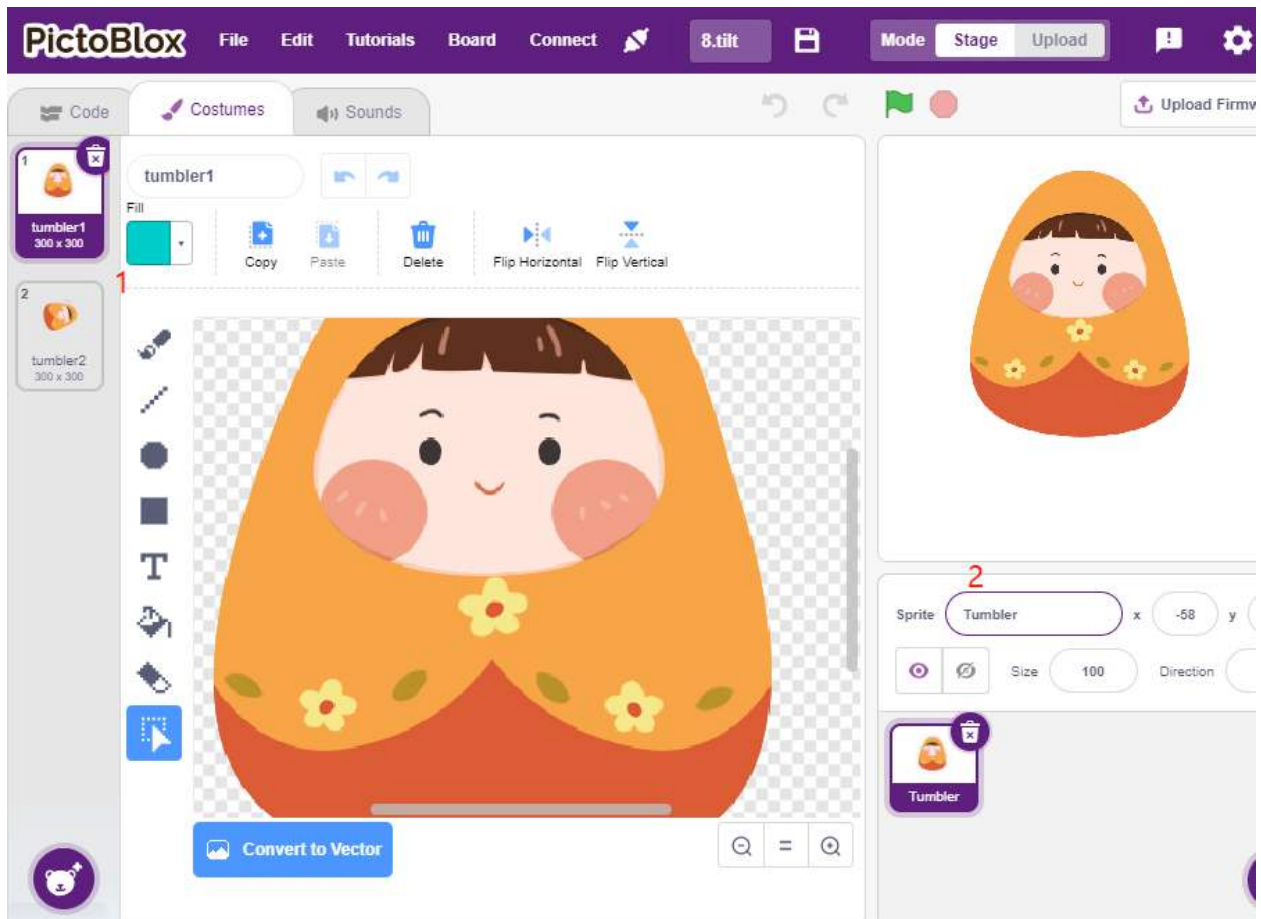
Select the **Tobi** sprite and go to the **Costumes** page. Select the icon in the bottom left corner and then select **Upload Costume**.



Then open `tumbler1.png` and `tumbler2.png` in the `sunfounder_vincent_kit_for_arduino\scratch\picture` path and make sure you have downloaded the relevant material from [github](#).



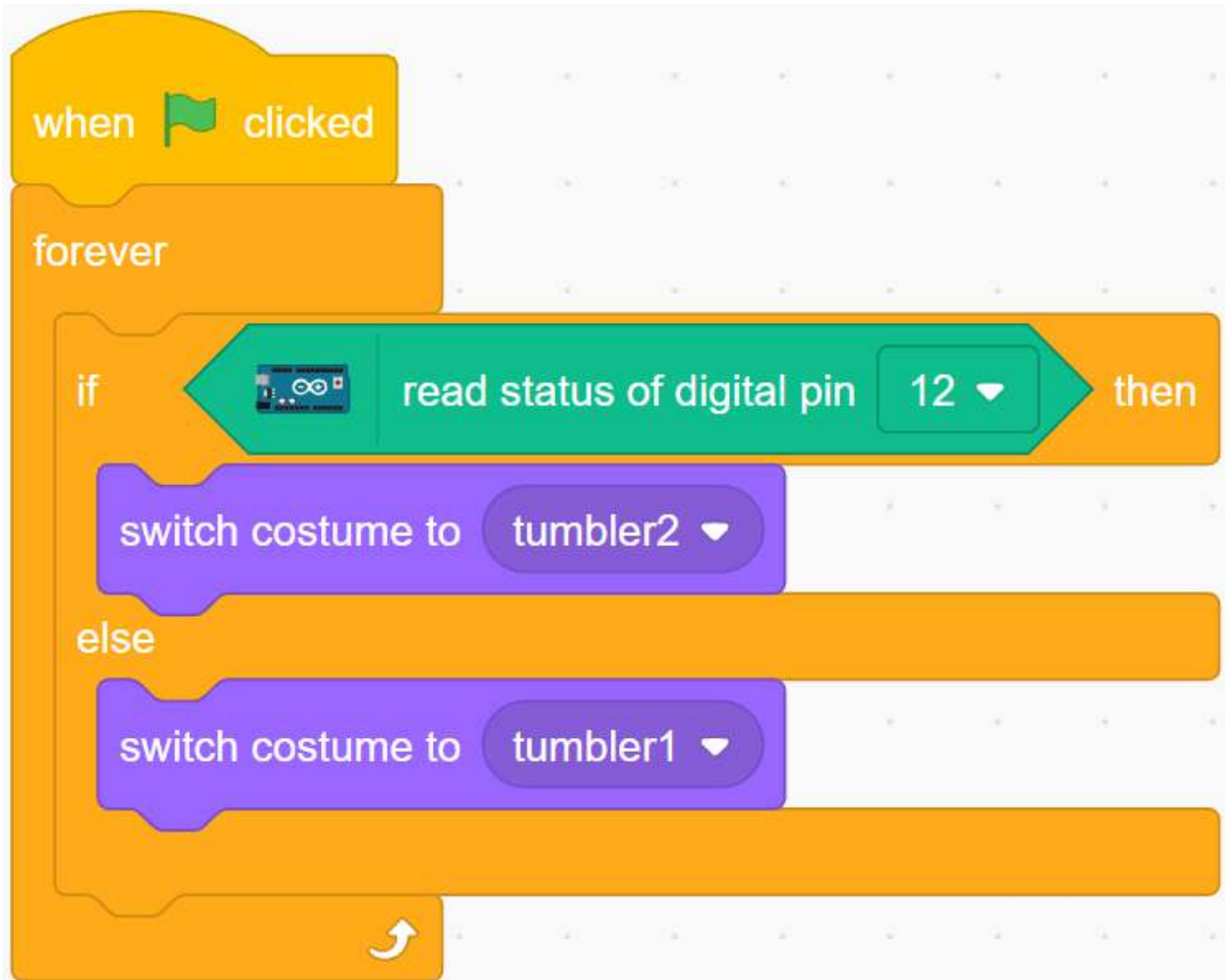
Remove the **Tobi** sprite related costume and change the name to **Tumbler**. Now that we have customized a new sprite Tumbler, we start scripting it.



2. Tilt the switch

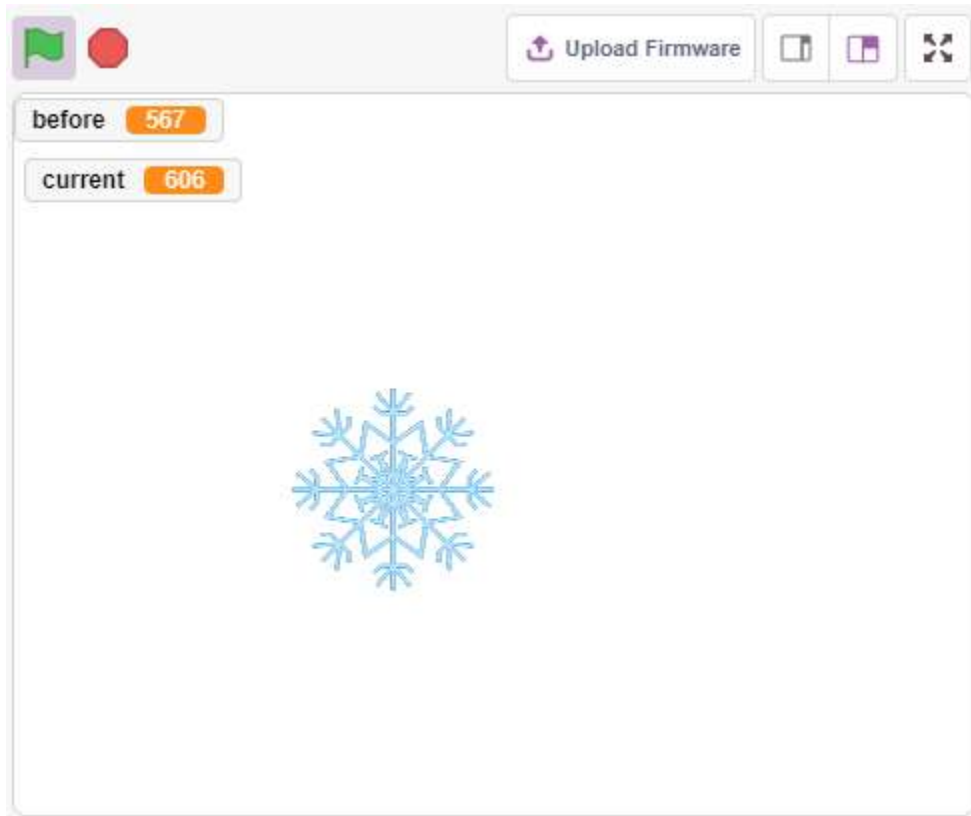
If the value of pin12 is read as 0 (the switch is tilted), switch the sprite costume to **tumbler2**, which is also in the tilted state. Otherwise, switch the sprite costume to **tumbler1**, upright.

- [if then else]: The block checks its boolean condition; if the condition is true, the code held inside the first C (space) will activate; if the if the condition is false, the code inside the second C will activate.
- [=]: The block is used to compare the equality of the values on the 2 sides of the equal sign, from the **Operators** platette.



3.11 2.8 Low Temperature Alarm

In this project, we will make a low temperature alarm system, when the temperature is below the threshold, the **Snowflake** sprite will appear on the stage.



3.11.1 You Will Learn

- Thermistor working principle
- Multivariable and Subtractive Operations

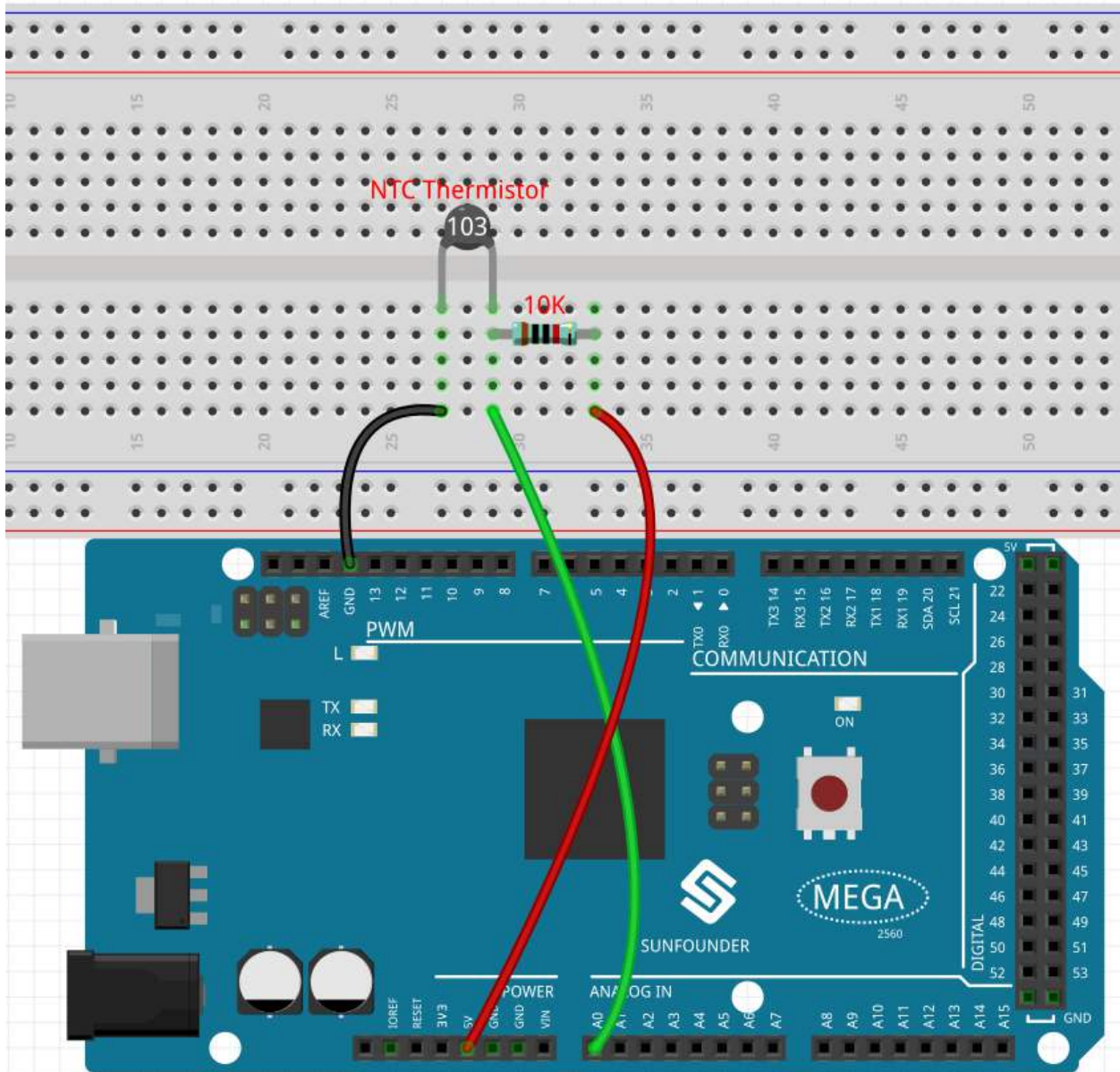
3.11.2 Build the Circuit

A thermistor is a type of resistor whose resistance is strongly dependent on temperature, more so than in standard resistors, and there are two types of resistors, PTC (resistance increases as temperature increases) and PTC (resistance decreases as temperature increases).

Build the circuit according to the following diagram.

One end of the thermistor is connected to GND, the other end is connected to A0, and a 10K resistor is connected in series to 5V.

The NTC thermistor is used here, so when the temperature rises, the resistance of the thermistor decreases, the voltage division of A0 decreases, and the value obtained from A0 decreases, and vice versa increases.

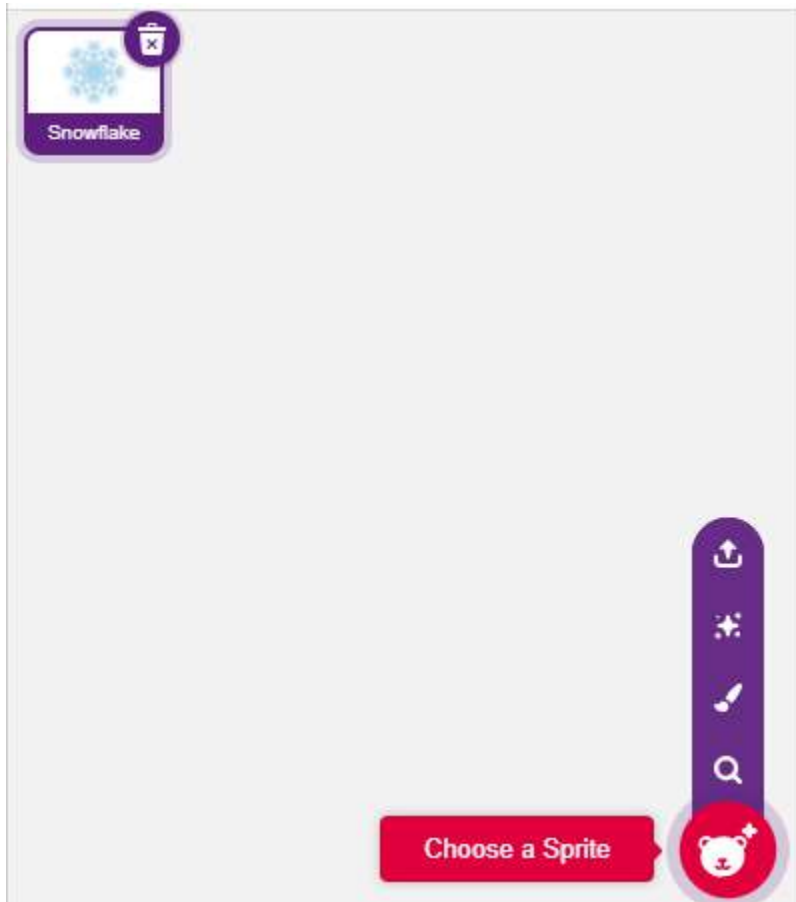


- *Breadboard*
- *Thermistor*
- *Resistor*

3.11.3 Programming

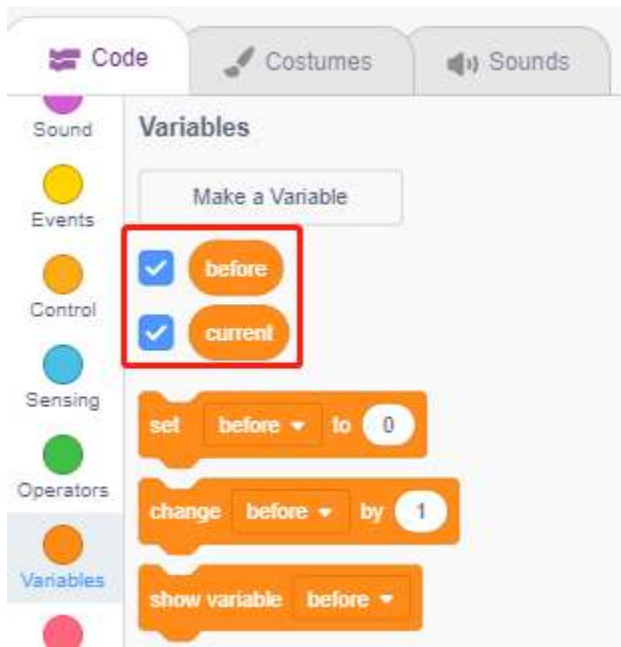
1. Select a sprite

Delete the default sprite, click the **Choose a Sprite** button in the lower right corner of the sprite area, enter **Snowflake** in the search box, and then click to add it.



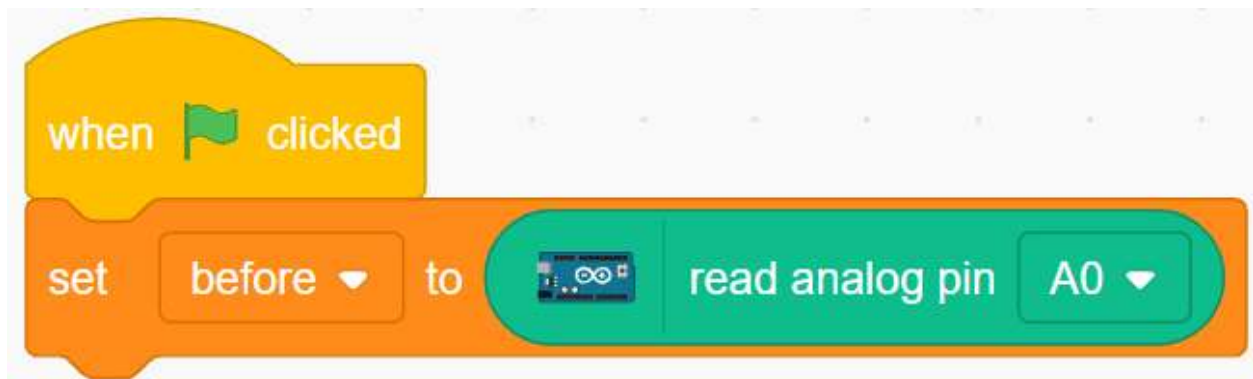
2. Create 2 variables

Create two variables, **before** and **current**, to store the value of A0 in different cases.



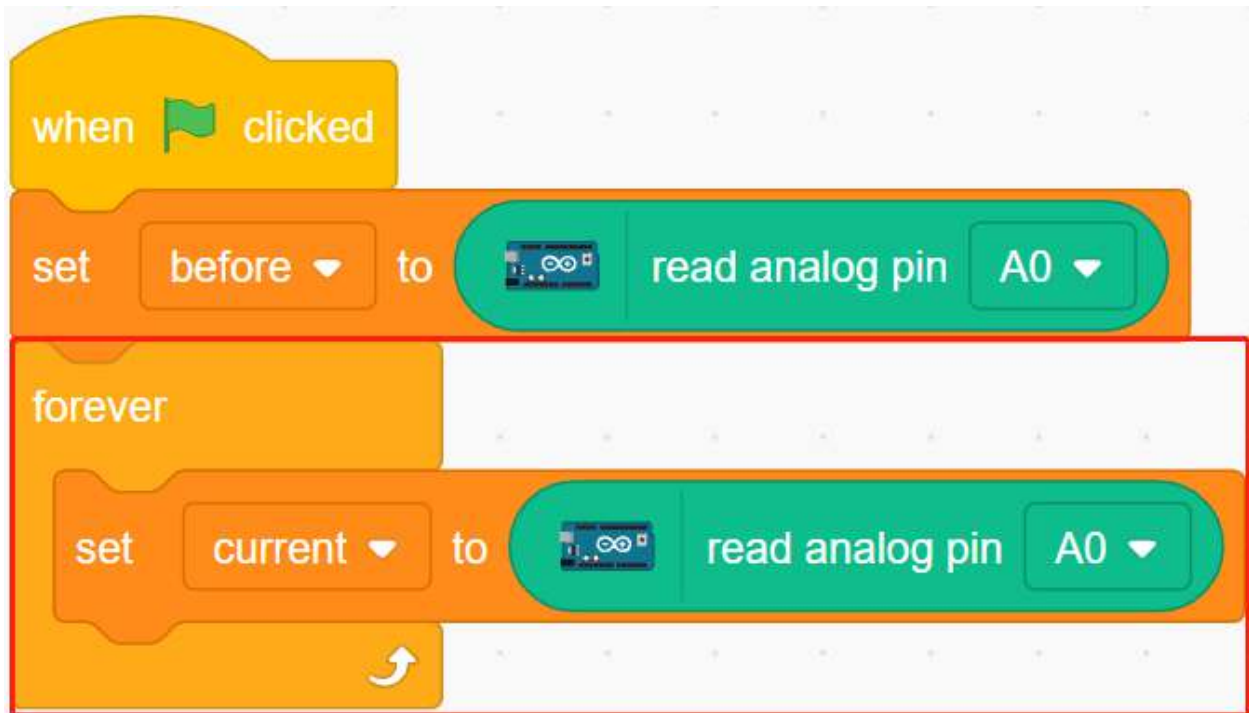
3. Read the value of A0

When the green flag is clicked, the value of A0 is read and stored in the variable **before**.



4. Read the value of A0 again

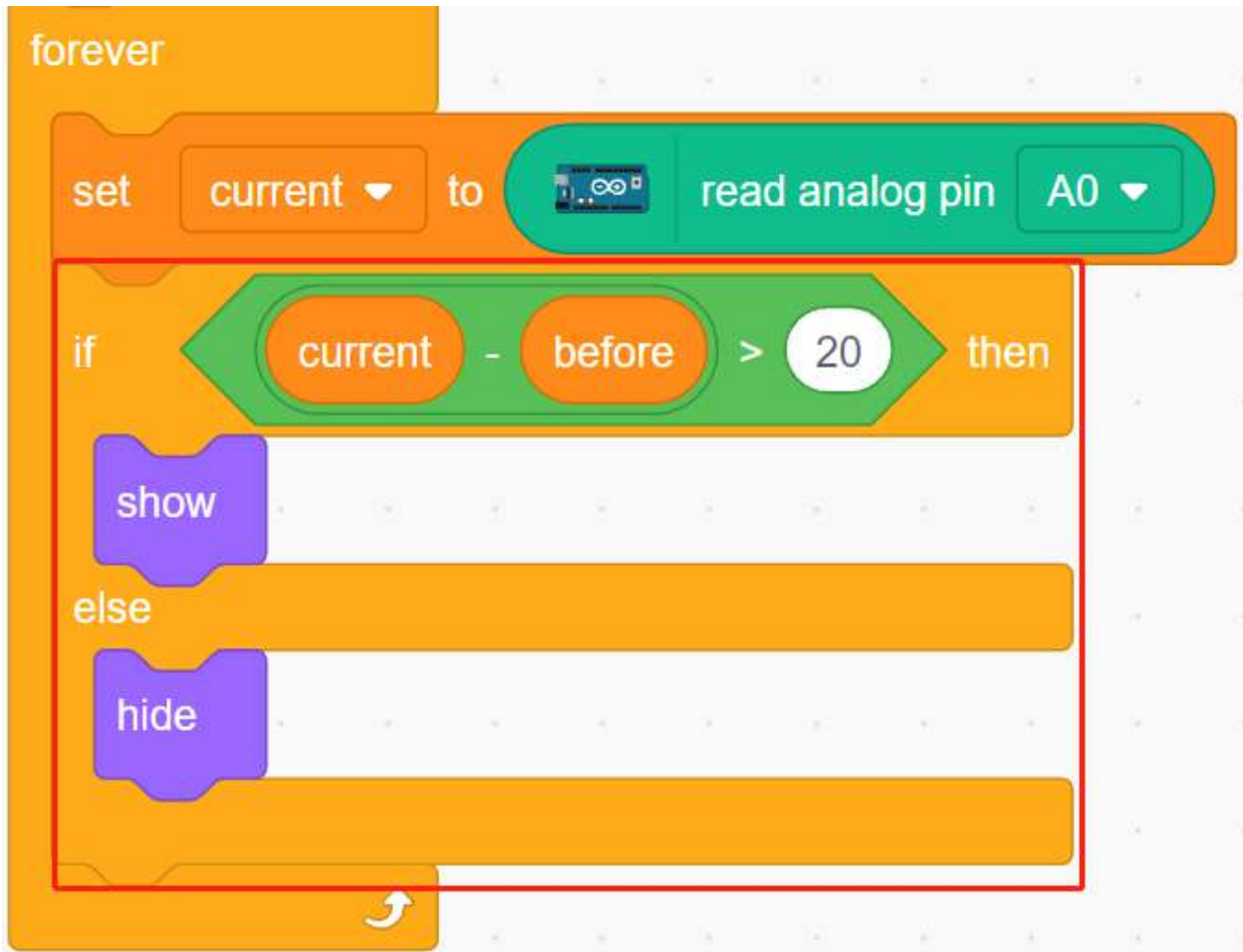
In [forever], read the value of A0 again and store it in the variable **current**.



5. Determining temperature changes

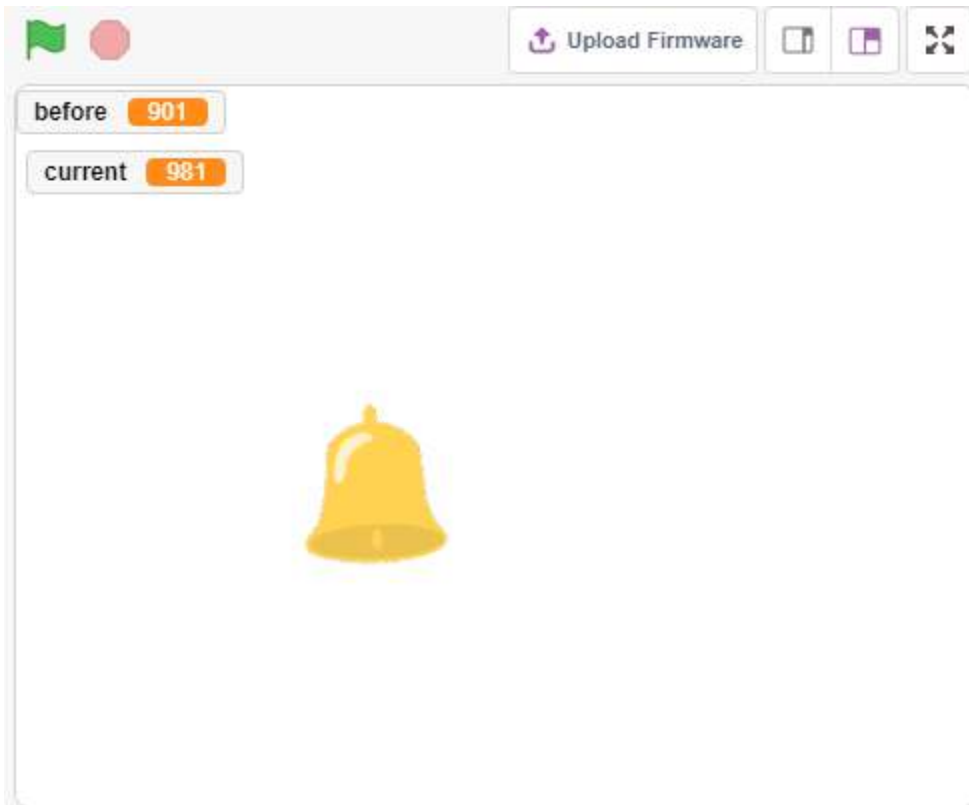
Using the [if else] block, determine if the current value of A0 is 50 greater than before, which represents a decrease in temperature. At this point let **Snowflake** sprite show, otherwise hide.

- [-] & [>]: subtraction and comparison operators from **Operators** palette.



3.12 2.9 Light Alarm Clock

In life, there are various kinds of time alarm clocks. Now let's make a light-controlled alarm clock. When morning comes, the brightness of light increases and this light-controlled alarm clock will remind you that it's time to get up.



3.12.1 You Will Learn

- Photoresistor working principle
- Stopping sound playback and stopping scripts from running

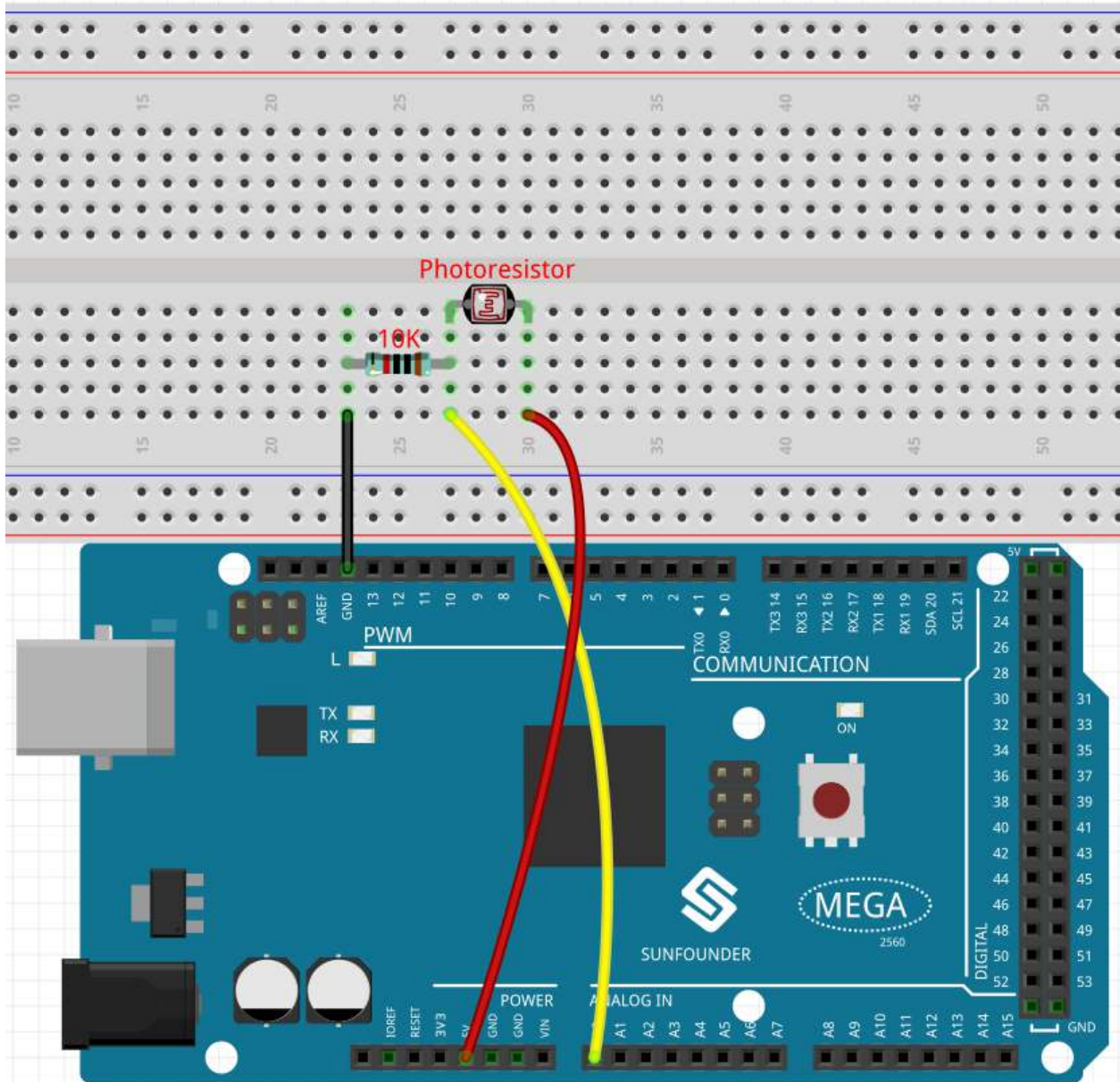
3.12.2 Build the Circuit

A photoresistor or photocell is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity.

Build the circuit according to the following diagram.

Connect one end of the photoresistor to 5V, the other end to A0, and connect a 10K resistor in series with GND at this end.

So when the light intensity increases, the resistance of a photoresistor decreases, the voltage division of the 10K resistor increases, and the value obtained by A0 becomes larger.

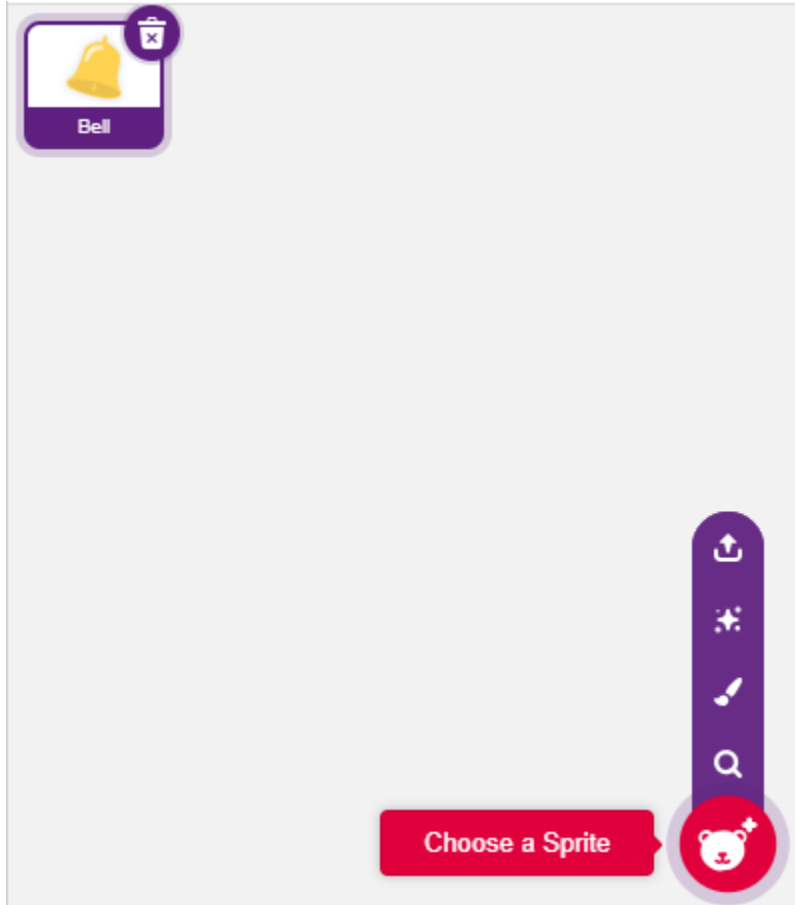


- Breadboard
- Photoresistor
- Resistor

3.12.3 Programming

1. Select a sprite

Delete the default sprite, click the **Choose a Sprite** button in the lower right corner of the sprite area, enter **bell** in the search box, and then click to add it.



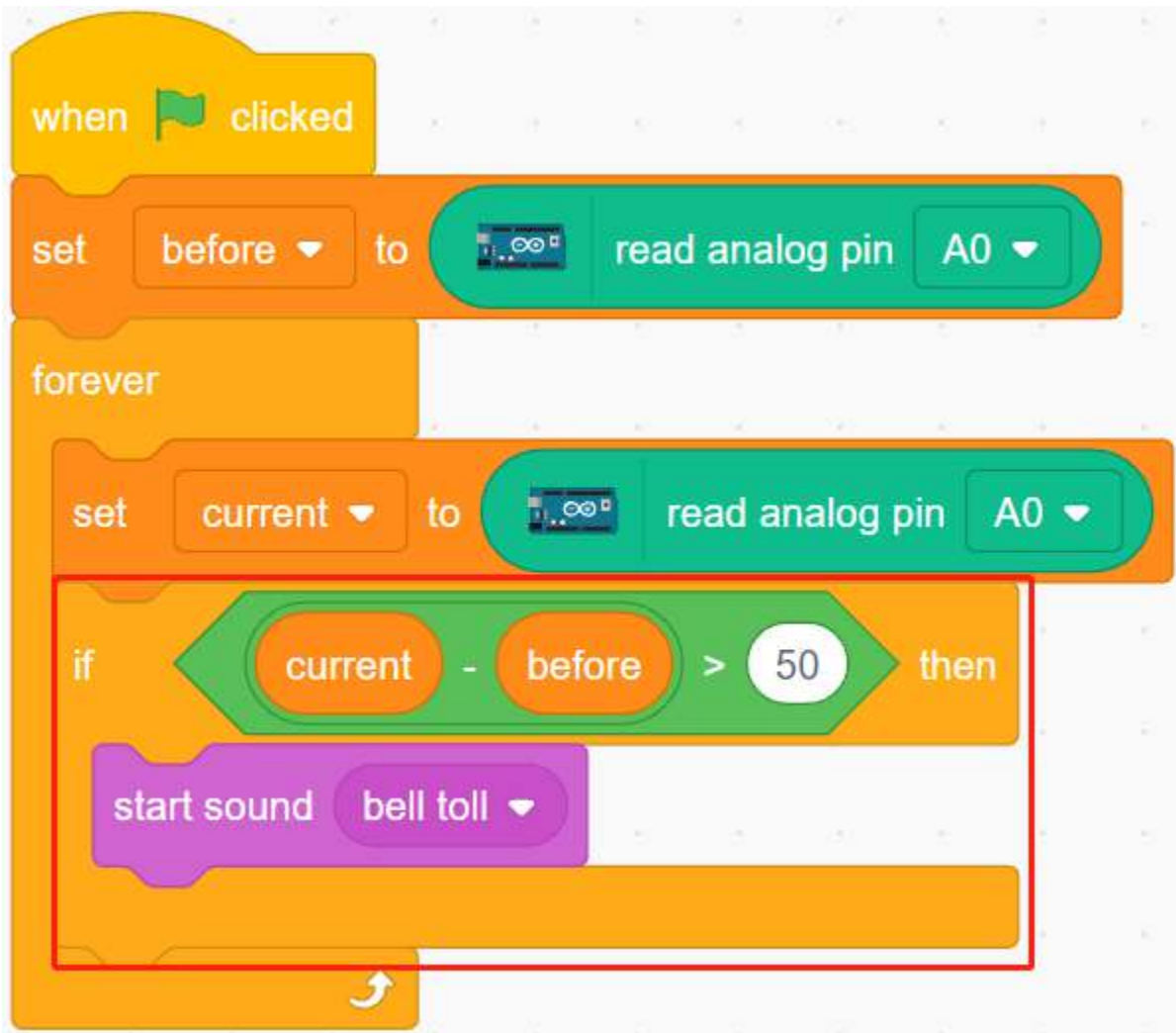
2. Read the value of A0

Create two variables **before** and **current**. When green flag is clicked, read the value of A0 and store it in variable **before** as a reference value. In [forever], read the value of A0 again, store it in the variable **current**.



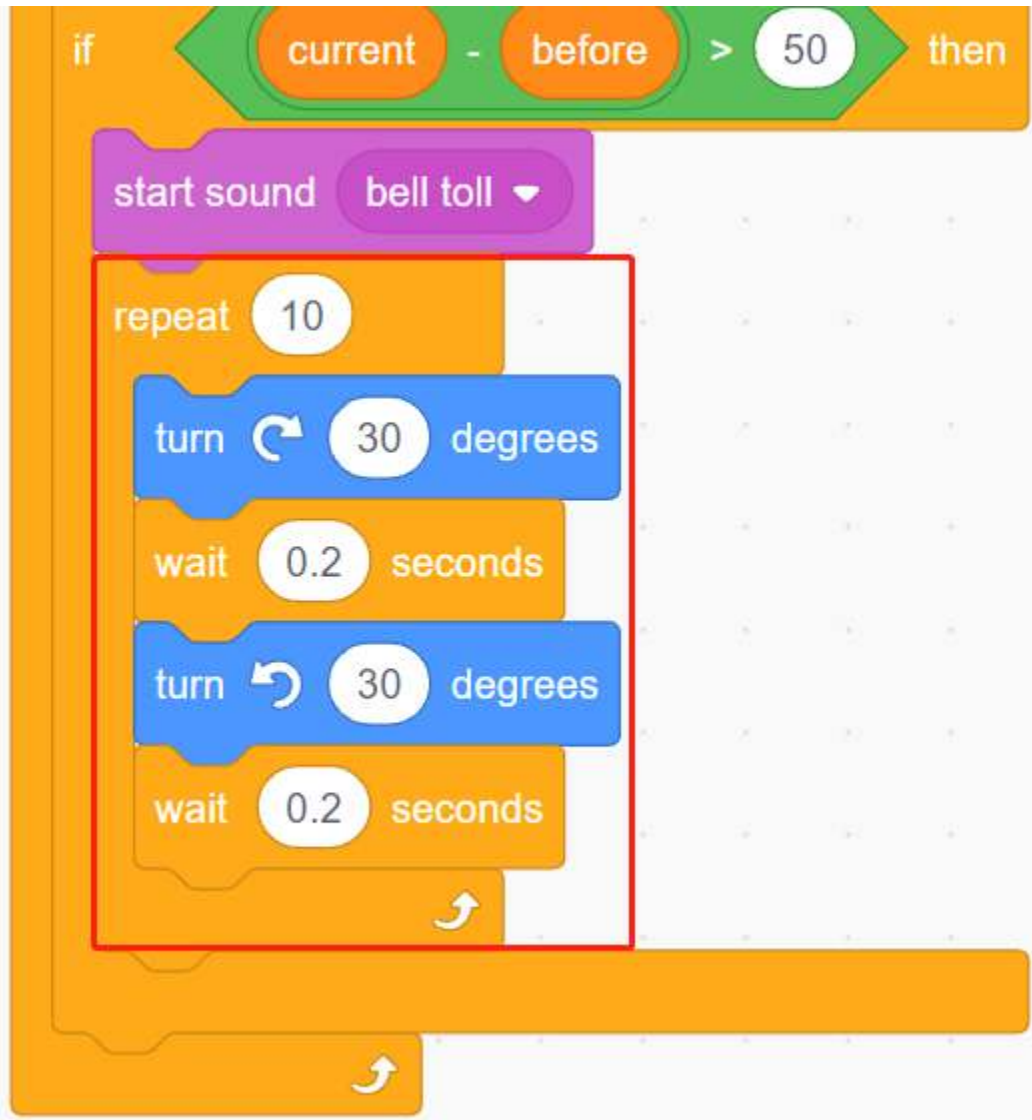
3. Make a sound

When the value of current A0 is greater than the previous 50, which represents the current light intensity is greater than the threshold, then let the sprite make a sound.



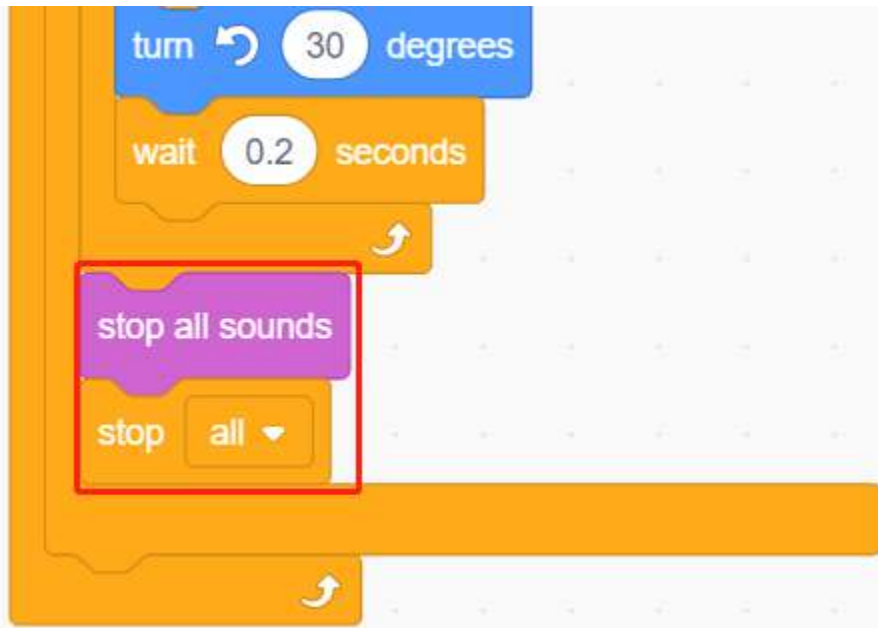
4. Turning the sprite

Use [turn block] to make the **bell** sprite turn left and right to achieve the alarm effect.



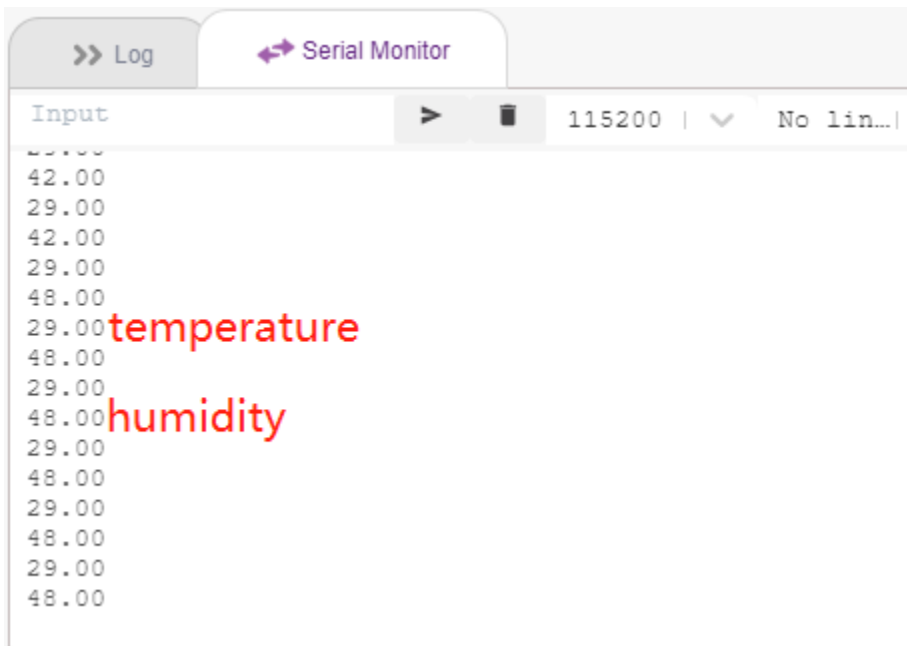
5. stop all

Stops the alarm when it has been ringing for a while.



3.13 2.10 Read Temperature and Humidity

Previous projects have been using stage mode, but some functions are only available in upload mode, such as serial communication function. In this project, we will print the temperature and humidity of the DHT11 using the Serial Monitor in *Upload Mode*.



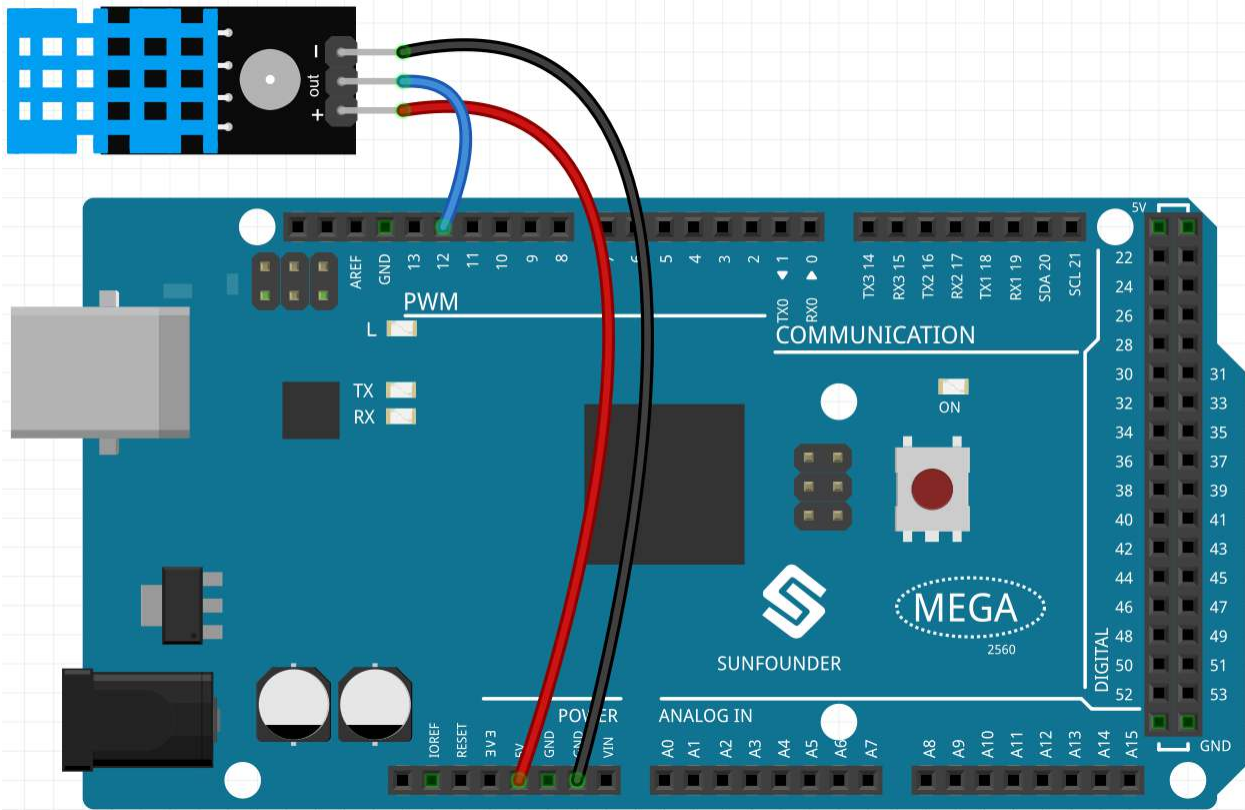
3.13.1 You Will Learn

- Get the temperature and humidity from the DHT11 module
- Serial Monitor for *Upload Mode*
- Add extension

3.13.2 Build the Circuit

The digital temperature and humidity sensor DHT11 is a composite sensor that contains a calibrated digital signal output of temperature and humidity.

Now build the circuit according to the following diagram.

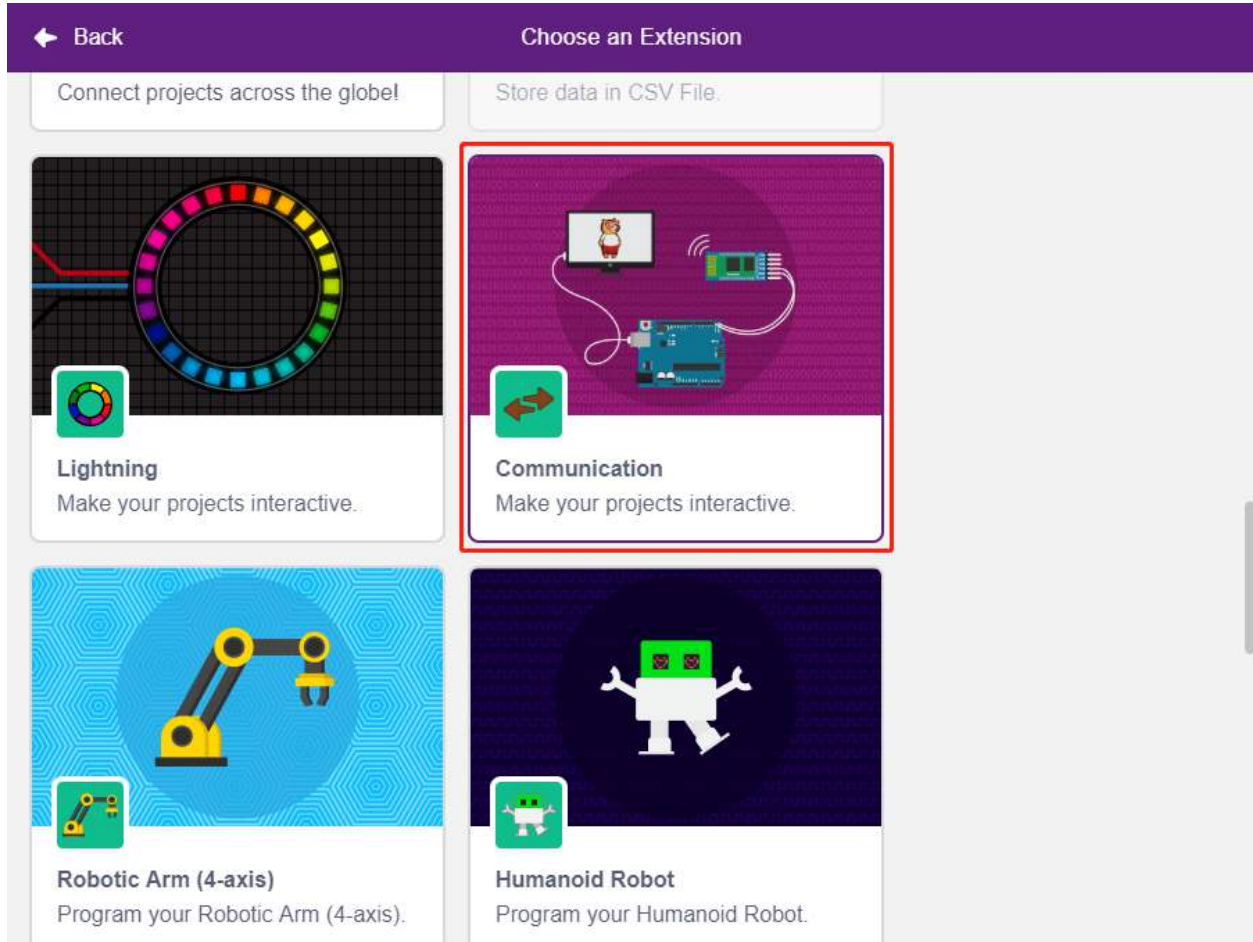


- *Breadboard*
- *Humiture Sensor Module*

3.13.3 Programming

1. Adding Extensions

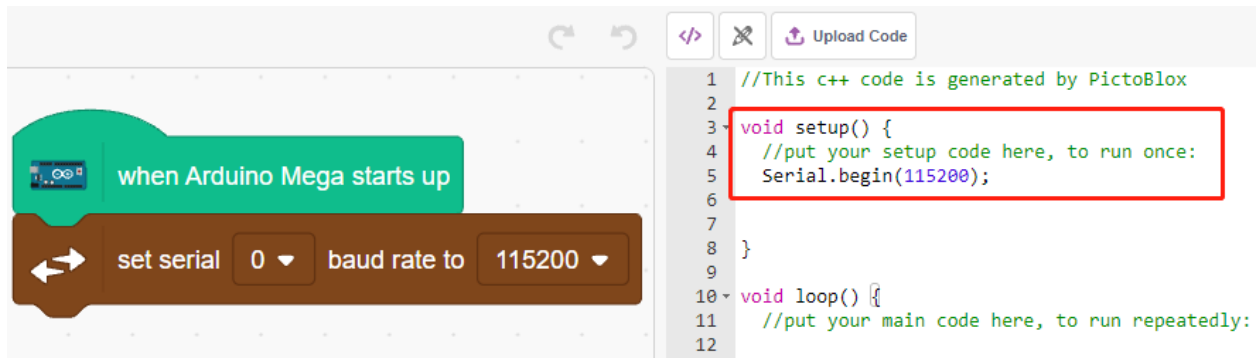
Switch to **Upload** mode, click the **Add Extension** button in the bottom left corner, then select **Communication** to add it, and it will appear at the end of the palette area.



2. Initializing the Arduino Mega and Serial Monitor

In **Upload** mode, start Arduino Mega and then set the serial port baud rate.

- [when Arduino Starts up]: In **Upload** mode, start Arduino Mega.
- [set serial baud rate to]: From the **Communications** palette, used to set the baud rate of serial port 0, default is 115200. If you are using Mega2560, then you can choose to set baud rate in serial port 0~3.



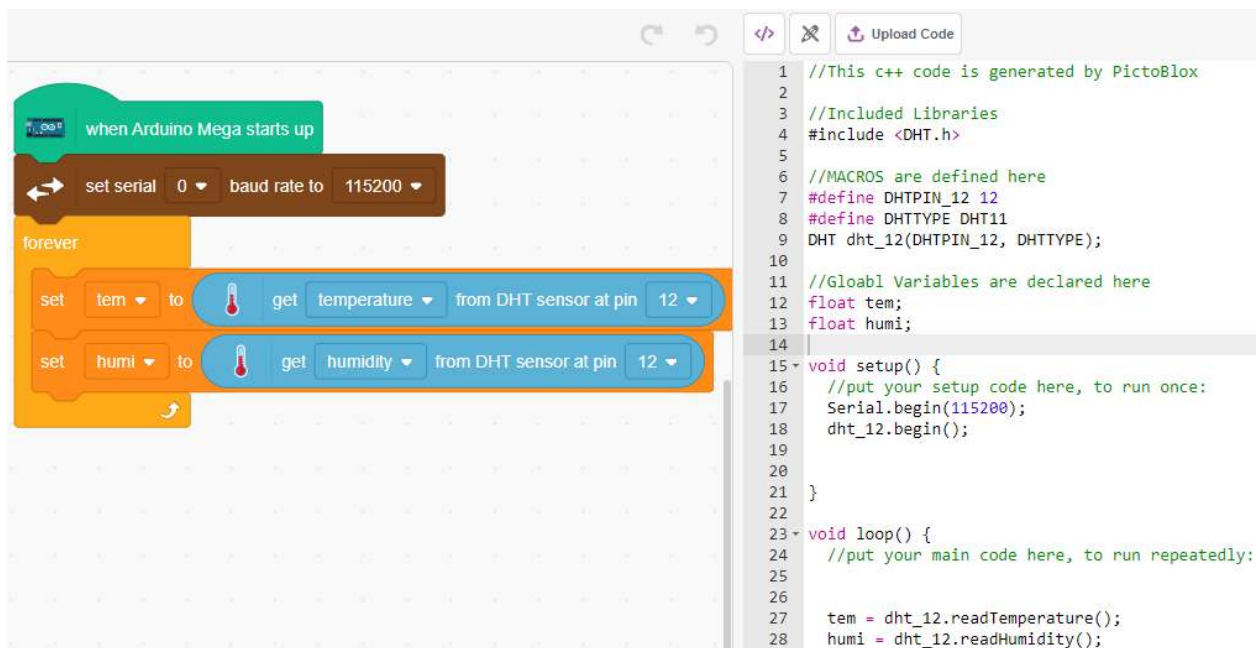
```

1 //This c++ code is generated by PictoBlox
2
3 void setup() {
4   //put your setup code here, to run once:
5   Serial.begin(115200);
6
7 }
8
9
10 void loop() {
11   //put your main code here, to run repeatedly:
12

```

3. Read temperature and humidity

Create 2 variables **tem** and **humi** to store the temperature and humidity respectively, the code will appear on the right side while you drag and drop the block.



```

1 //This c++ code is generated by PictoBlox
2
3 //Included Libraries
4 #include <DHT.h>
5
6 //MACROS are defined here
7 #define DHTPIN_12 12
8 #define DHTTYPE DHT11
9 DHT dht_12(DHTPIN_12, DHTTYPE);
10
11 //Global Variables are declared here
12 float tem;
13 float humi;
14
15 void setup() {
16   //put your setup code here, to run once:
17   Serial.begin(115200);
18   dht_12.begin();
19
20 }
21
22
23 void loop() {
24   //put your main code here, to run repeatedly:
25
26
27   tem = dht_12.readTemperature();
28   humi = dht_12.readHumidity();

```

4. Print them on the Serial Monitor

Write the read temperature and humidity to the Serial Monitor. To avoid transferring too fast and causing PictoBlox to jam, use the [wait seconds] block, to add some time interval for the next print.



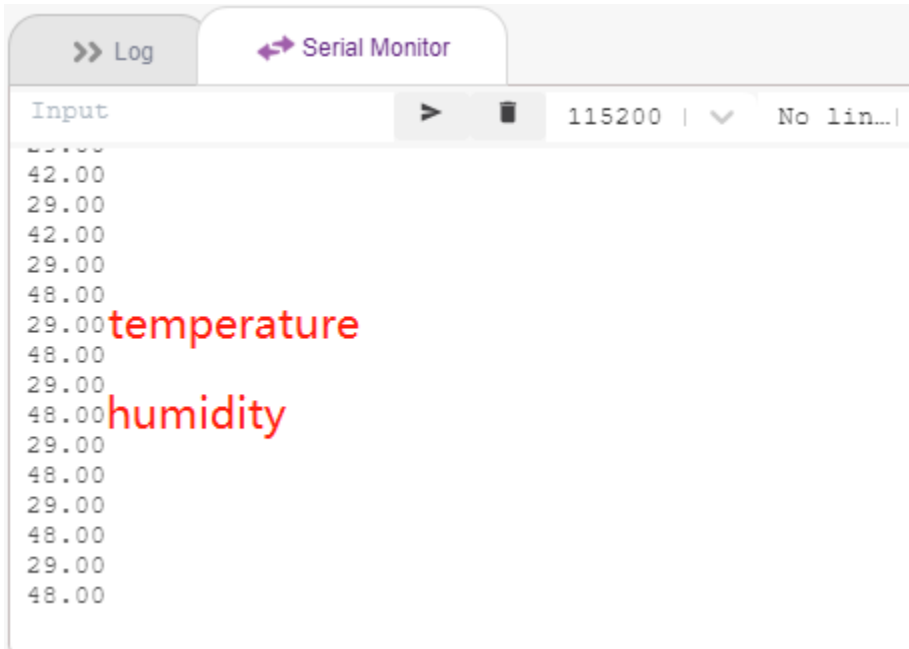
5. Uploading code

Unlike the **Stage** mode, the code in **Upload** mode needs to be uploaded to the Arduino board using the **Upload Code** button to see the effect. This also allows you to unplug the USB cable and still have the program running.



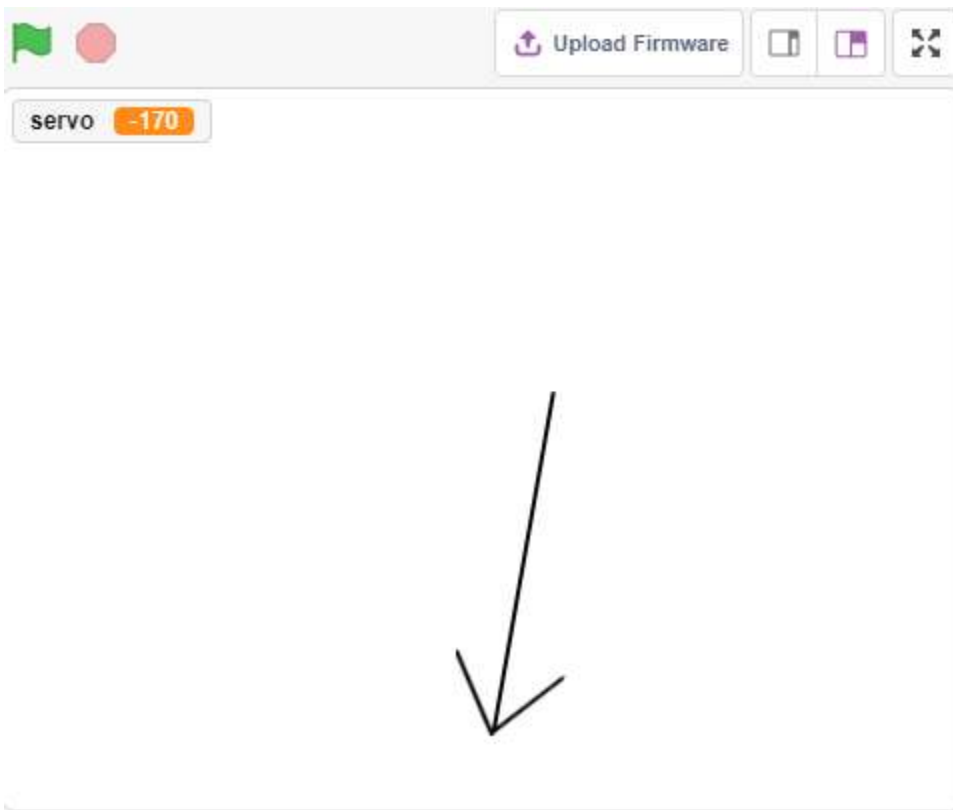
6. Turn on the serial monitor

Now open the **Serial Monitor** to see the temperature and humidity.



3.14 2.11 Pendulum

In this project, we will make an arrow pendulum while the servo will follow the rotation.



3.14.1 You Will Learn

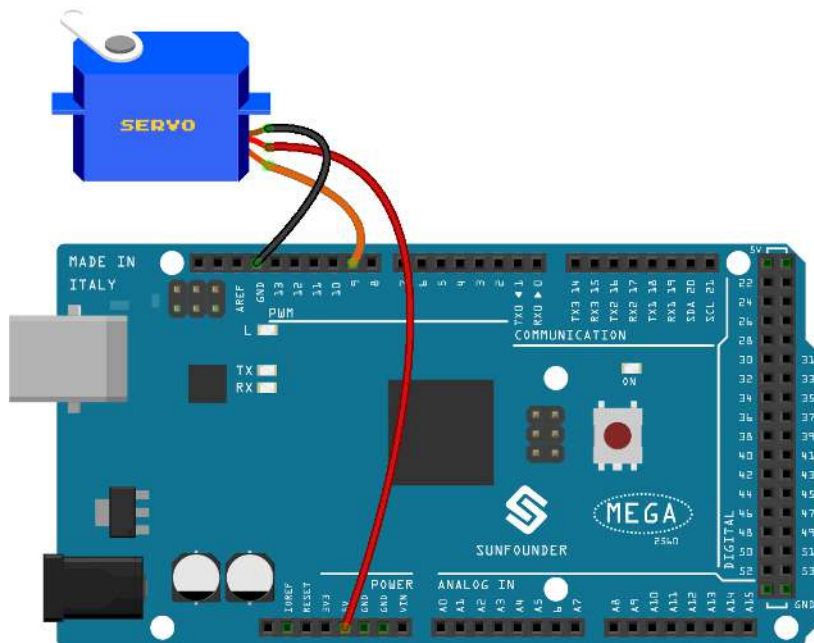
- How the servo works and the angle range
- Draw a sprite and put the center point on the tail.

3.14.2 Build the Circuit

A servo is a geared motor that can only rotate 180 degrees. It is controlled by sending electrical pulses from your circuit board. These pulses tell the servo what position it should move to.

The servo has three wires: the brown wire is GND, the red one is VCC (connect to 3.3V), and the orange one is the signal wire. The angle range is 0-180.

Now build the circuit according to the diagram below.

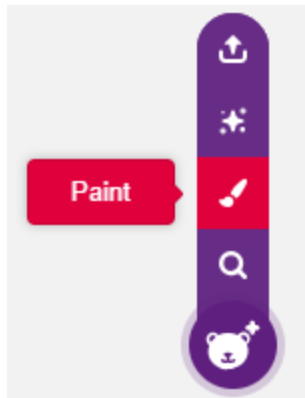


- *Breadboard*
- *Servo*

3.14.3 Programming

1. Paint a sprite

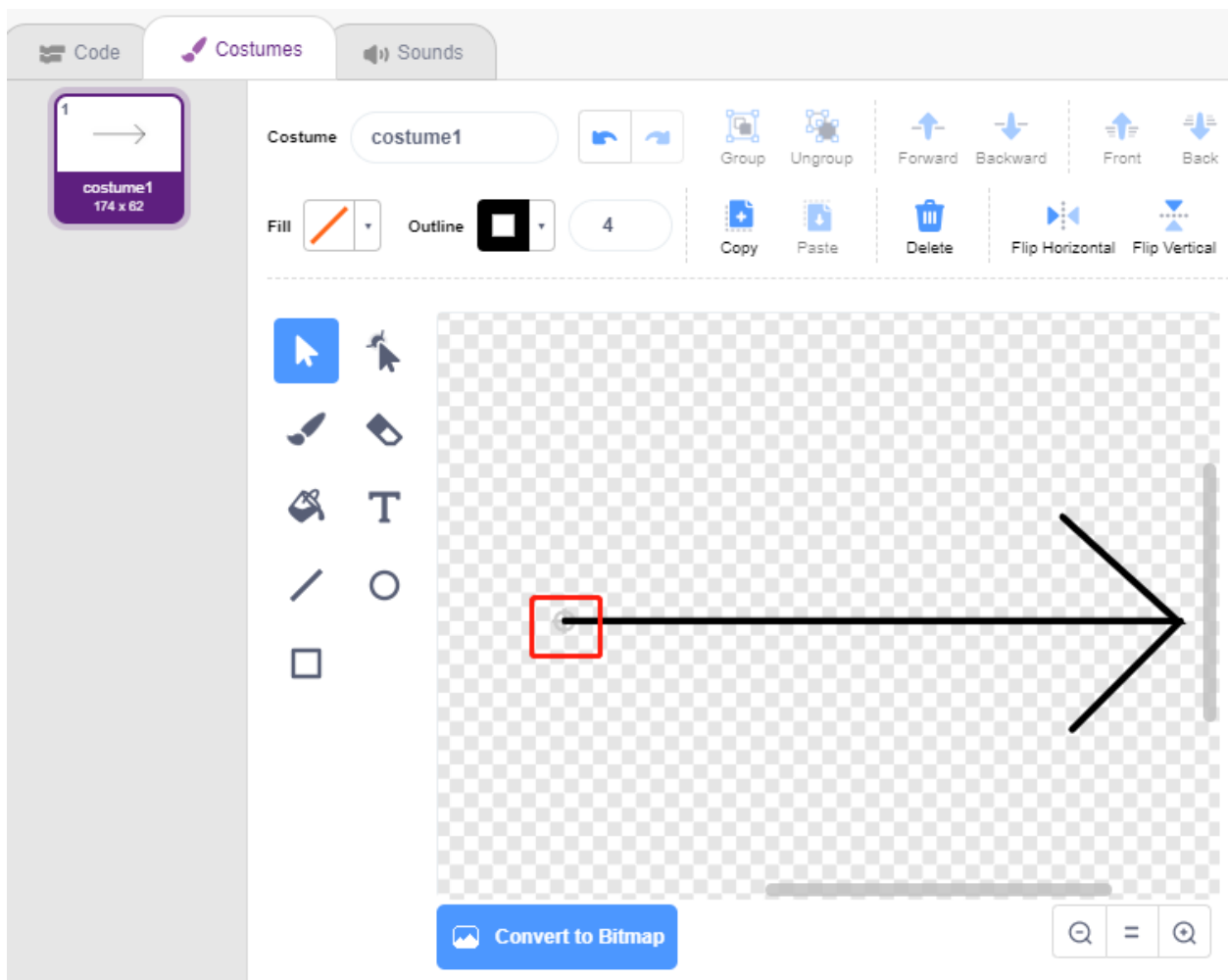
Delete the default sprite, select the Sprite button and click **Paint**, a blank sprite **Sprite1** will appear.



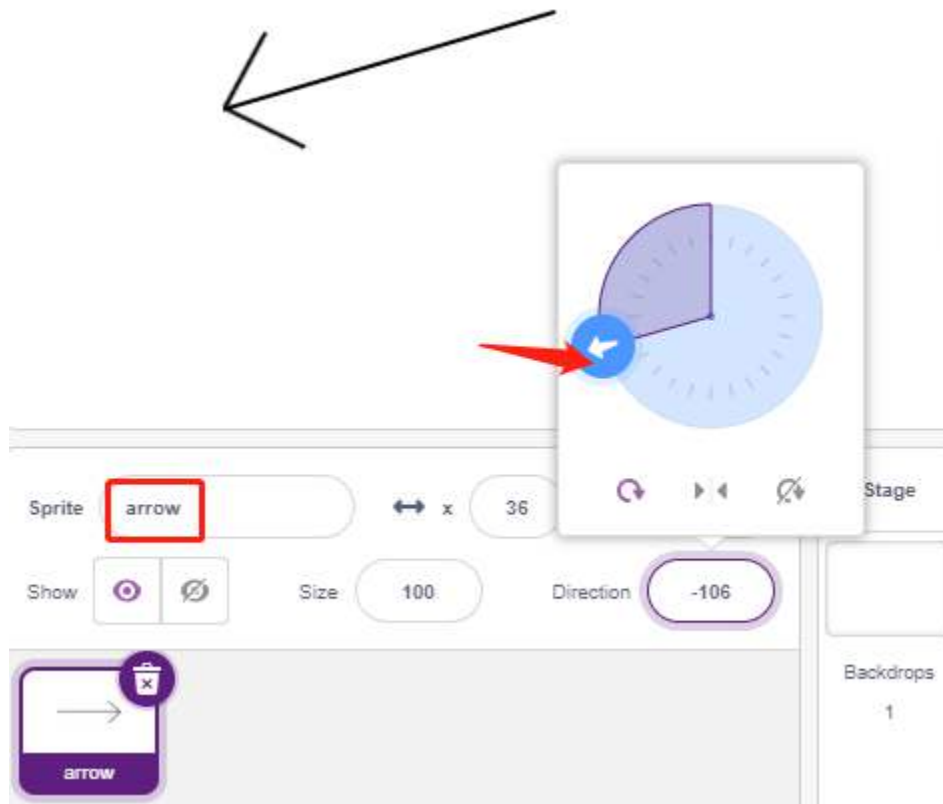
On the open **Costumes** page, use the **Line tool** to draw an arrow.

Note:

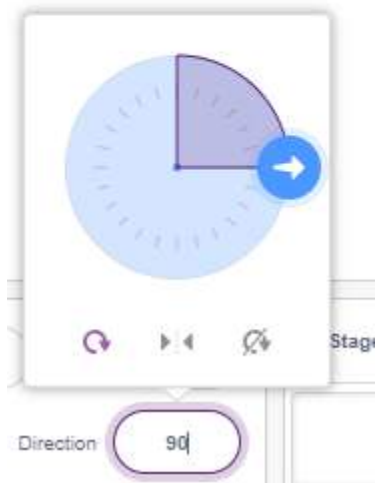
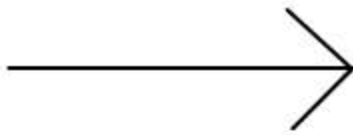
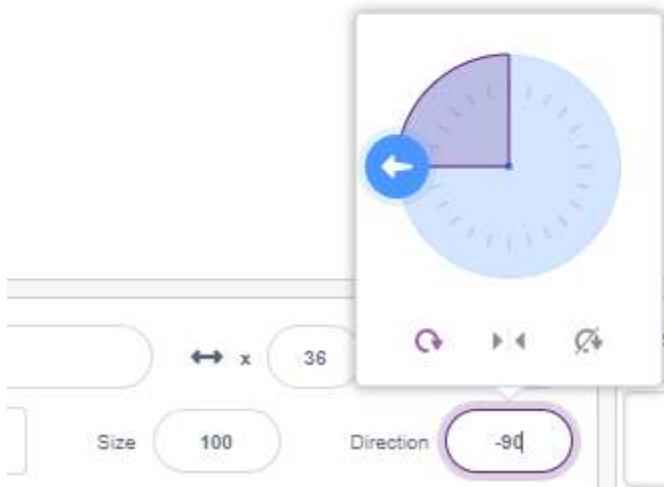
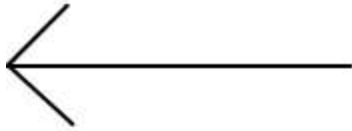
- Be sure to start drawing the arrow from the center of the canvas outward so that the arrow is turning in a circle with the center point as the origin.
- Hold Shift to make the line angle straight or 45 degrees.



After drawing, the **arrow** sprite will be displayed on the stage, name it **arrow**. Then click on the number after **Direction**, a circular dial will appear, now drag this arrow and see if the **arrow** sprite on the stage turns with the tail as the origin.

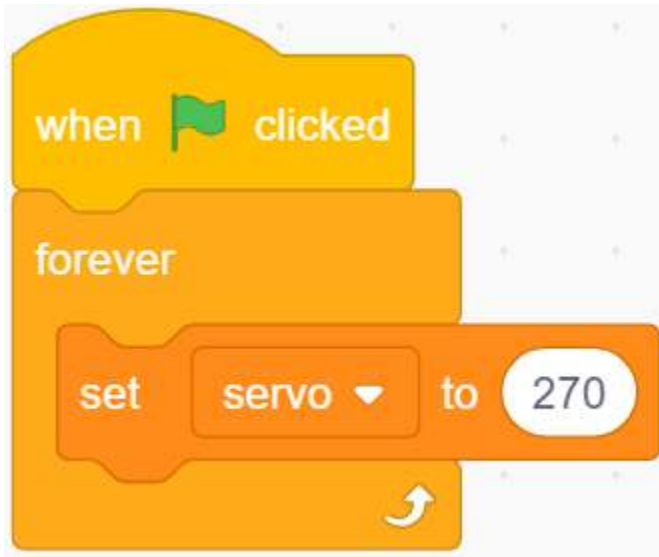


To make the **arrow** sprite swing from the left to the right, the angle range is -90 to -180, 180 to 90.



2. Creating a variable.

Create a variable called **servo**, which stores the angle value and sets the initial value to 270.



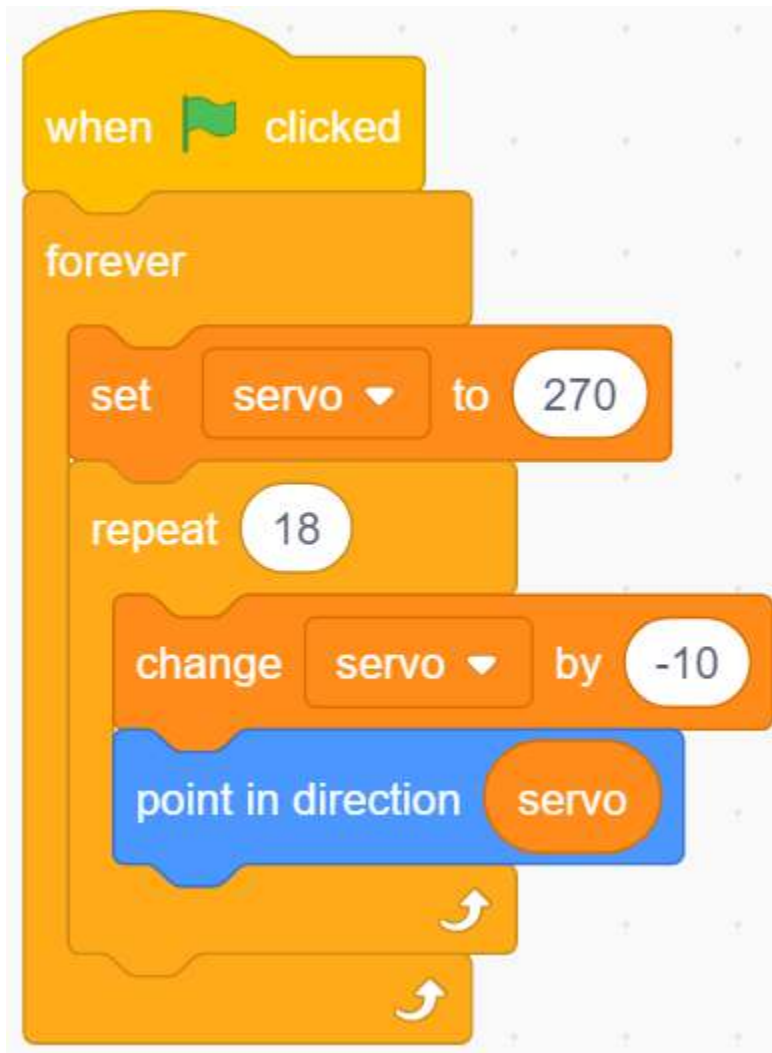
3. Swing from the left to the right

Now let the **arrow** sprite swing from the left -90 degree position to the right 90 degree position.

With [repeat] block, add -10 to the variable each time, and you'll get to 90 degrees in 18 passes. Then use [point in block] to make the arrow sprite turn to these angles.

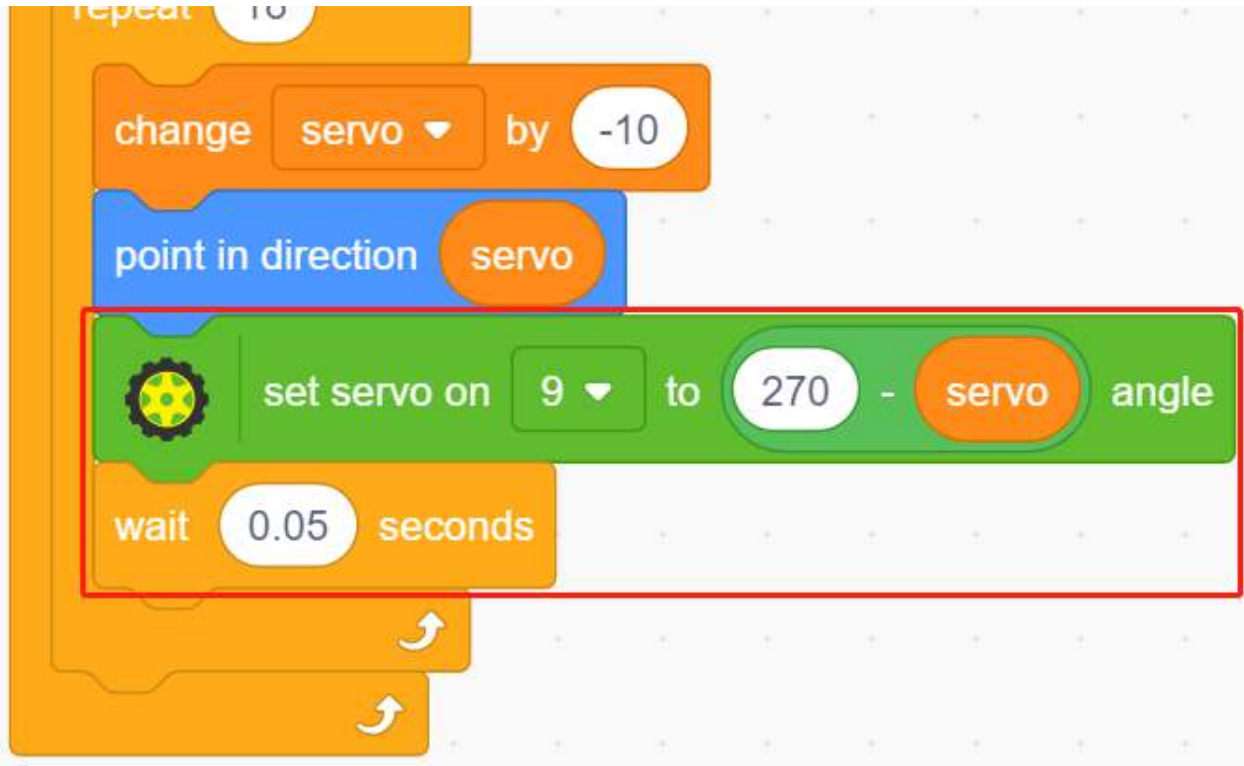
Since the sprite rotation angle is -180 ~ 180, angles outside this range are converted by the following conditions.

- If angle > 180, then angle -360.



4. Turning the Servo

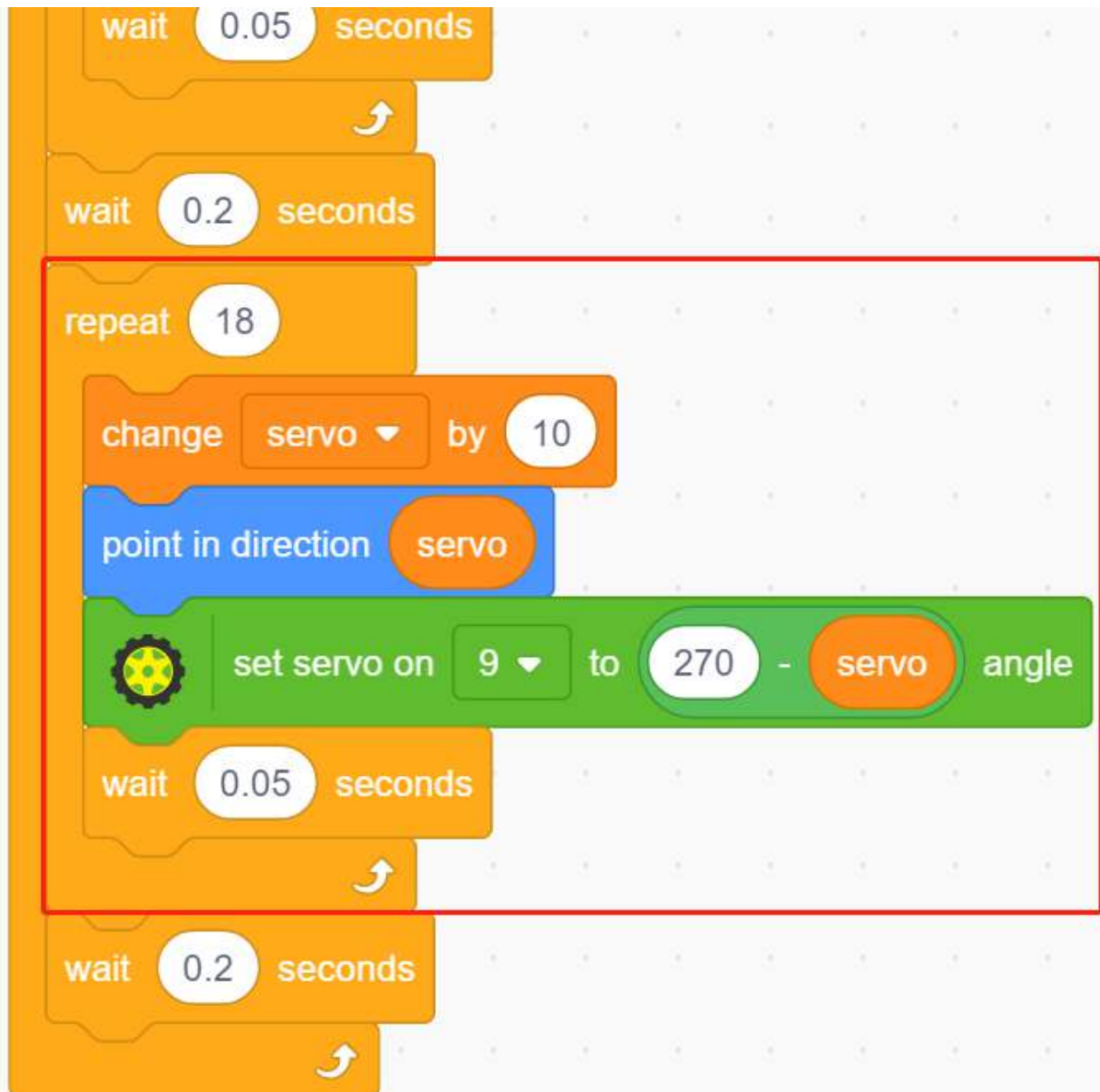
When you click on the green flag, you will see the arrow quickly turn to the right and then back to the left, so use a [wait seconds] block here to make the rotation slower. Also use the [set servo on to angle] block to make the servo connected to the Arduino board turn to a specific angle.



5. Swinging from right to left

By the same method, make the servo and **arrow** sprite slowly rotate from the right to the left.

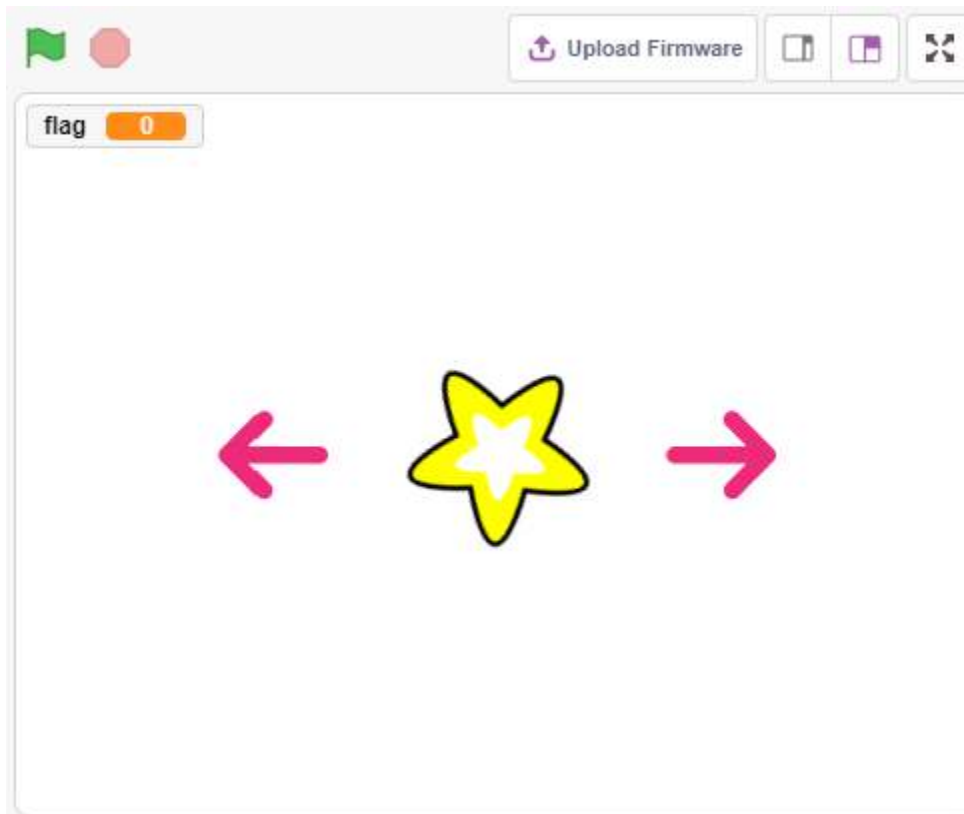
- If angle > 180, then angle -360.



3.15 2.12 Rotating Fan

In this project, we will make a spinning star sprite and fan.

Clicking on the left and right arrow sprites on the stage will control the clockwise and counterclockwise rotation of the motor and star sprite, click on the star sprite to stop the rotation.



3.15.1 You Will Learn

- Motor working principle
- Broadcast function
- Stop other script in sprite block

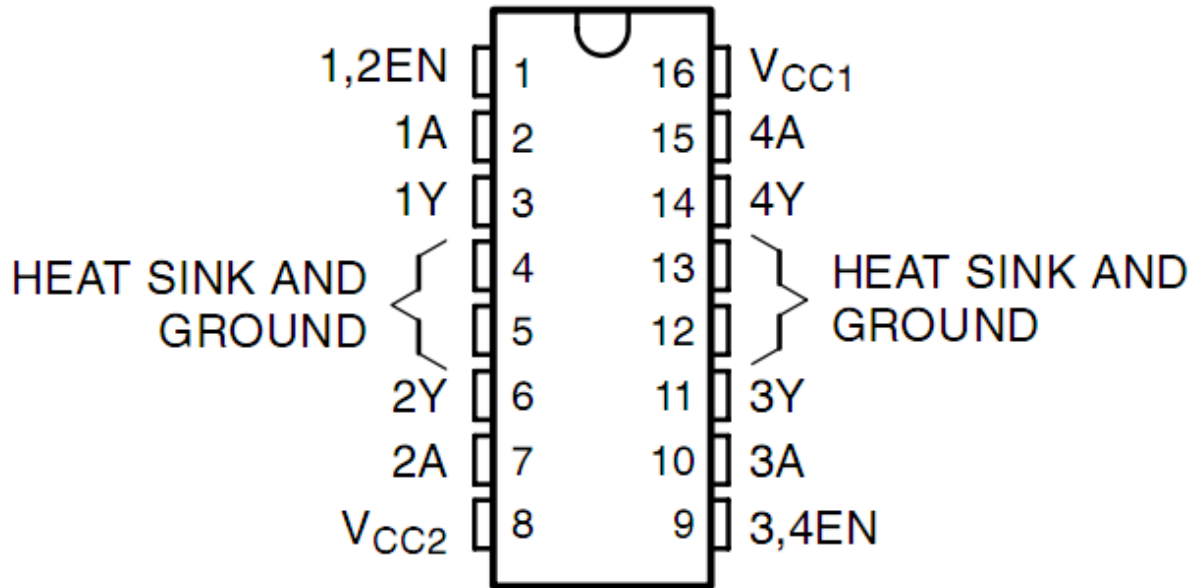
3.15.2 Build the Circuit

In this project, the motor driver chip *L293D* is used to make the motor rotate.

L293D is a 4-channel motor driver integrated by chip with high voltage and high current.

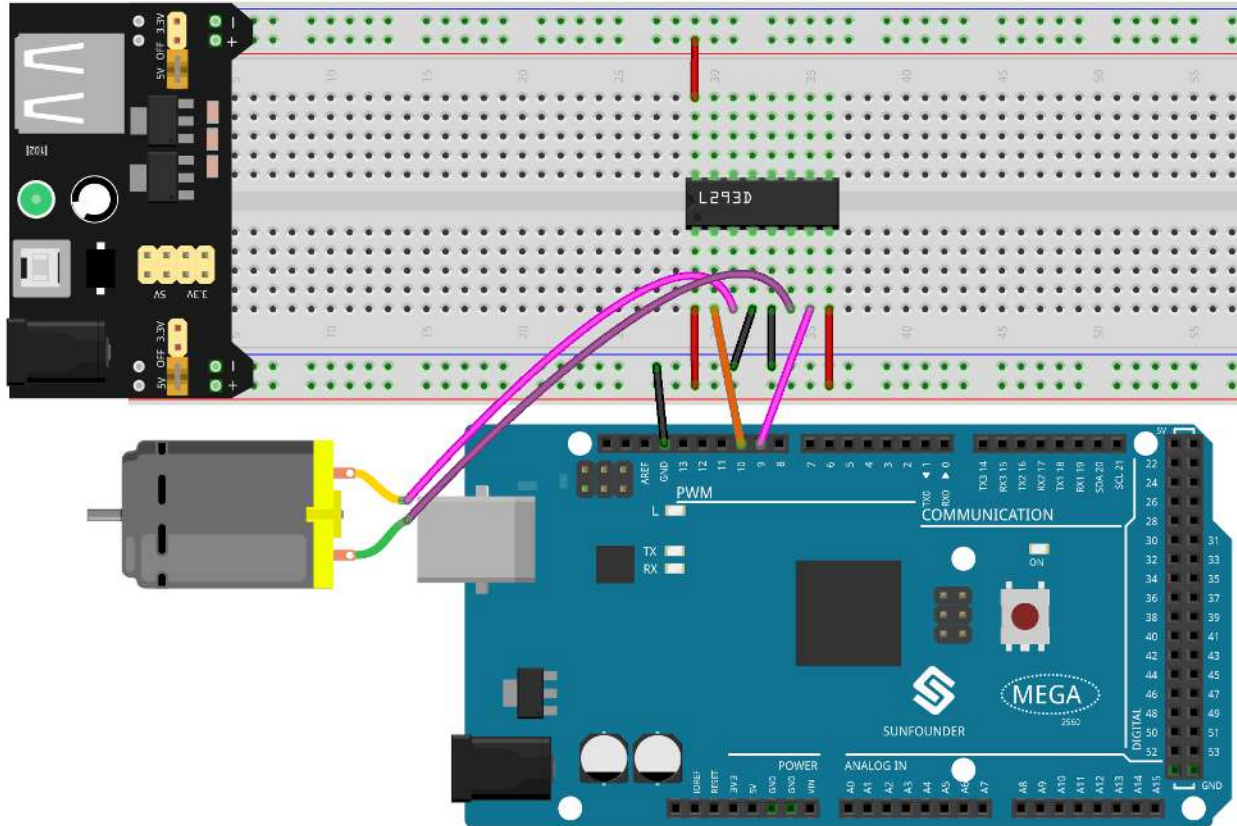
Its pinouts are shown below:

Pin **EN** is an enable pin and only works with high level; **A** stands for input and **Y** for output. When pin **EN** is High level, if **A** is High, **Y** outputs high level; if **A** is Low, **Y** outputs Low level. When pin **EN** is Low level, the *L293D* does not work.



Now build the circuit according to the following diagram.

- The Enable pin 1,2EN of the L293D are connected to 3.3V already, so L293D is always in the working state.
- Connect pin 1A and 2A to pin 9 and 10 of the control board respectively.
- The two pins of the motor are connected to pin 1Y and 2Y respectively.
- When pin 10 is set as High level and pin 9 as Low, the motor will start to rotate towards one direction.
- When the pin 10 is Low and pin 9 is High, it rotates in the opposite direction.



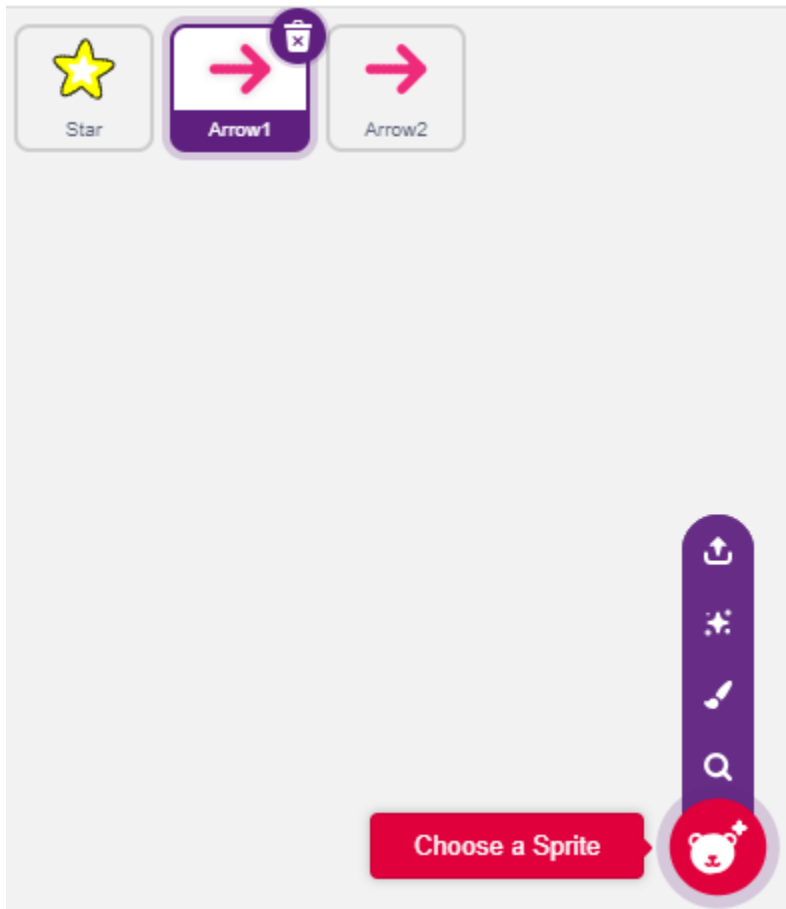
- Breadboard
- DC Motor
- L293D

3.15.3 Programming

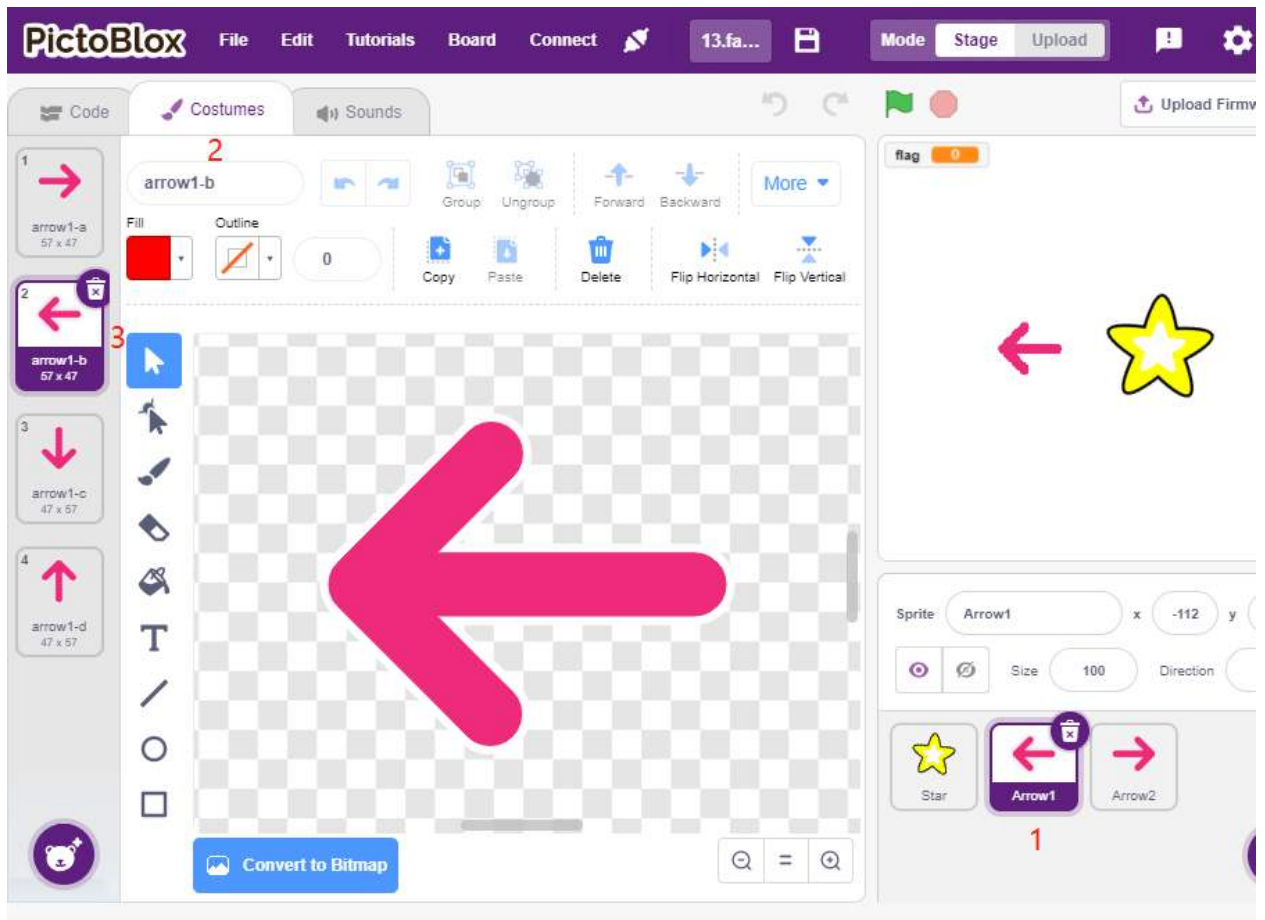
The effect we want to achieve is to use 2 arrow sprites to control the clockwise and counterclockwise rotation of the motor and the star sprite respectively, clicking on the star sprite will stop the motor from rotating.

1. Add sprites

Delete the default sprite, then select the **Star** sprite and the **Arrow1** sprite, and copy **Arrow1** once.



In the **Costumes** option, change the **Arrow1** sprite to a different direction costume.



Adjust the size and position of the sprite appropriately.

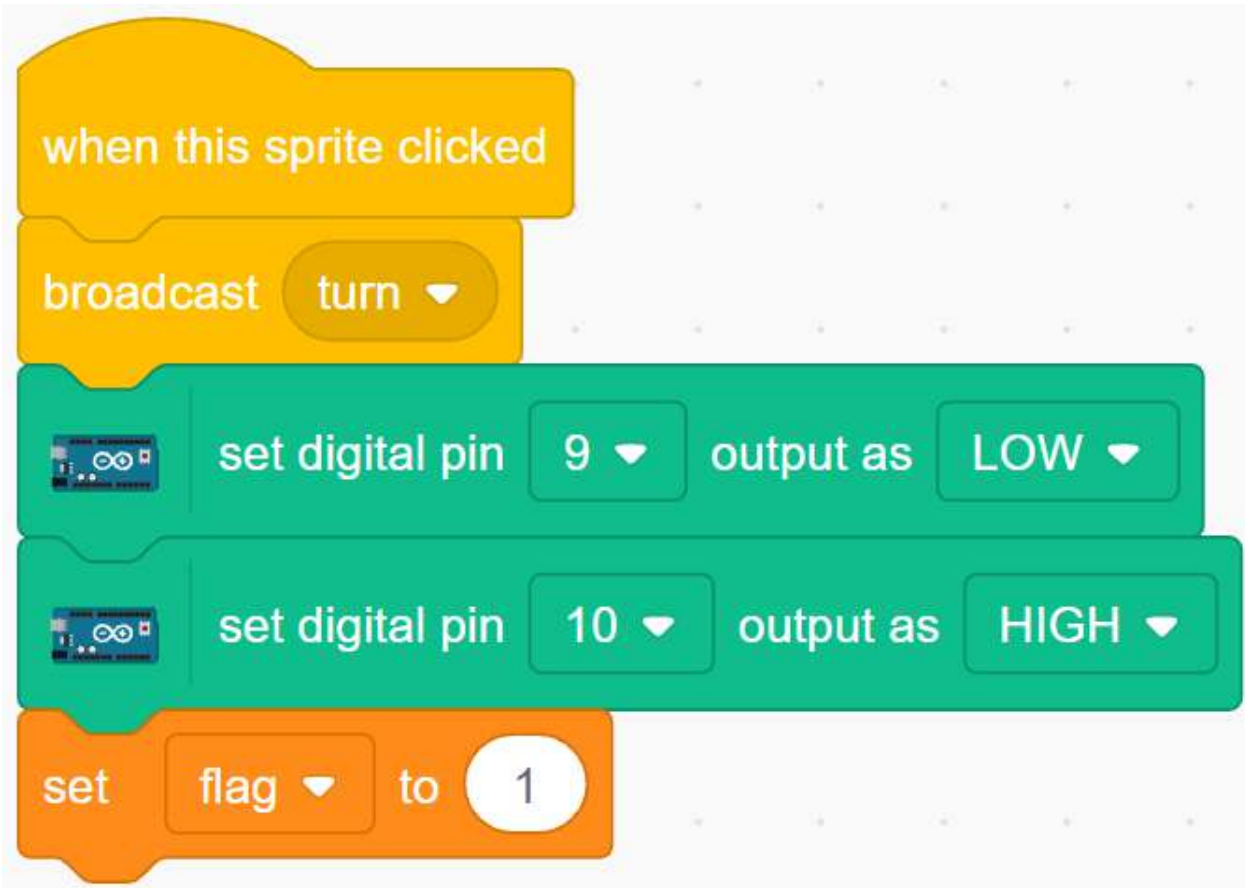


2. Left arrow sprite

When this sprite is clicked, it broadcasts a message - turn, then sets digital pin 9 to low and pin 10 to high, and sets the variable **flag** to 1. If you click the left arrow sprite, you will find that the motor turns counterclockwise, if your turn is clockwise, then you swap the positions of pin 9 and pin 10.

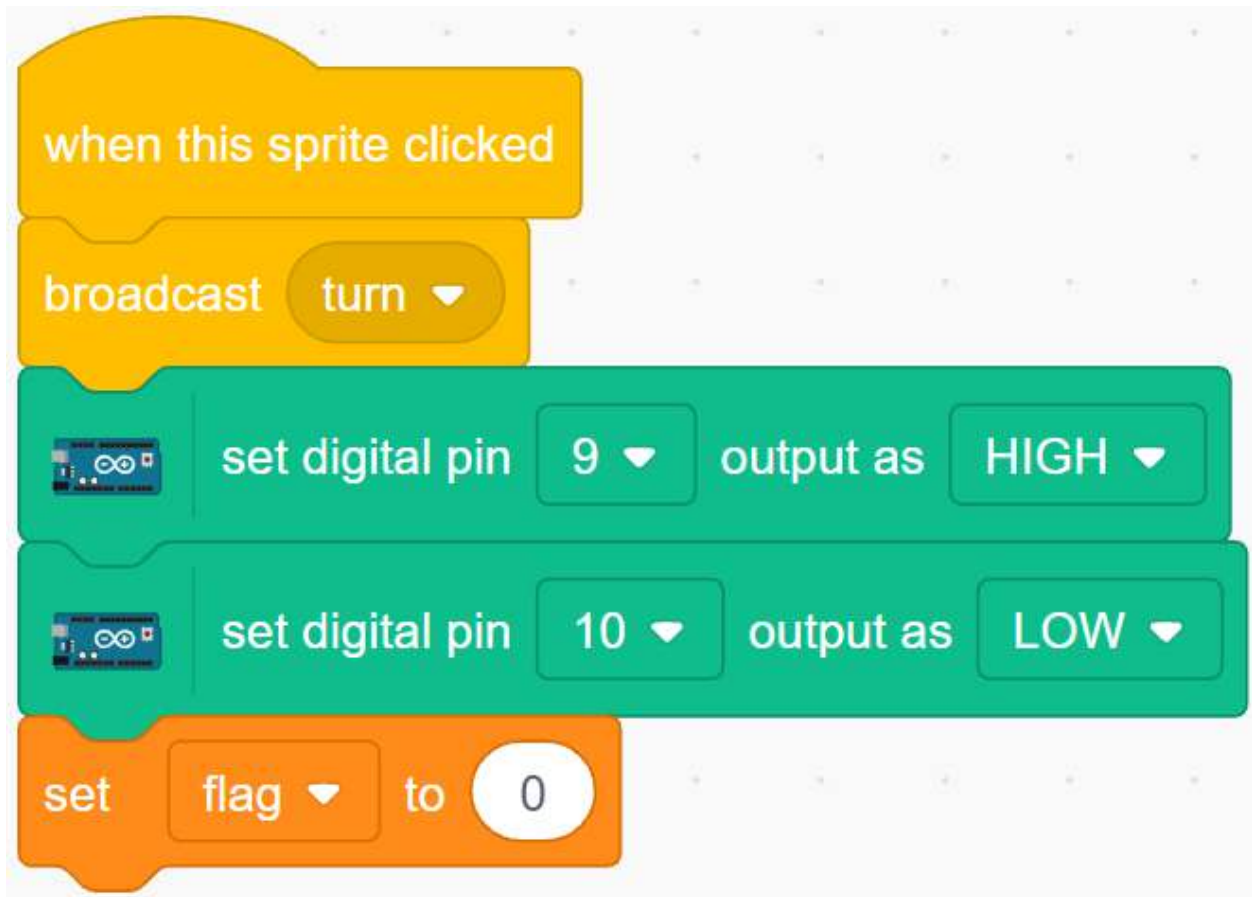
There are 2 points to note here.

- [broadcast]: from the **Events** palette, used to broadcast a message to the other sprites, when the other sprites receive this message, it will perform a specific event. For example, here is **turn**, when the **star** sprite receives this message, it executes the rotation script.
- variable flag: The direction of rotation of the star sprite is determined by the value of flag. So when you create the **flag** variable, you need to make it apply to all sprites.



3. right-arrow sprite

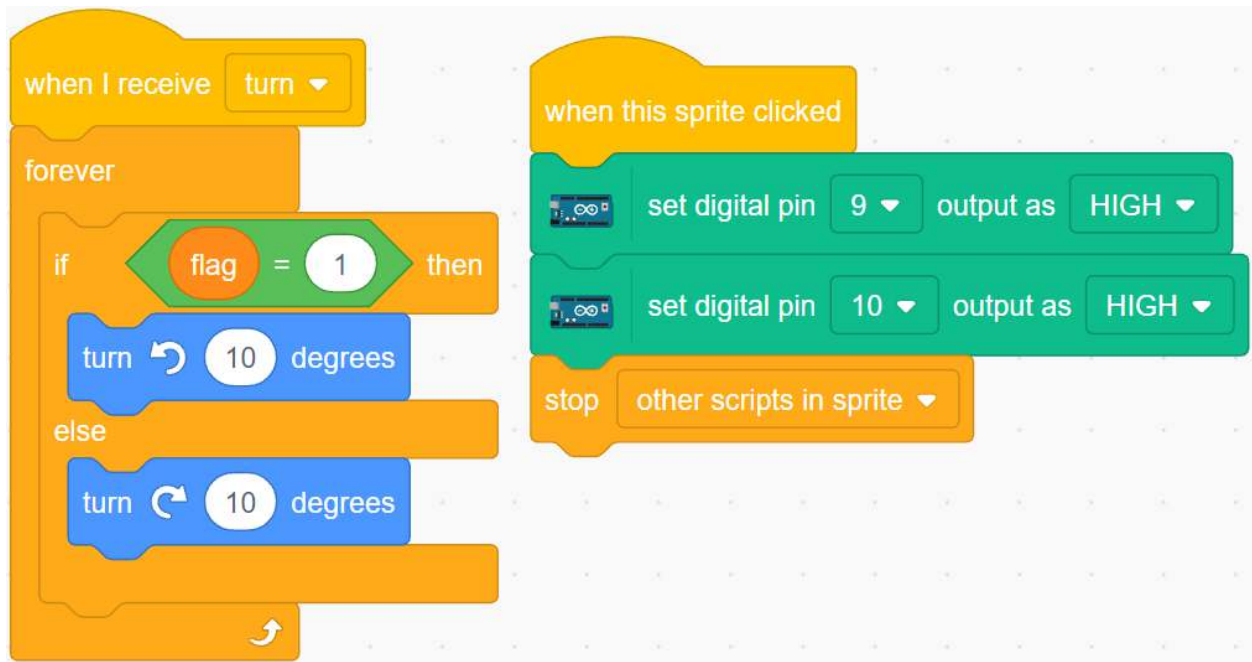
When this sprite is clicked, broadcast a message turn, then set digital pin 9 high and pin 10 low to make the motor turn clockwise and set the **flag** variable to 0.



4. star sprite

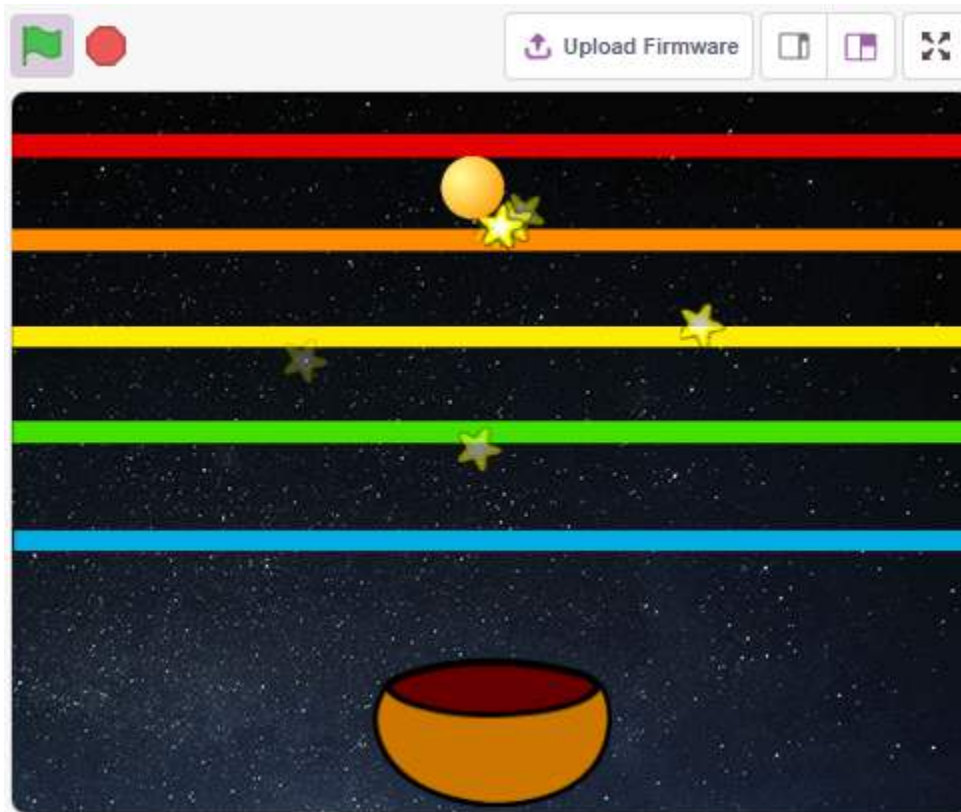
There are 2 events included here.

- When the **star** sprite receives the broadcasted message turn, it determines the value of flag; if flag is 1, it turns 10 degrees to the left, otherwise it reverses. Since it is in [FOREVER], it will keep turning.
- When this sprite is clicked, set both pins of the motor to high to make it stop rotating and stop the other scripts in this sprite.



3.16 2.13 Blow Ball

In this project, we use sound sensor to make the ball on the stage fly upwards. Blow into the sound sensor module, the more vibration it feels, the higher the ball can fly. When the ball touches the string, it makes a nice sound as well as a twinkling starlight.



3.16.1 You Will Learn

- How the Sound module works and the angle range
- Fill the sprite with colors
- Touch between the sprites

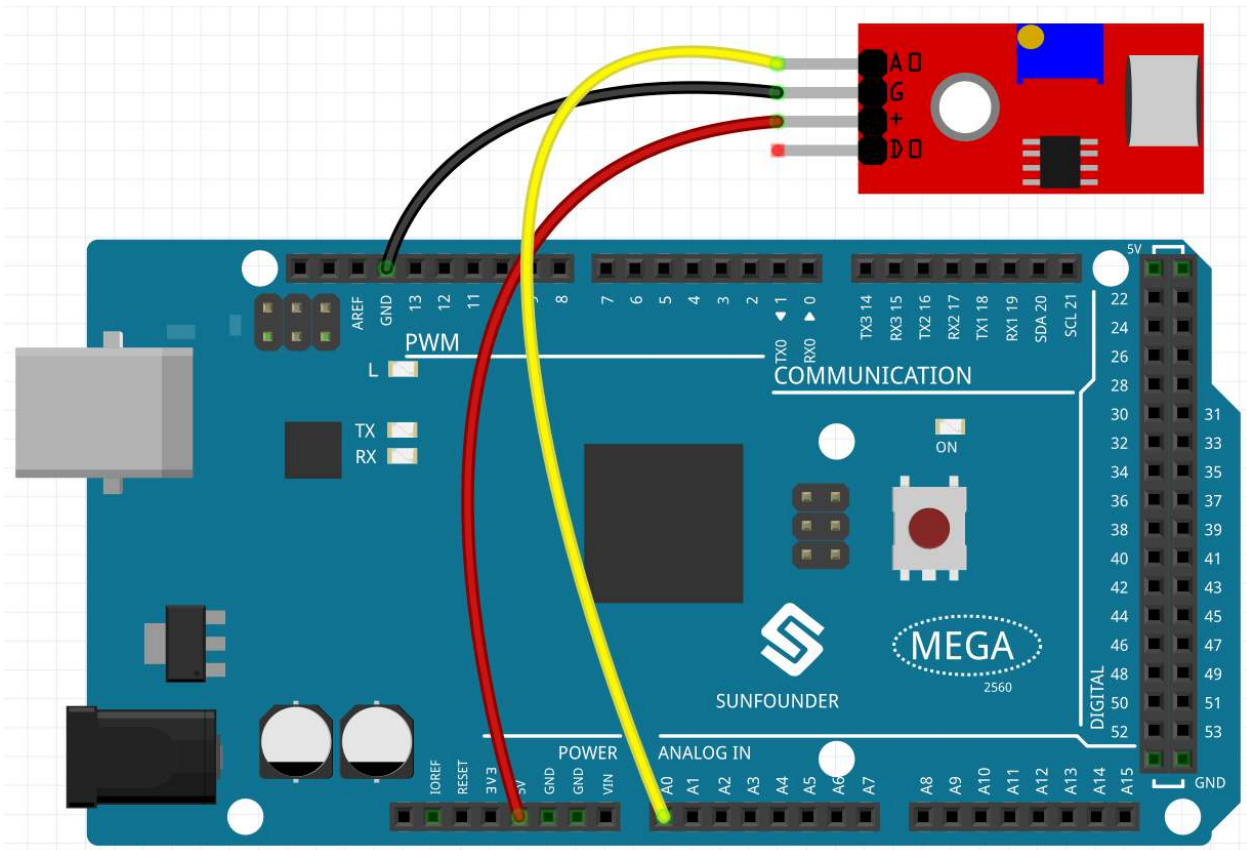
3.16.2 Build the Circuit

A sound sensor is defined as a module that detects sound waves through its intensity and converting it to electrical signals.

This module has two outputs:

- **AO:** analog output, real-time output voltage signal of microphone.
- **DO:** when the intensity of the sound reaches a certain threshold, the output is a high or low level signal. The threshold sensitivity can be achieved by adjusting the potentiometer.

Here we have used only the AO pin, now build the circuit according to the following diagram.



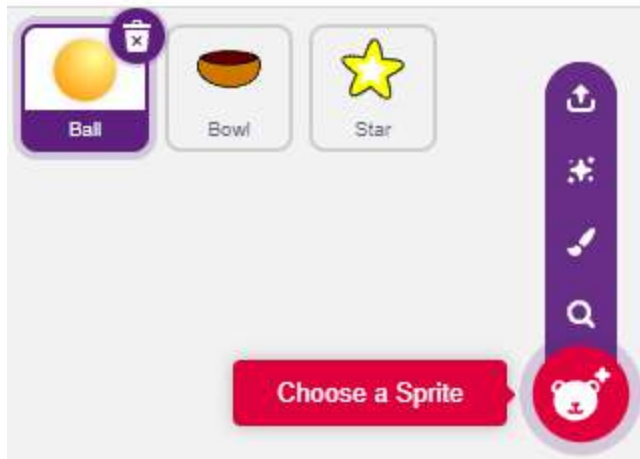
- *Breadboard*
- *Sound Sensor Module*

3.16.3 Programming

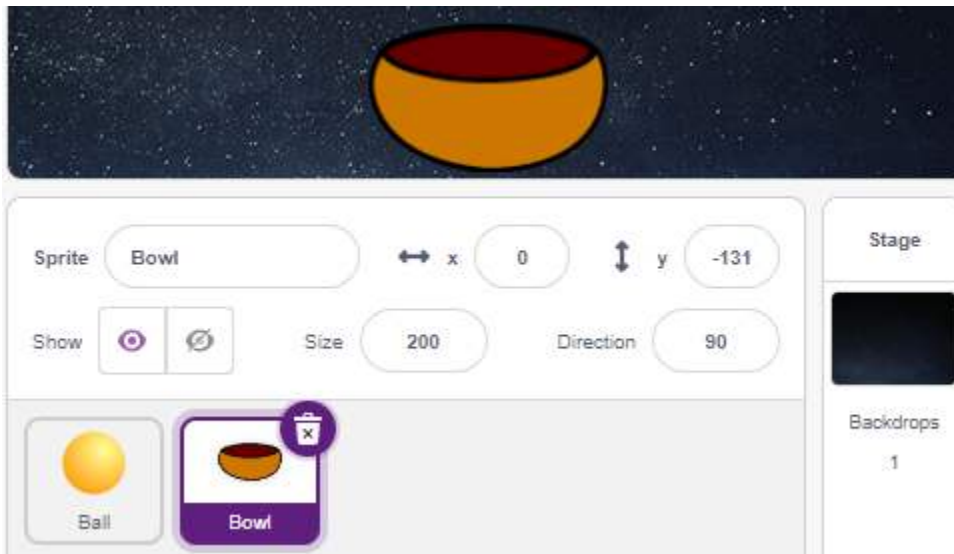
The effect we want to get is that when you blow into the sound sensor, the ball sprite on the stage keeps going up, and if you stop blowing, it will fall on the bowl sprite. If it touches the Line sprite while walking up or falling down, it will make a musical sound and emit star sprites in all directions.

1. Select sprite and backdrop

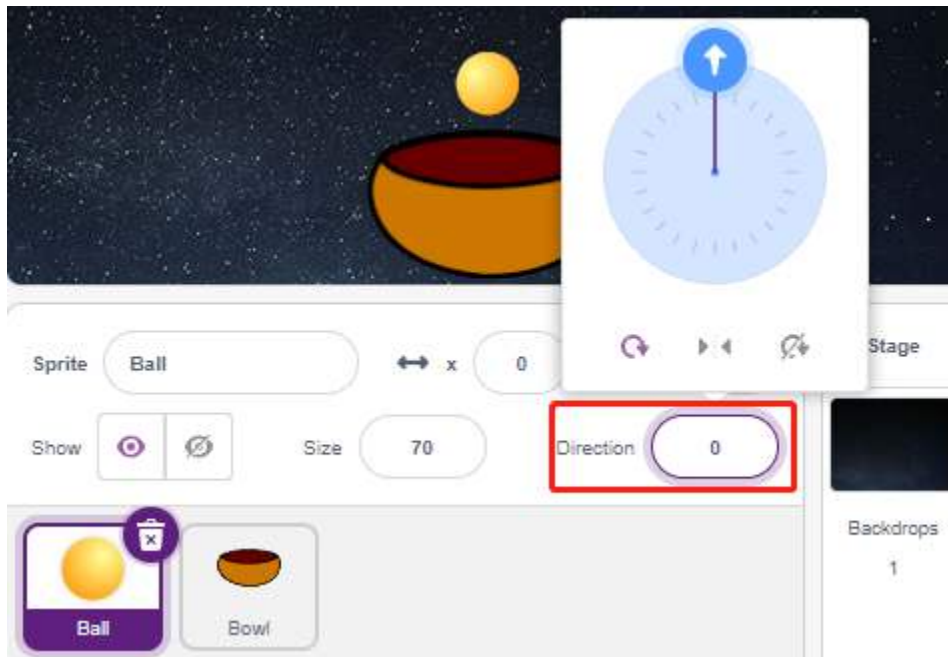
Delete the default sprite, select the **Ball**, **Bowl** and **Star** sprite.



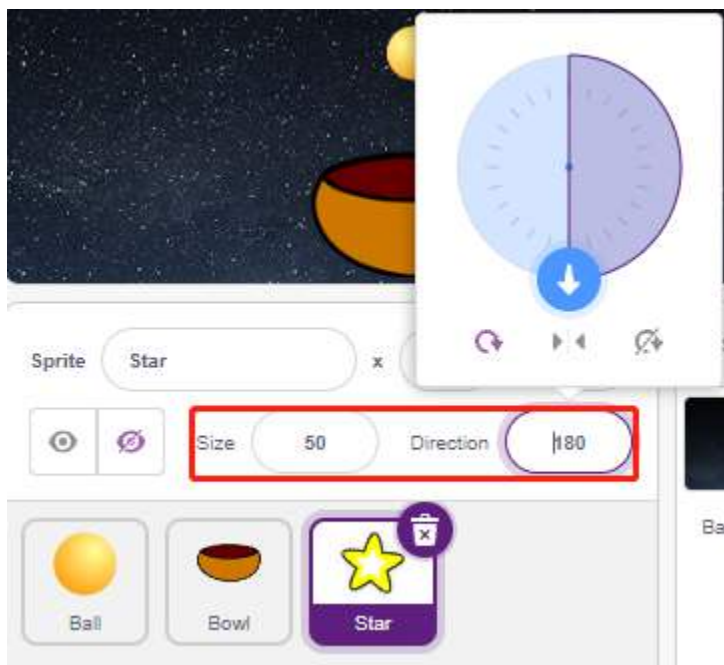
Move the **Bowl** sprite to the bottom center of the stage and enlarge its size.



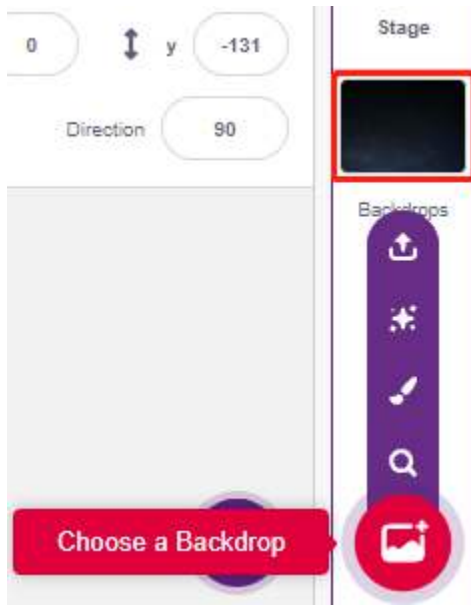
Because we need to move it upwards, so set direction of **Ball** sprite to 0.



Set the size and direction of the **Star** sprite to 180 because we need it to fall down, or you can change it to another angle.

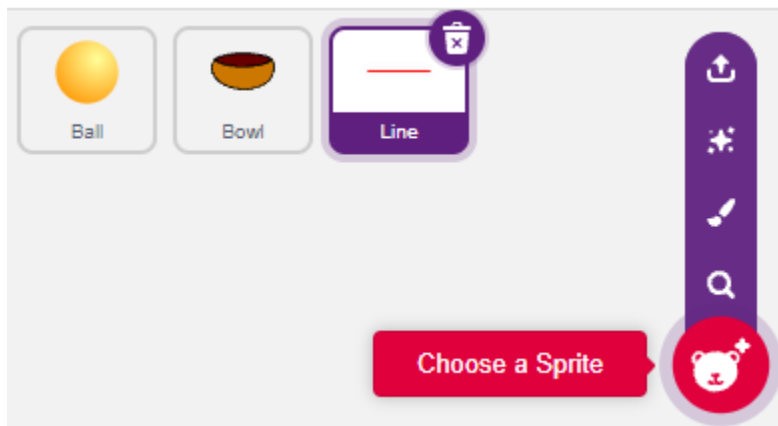


Now add the **Stars** backdrop.

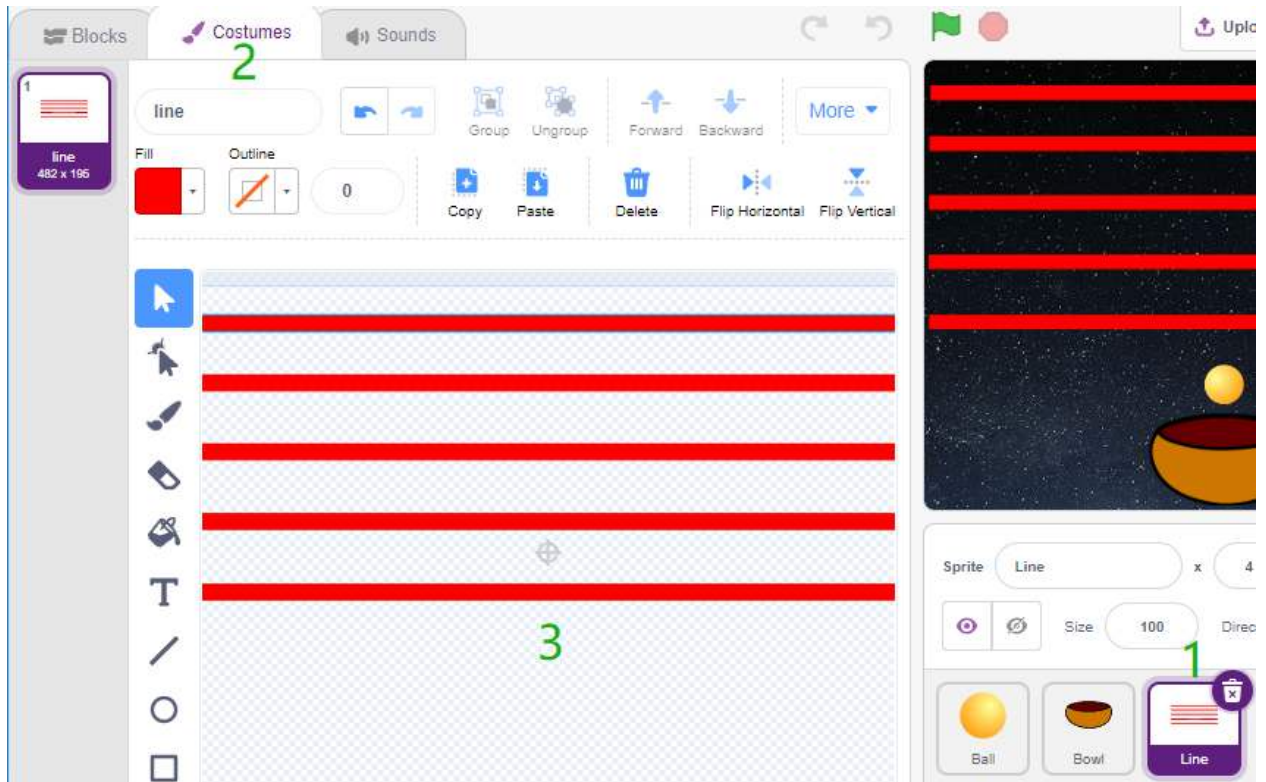


2. Draw a Line sprite

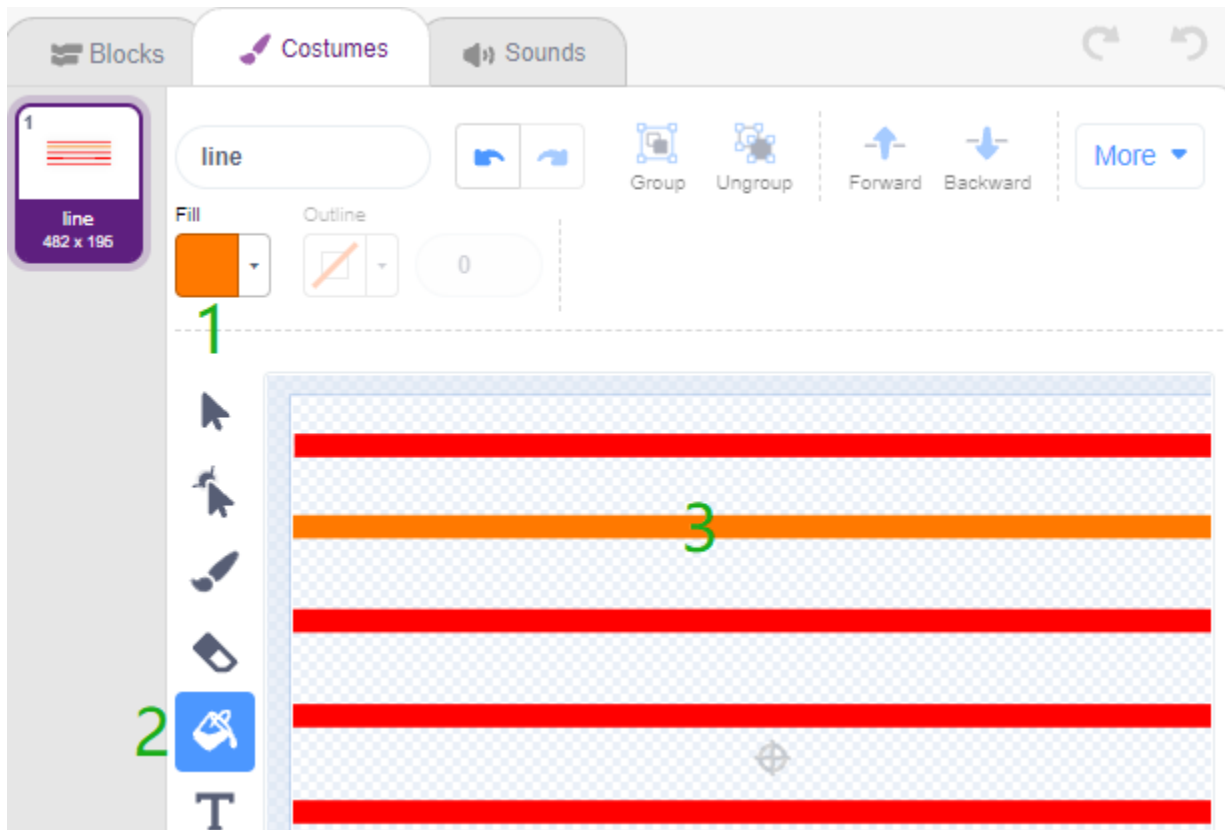
Add a Line sprite.



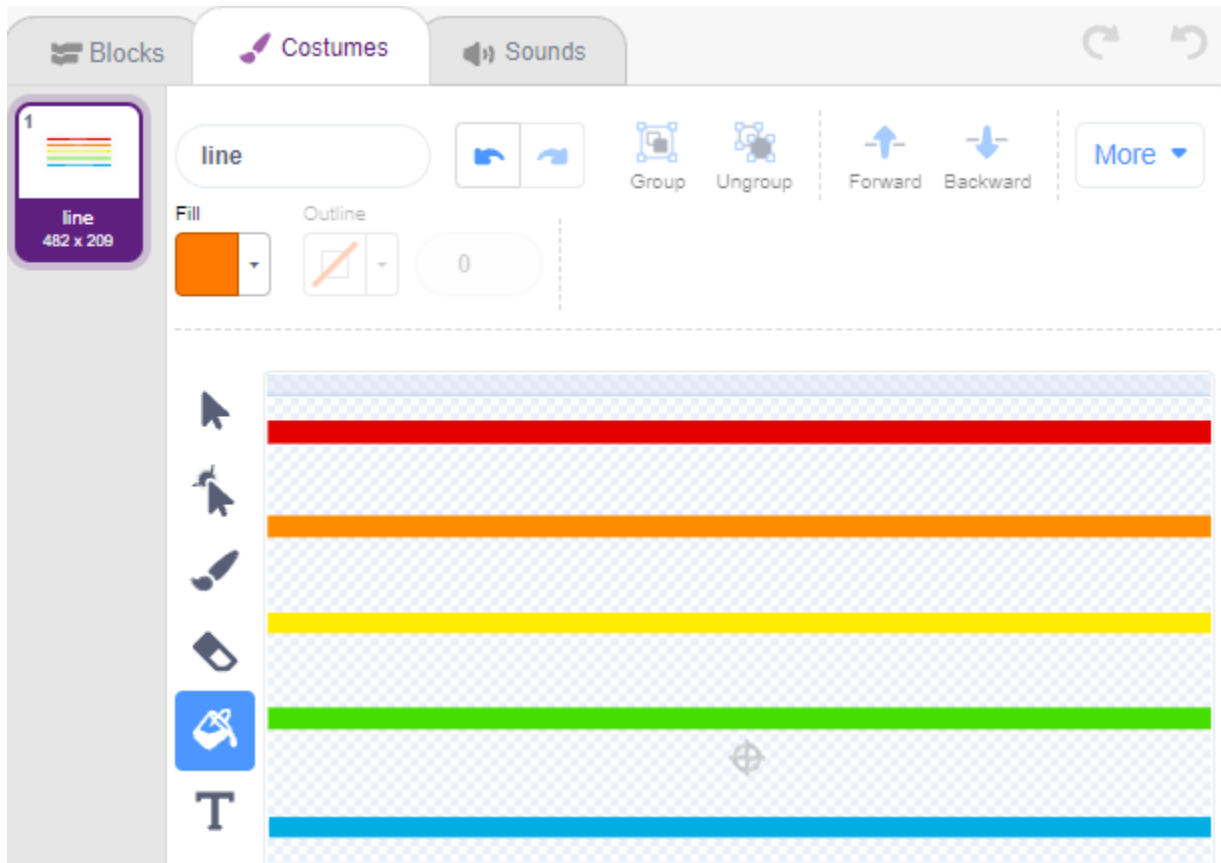
Go to the **Costumes** page of the **Line** sprite, reduce the width of the red line on the canvas slightly, then copy it 5 times and align the lines.



Now fill the lines with different colors. First choose a color you like, then click on the **Fill** tool and move the mouse over the line to fill it with color.

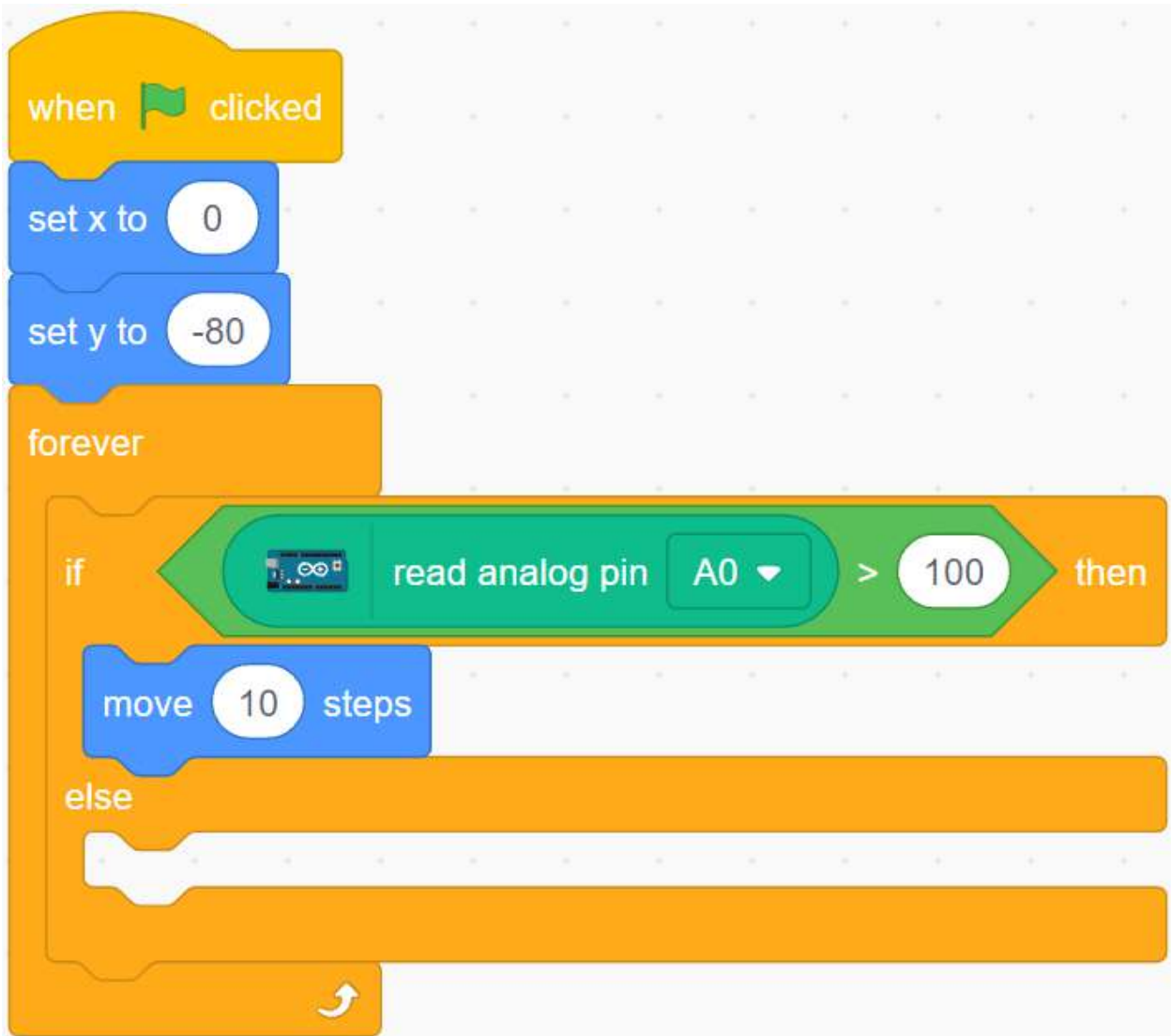


Follow the same method to change the color of the other lines.

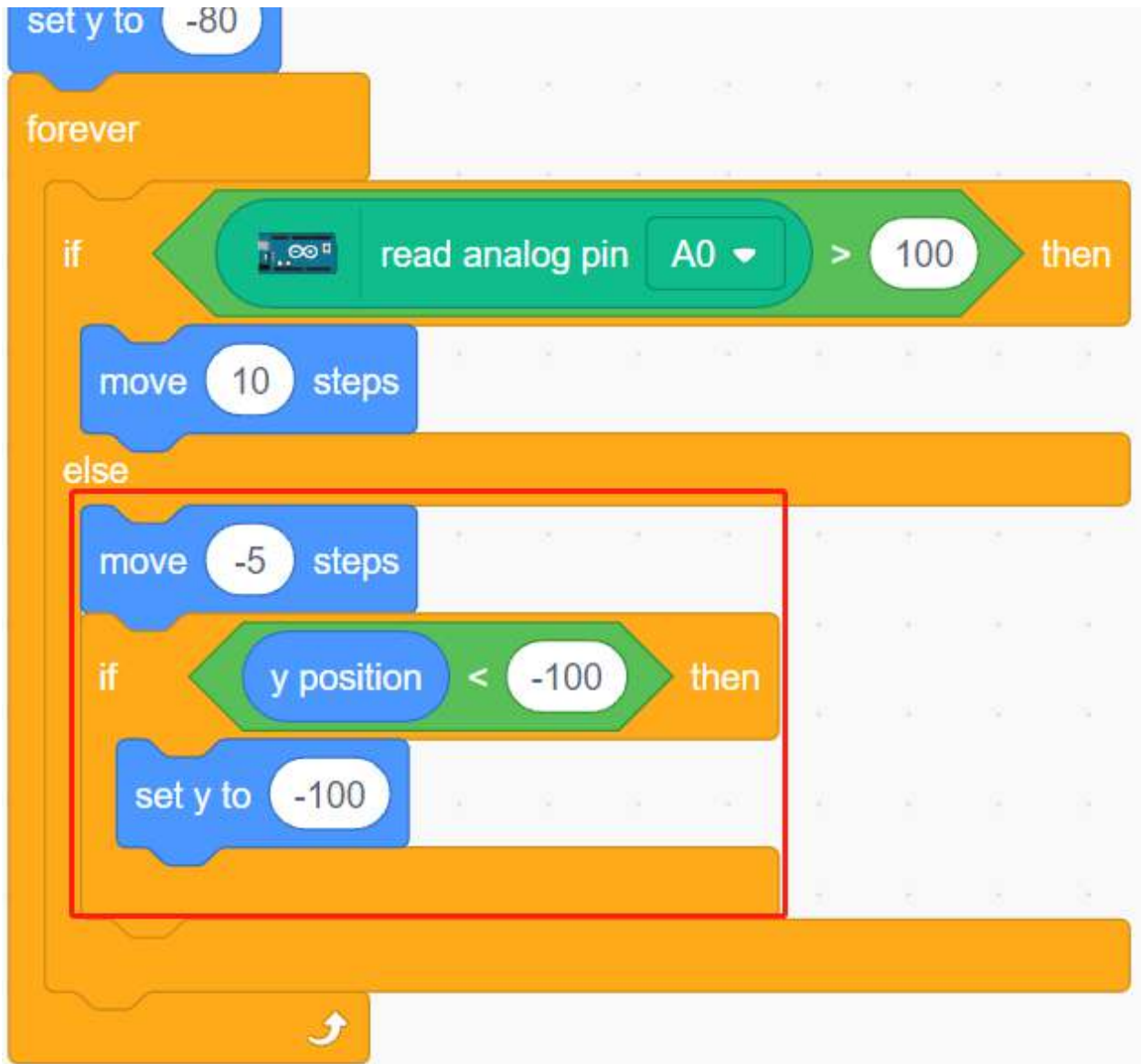


3. Scripting the Ball sprite

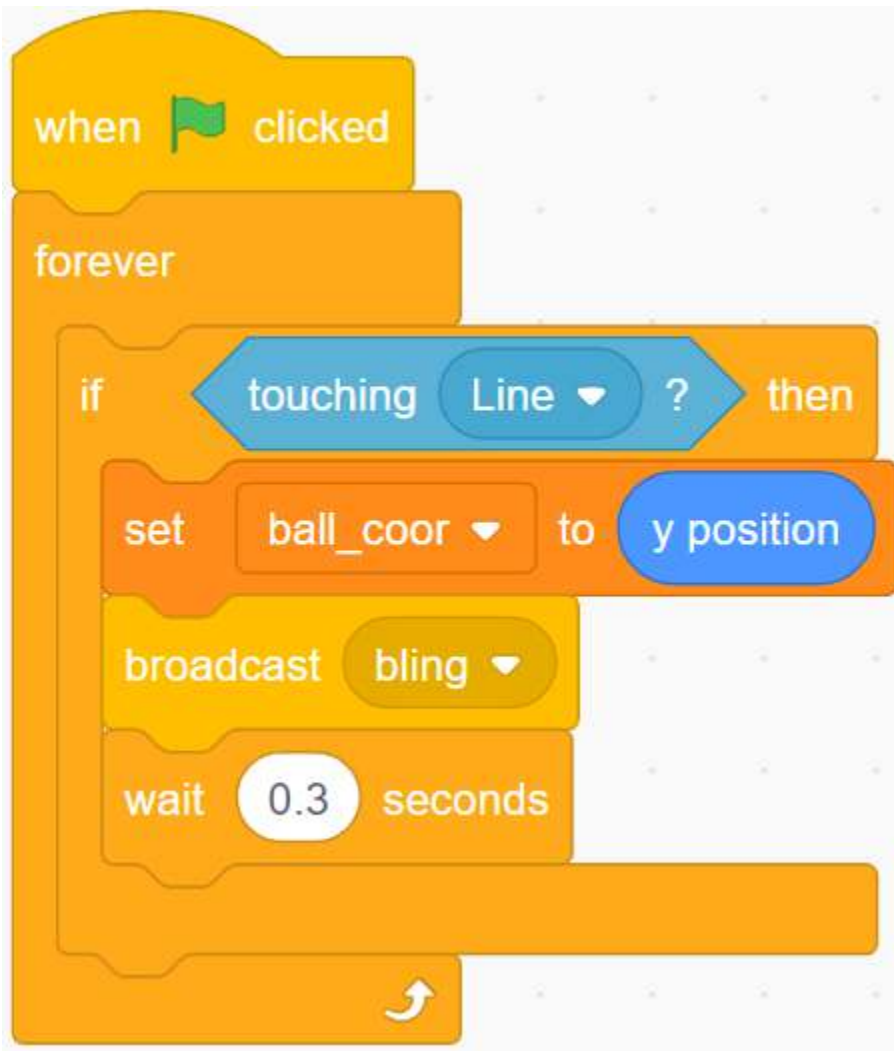
Set the initial position of the **Ball** sprite, then when the value of the sound sensor is greater than 100 (it can be any other value, depending on your current environment), let the Ball move up.



Otherwise, the **Ball** sprite will fall and limit its Y coordinate to a minimum of -100. This can be modified to make it look like it is falling on the Bowl sprite.

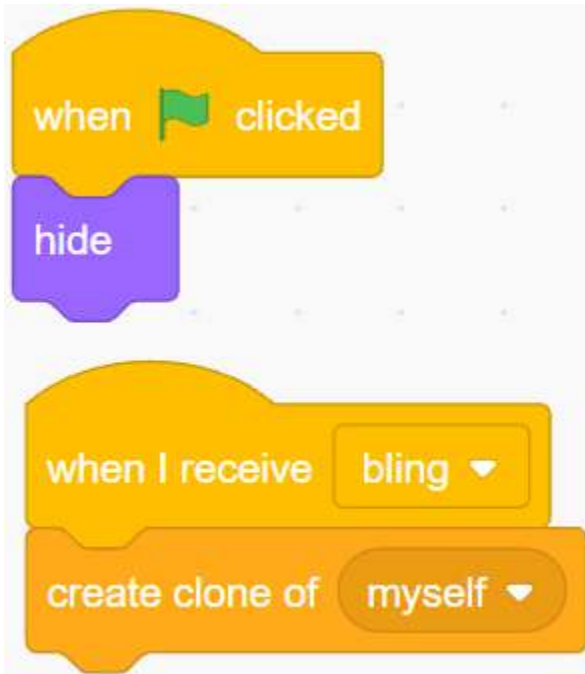


When the **Line** sprite is hit, the current Y coordinate is saved to the variable **ball_coor** and a **Bling** message is broadcast.

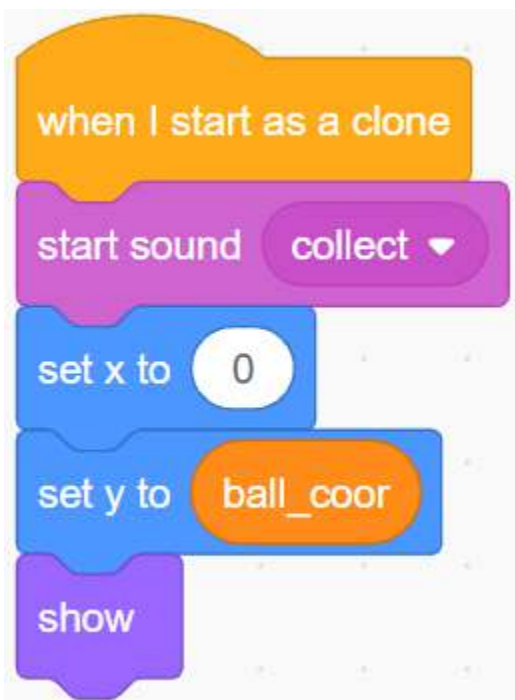


4. Scripting the Star sprite

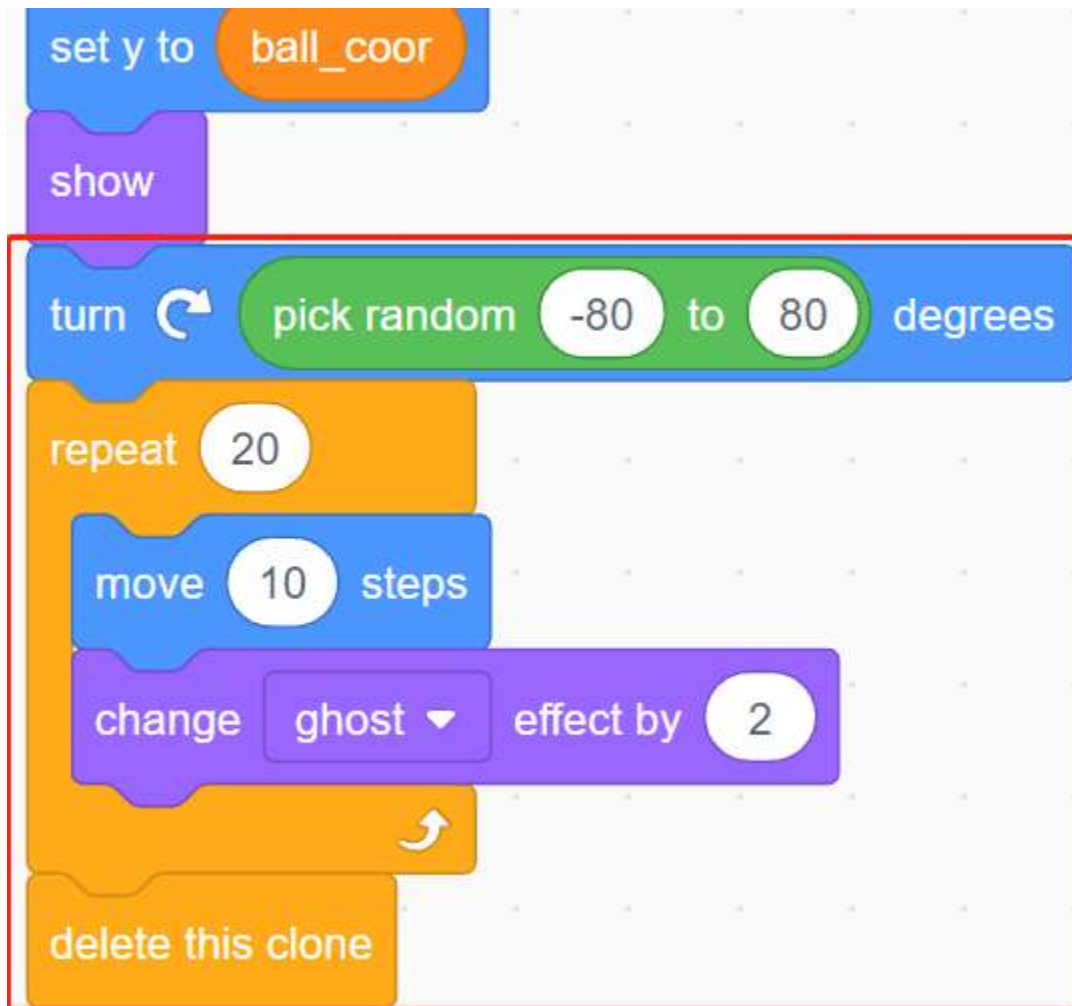
When the script starts, first hide the **Star** sprite. When the **Bling** message is received, clone the **Star** sprite.



When the **Star** sprite appears as a clone, play the sound effect and set its coordinates to be in sync with the **Ball** sprite.



Create the effect of the **Star** sprite appearing, and adjust it as needed.

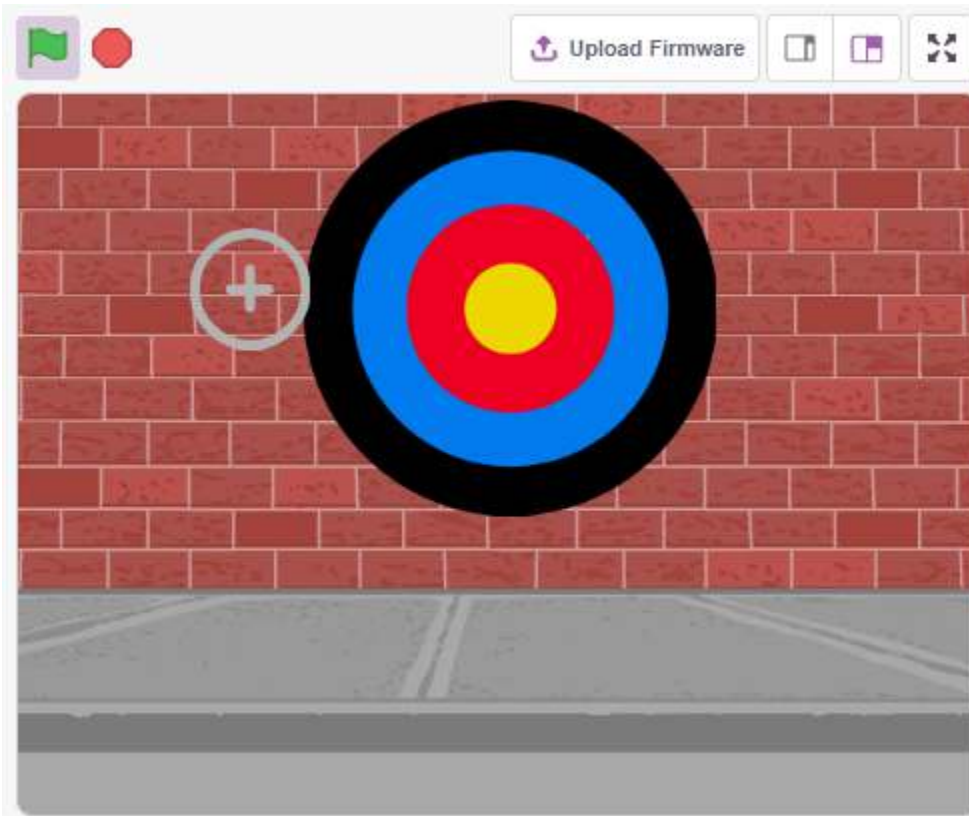


3.17 2.14 GAME - Shooting

Have you seen those shooting games on TV? The closer a contestant shoots a bullet on the target to the bullseye, the higher his score.

Today we are also doing a shooting game in Scratch. In the game, let the Crosshair shoot as far as possible to the bullseye to get a higher score.

Click on the green flag to start. Use the Obstacle Avoidance module to shoot an bullet.



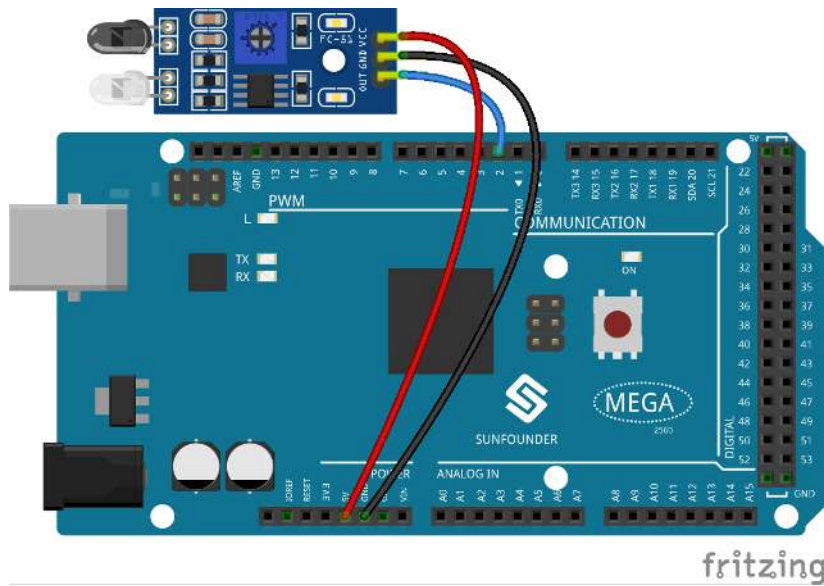
3.17.1 You Will Learn

- How the Obstacle Avoidance module works and the angle range
- Paint different sprites
- Touch colors

3.17.2 Build the Circuit

The obstacle avoidance module is a distance-adjustable infrared proximity sensor whose output is normally high and low when an obstacle is detected.

Now build the circuit according to the diagram below.

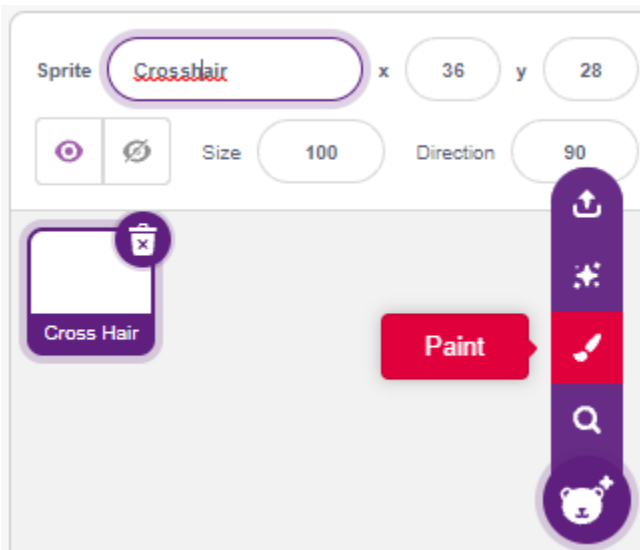


- Breadboard
- Obstacle Avoidance Module

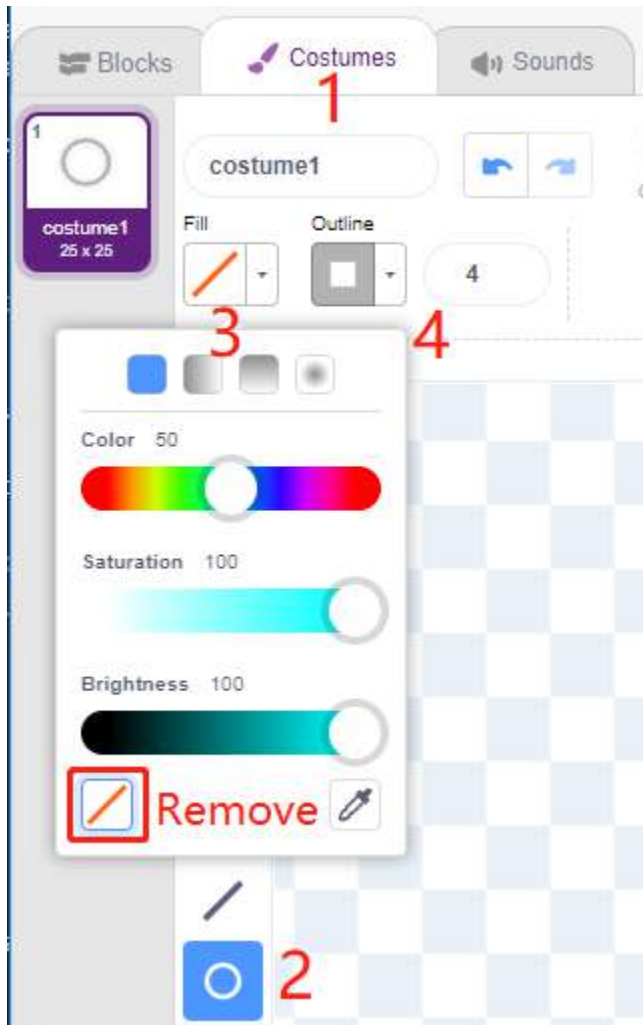
3.17.3 Programming

1. Paint the Crosshair sprite

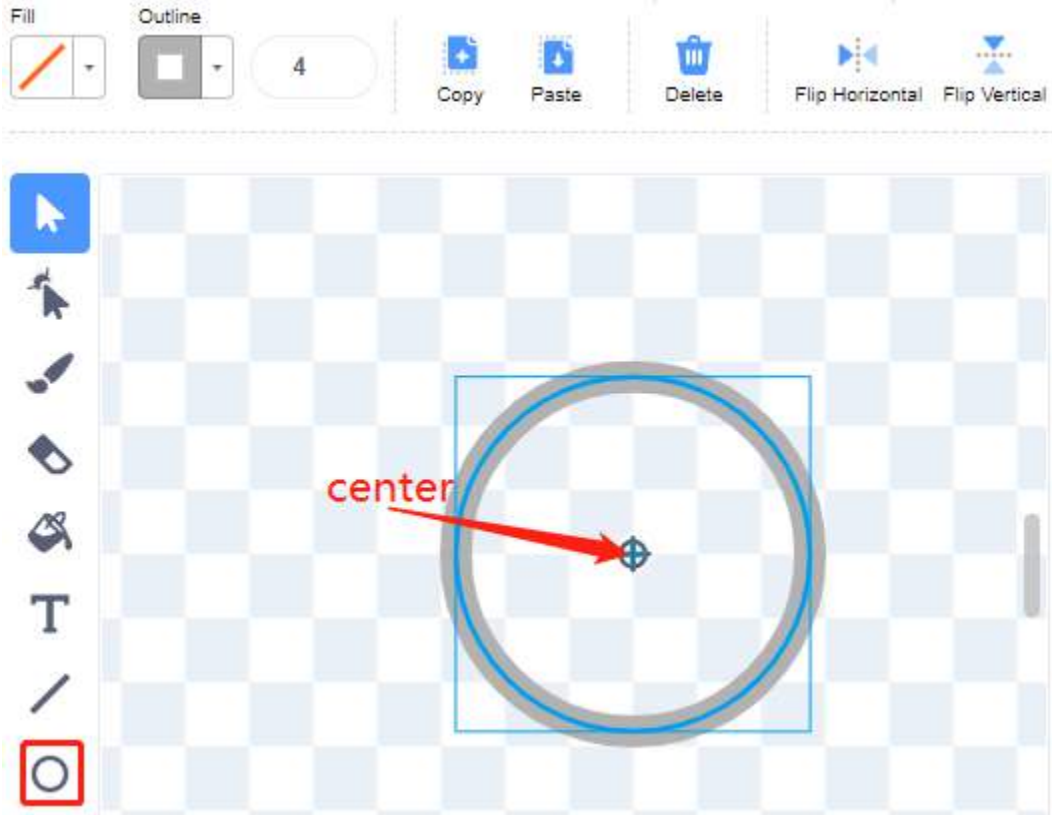
Delete the default sprite, select the **Sprite** button and click **Paint**, a blank sprite **Sprite1** will appear and name it **Crosshair**.



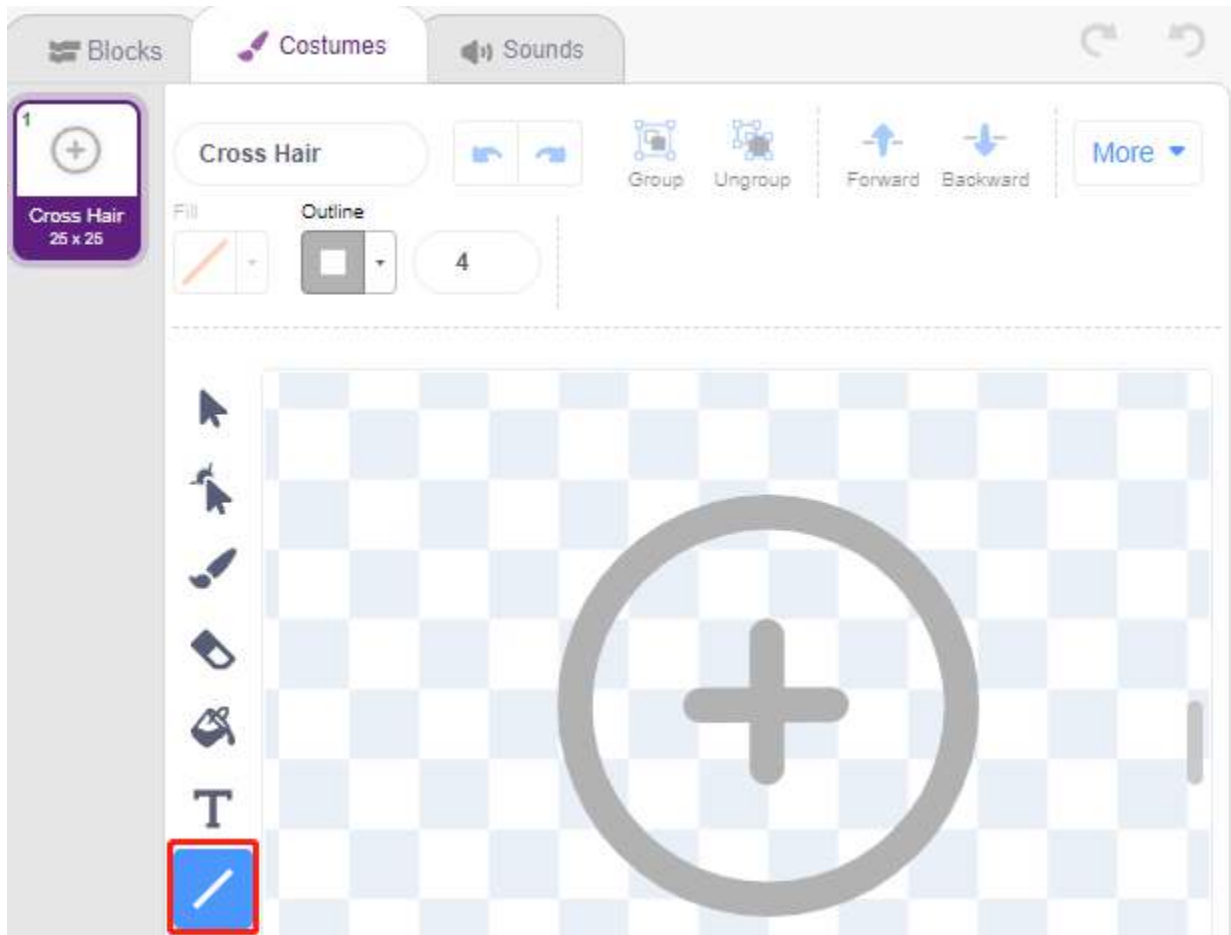
Go to the **Crosshair** sprite's **Costumes** page. Click on the **Circle** tool, remove the fill color, and set the color and width of the outline.



Now draw a circle with the **Circle** tool. After drawing, you can click to the **Select** tool and move the circle so that the original point is aligned with the center of the canvas.

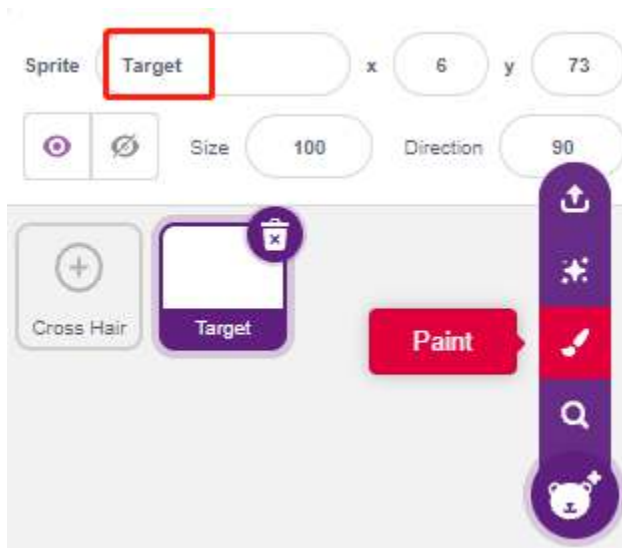


Using the **Line** tool, draw a cross inside the circle.

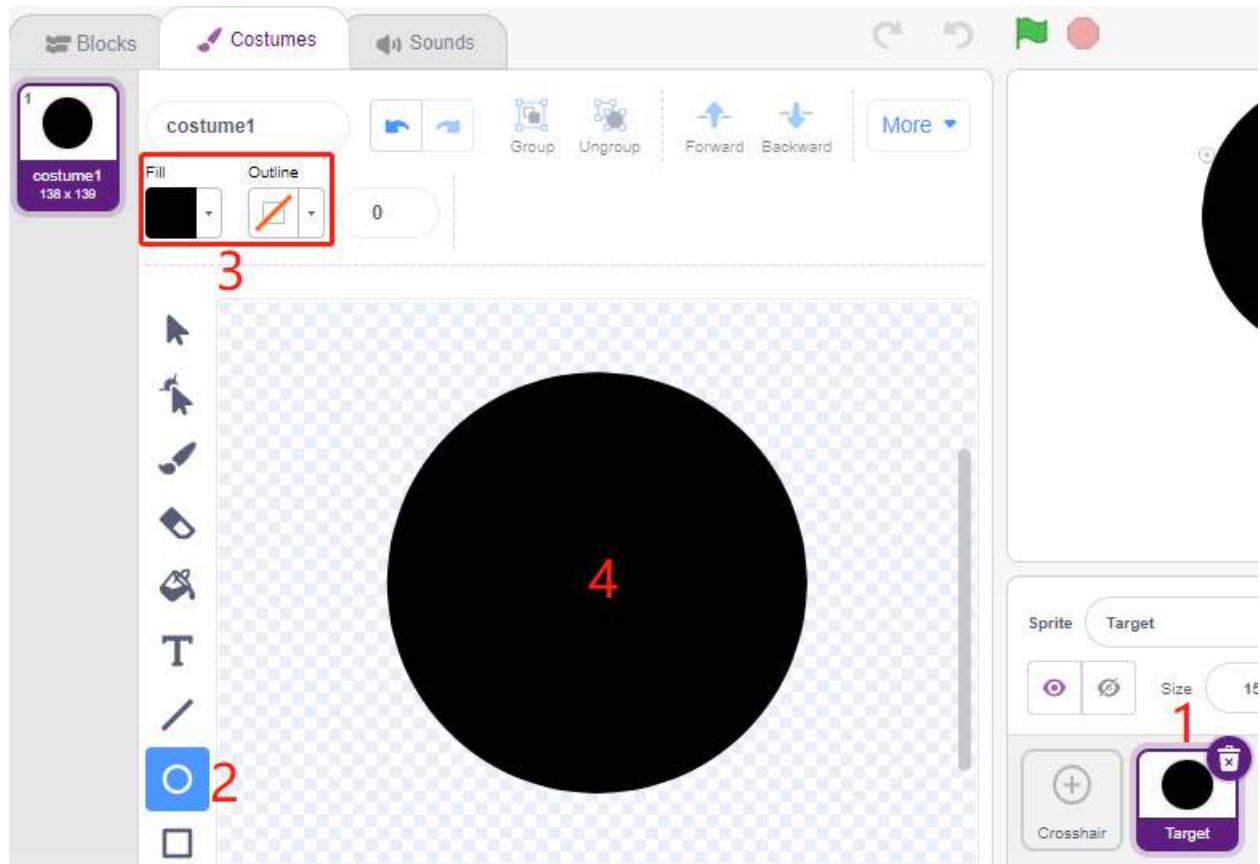


Paint the Target sprite

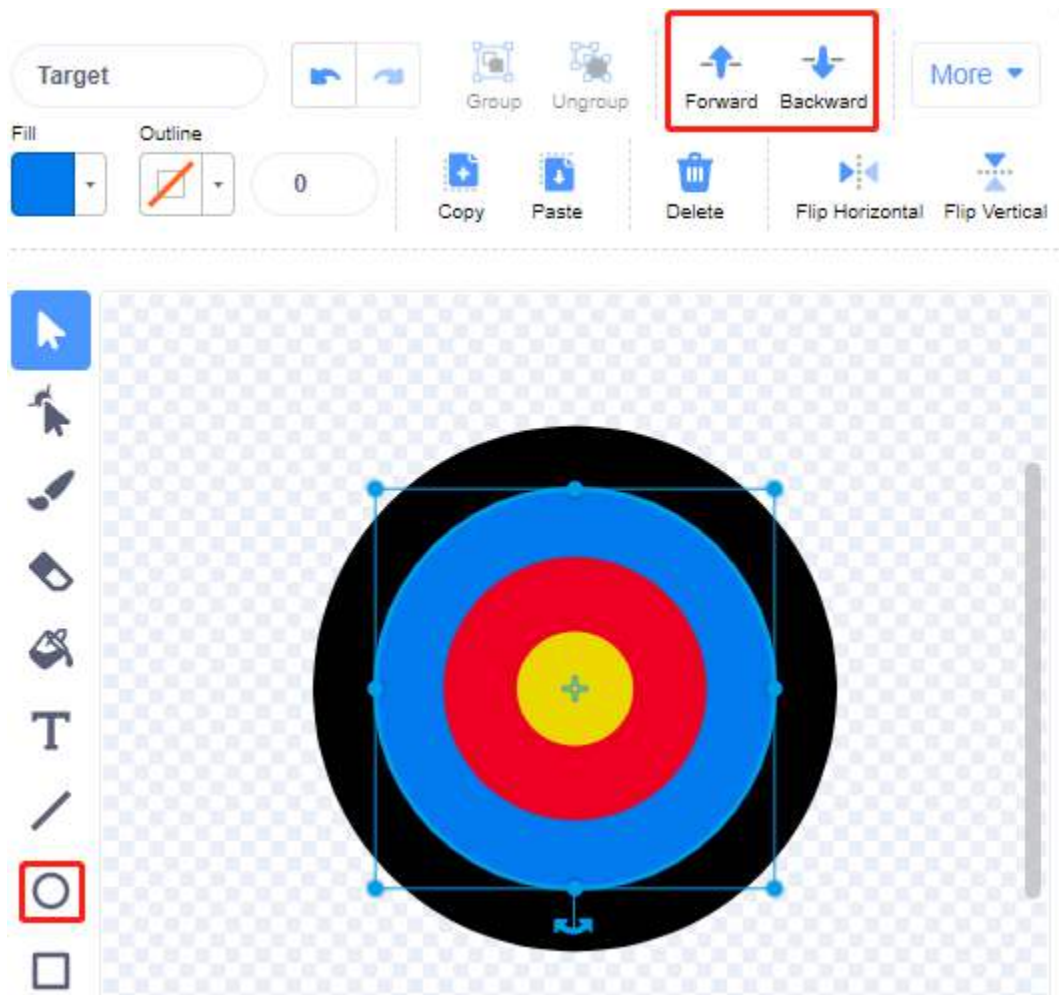
Create a new sprite called **Target** sprite.



Go to the Costumes page of the **Target** sprite, click on the **Circle** tool, select a fill color and remove the Outline and paint a large circle.

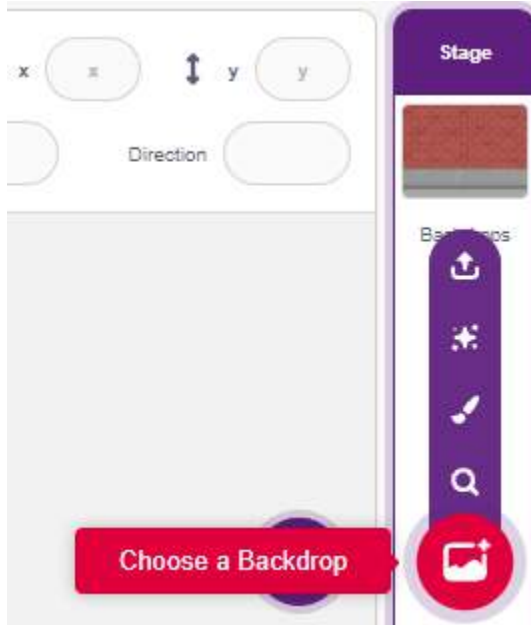


Use the same method to draw additional circles, each with a different color, and you can use the **Forward** or **Backward** tool to change the position of the overlapping circles. Note that you also need to select the tool to move the circles, so that the origin of all the circles and the center of the canvas are aligned.



3. Add a backdrop

Add a suitable background which preferably does not have too many colors and does not match the colors in the **Target** sprite. Here I have chosen **Wall1** backdrop.

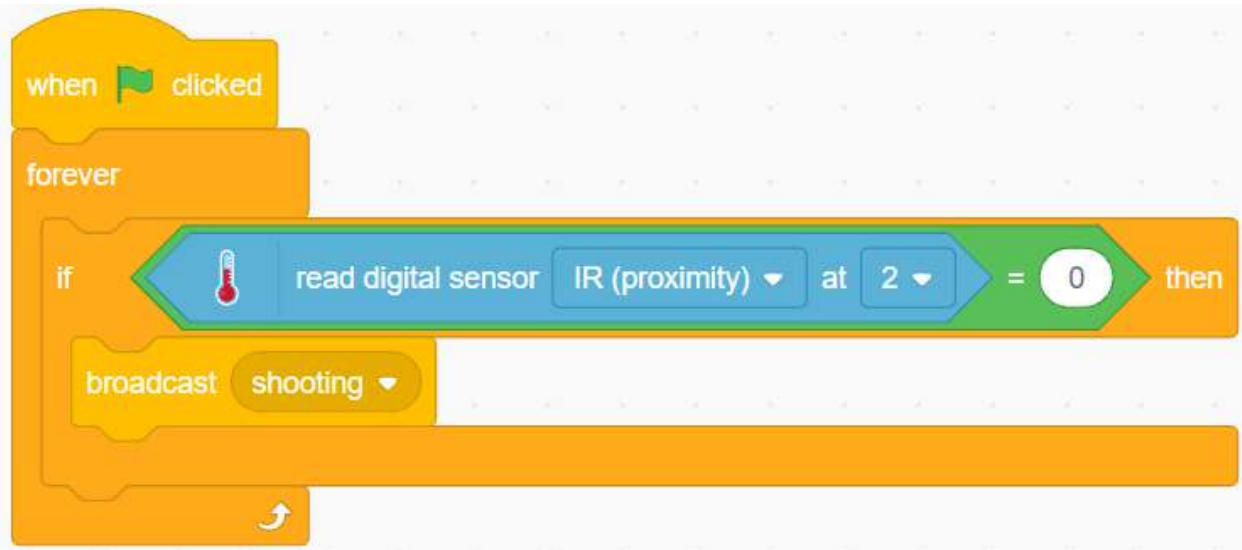


4. Script the Crosshair sprite

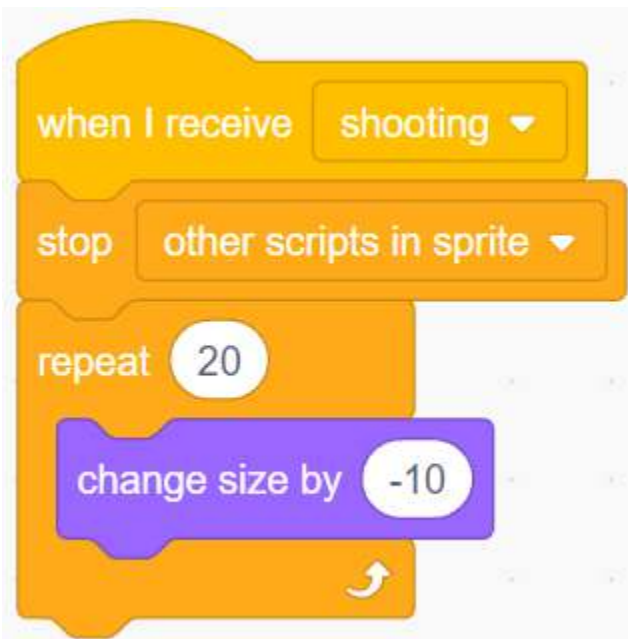
Set the random position and size of the **Crosshair** sprite, and let it move randomly.



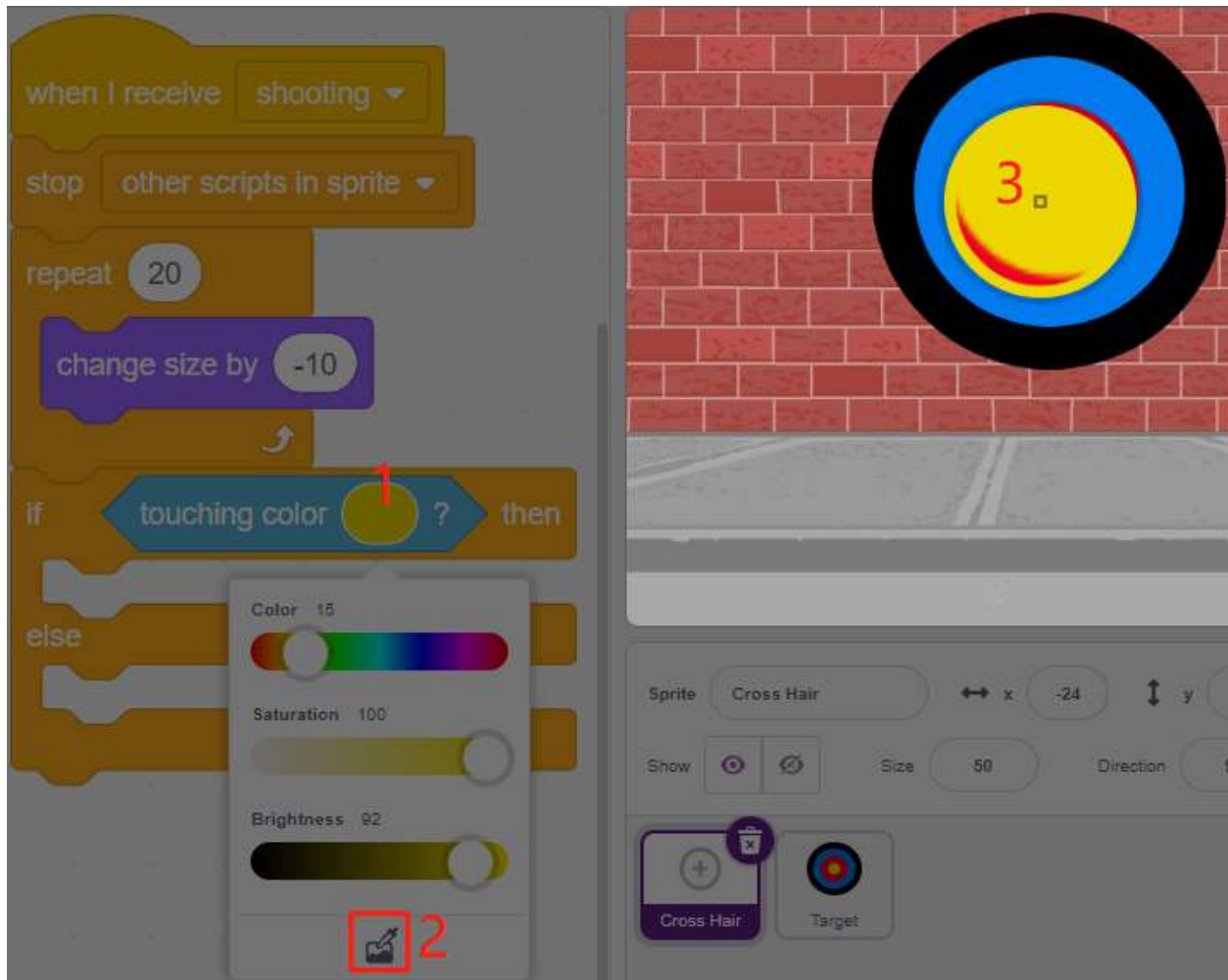
When a hand is placed in front of the obstacle avoidance module, it will output a low level as a transmit signal.



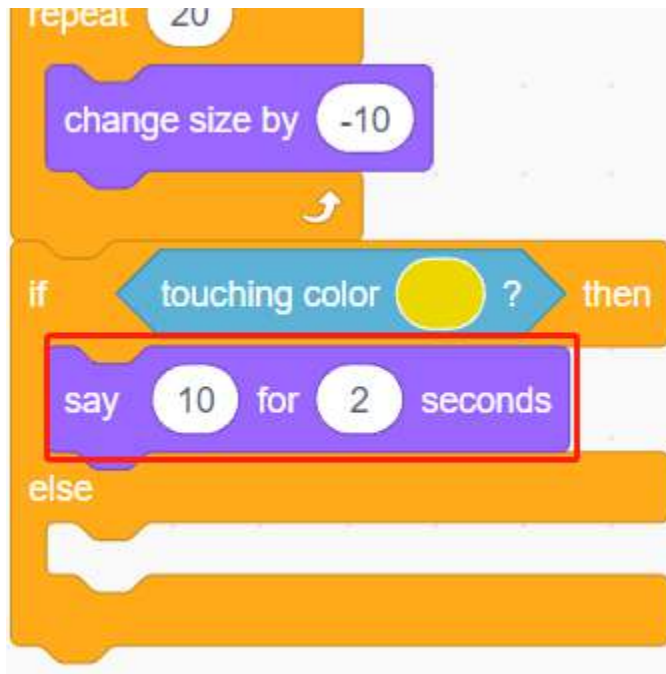
When the **shooting** message is received, the sprite stops moving and slowly shrinks, thus simulating the effect of a bullet being shot.



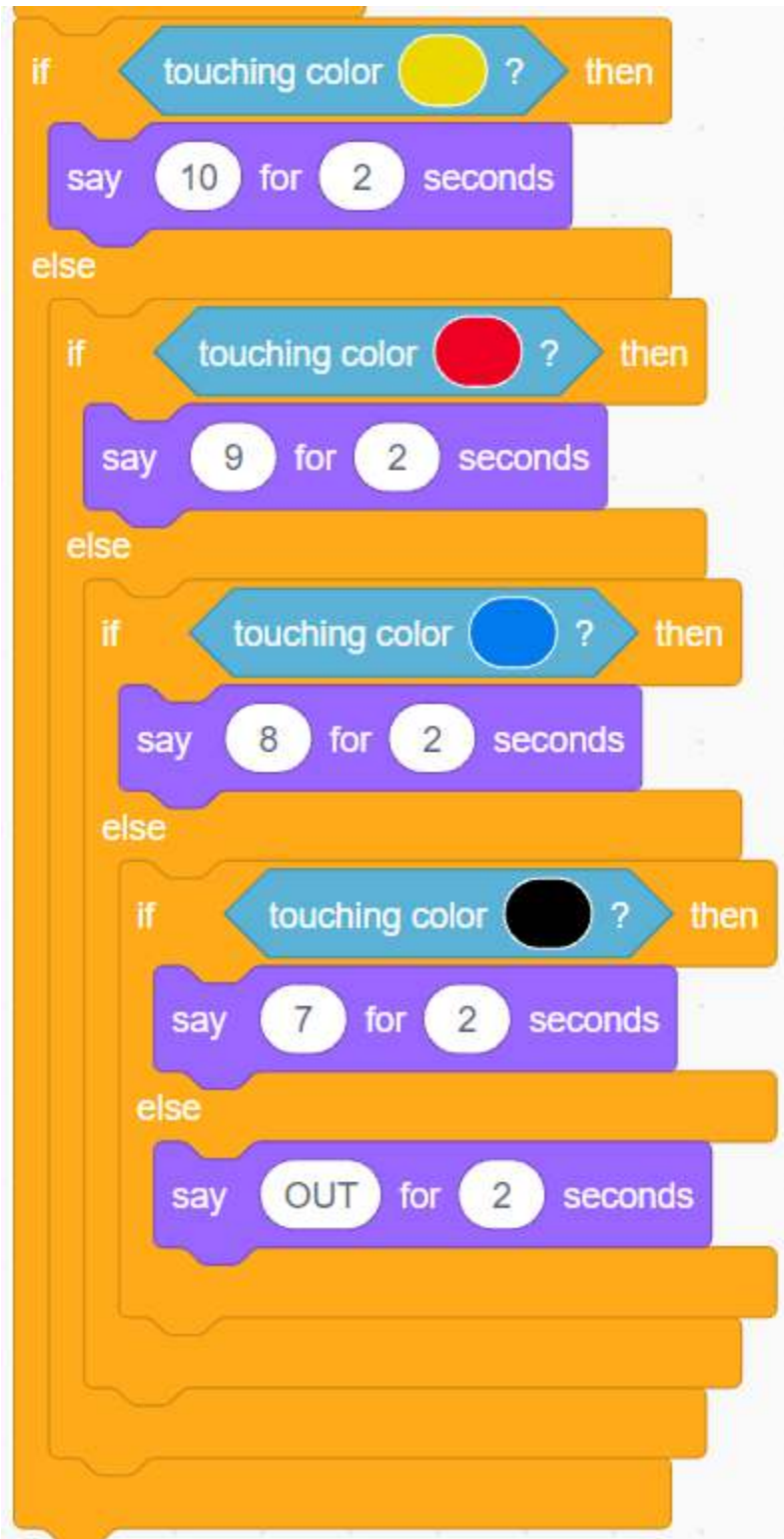
Use the [Touch color ()] block to determine the position of the shot.



When the shot is inside the yellow circle, 10 is reported.



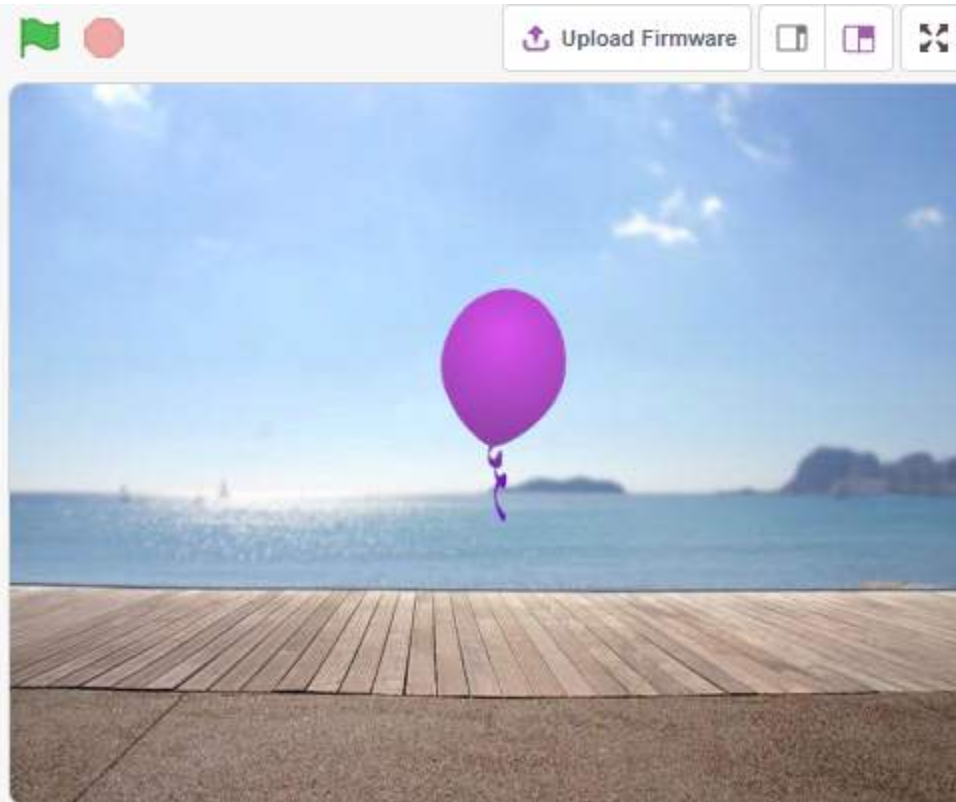
Use the same method to determine the position of the bullet shot, if it is not set on the **Target** sprite, it means it is out of the circle.



3.18 2.15 GAME - Inflating the Balloon

Here, we will play a game of ballooning.

After clicking the green flag, the balloon will become bigger and bigger. If the balloon is too big, it will be blown up; if the balloon is too small, it will fall down; you need to judge when to touch on the touch module to make it fly upwards.



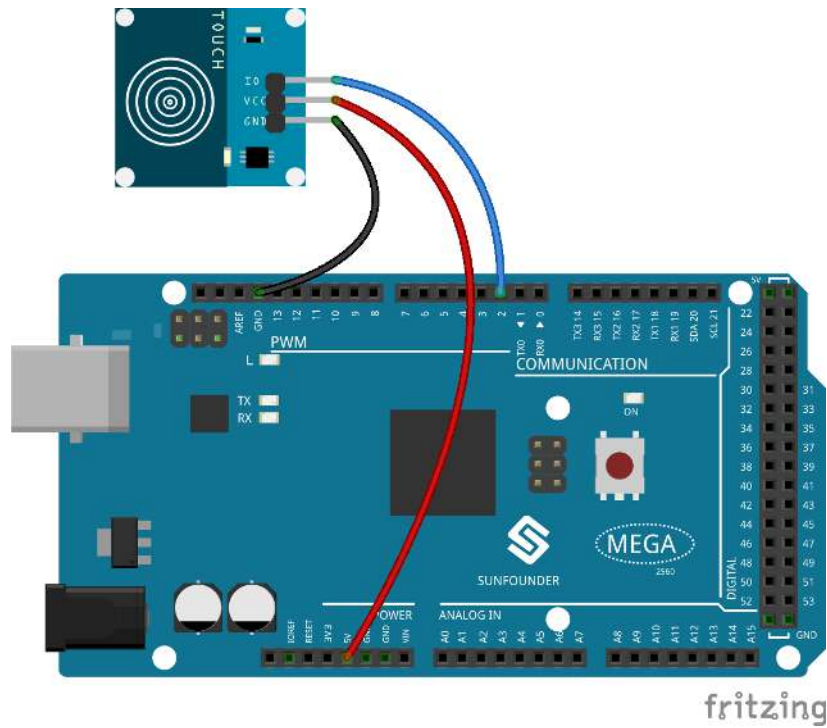
3.18.1 You Will Learn

- How the Touch module works and the angle range
- Paint costume for the sprite

3.18.2 Build the Circuit

This module is a capacitive touch switch module based on a touch sensor IC (TTP223B). In the normal state, the module outputs a low level with low power consumption; when a finger touches the corresponding position, the module outputs a high level and becomes low level again after the finger is released.

Now build the circuit according to the diagram below.

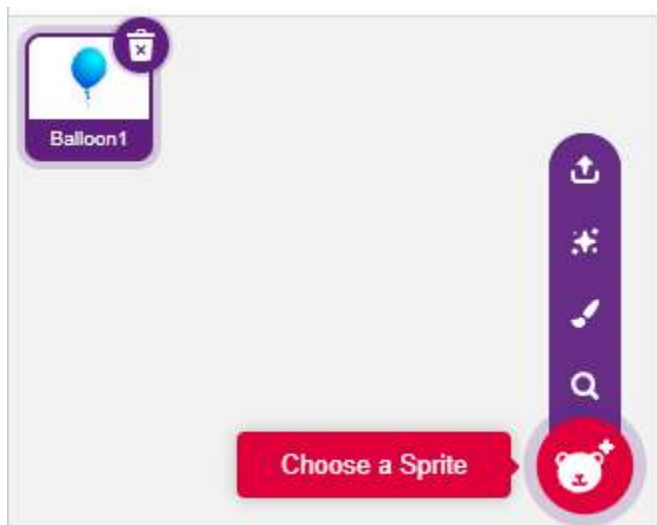


- Breadboard
- Touch Switch Module

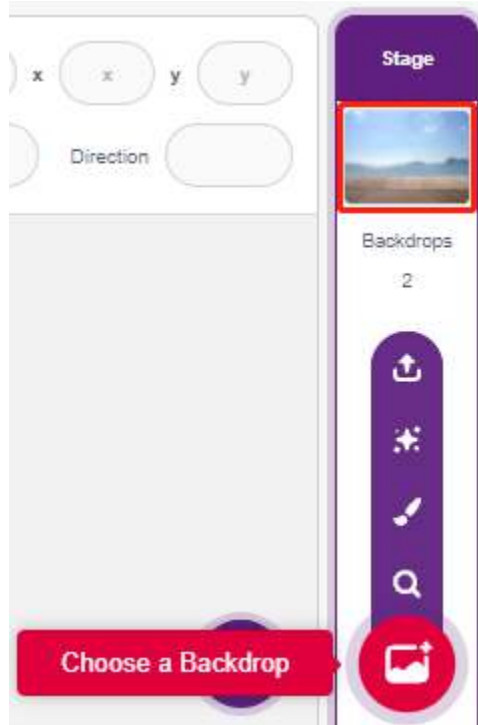
3.18.3 Programming

1. Add a sprite and a backdrop

Delete the default sprite, click the **Choose a Sprite** button in the lower right corner of the sprite area, then select the **Balloon1** sprite.



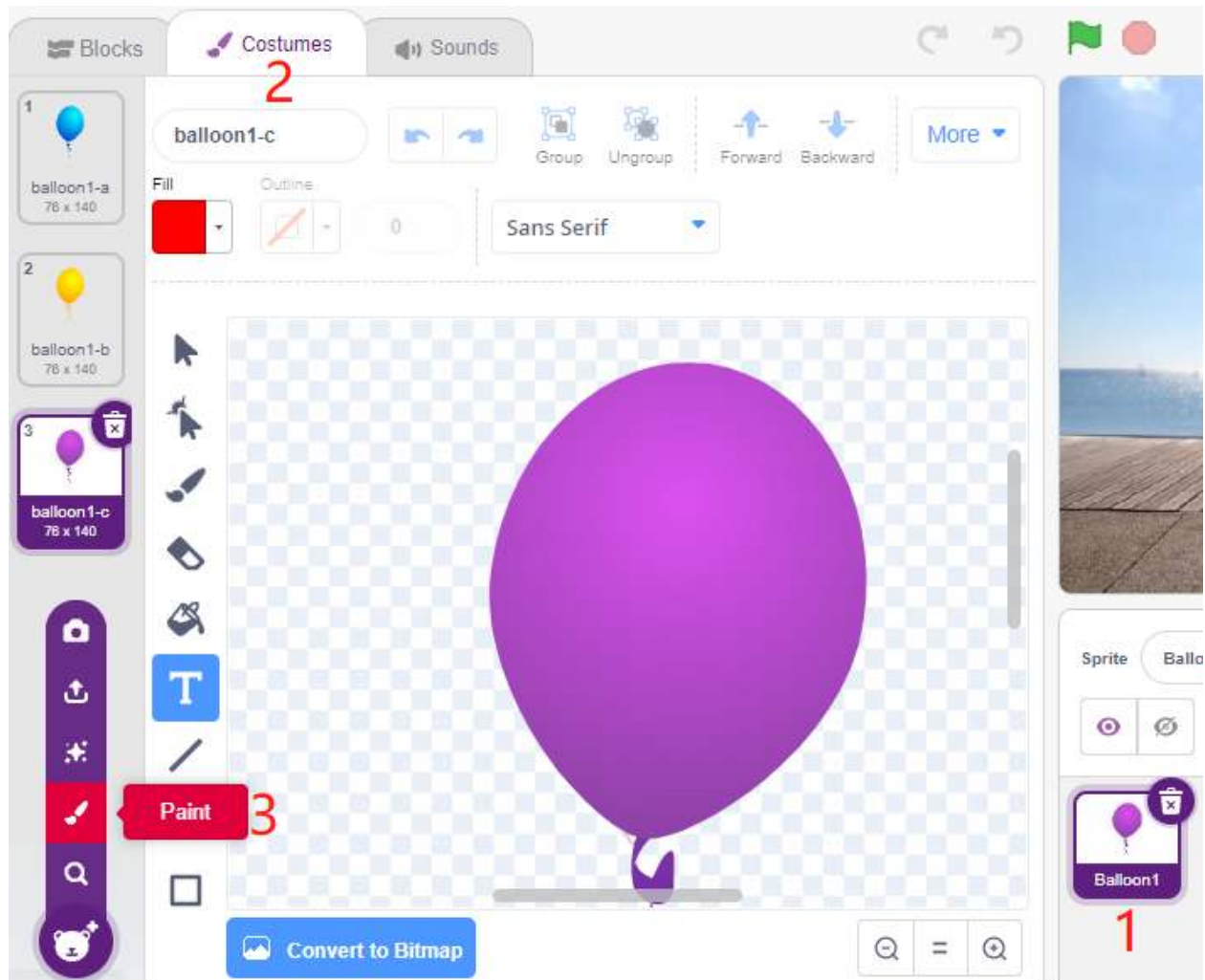
Add a **Boardwalk** backdrop via the **Choose a backdrop** button, or other backbackdrops you like.



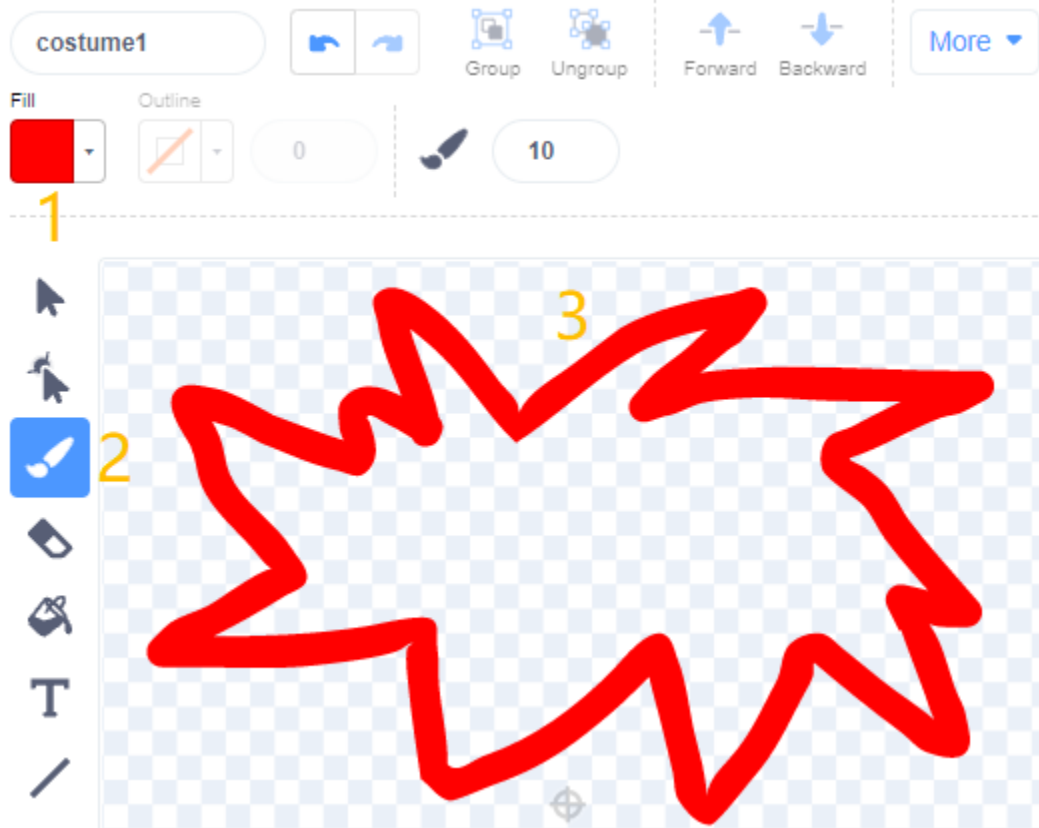
2. Paint a costume for the Balloon1 sprite

Now let's draw an exploding effect costume for the balloon sprite.

Go to the **Costumes** page for the **Balloon1** sprite, click the **Choose a Costume** button in the bottom left corner, and select **Paint** to bring up a blank **Costume**.



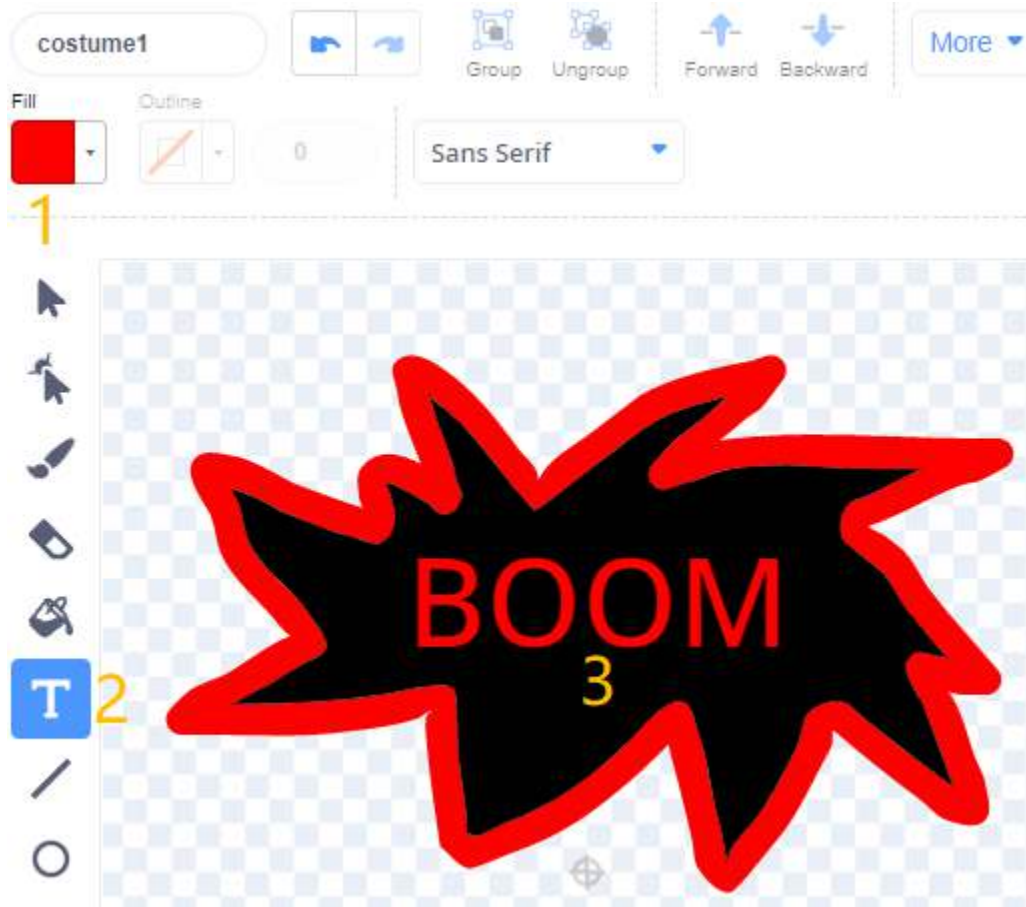
Select a color and then use the **Brush** tool to draw a pattern.



Select a color again, click the Fill tool, and move the mouse inside the pattern to fill it with a color.

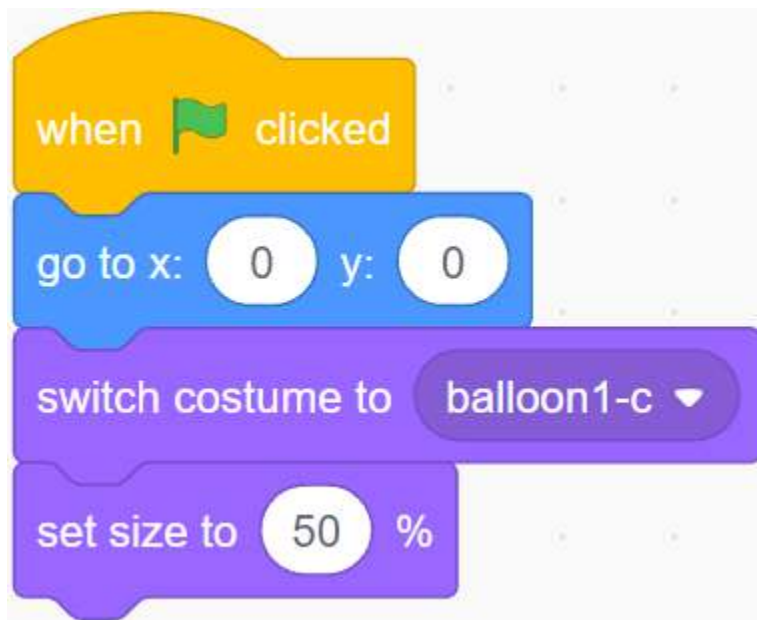


Finally, write the text BOOM, so that an explosion effect costume is complete.

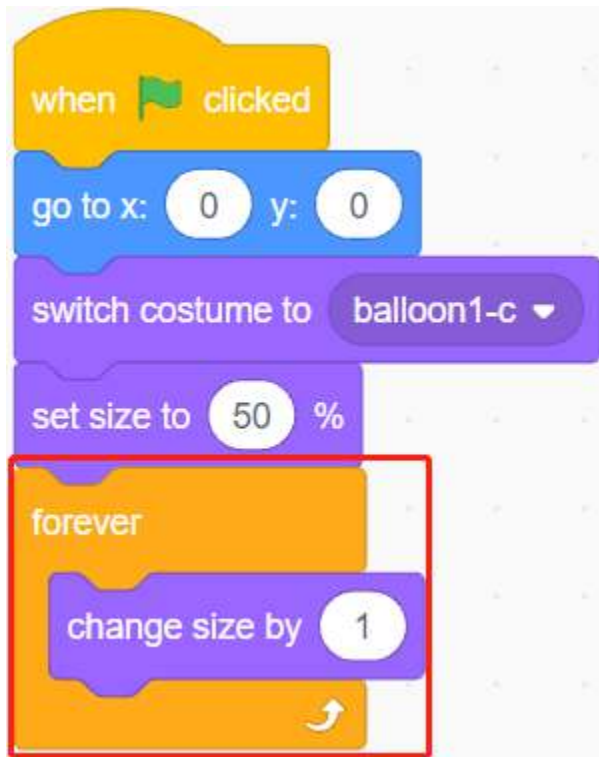


3. Scripting the Balloon sprite

Set the initial position and size of the **Balloon1** sprite.

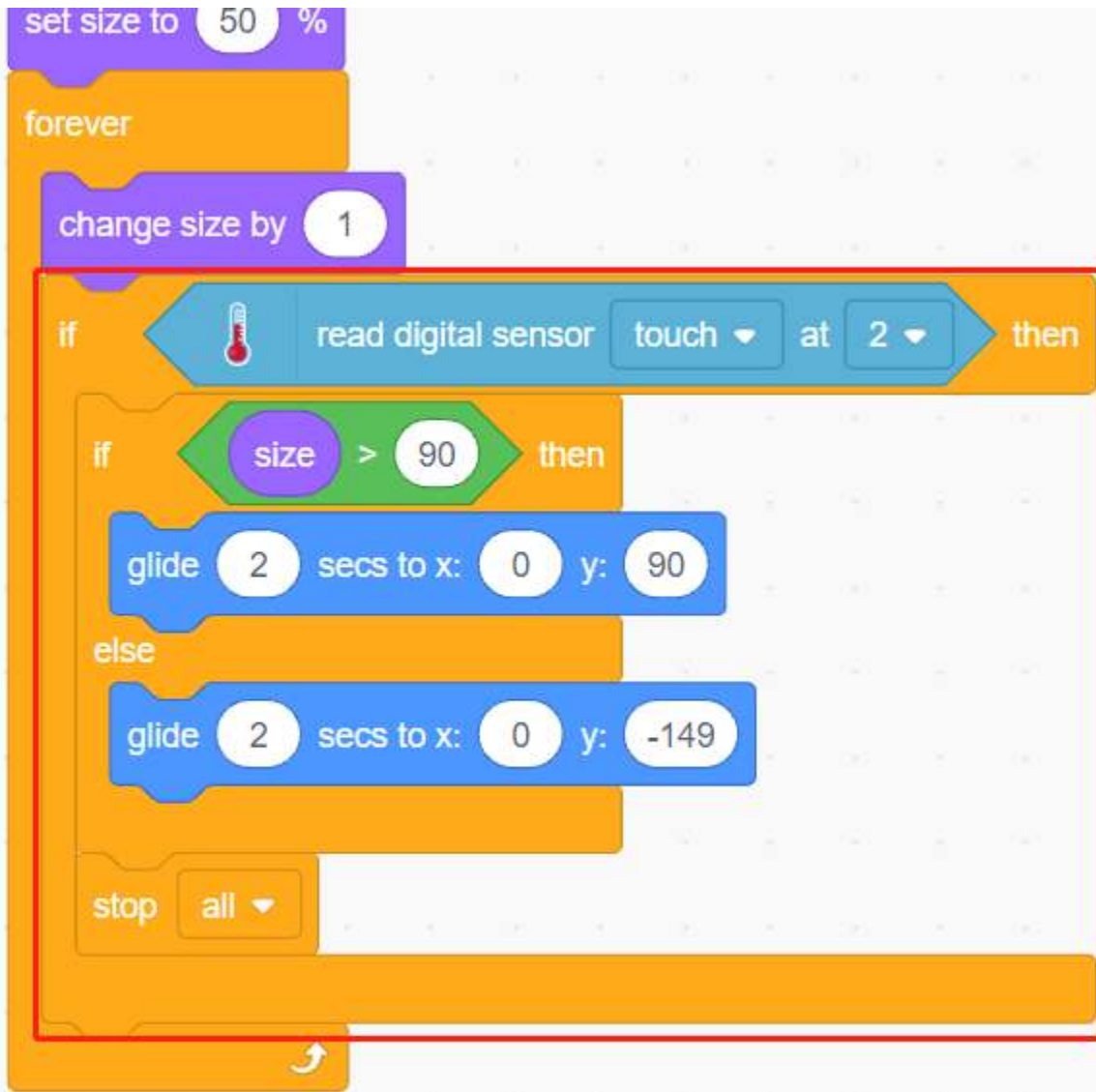


Then let the **Balloon** sprite slowly get bigger.

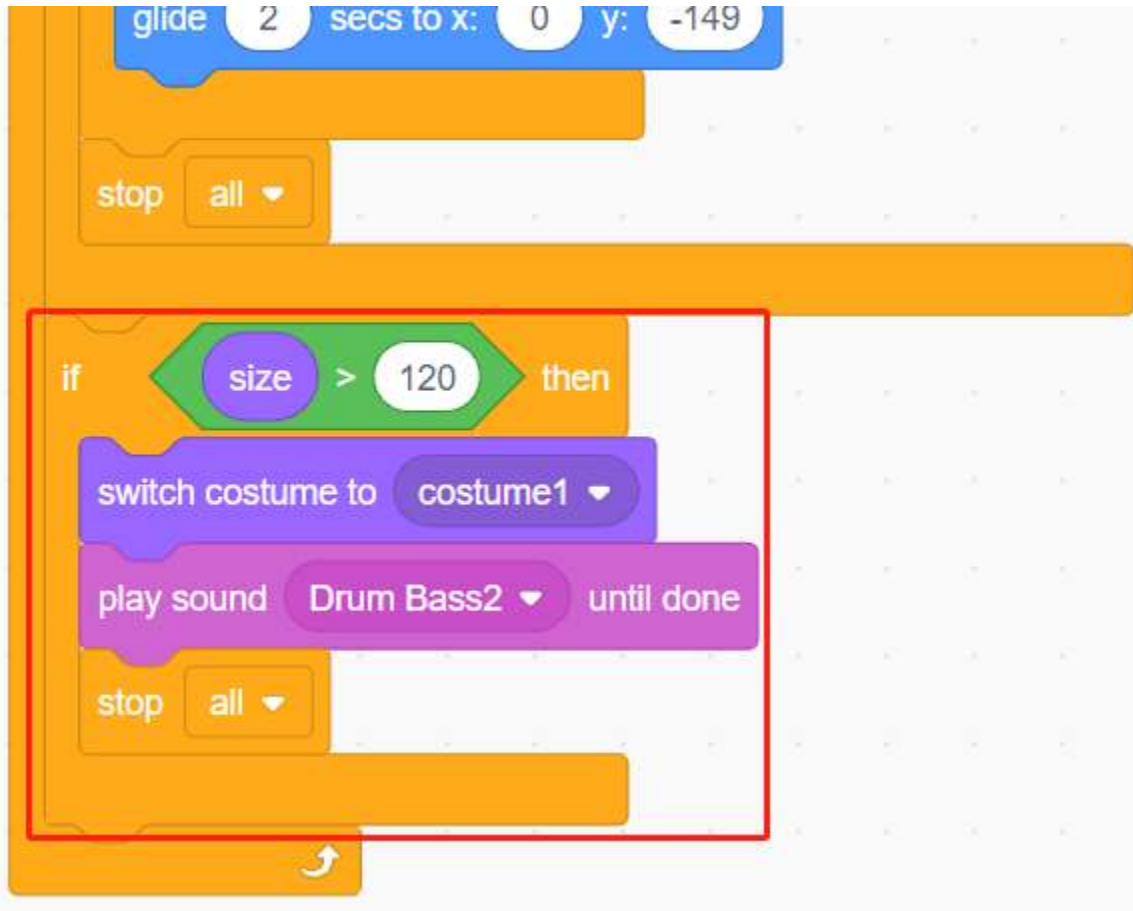


When the touch module is touched (value is 1), the size of the **Balloon1** sprite stops getting bigger.

- When the size is less than 90, it will fall (y coordinate decreases).
- When the size is bigger than 90 and smaller than 120, it will fly to the sky (y coordinate increases).



If the touch module has not been touched, the balloon slowly gets bigger and when the size is bigger than 120, it will explode (switch to the explode effect costume).

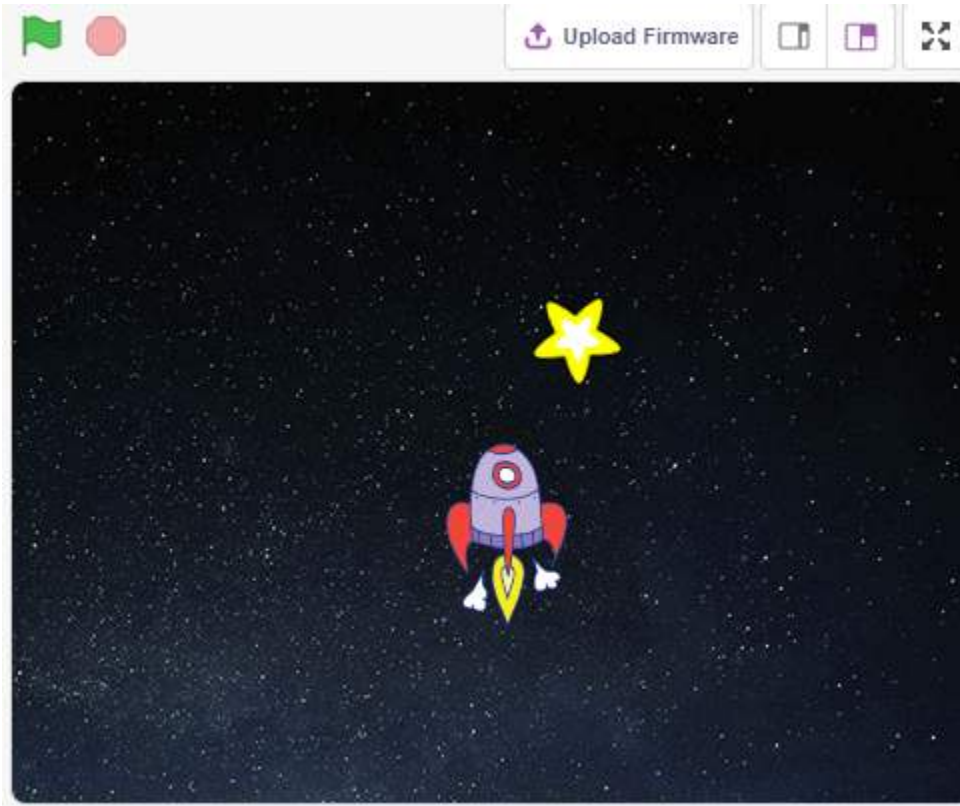


3.19 2.16 GAME - Star-Crossed

In the next projects, we will play some fun mini-games in PictoBlox.

Here we use Joystick module to play a Star-Crossed game.

After the script is run, stars will appear randomly on the stage, you need to use Joystick to control Rocketship to avoid the stars, if you touch it, the game will be over.



3.19.1 You Will Learn

- How Joystick module works
- Set the x and y coordinates of the sprite

3.19.2 Build the Circuit

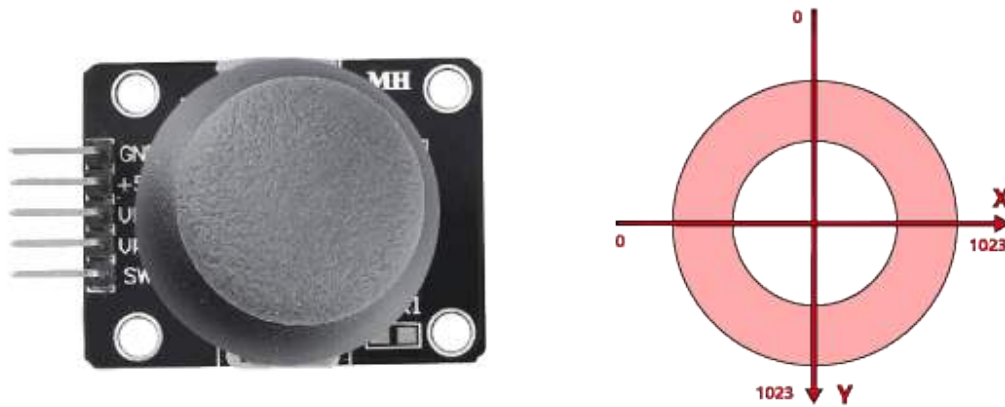
A joystick is an input device consisting of a stick that pivots on a base and reports its angle or direction to the device it is controlling. Joysticks are often used to control video games and robots.

In order to communicate a full range of motion to the computer, a joystick needs to measure the stick's position on two axes – the X-axis (left to right) and the Y-axis (up and down).

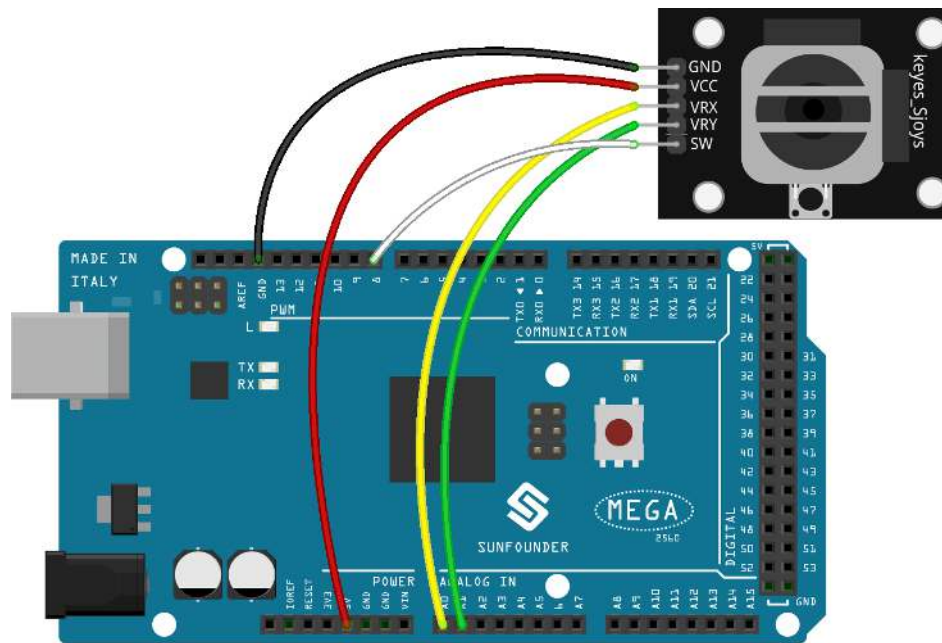
The motion coordinates of the joystick are shown in the following figure.

Note:

- The x coordinate is from left to right, the range is 0-1023.
 - y coordinate is from top to bottom, range is 0-1023.
-



Now build the circuit according to the following diagram.



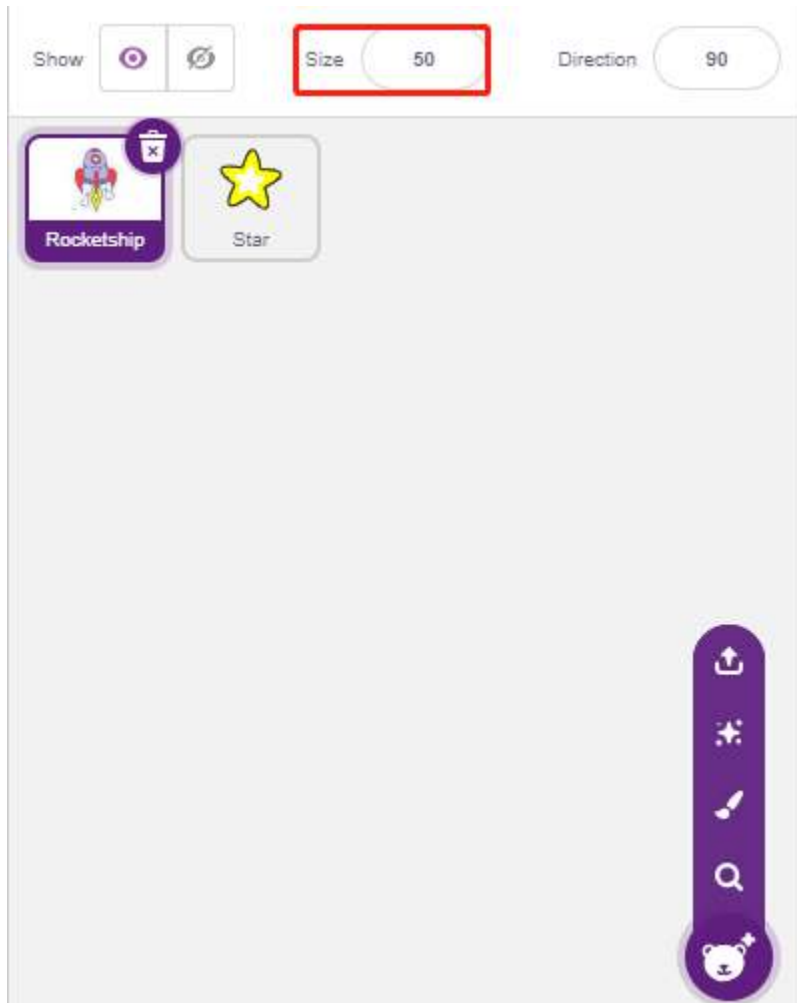
- Breadboard
- Joystick Module

3.19.3 Programming

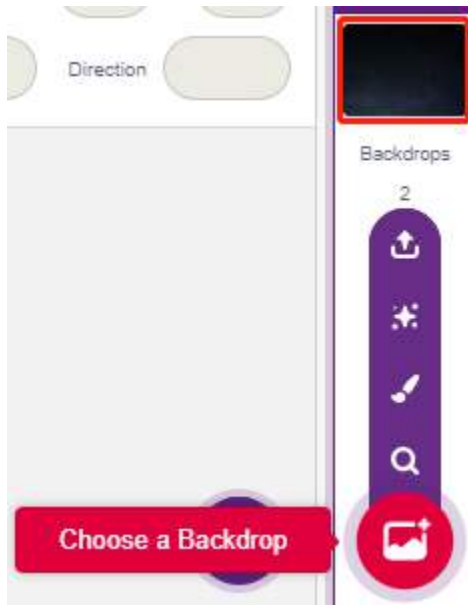
The whole script is to achieve the effect that when the green flag is clicked, the **Stars** sprite moves in a curve on the stage and you need to use the joystick to move the **Rocketship**, so that it will not be touched by the **Star** sprite.

1. Add sprites and backdrops

Delete the default sprite, and use the **Choose a Sprite** button to add the **Rocketship** sprite and the **Star** sprite. Note that the **Rocket** sprite size is set to 50%.



Now add the **Stars** backdrop by **Choose a Backdrop**.

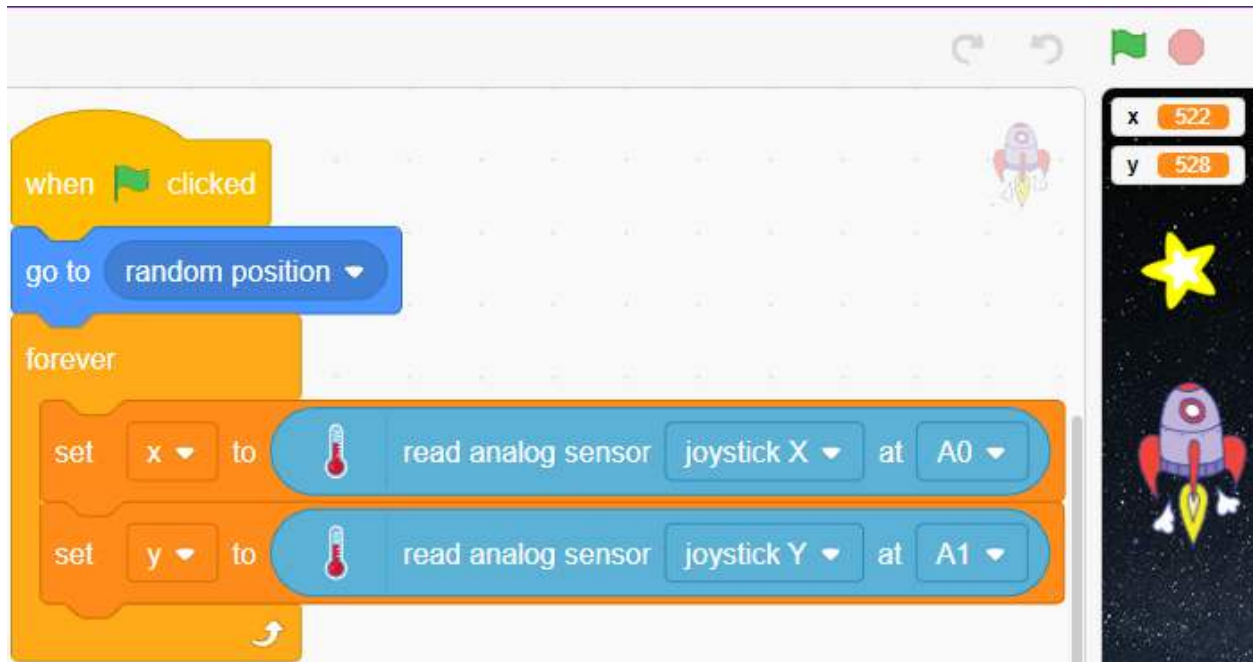


2. Scripting for Rocketship

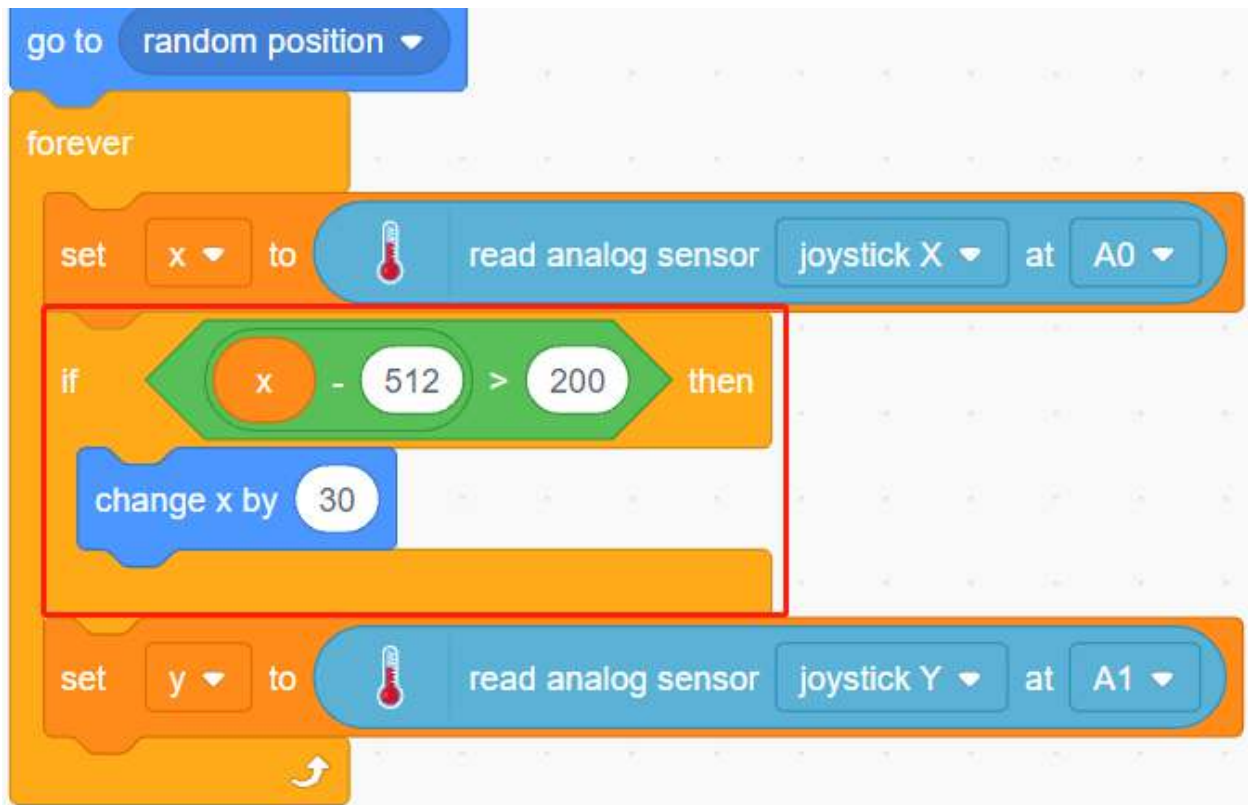
The **Rocketship** sprite is to achieve the effect that it will appear at a random position and then be controlled by the joystick to move it up, down, left, and right.

The workflow is as follows.

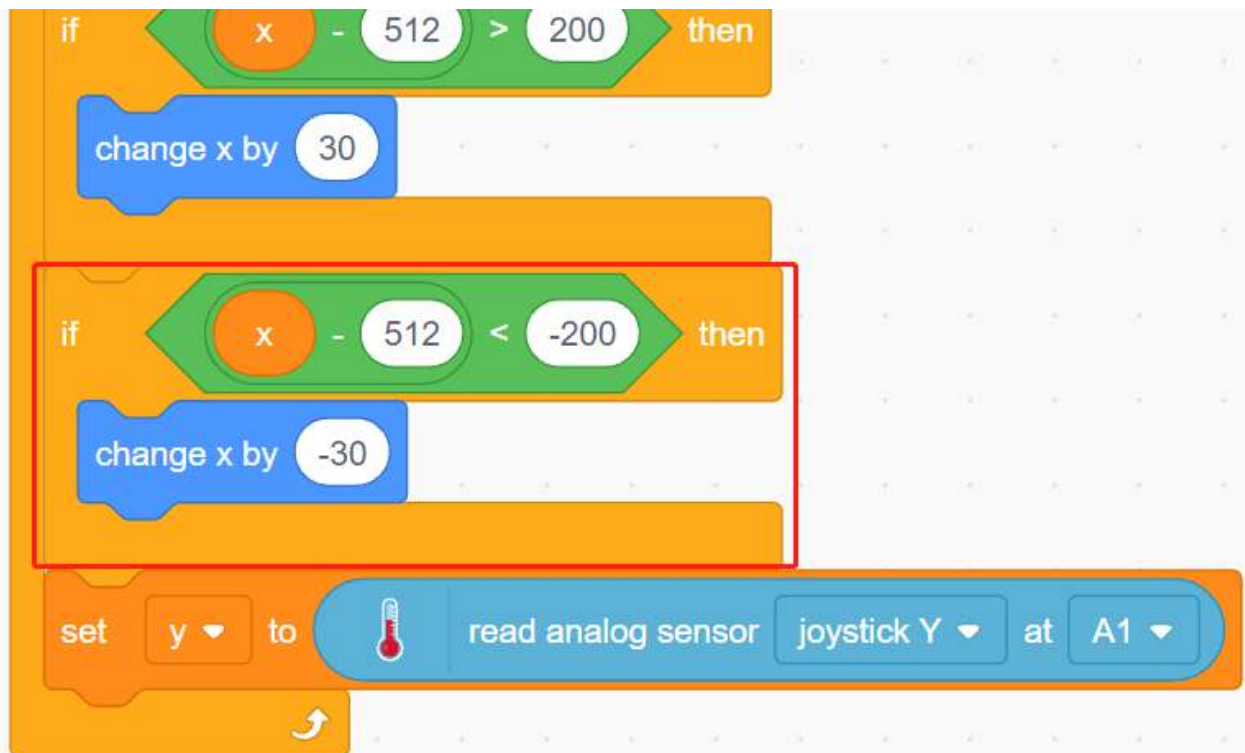
- When the green flag is clicked, have the sprite go to a random location and create 2 variables **x** and **y**, which store the values read from A0 (VRX of Joystick) and A1 (VRY of Joystick), respectively. You can let the script run, toggling the joystick up and down, left and right, to see the range of values for **x** and **y**.



- The value of A0 is in the range 0-1023 (the middle is about 512). Use $x-512 > 200$ to determine if Joystick is toggling to the right, and if so, make the x coordinate of the sprite +30 (to move the sprite to the right).

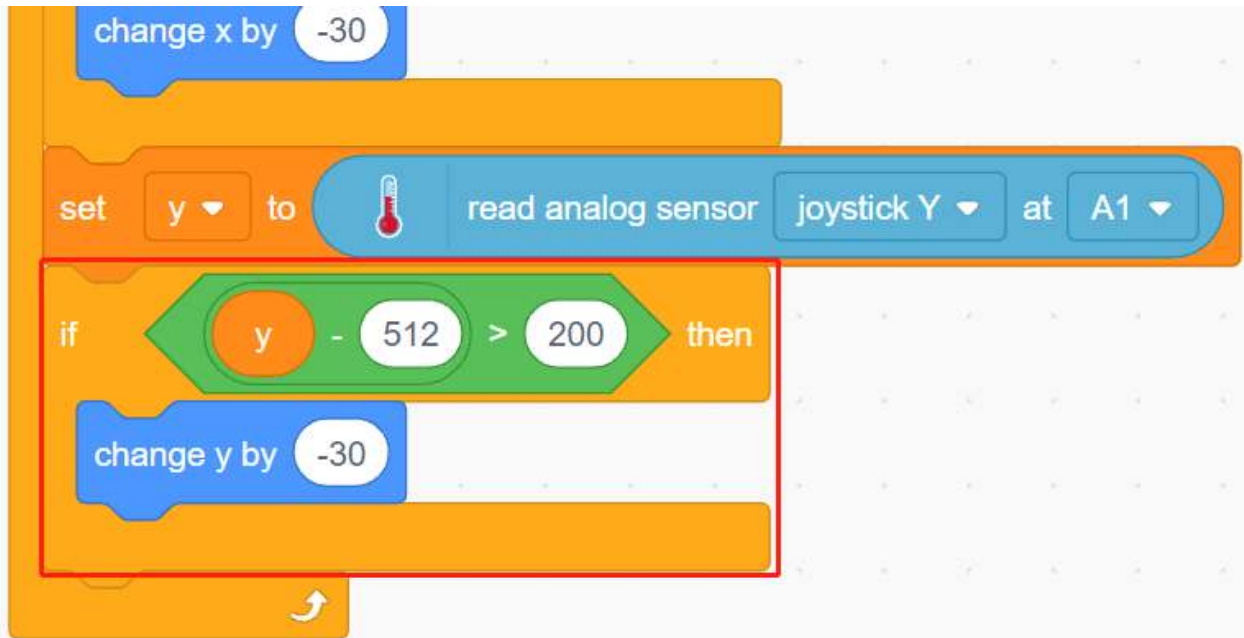


- If the Joystick is toggled to the left ($x - 512 < -200$), let the x coordinate of the sprite be -30 (let the sprite move to the left).

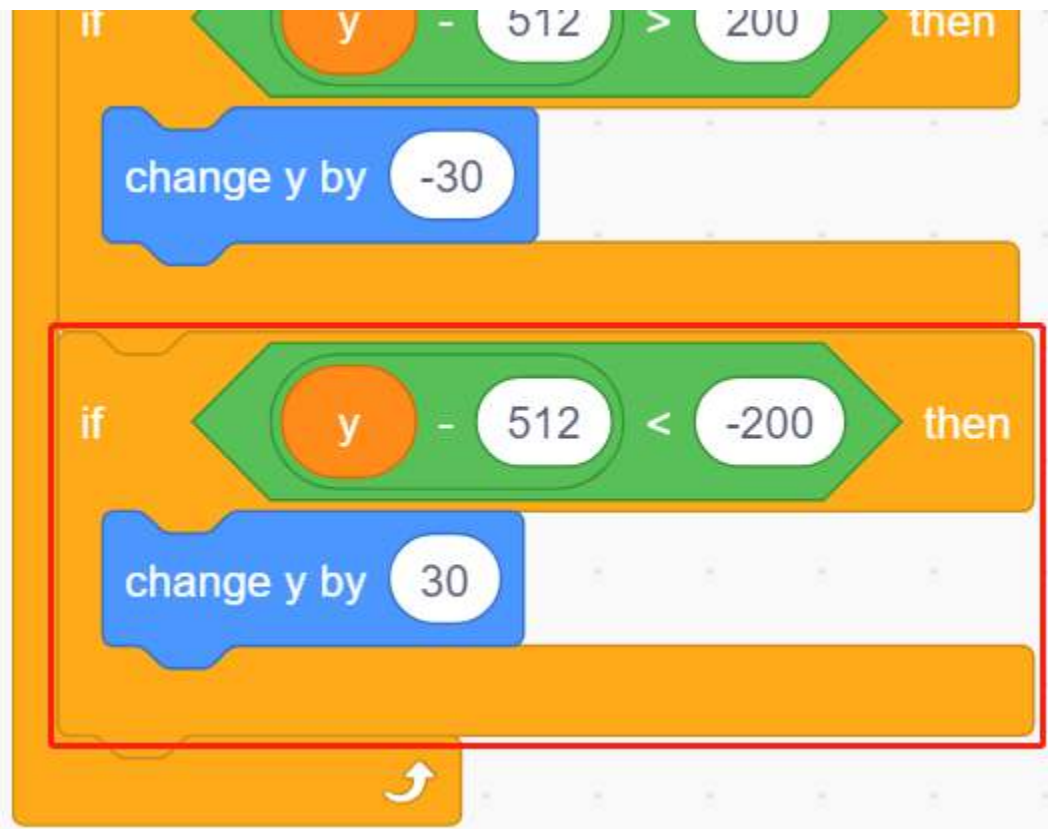


- Since the Joystick's y coordinate is from up (0) to down (1023), and the sprite's y coordinate is from down to

up. So in order to move the Joystick upwards and the sprite upwards, the y-coordinate must be -30 in the script.



- If the joystick is flicked down, the y-coordinate of the sprite is +30.



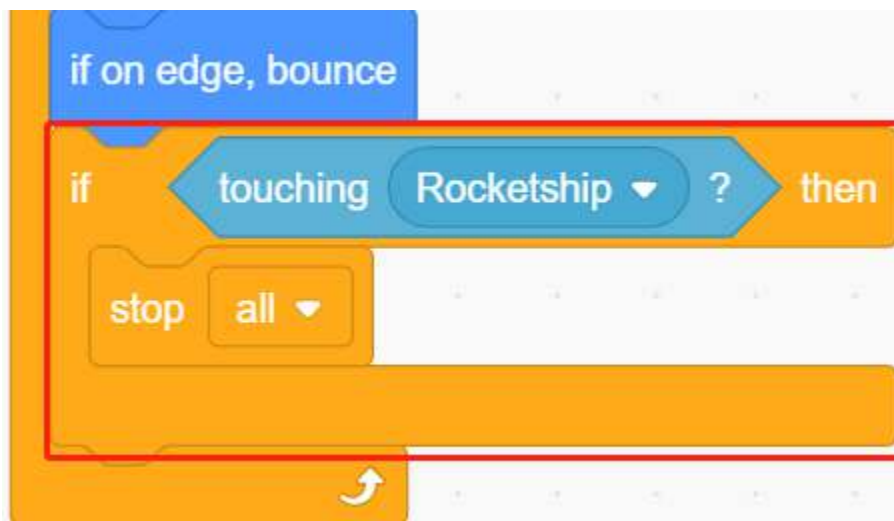
3. Scripting for Star

The effect to be achieved by the **Star** sprite is to appear at a random location, and if it hits **Rocketship**, the script stops running and the game ends.

- When the green flag is clicked and the sprite goes to a random location, the [turn degrees] block is to make the **Star** sprite move forward with a bit of an angle change so you can see that it is moving in a curve and if on edge, bounce.



- If the sprite touches the **Rocketship** sprite while it's moving, stop the script from running.



3.20 2.17 GAME - Eat Apple

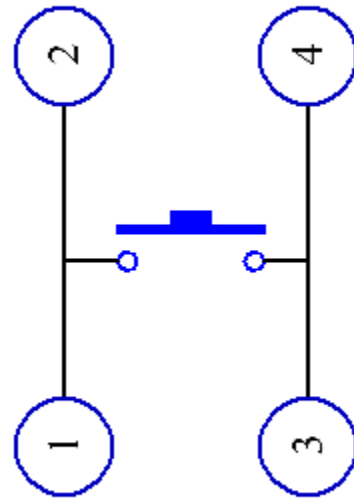
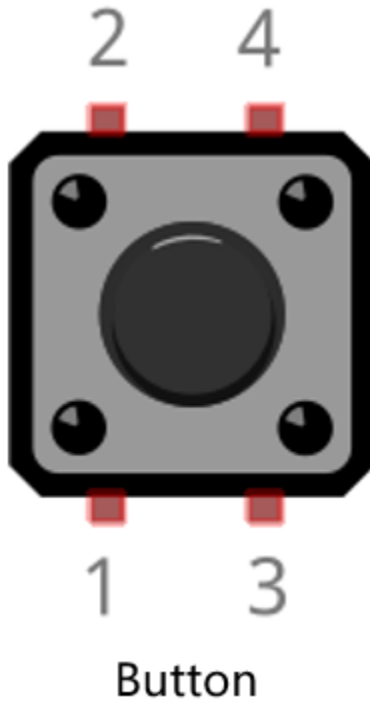
In this project, we play a game that uses button to control Beetle to eat apple.

When the green flag is clicked, press the button and Beetle will rotate, press the button again and Beetle stops running and goes forward at that angle. You need to control the angle of Beetle so that it moves forward without touching the black line on the map until it eats the apple. If it touches the black line, the game is over.



3.20.1 Build the Circuit

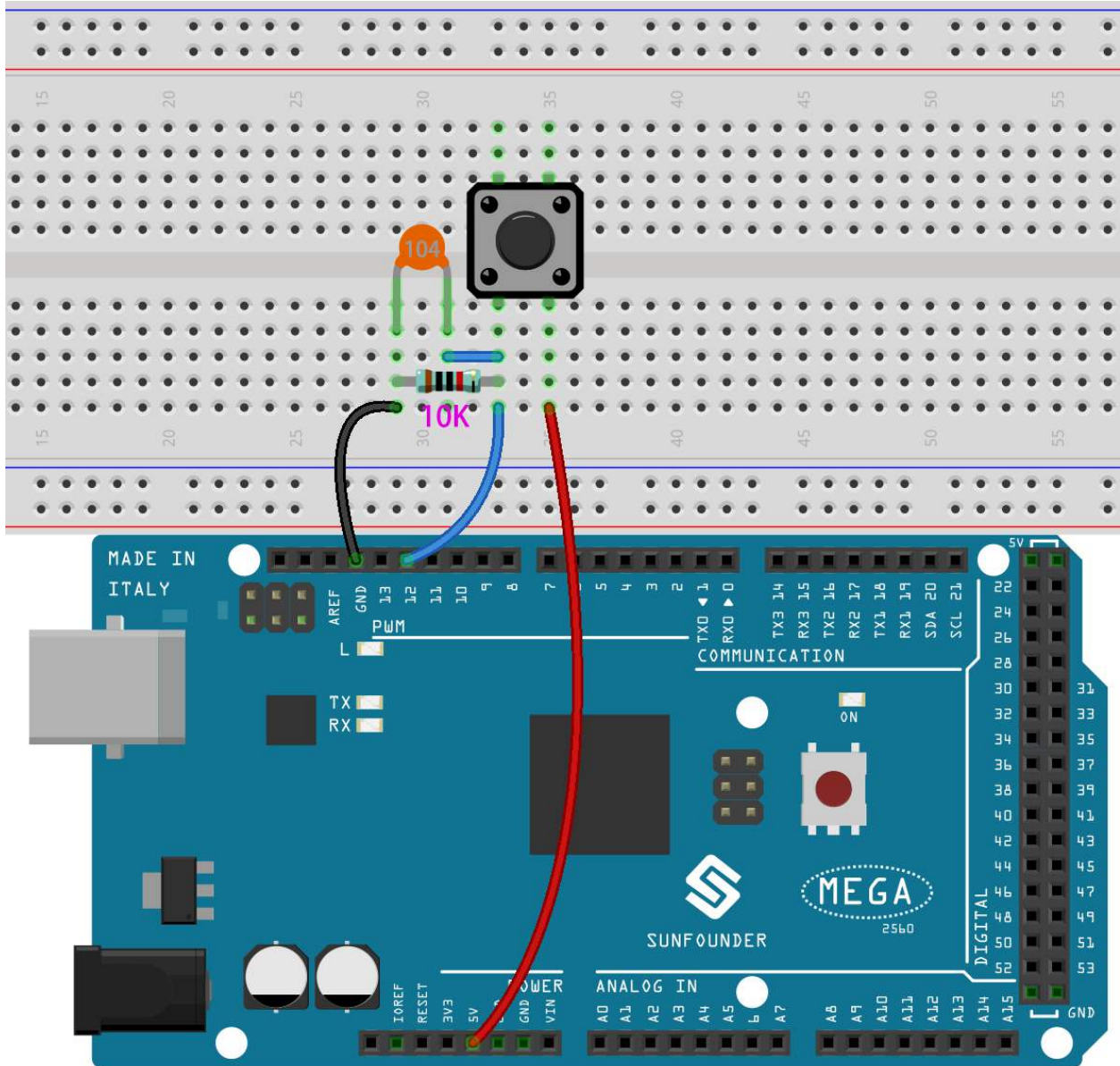
The button is a 4-pin device, since the pin 1 is connected to pin 2, and pin 3 to pin 4, when the button is pressed, the 4 pins are connected, thus closing the circuit.



Internal Structure

Build the circuit according to the following diagram.

- Connect one of the pins on the left side of the button to pin 12, which is connected to a pull-down resistor and a 0.1 μ F (104) capacitor (to eliminate jitter and output a stable level when the button is working).
- Connect the other end of the resistor and capacitor to GND, and one of the pins on the right side of the button to 5V.



- Breadboard
- Button
- Resistor
- Capacitor

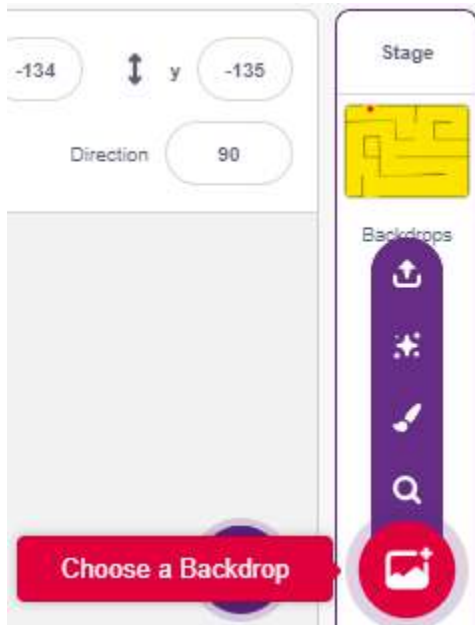
3.20.2 Programming

The effect we want to achieve is to use the button to control the direction of the **Beetle** sprite to move forward and eat the apple without touching the black line on the **Maze** backdrop, which will switch the backdrop when eaten.

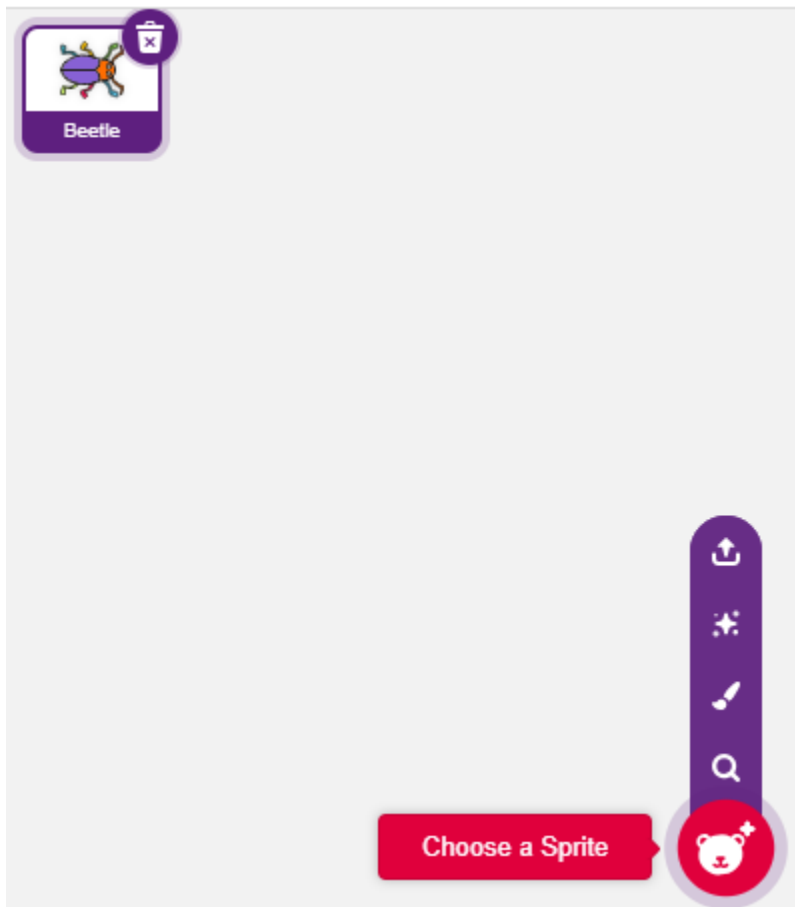
Now add the relevant backdrops and sprites.

1. Adding backdrops and sprites

Add a **Maze** backdrop via the **Choose a backdrop** button.



Delete the default sprite, then select the **Beetle** sprite.



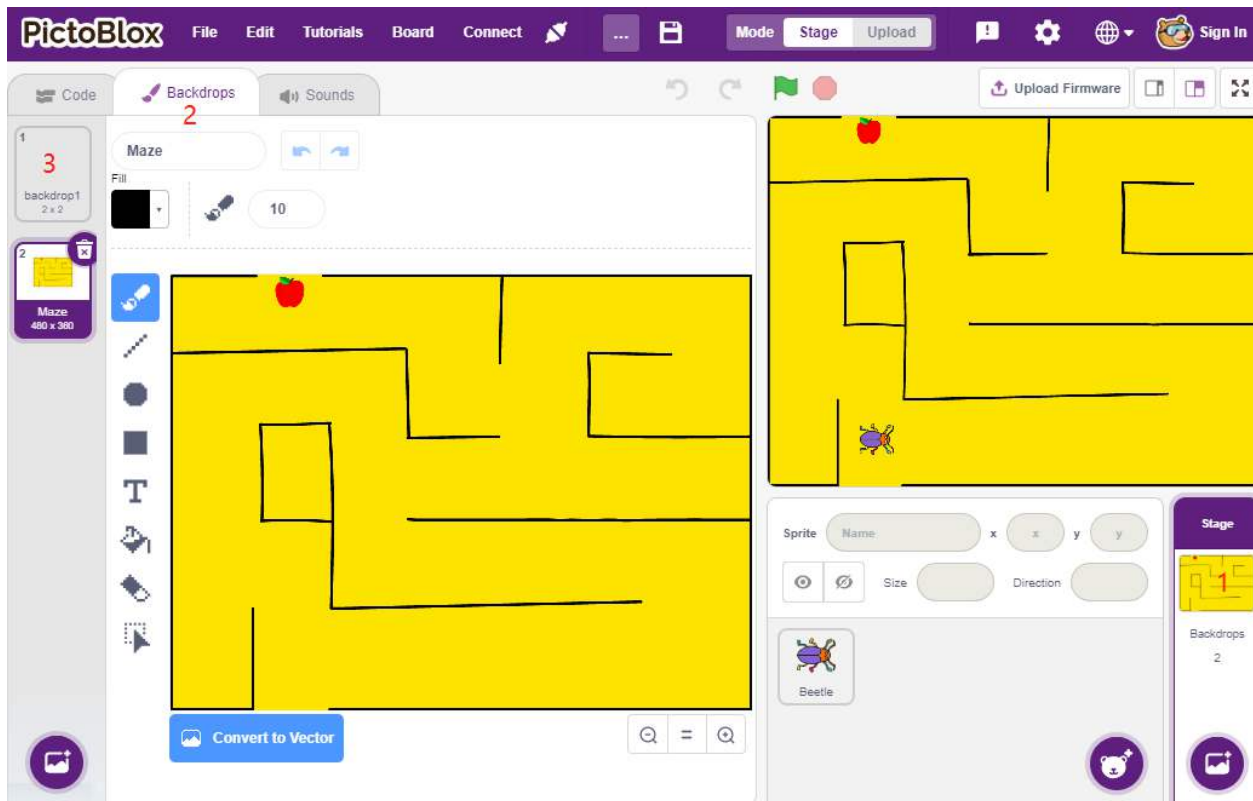
Place the **Beetle** sprite at the entrance of the **Maze** backdrop, remembering the x,y coordinate values at this point, and resize the sprite to 40%.



2. Draw a backdrop

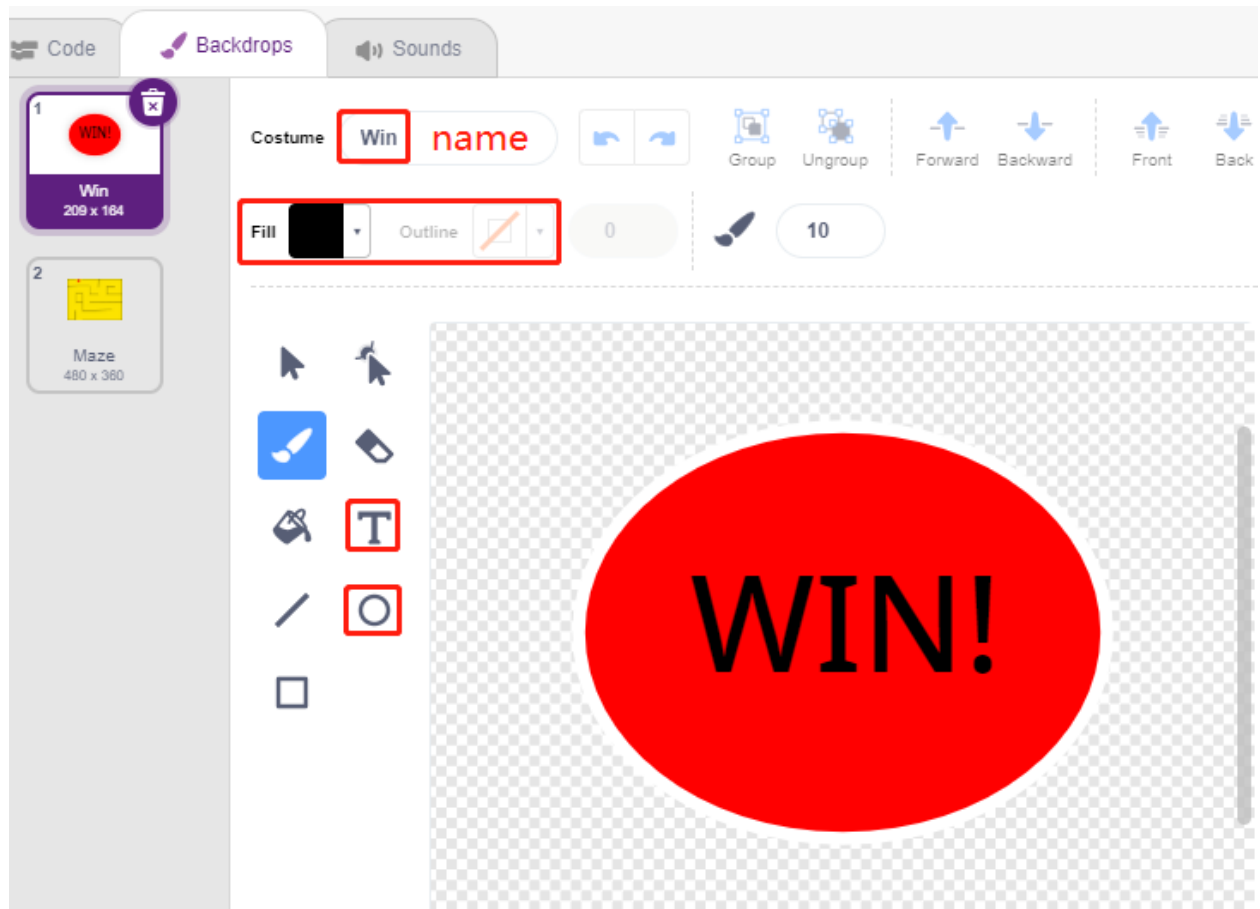
Now it's time to simply draw a backdrop with the WIN! character appearing on it.

First click on the backdrop thumbnail to go to the **Backdrops** page and click on the blank backdrop1.



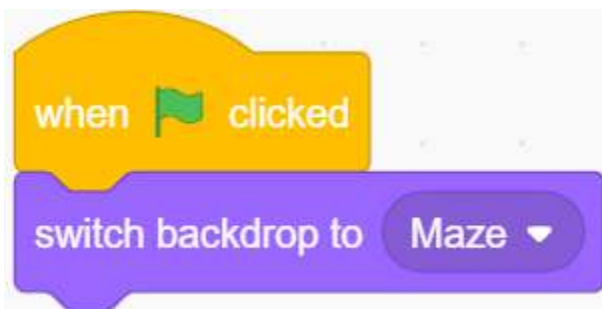
Now start drawing, you can refer to the picture below to draw, or you can draw a backdrop on your own, as long as the expression is winning.

- Using the **Circle** tool, draw an ellipse with the color set to red and no outline.
- Then use the **Text** tool, write the character “WIN!”, set the character color to black, and adjust the size and position of the character.
- Name the backdrop as **Win**.



3. Scripting for the backdrop

The backdrop needs to be switched to **Maze** every time the game starts.



4. Writing scripts for the sprite Beetle

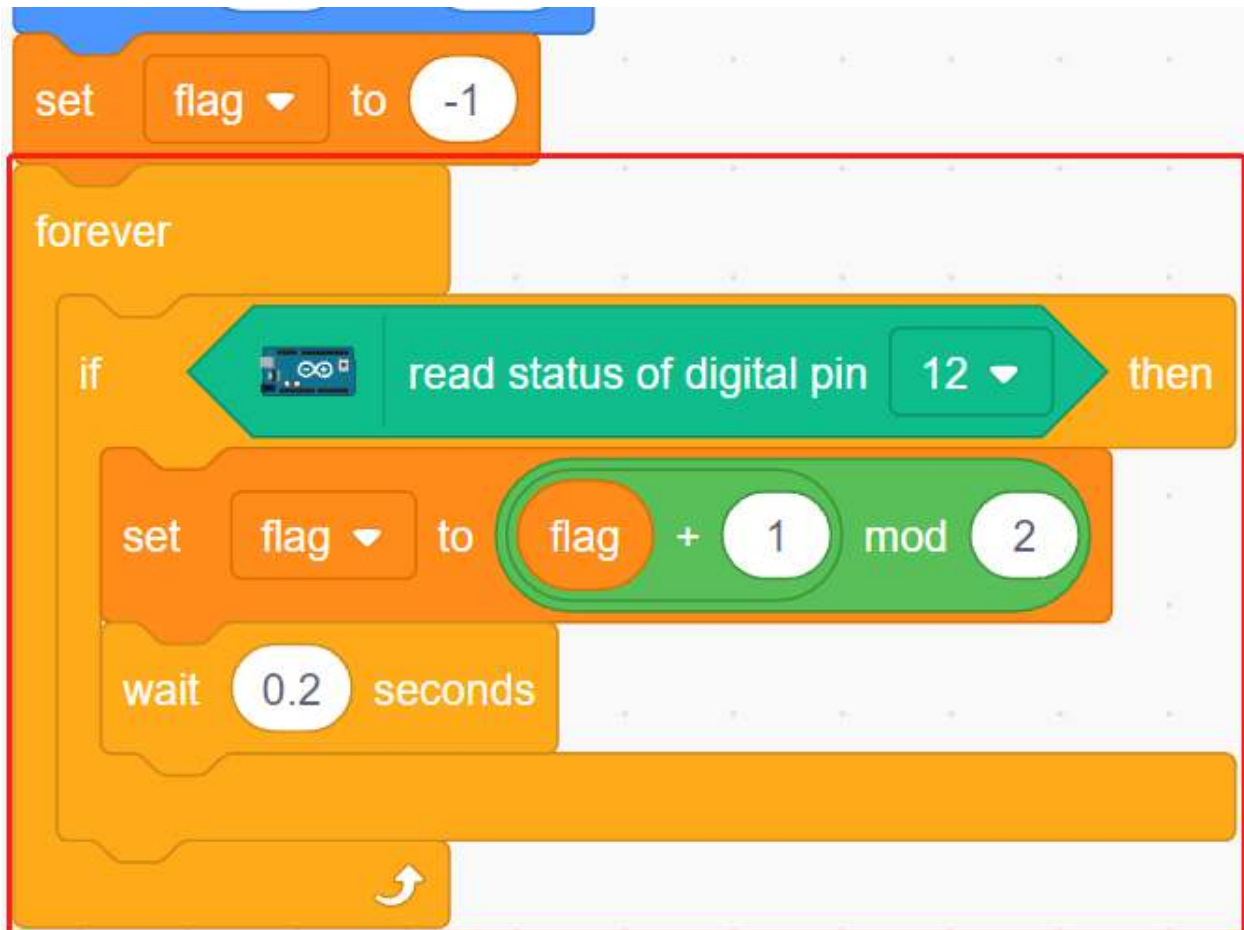
Now write a script for the sprite **Beetle** to be able to move forward and turn direction under the control of a button. The workflow is as follows.

- When the green flag is clicked, set the **Beetle** angle to 90, and the position to (-134, -134), or replace it with the coordinate value of your own placed position. Create the variable **flag** and set the initial value to -1.

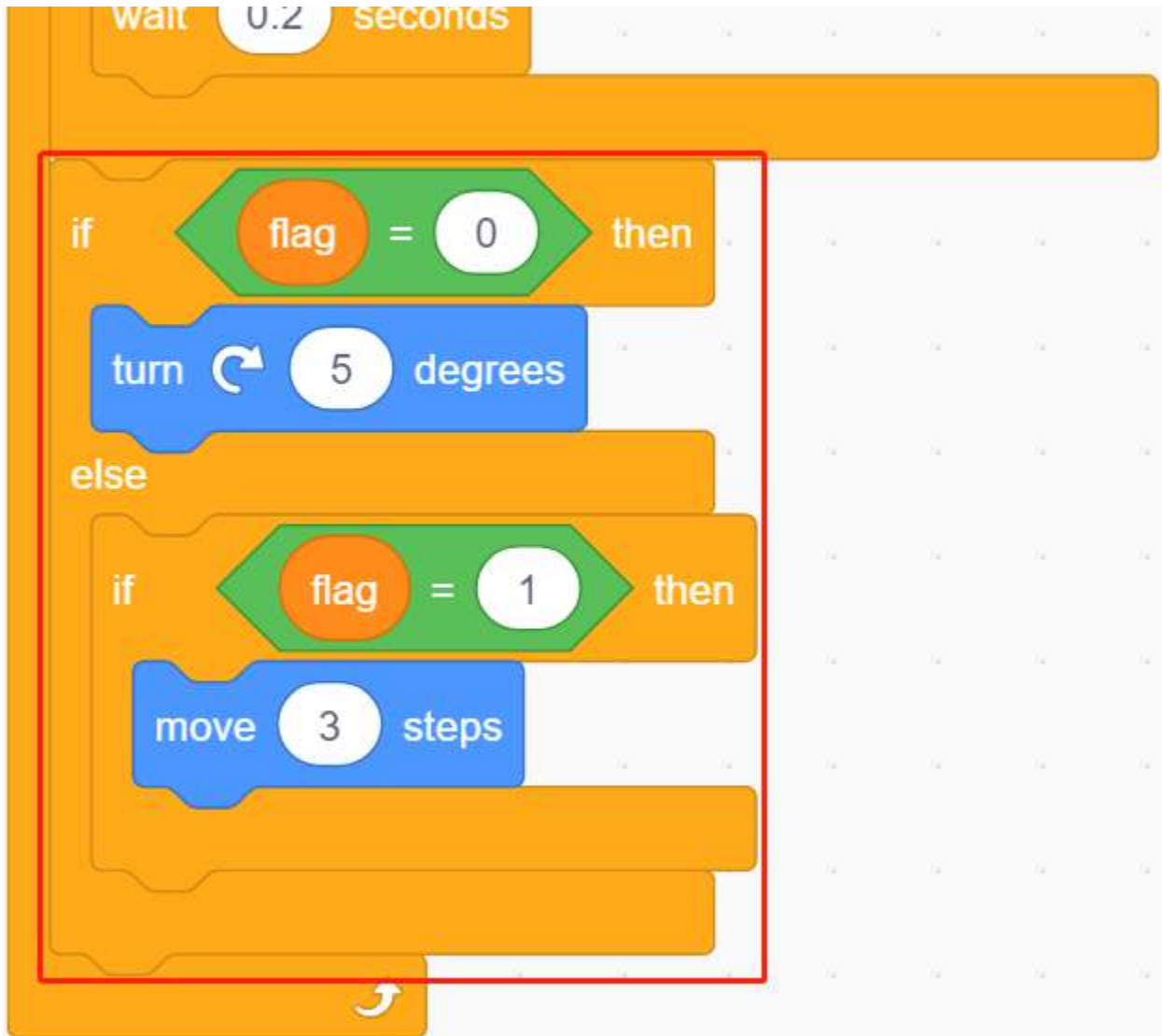


Next, in the [forever] block, four [if] blocks are used to determine various possible scenarios.

- If the key is 1 (pressed), use the [mod] block to toggle the value of the variable **flag** between 0 and 1 (alternating between 0 for this press and 1 for the next press).

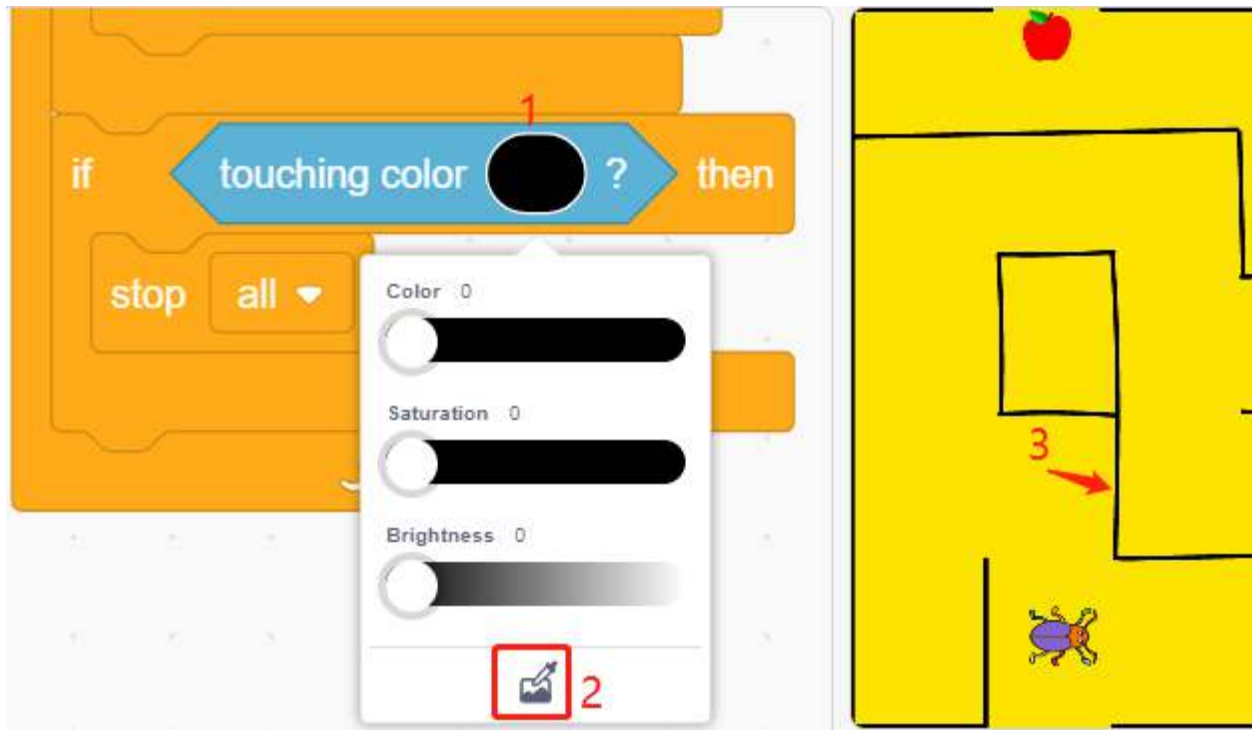


- If flag=0 (this key press), let the **Beetle** sprite turn clockwise. Then determine if flag is equal to 1 (key pressed again), the **Beetle** sprite moves forward. Otherwise, it keeps turning clockwise.

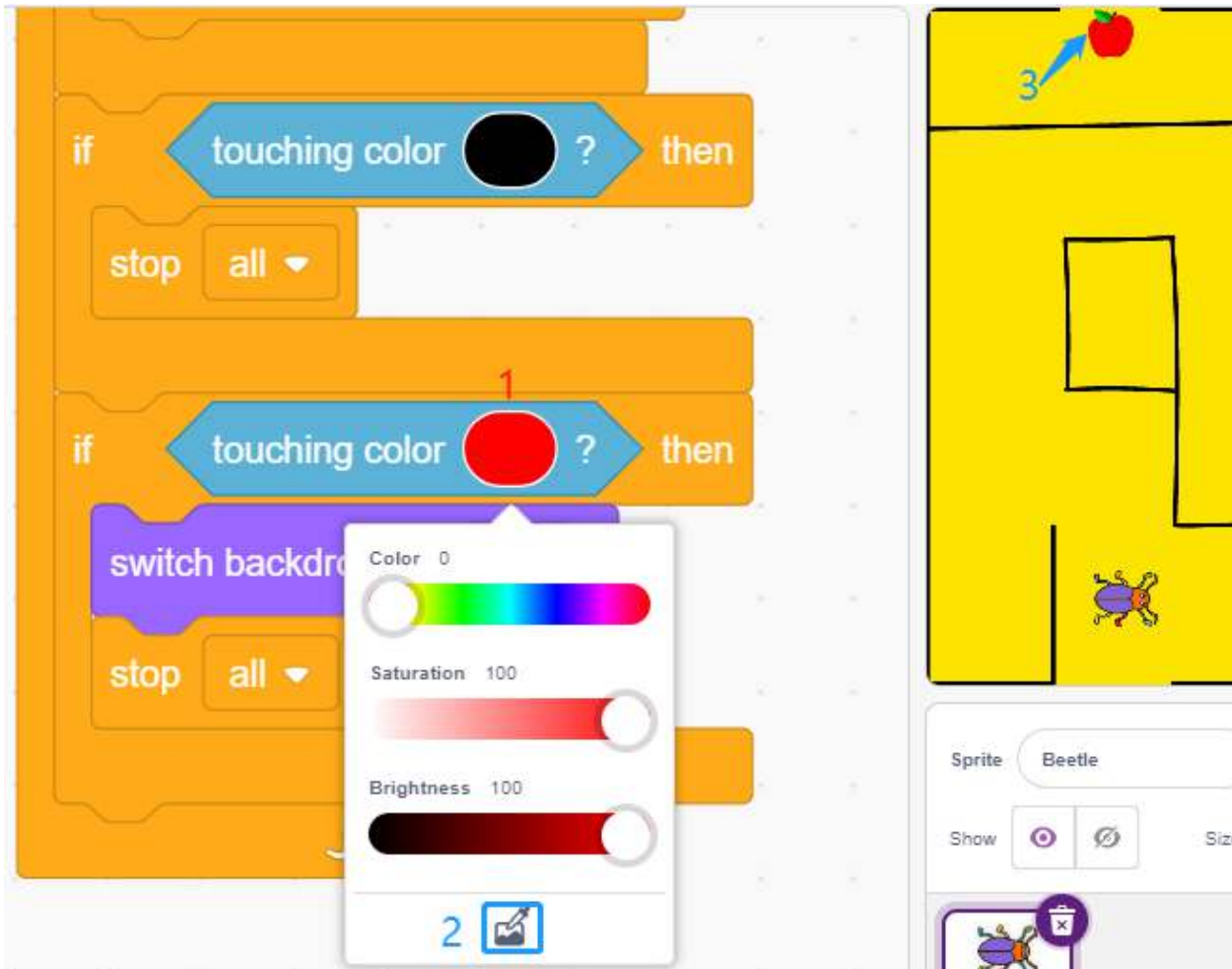


- If the Beetle sprite touches black (the black line on the **Maze** backdrop), the game ends and the script stops running.

Note: You need to click on the color area in the [Touch color] block, and then select the eyedropper tool to pick up the color of the black line on the stage. If you choose a black arbitrarily, this [Touch color] block will not work.



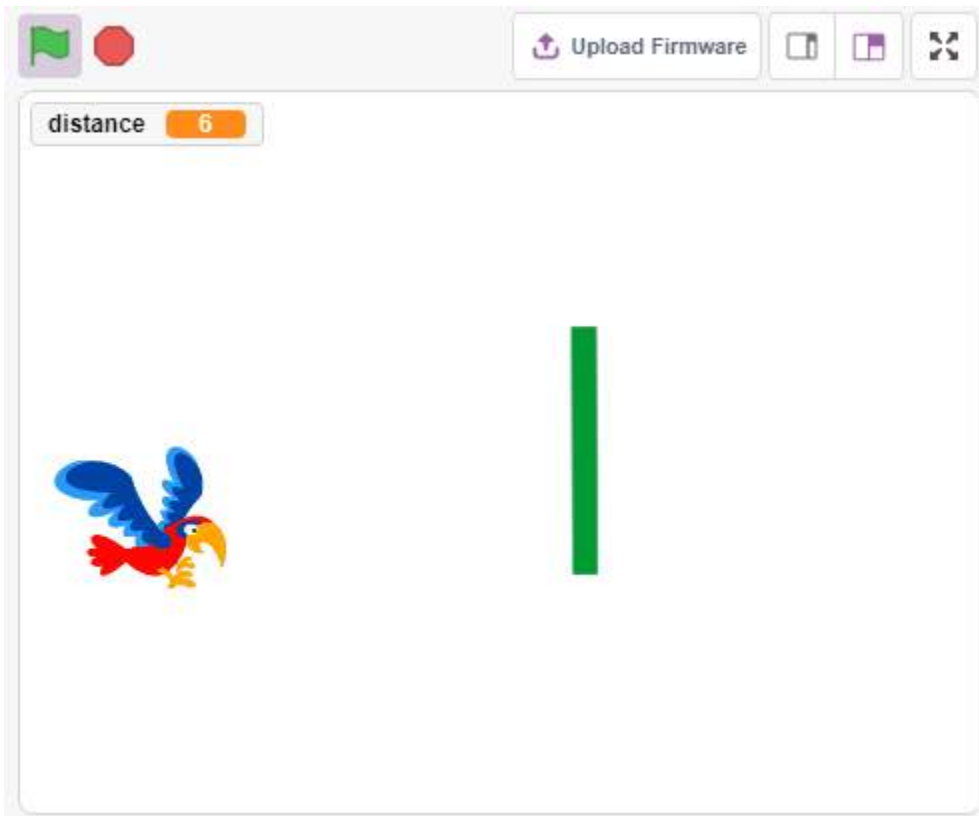
- If Beetle touches red (Also use the straw tool to pick up the red color of the apple), the backdrop will be switched to **Win**, which means the game succeeds and stops the script from running.



3.21 2.18 GAME - Flappy Parrot

Here we use the ultrasonic module to play a flappy parrot game.

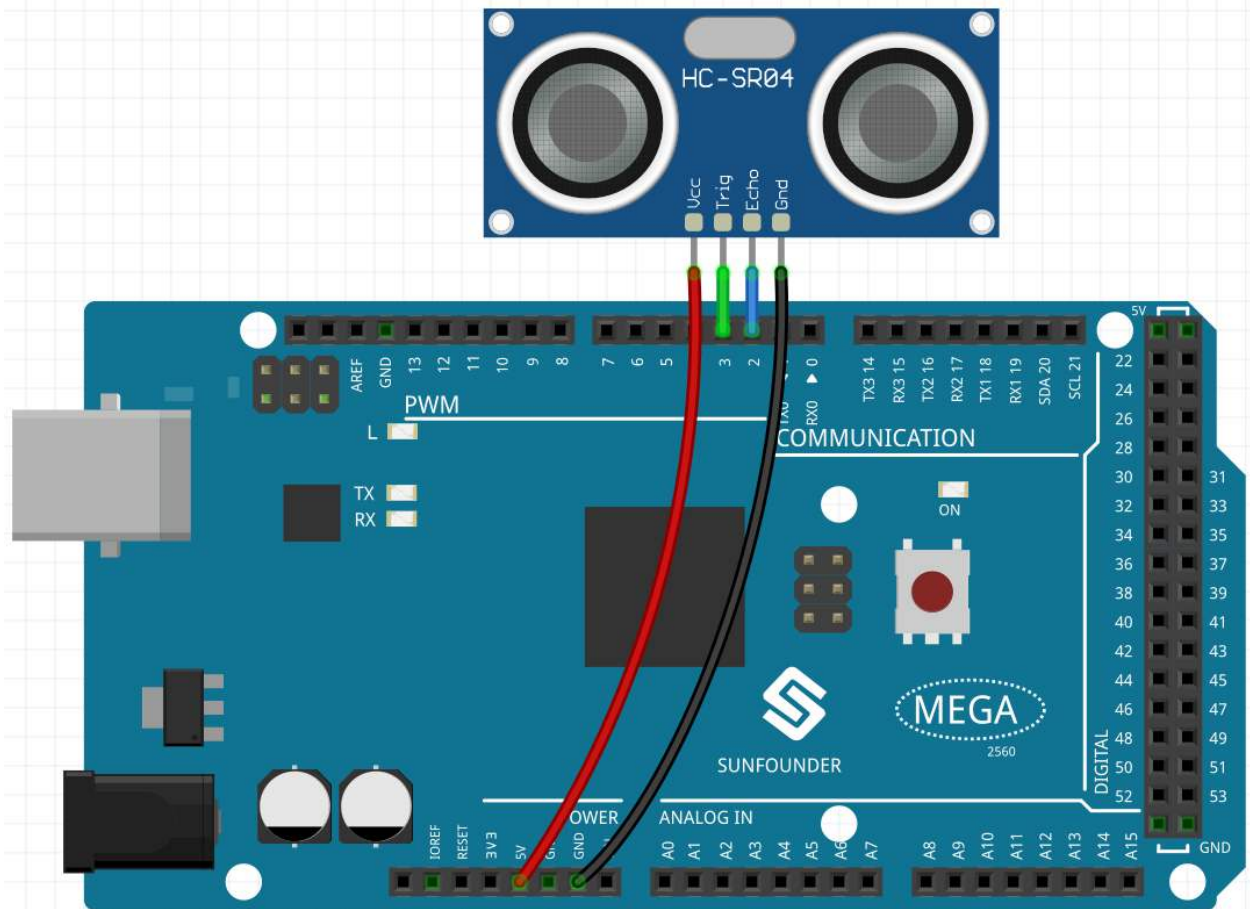
After the script runs, the green bamboo will slowly move from the right to the left at a random height. Now place your hand on top of the ultrasonic module, if the distance between your hand and the ultrasonic module is less than 10, the parrot will fly upwards, otherwise it will fall downwards. You need to control the distance between your hand and the ultrasonic module so that the Parrot can avoid the green bamboo (Paddle), if it touches it, the game is over.



3.21.1 Build the Circuit

An ultrasonic sensor module is an instrument that measures the distance to an object using ultrasonic sound waves. It has two probes. One is to send ultrasonic waves and the other is to receive the waves and transform the time of sending and receiving into a distance, thus detecting the distance between the device and an obstacle.

Now build the circuit according to the following diagram.



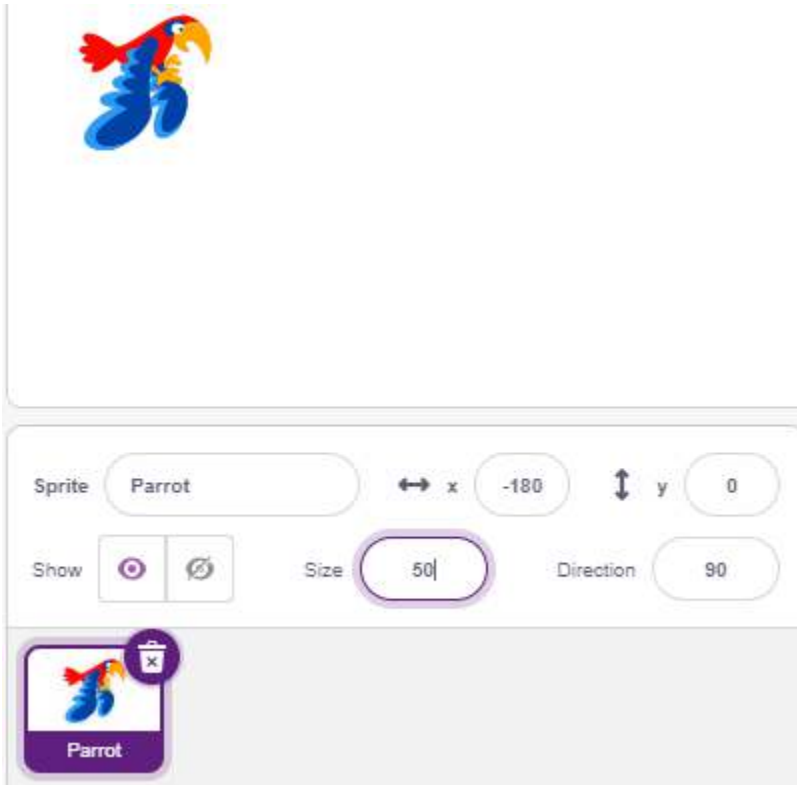
- *Breadboard*
- *Ultrasonic Module*

3.21.2 Programming

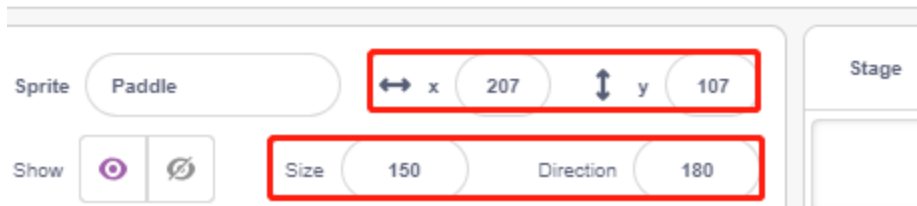
The effect we want to achieve is to use the ultrasonic module to control the flight height of the sprite **Parrot**, while avoiding the **Paddle** sprite.

1. Add a sprite

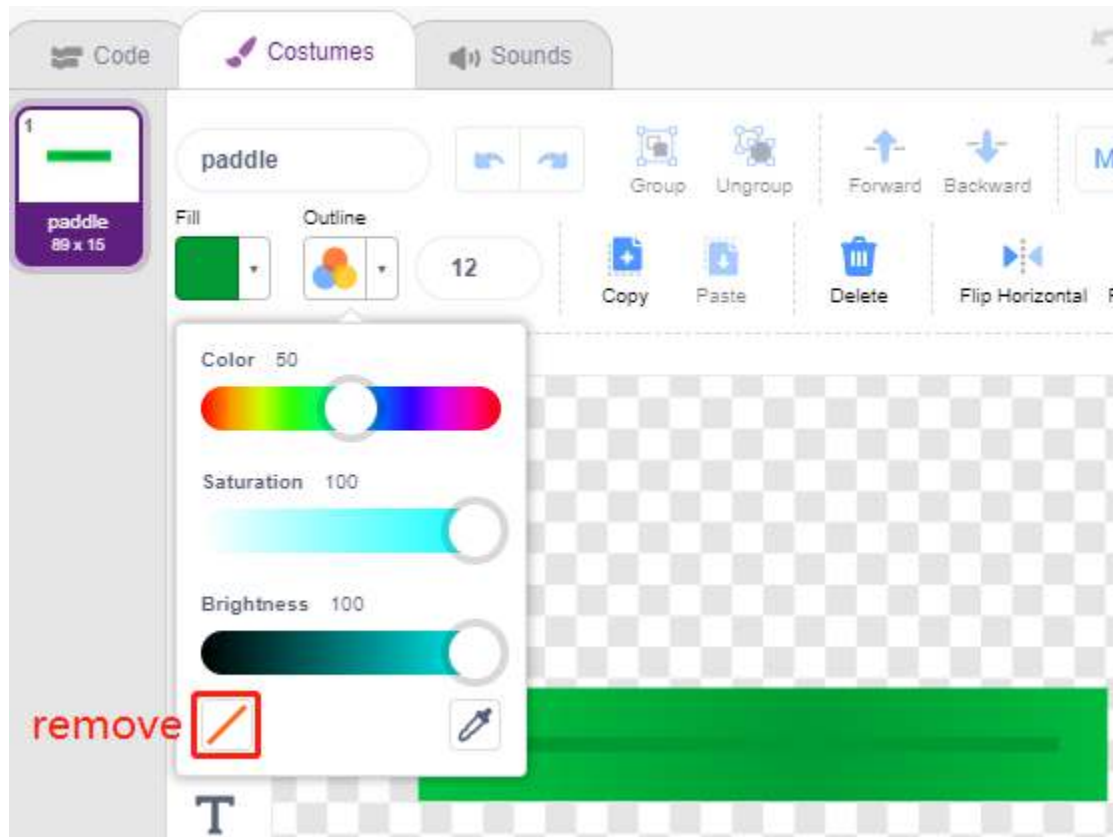
Delete the default sprite, and use the **Choose a Sprite** button to add the **Parrot** sprite. Set its size to 50%, and move its position to the left center.



Now add the **Paddle** sprite, set its size to 150%, set its angle to 180, and move its initial position to the top right corner.



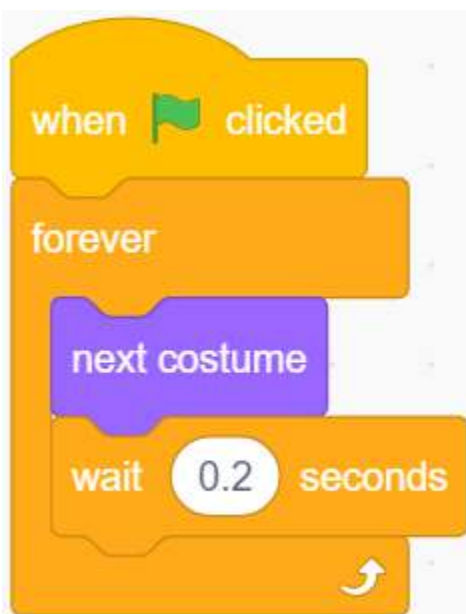
Go to the **Costumes** page of the **Paddle** sprite and remove the Outline.



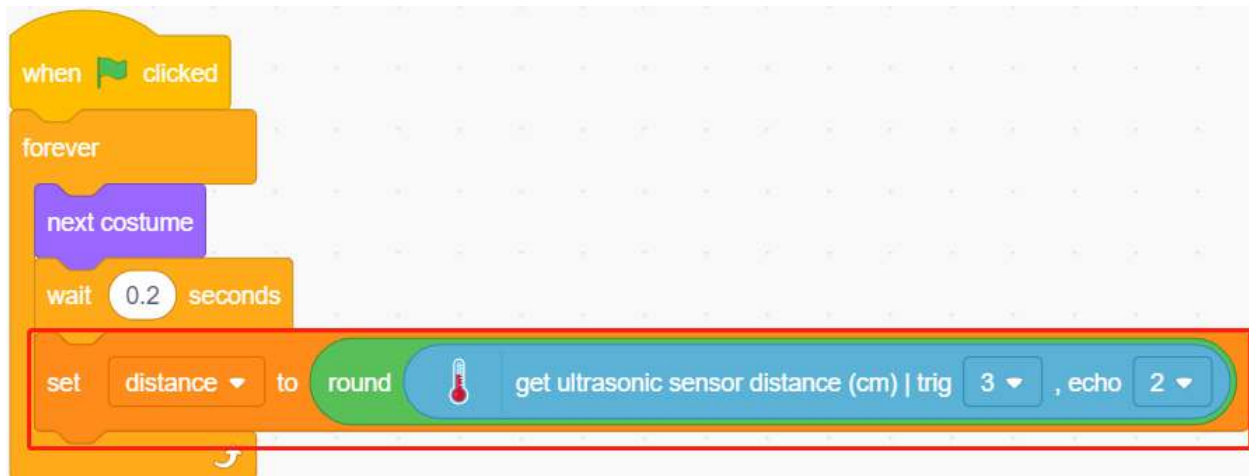
2. Scripting for the Parrot Sprite

Now script the **Parrot** sprite, which is in flight and the flight altitude is determined by the detection distance of the ultrasonic module.

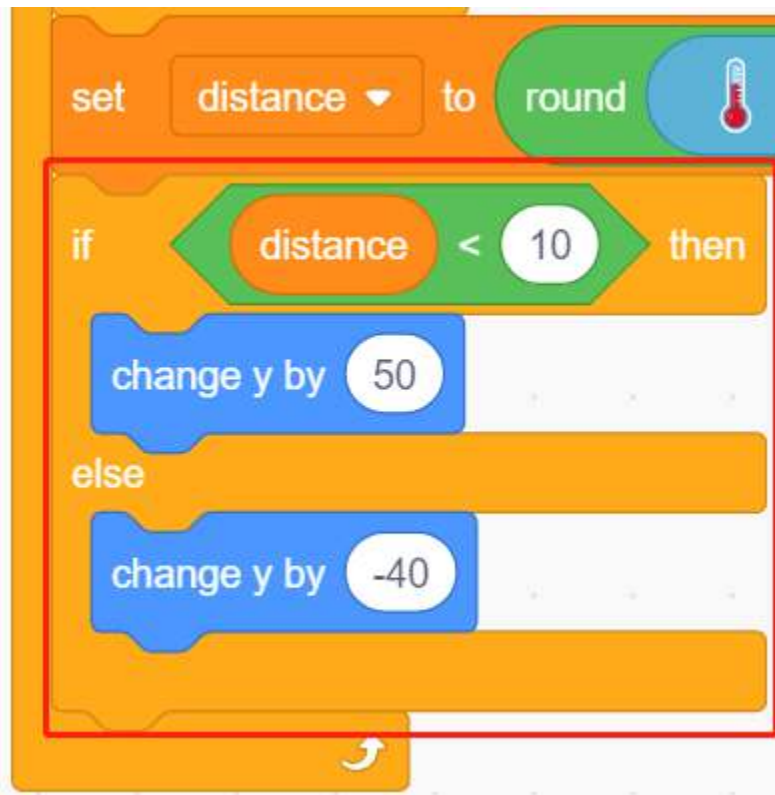
- When the green flag is clicked, switch the costume every 0.2s so that it is always in flight.



- Read the value of the ultrasonic module and store it in the variable **distance** after rounding it with the [round] block.



- If the ultrasonic detection distance is less than 10cm, let the y coordinate increase by 50, the **Parrot** sprite will fly upwards. Otherwise, the y-coordinate value is decreased by 40, **Parrot** will fall down.



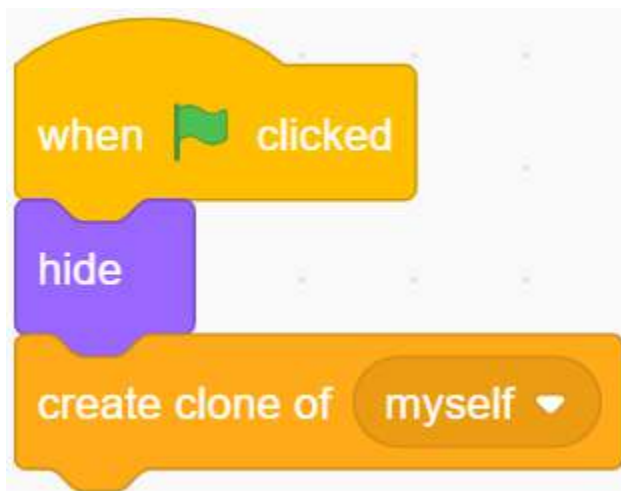
- If the **Parrot** sprite touches the **Paddle** sprite, the game ends and the script stops running.



3. Scripting for the Paddle sprite

Now write the script for the **Paddle** sprite, which needs to appear randomly on the stage.

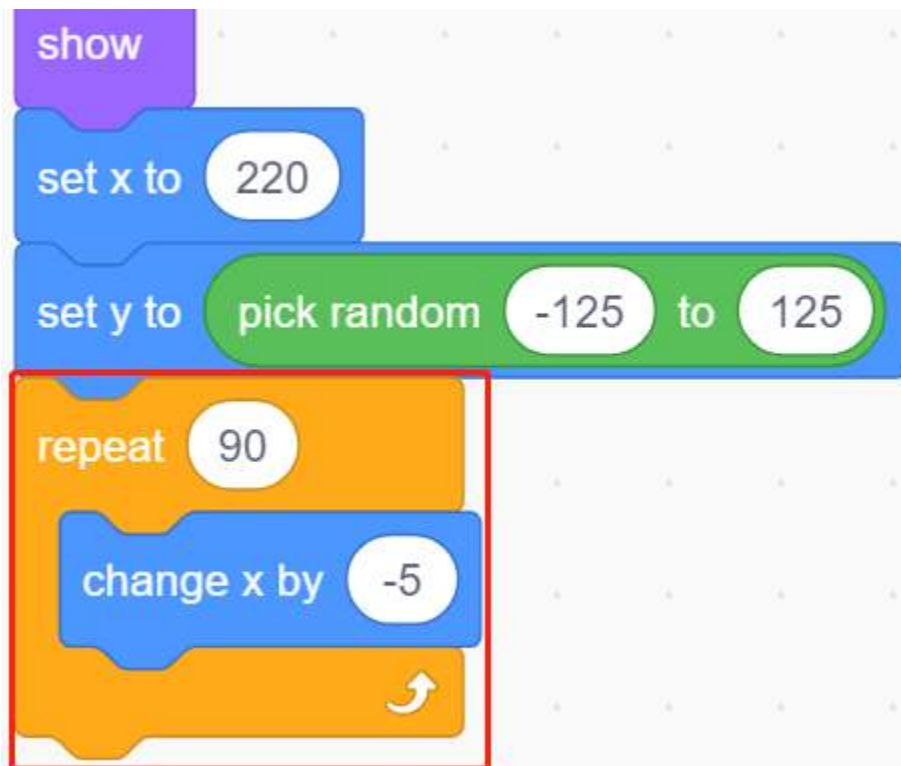
- Hide the sprite **Paddle** when the green flag is clicked, and clone itself at the same time. The [create clone of] block is a control block and a stack block. It creates a clone of the sprite in the argument. It can also clone the sprite it is running in, creating clones of clones, recursively.



- When **Paddle** is presented as a clone, its position is 220 (rightmost) for the x-coordinate and its y-coordinate at (-125 to 125) random (height random).



- Use the [repeat] block to make its x coordinate value slowly decrease, so you can see the clone of the **Paddle** sprite slowly move from the right to the left until it disappears.



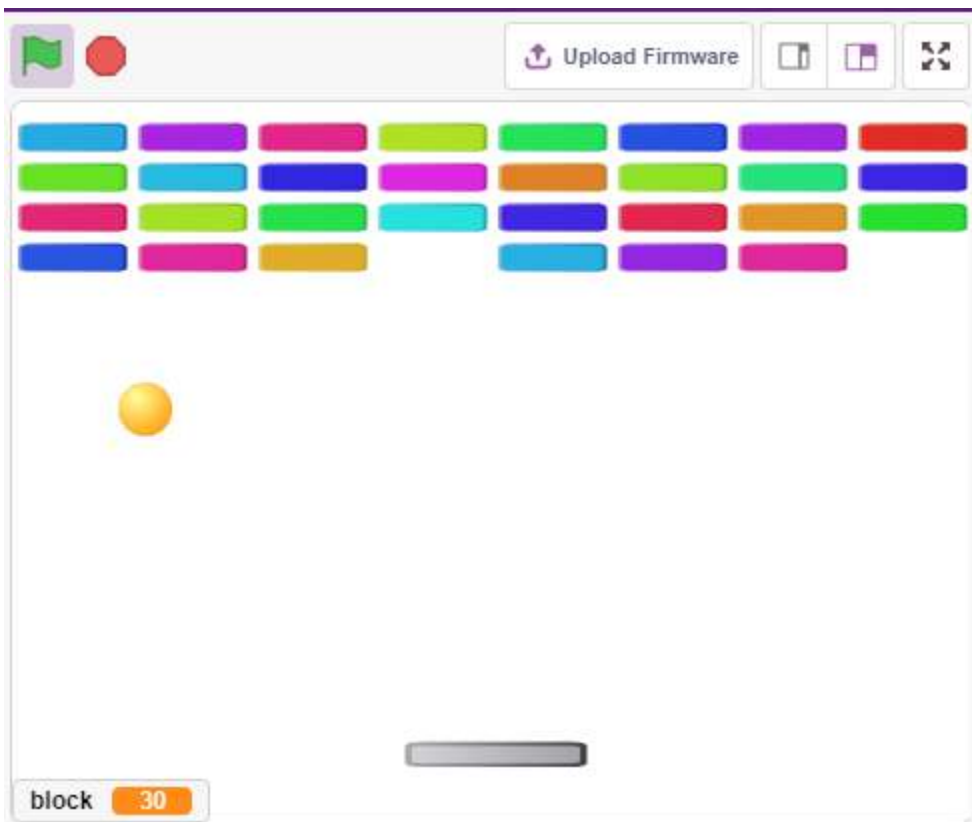
- Re-clone a new **Paddle** sprite and delete the previous clone.



3.22 2.19 GAME - Breakout Clone

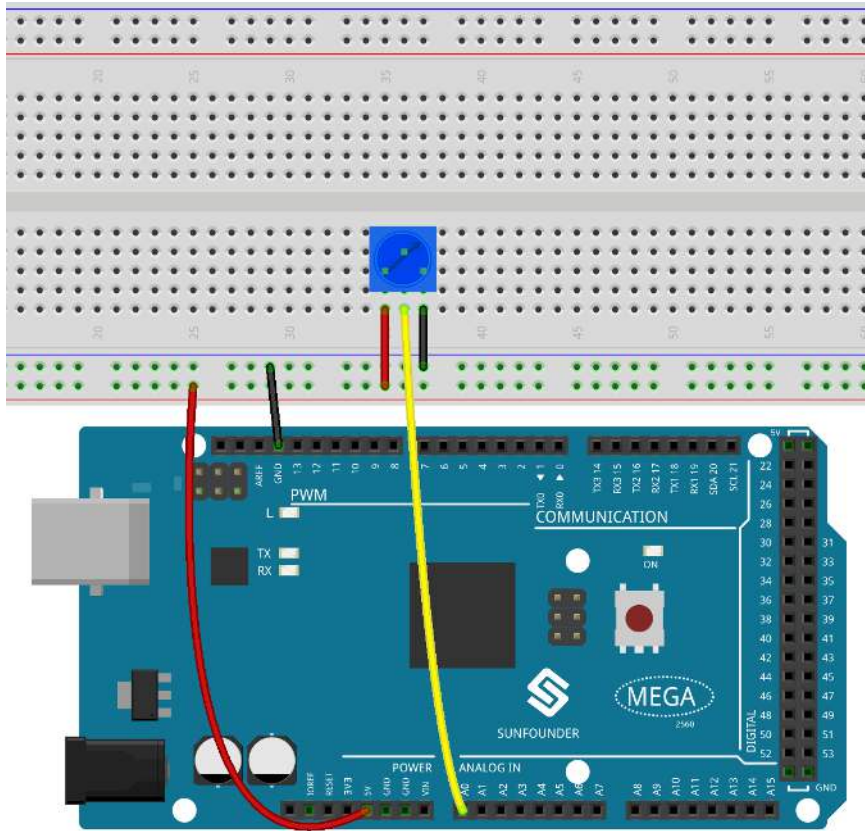
Here we use the potentiometer to play a Breakout Clone game.

After clicking the green flag, you need to use the potentiometer to control the paddle on the stage to catch the ball so that it can go up and hit the bricks, all the bricks disappear then the game is won, if you don't catch the ball, the game is lost.



3.22.1 Build the Circuit

The potentiometer is a resistive element with 3 terminals, the 2 side pins are connected to 5V and GND, and the middle pin is connected to A0. After the conversion by the ADC converter of the Arduino board, the value range is 0-1023.



- *Breadboard*
- *Potentiometer*

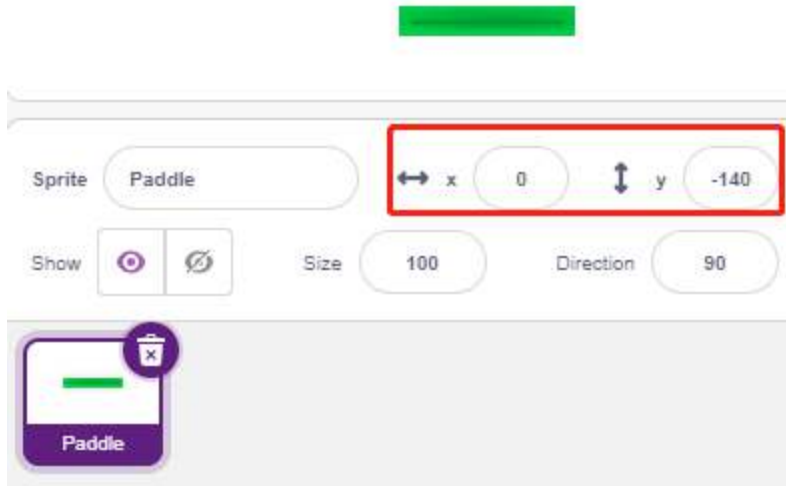
3.22.2 Programming

There are 3 sprites on the stage.

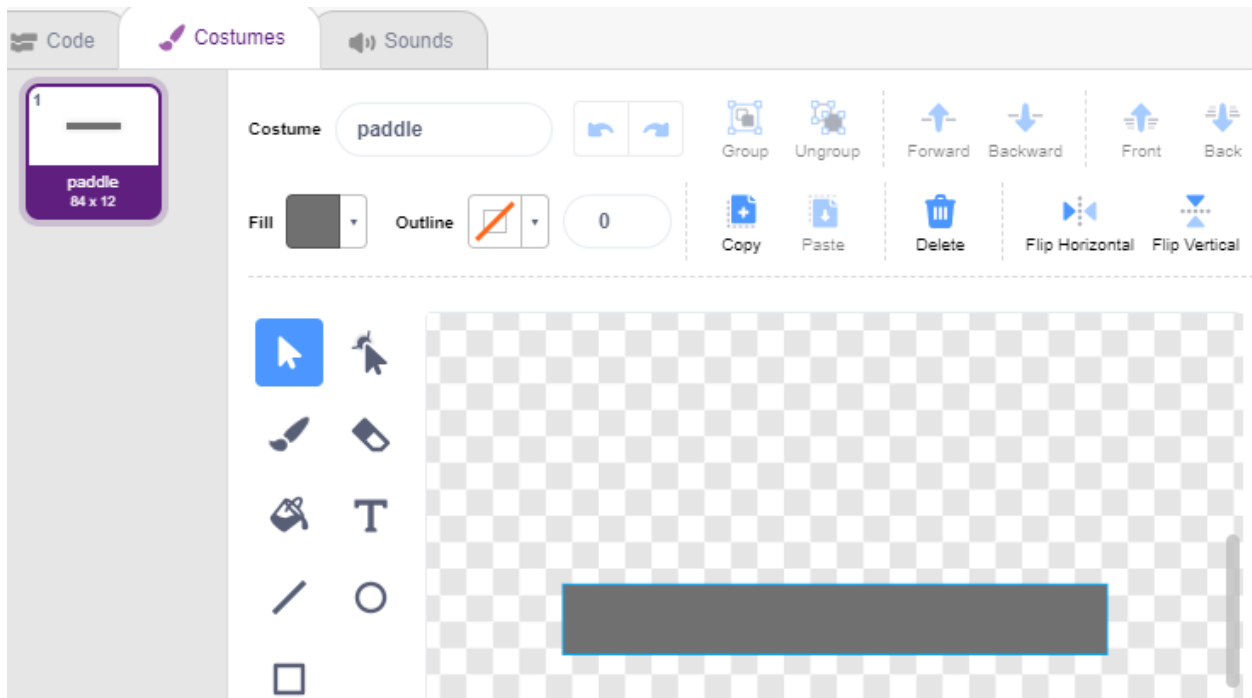
1. Paddle sprite

The effect to be achieved by the **Paddle** is that the initial position is in the middle of the bottom of the stage, and it is controlled by a potentiometer to move it to the left or to the right.

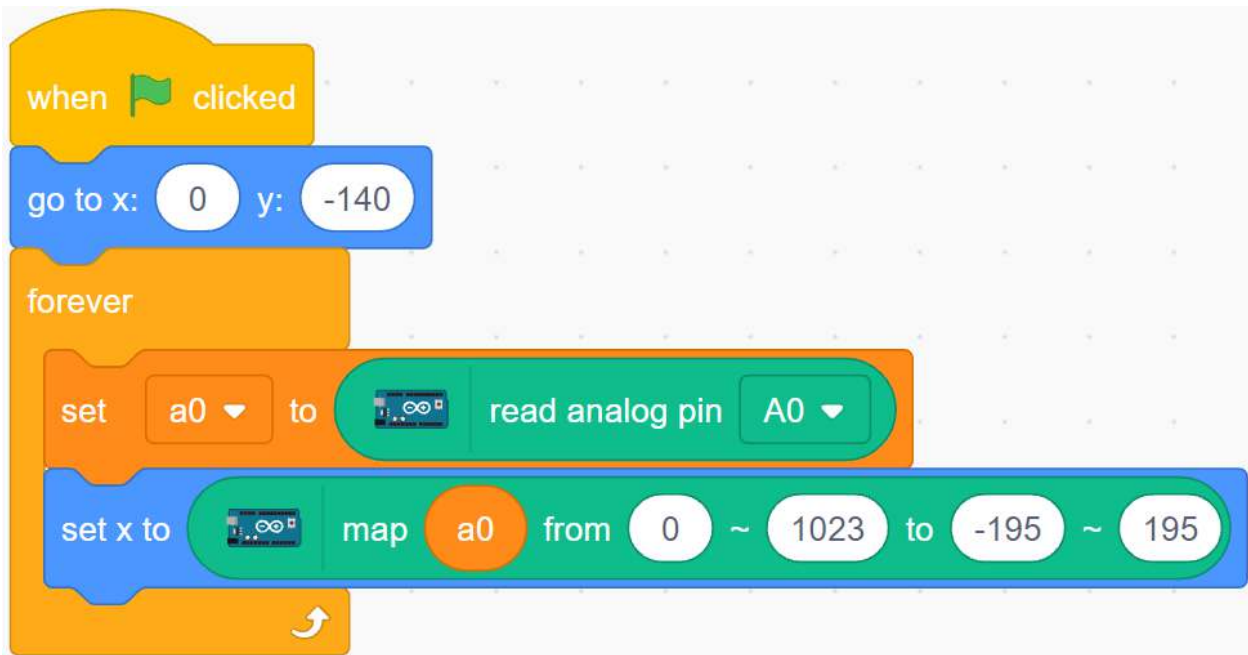
- Delete the default sprite, use the **Choose a Sprite** button to add the **Paddle** sprite, and set its x and y to (0, -140).



- Go to the **Costumes** page, remove the Outline and change its color to dark gray.



- Now script the **Paddle** sprite to set its initial position to (0, -140) when the green flag is clicked, and read the value of A0 (potentiometer) into the variable **a0**. Since the **Paddle** sprite moves from left to right on the stage at x-coordinates -195~195, you need to use the [map] block to map the variable **a0** range 0~1023 to -195~195.

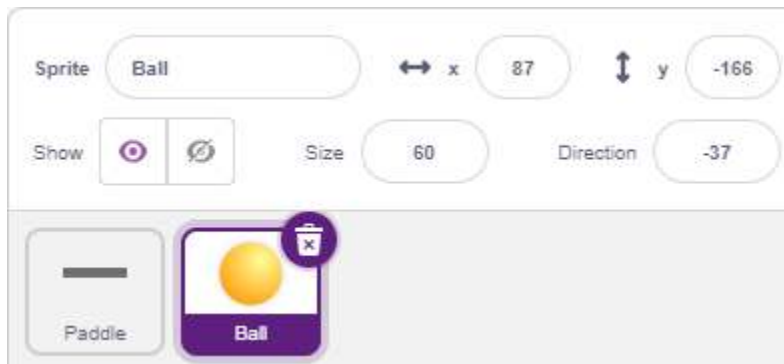


- Now you can rotate the potentiometer to see if the **Paddle** can move left and right on the stage.

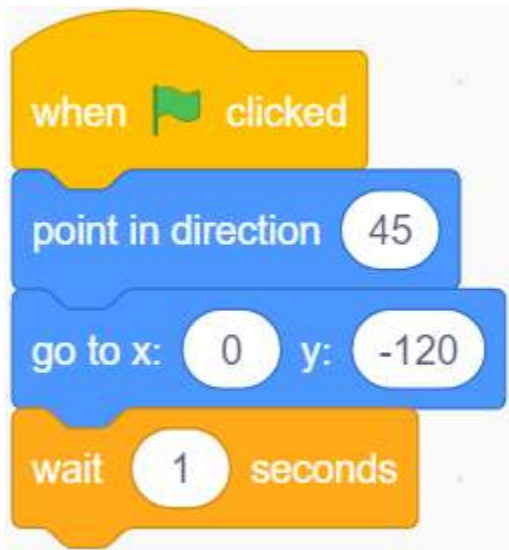
2. Ball sprite

The effect of the ball sprite is that it moves around the stage and bounces when it touches the edge; it bounces down if it touches the block above the stage; it bounces up if it touches the Paddle sprite during its fall; if it doesn't, the script stops running and the game ends.

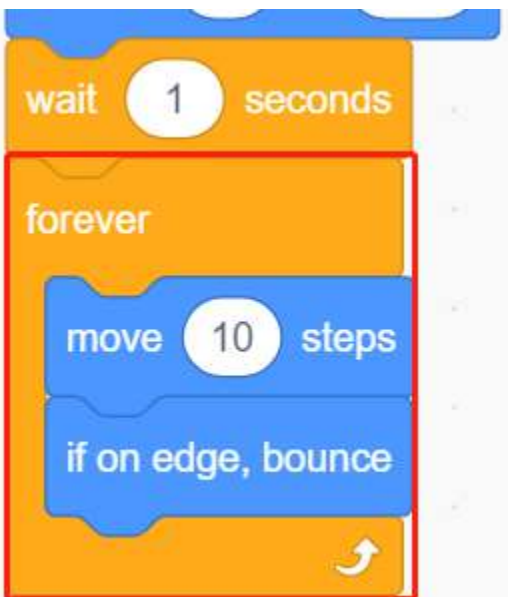
- Add **Ball** sprite.



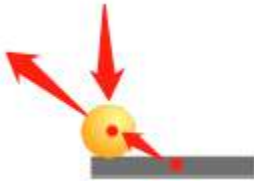
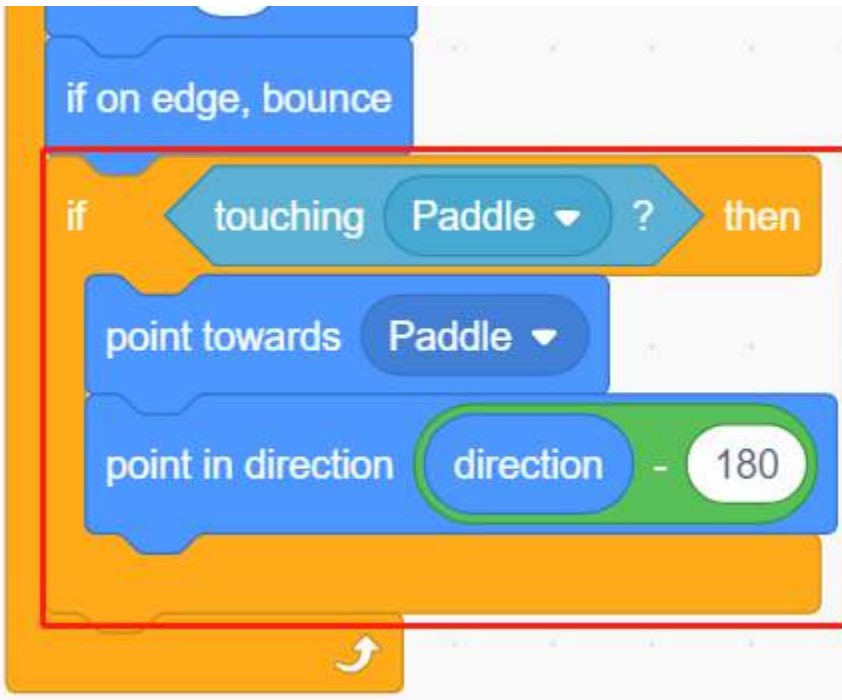
- When the green flag is clicked, set the angle of the **Ball** sprite to 45° and set the initial position to (0, -120).



- Now let the **Ball** sprite move around the stage and bounce when it touches the edge, and you can click on the green flag to see the effect.



- When the **Ball** sprite touches the **Paddle** sprite, do a reflection. The easy way to do this is to let the angle be directly inverted, but then you'll find that the path of the ball is completely fixed, which is too boring. Therefore, we use the center of the two sprites to calculate and make the ball bounce in the opposite direction of the center of the baffle.



- When the **Ball** sprite falls to the edge of the stage, the script stops running and the game ends.



3. Block1 sprite

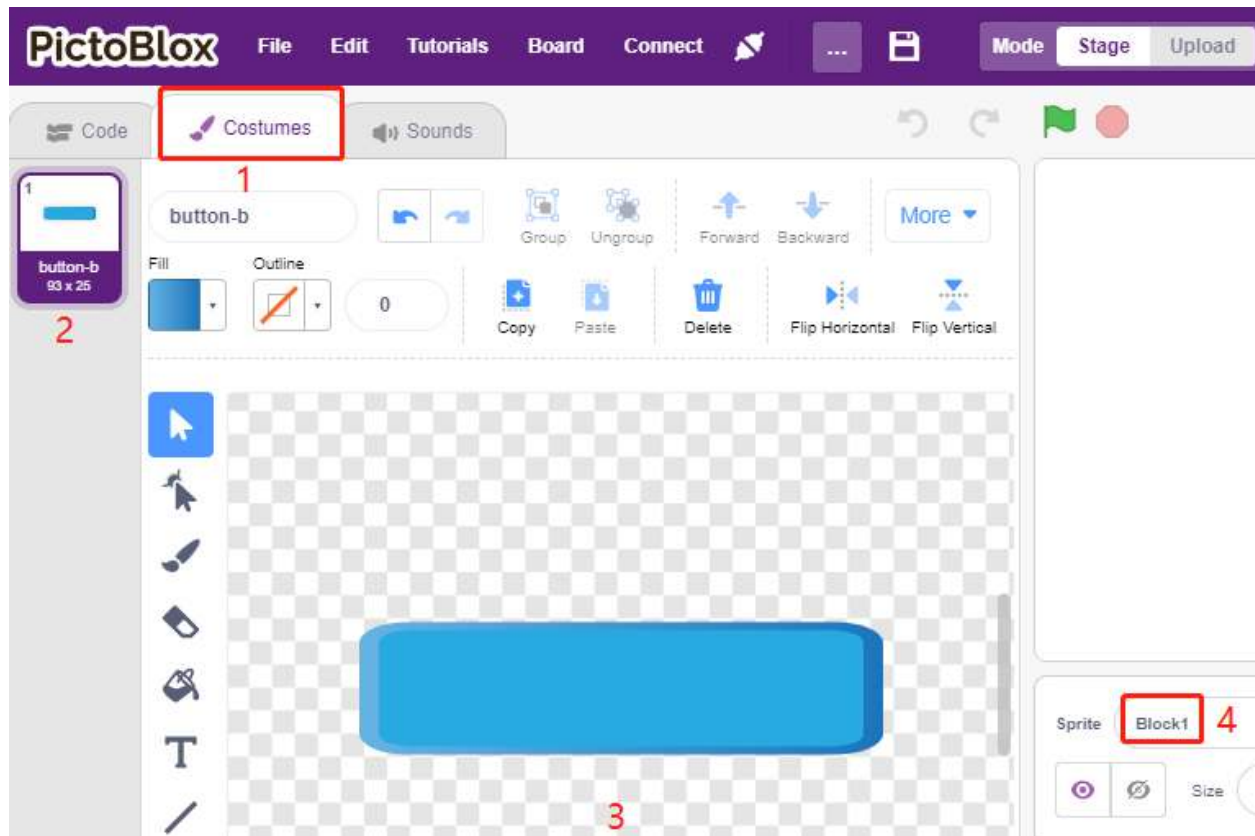
The **Block1** sprite is to appear with the effect of cloning 4x8 of itself above the stage in a random color, and deleting a clone if it is touched by the **Ball** sprite.

The **Block1** sprite is not available in the **PictoBlox** library, you need to draw it yourself or modify it with an existing sprite. Here we are going to modify it with the **Button3** sprite.

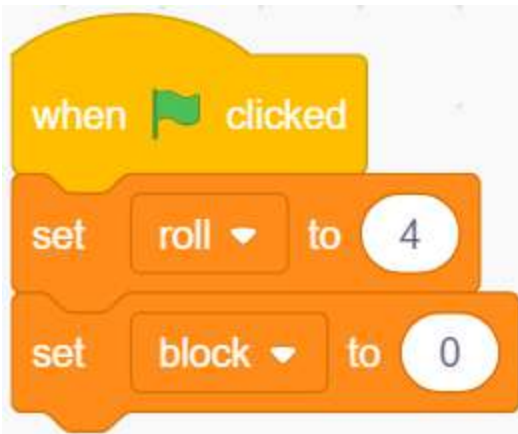
- After adding the **Button3** sprite, go to the **Costumes** page. Now delete **button-a** first, then reduce both the width and height of **button-b**, and change the sprite name to **Block1**, as shown in the following image.

Note:

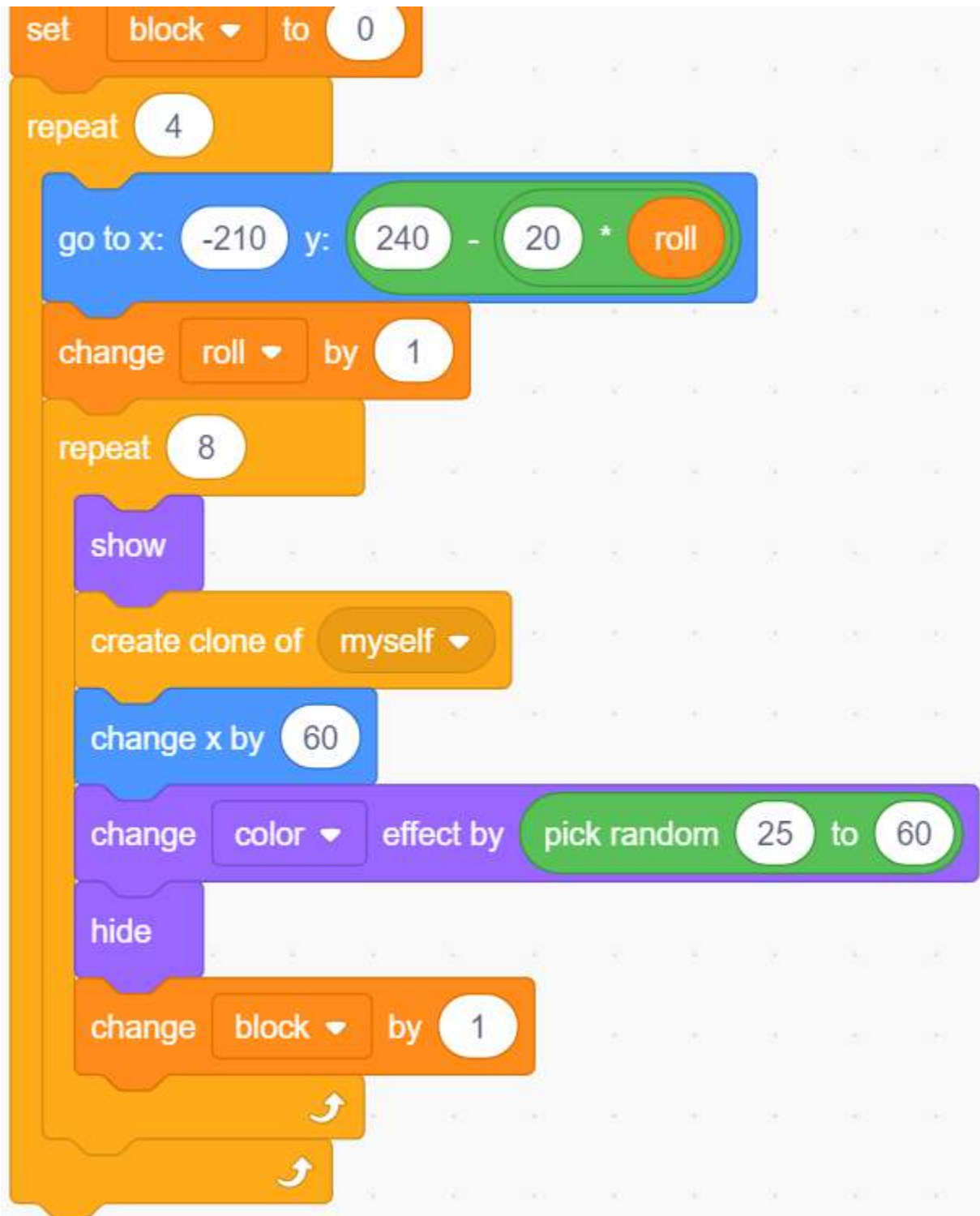
- For the width of **Block1**, you can probably simulate it on the screen to see if you can put down 8 in a row, if not, then reduce the width appropriately.
- In the process of shrinking the **Block1** sprite, you need to keep the center point in the middle of the sprite.



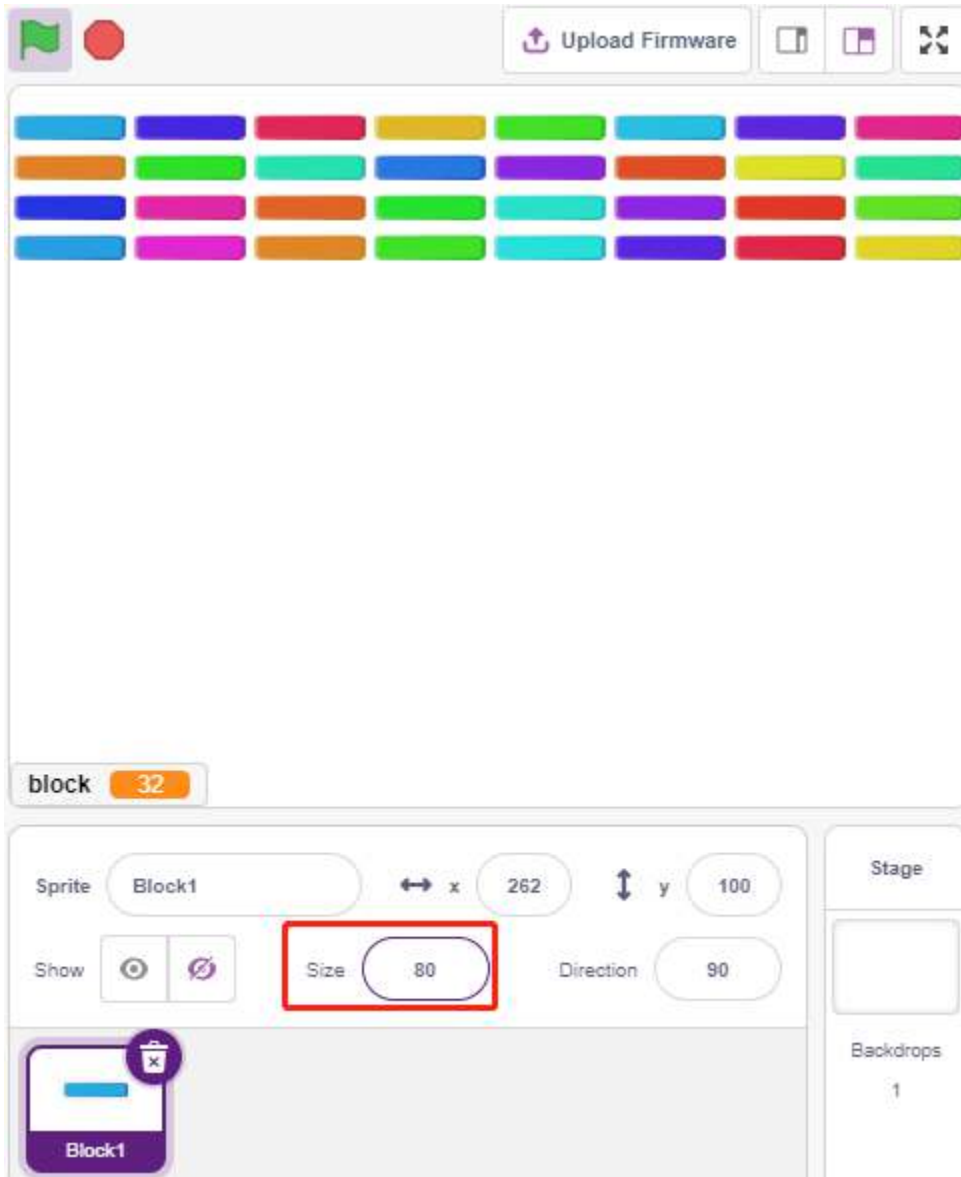
- Now create 2 variables first, **block** to store the number of blocks and **roll** to store the number of rows.



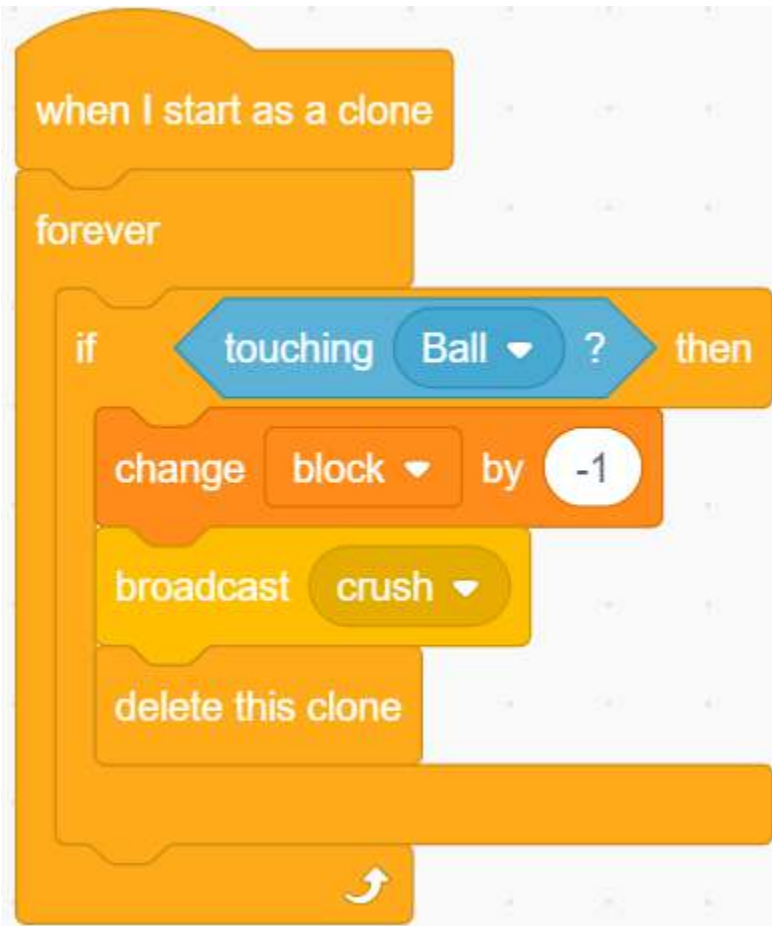
- We need to make a clone of the **Block1** sprite, so that it displays from left to right, top to bottom, one by one, 4x8 in total, with random colors.



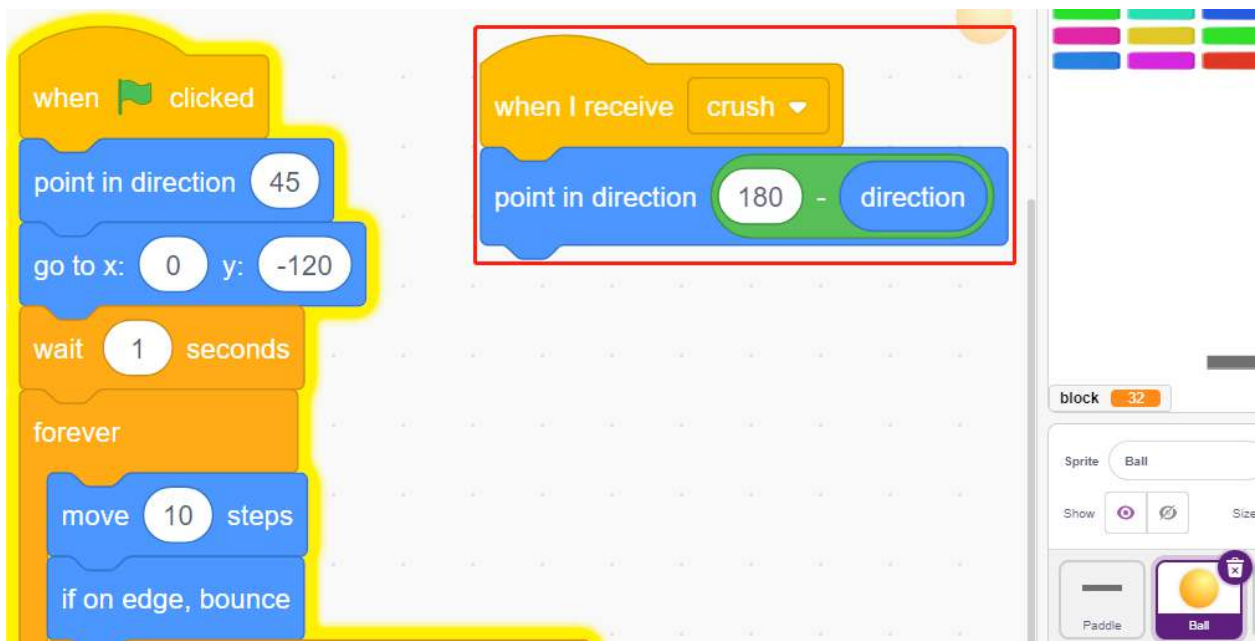
- After the script is written, click on the green flag and look at the display on the stage, if it is too compact or too small, you can change the size.



- Now write the trigger event. If the cloned **Block1** sprite touches the **Ball** sprite, delete the clone and broadcast the message **crush**.



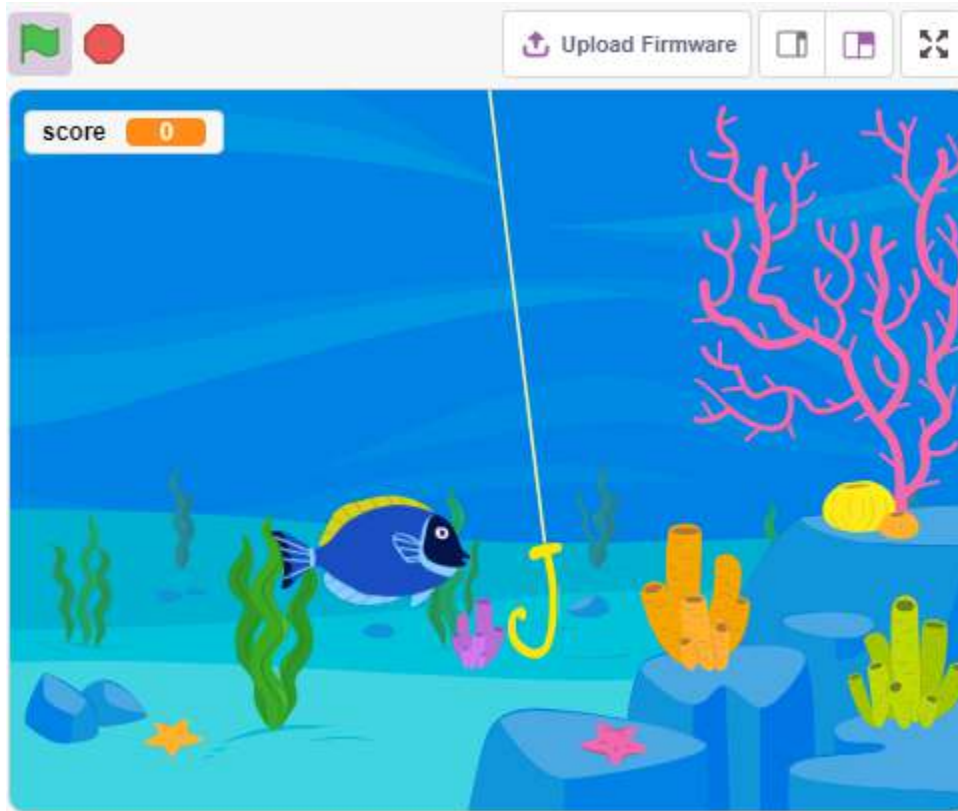
- Back to the **Ball** sprite, when the broadcast **crush** is received (the **Ball** sprite touches the clone of **Block1** sprite), the **Ball** is popped from the opposite direction.



3.23 2.20 GAME - Fishing

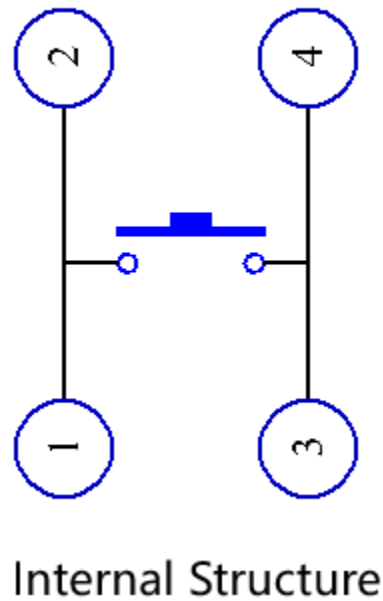
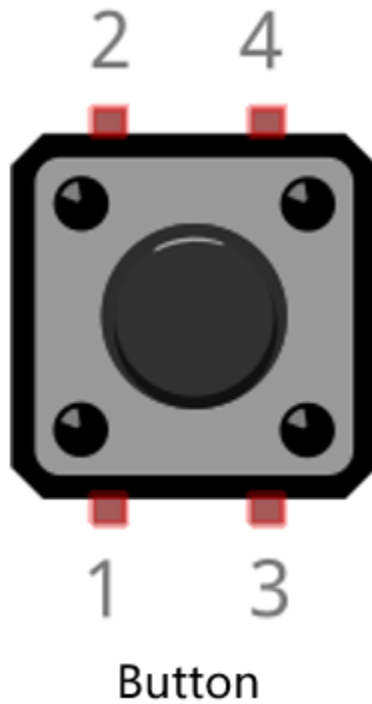
Here, we play a fishing game with a button.

When the script is running, the fish swim left and right on the stage, you need to press the button when the fish is almost close to the hook (it is recommended to press it for a longer time) to catch the fish, and the number of fish caught will be recorded automatically.



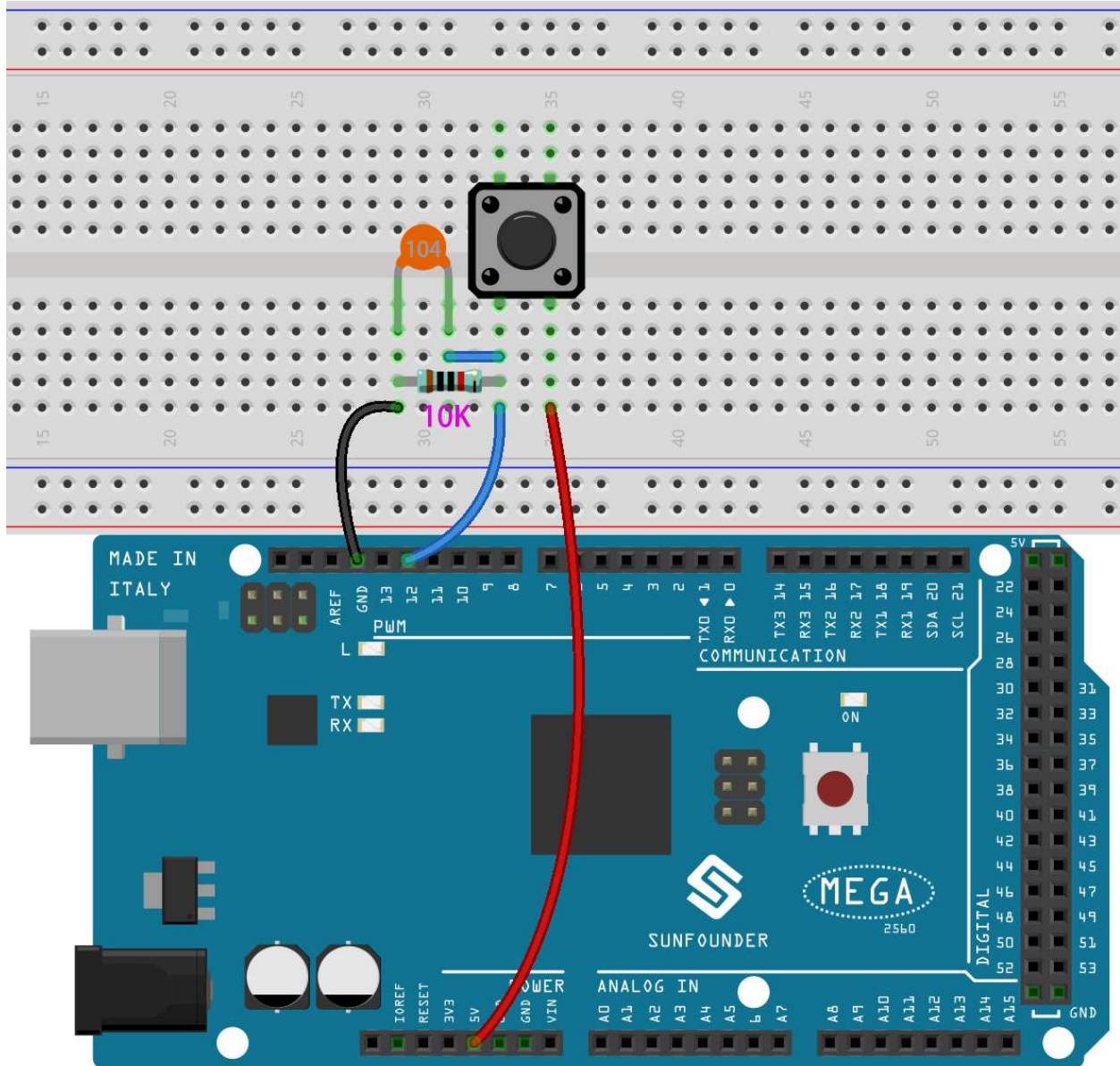
3.23.1 Build the Circuit

The button is a 4-pin device, since the pin 1 is connected to pin 2, and pin 3 to pin 4, when the button is pressed, the 4 pins are connected, thus closing the circuit.



Build the circuit according to the following diagram.

- Connect one of the pins on the left side of the button to pin 12, which is connected to a pull-down resistor and a 0.1 μ F (104) capacitor (to eliminate jitter and output a stable level when the button is working).
- Connect the other end of the resistor and capacitor to GND, and one of the pins on the right side of the button to 5V.



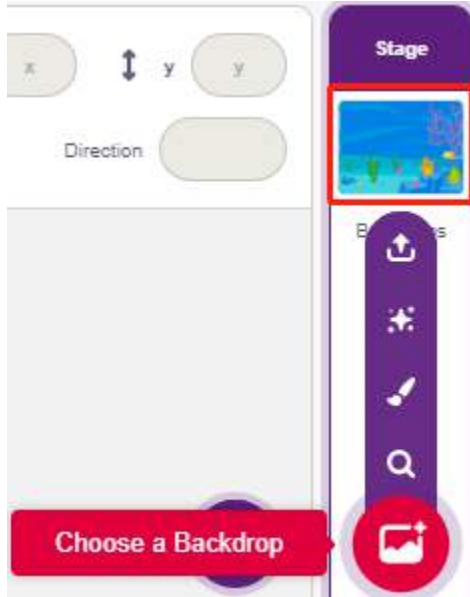
- Breadboard
- Button
- Resistor
- Capacitor

3.23.2 Programming

We need to select an **Underwater** backdrop first, then add a **Fish** sprite and let it swim back and forth on the stage. Then draw a **Fishhook** sprite and control it by a button to start fishing. When the **Fish** sprite touches the **Fishhook** sprite in the hooked state (turns red), it will be hooked.

1. Adding a backdrop

Use the **Choose a Backdrop** button to add an **Underwater** backdrop.

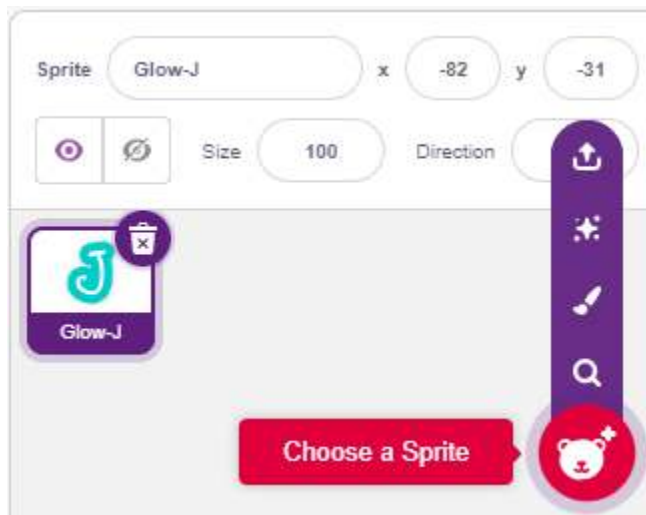


2. Fishhook sprite

The **Fishhook** sprite is intended to have the effect that it normally stays underwater in a yellow state; when the button is pressed, it is in a fishing state (red) and it moves above the stage.

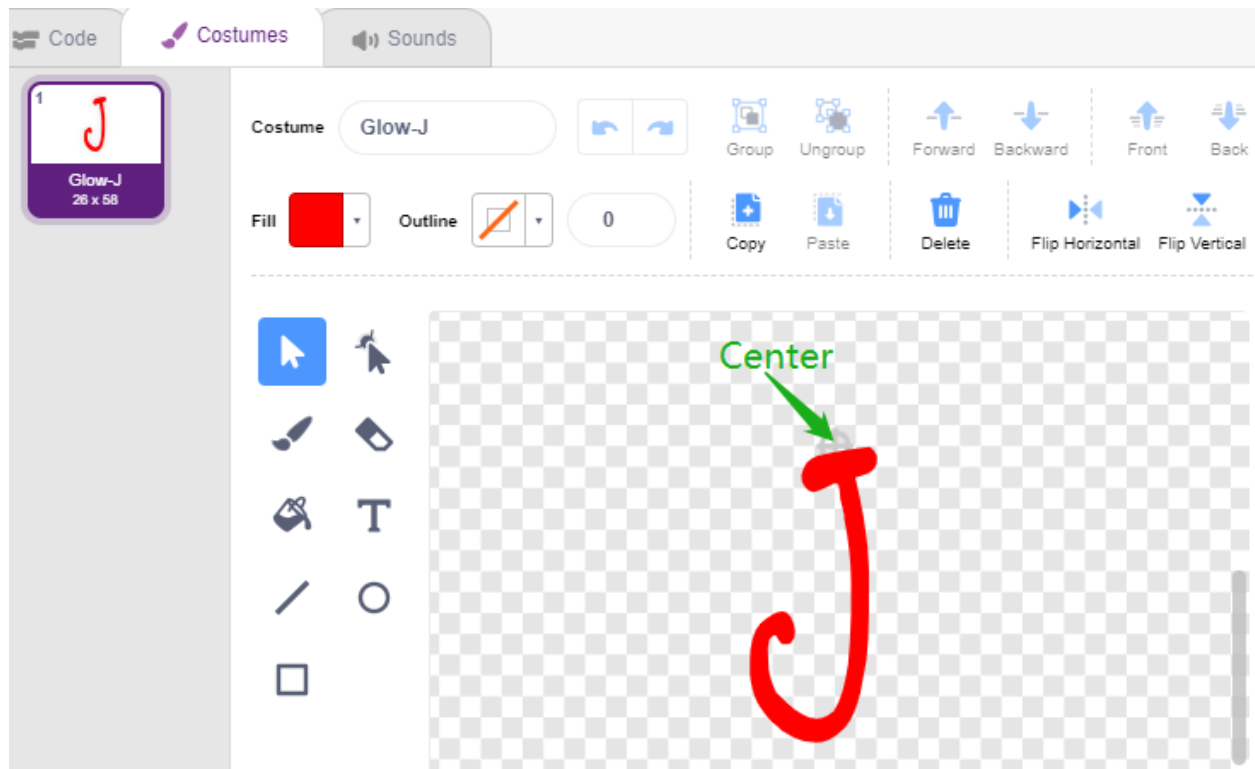
There is no **Fishhook** sprite in Pictoblox, we can modify the **Glow-J** sprite to look like a fishhook.

- Add the **Glow-J** sprite via **Choose a Sprite**.

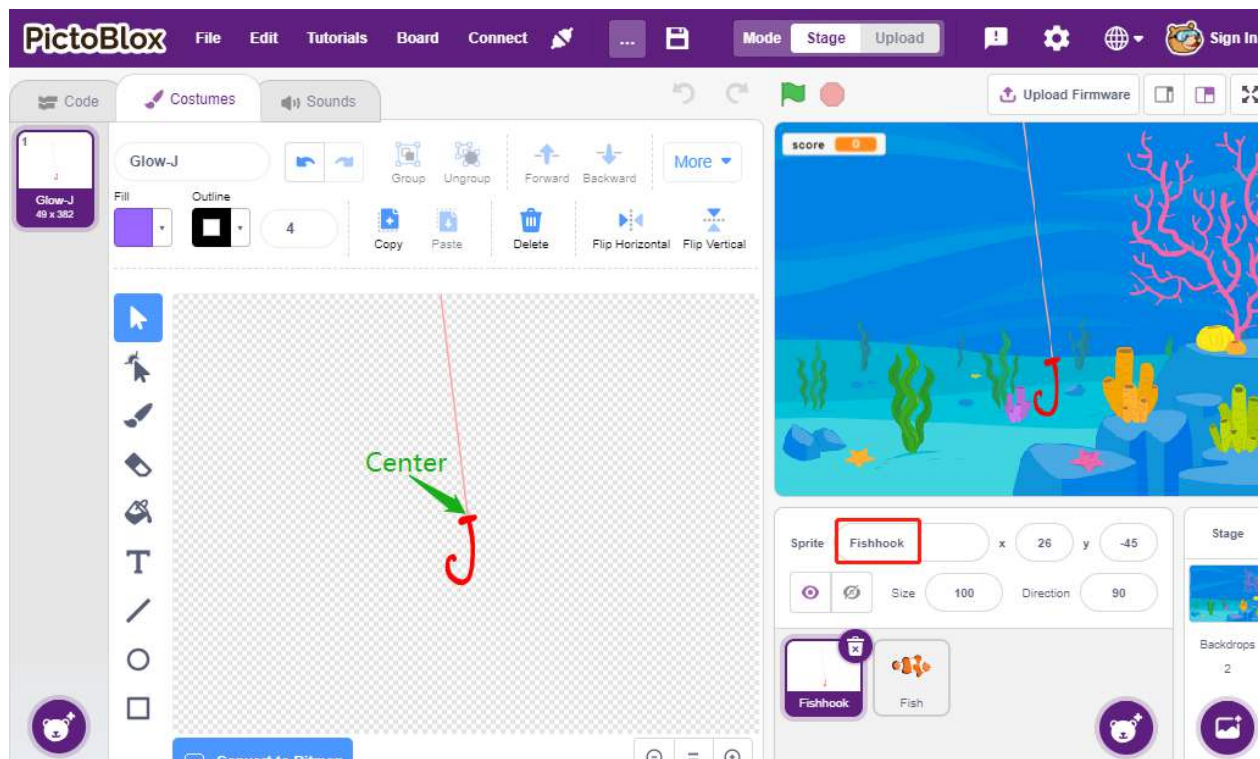


- Now go to the **Costumes** page of the **Glow-J** sprite, select Cyan's fill in the screen and remove it. Then change the J color to red and also reduce its width. The most important point to note is that you need to have the top of

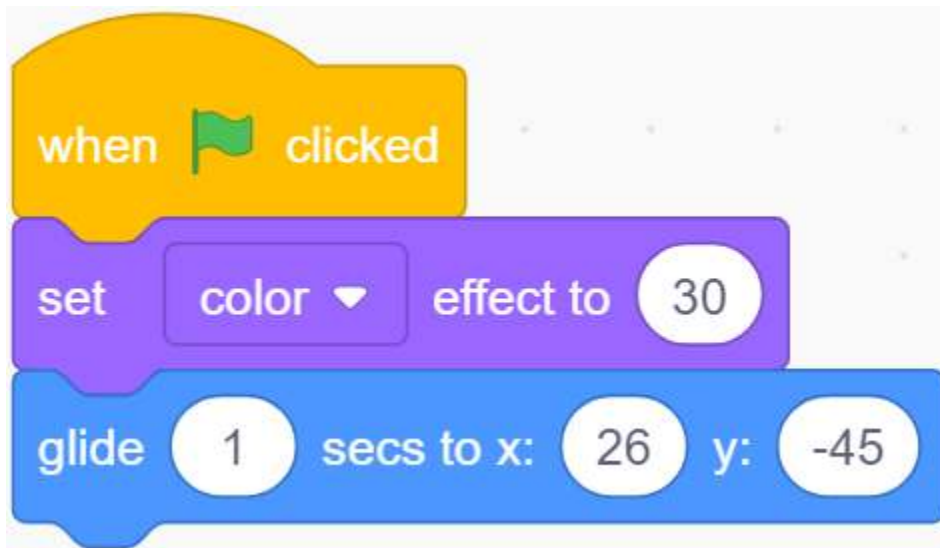
it just at the center point.



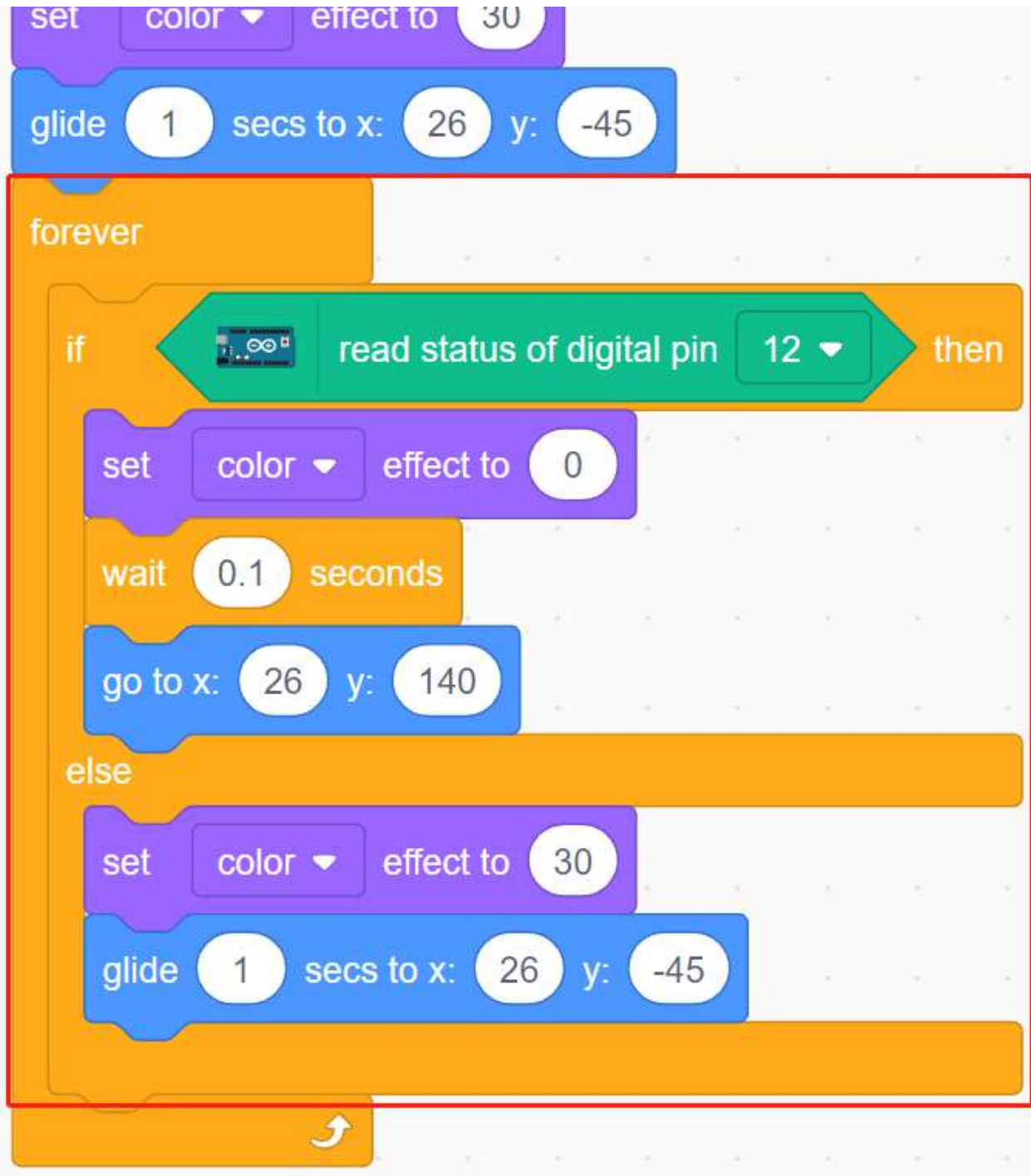
- Use the **Line** tool to draw a line as long as possible from the center point up (line out of the stage). Now that the sprite is drawn, set the sprite name to **Fishhook** and move it to the right position.



- When the green flag is clicked, set the sprite's color effect to 30 (yellow), and set its initial position.



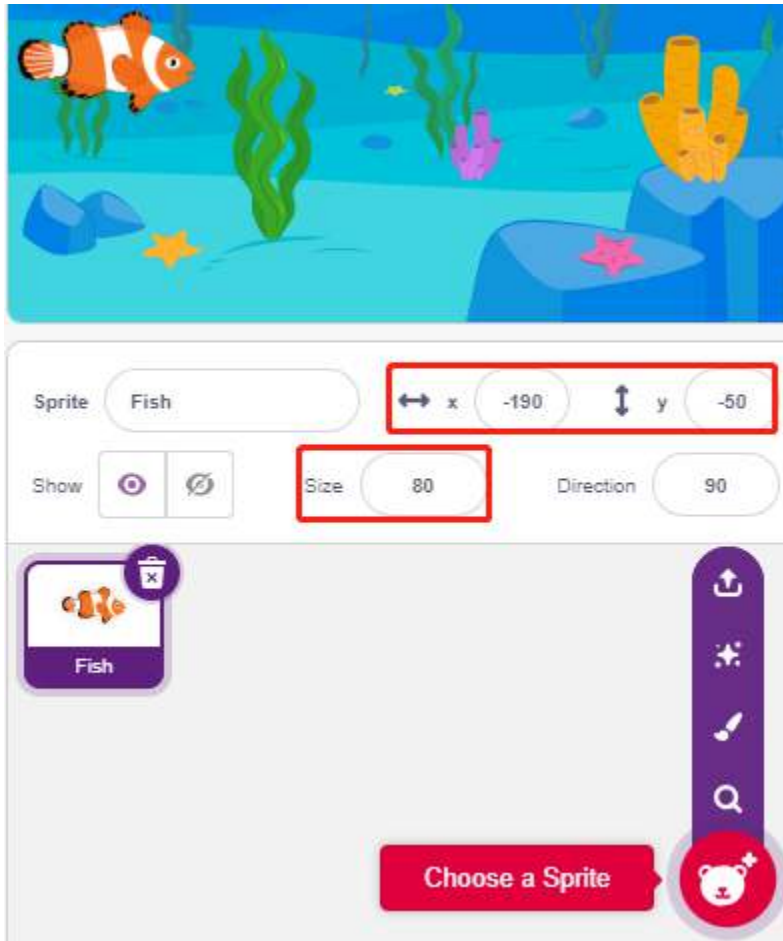
- If the button is pressed, set the color effect to 0 (red, start fishing state), wait for 0.1 and then move the **Fishhook** sprite to the top of the stage. Release the button and let the **Fishhook** return to its initial position.



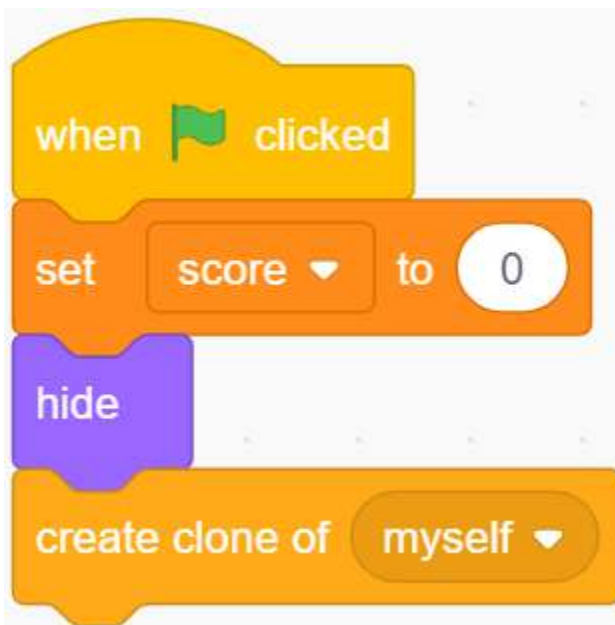
3. Fish sprite

The effect to be achieved by the **Fish** sprite is to move left and right on the stage, and when it encounters a **Fishhook** sprite in the fishing state, it shrinks and moves to a specific position and then disappears, and then clones a new **fish** sprite again.

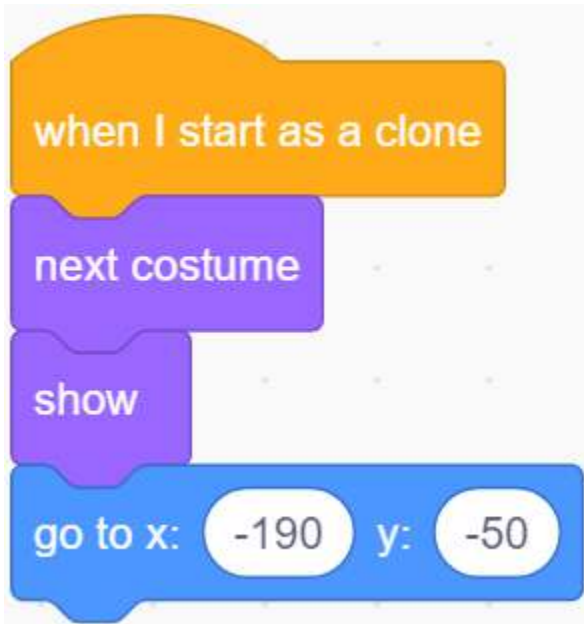
- Now add the **fish** sprite and adjust its size and position.



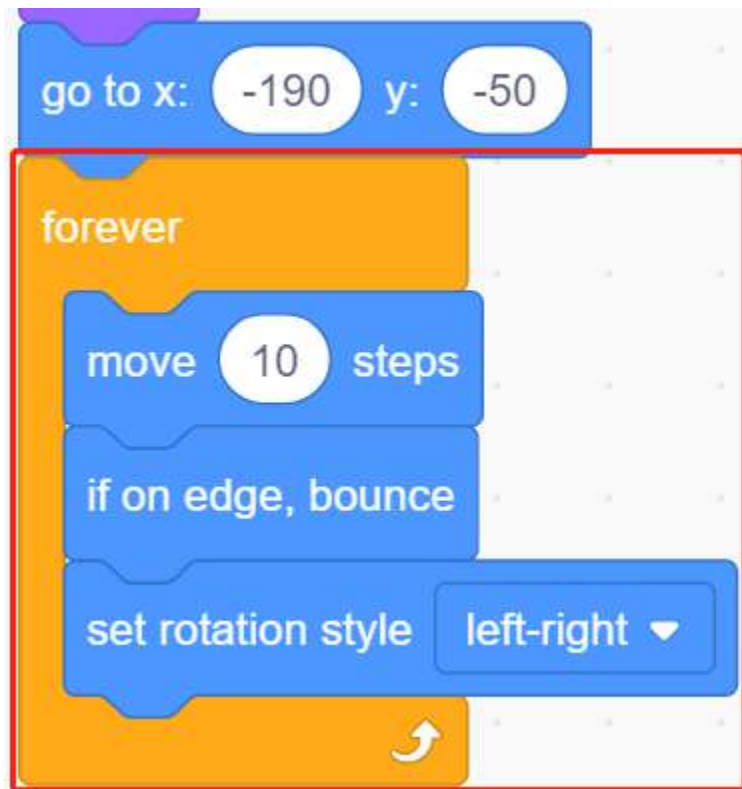
- Create a variable **score** to store the number of fish caught, hide this sprite and clone it.



- Show the clone of the **fish** sprite, switch its costume and finally set the initial position.



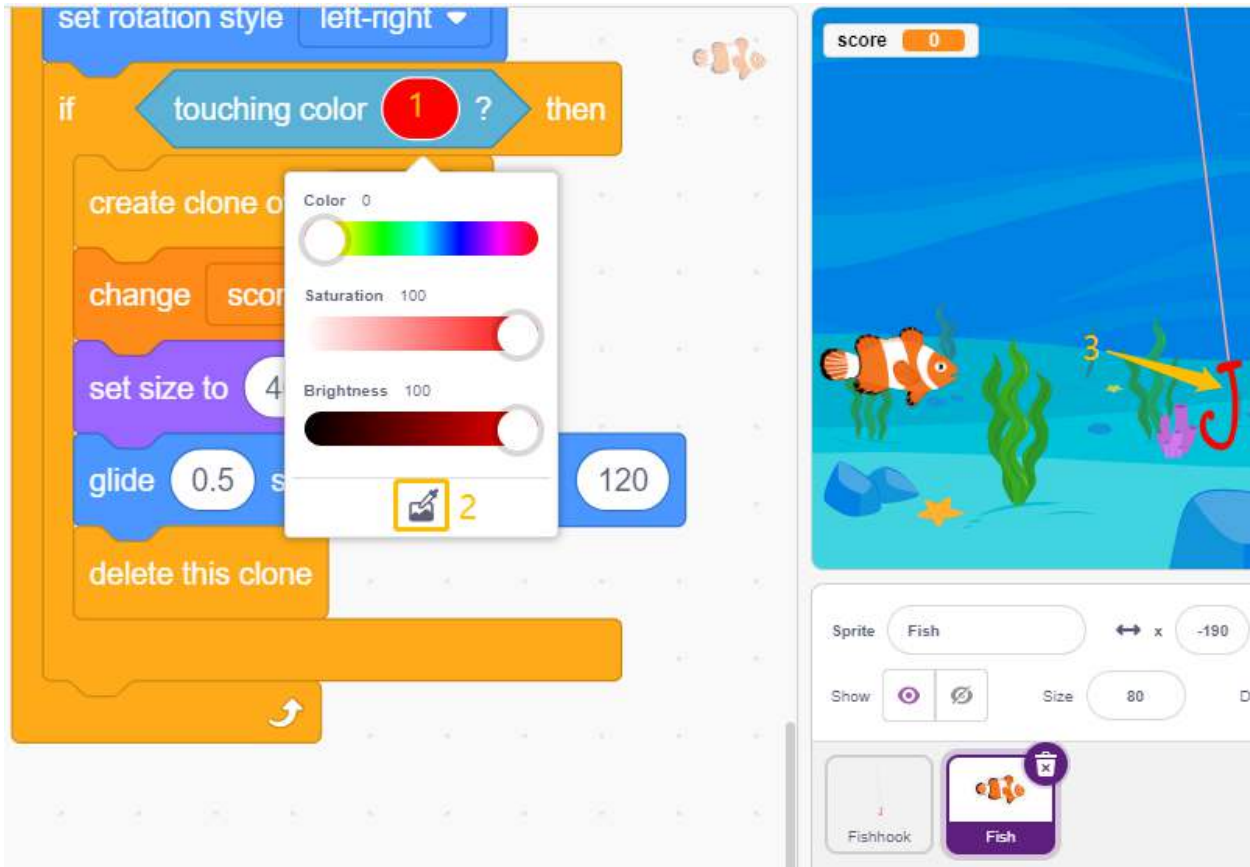
- Make the **fish** sprite's clone move left and right and bounce back when it touches the edge.



- The **fish** sprite (of the clone) will not react when it passes the **Fishhook** sprite; when it touches the **Fishhook** sprite in the fishing state (turns red), it will be caught, at which point the score (variable score) +1, and it will also show a score animation (shrinks 40%, quickly moves to the position of the scoreboard and disappears). At the same time, a new fish is created (a new fish sprite clone) and the game continues.

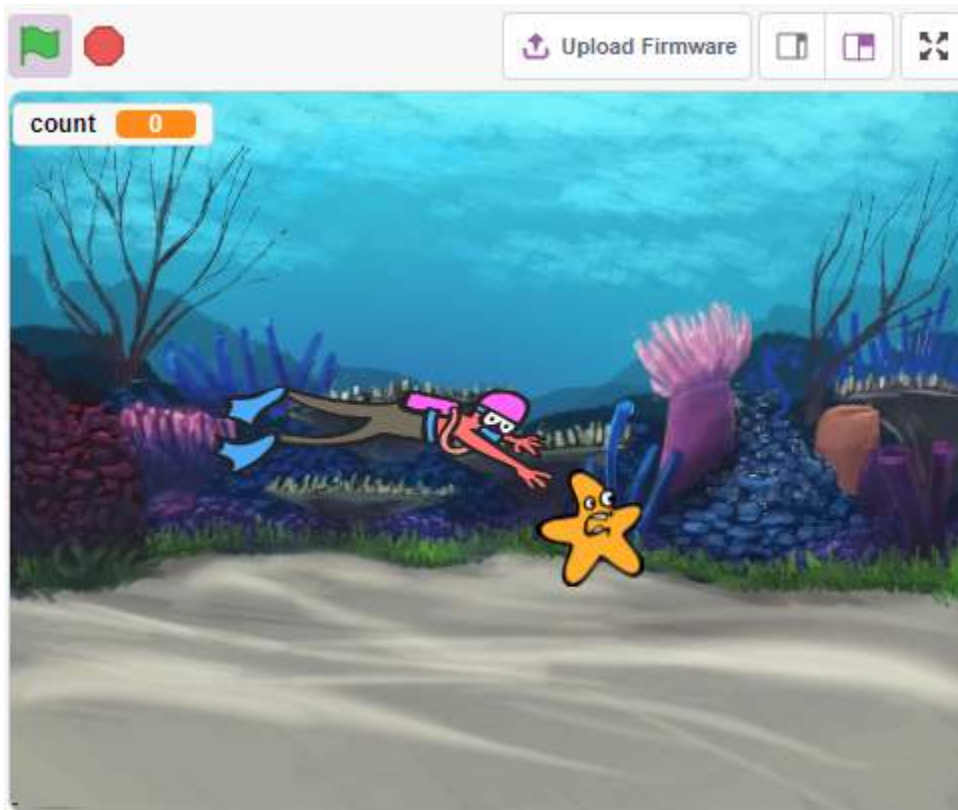
Note: You need to click on the color area in the [Touch color] block, and then select the eyedropper tool to pick up

the red color of the **Fishhook** sprite on the stage. If you choose a color arbitrarily, this [Touch color] block will not work.



3.24 2.21 Catching Starfish

Here, let's make a game to catch starfish. At the start of the script, a starfish is swimming comfortably on the stage, and a diver is also swimming to the right and left. You need to control the depth of the water level sensor module in the water in order to let the diver on the stage catch the starfish, and for each starfish caught, the count will be increased by 1.

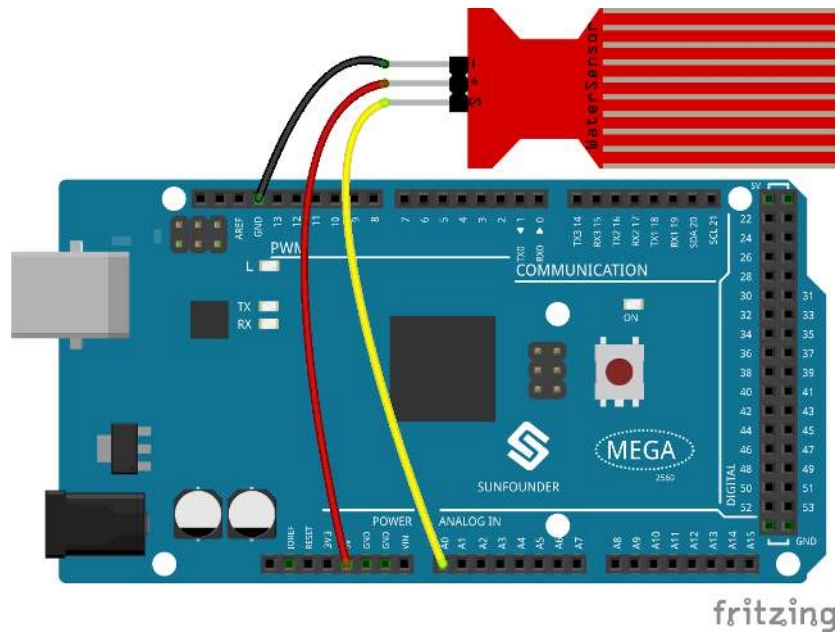


3.24.1 Build the Circuit

The Water Level Sensor module is an easy-to-use, compact and cost effective water level/drop detection sensor that measures the water level by a series of exposed parallel wire traces to determine the size of the water drop/volume.

The more water the sensor is immersed in, the greater the value of the S pin output.

Now build the circuit according to the diagram below.

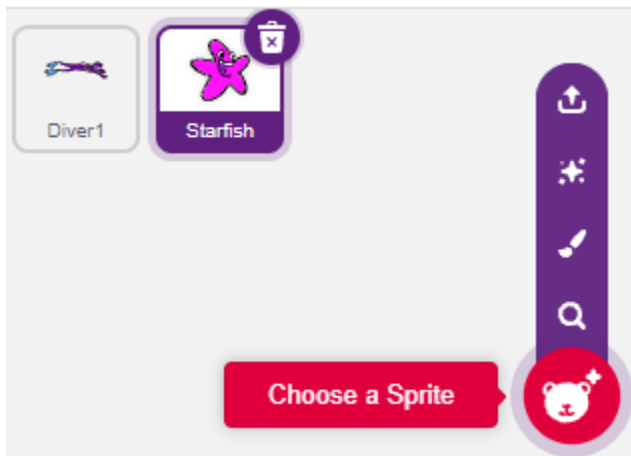


- Breadboard
- Water Level Sensor Module

3.24.2 Programming

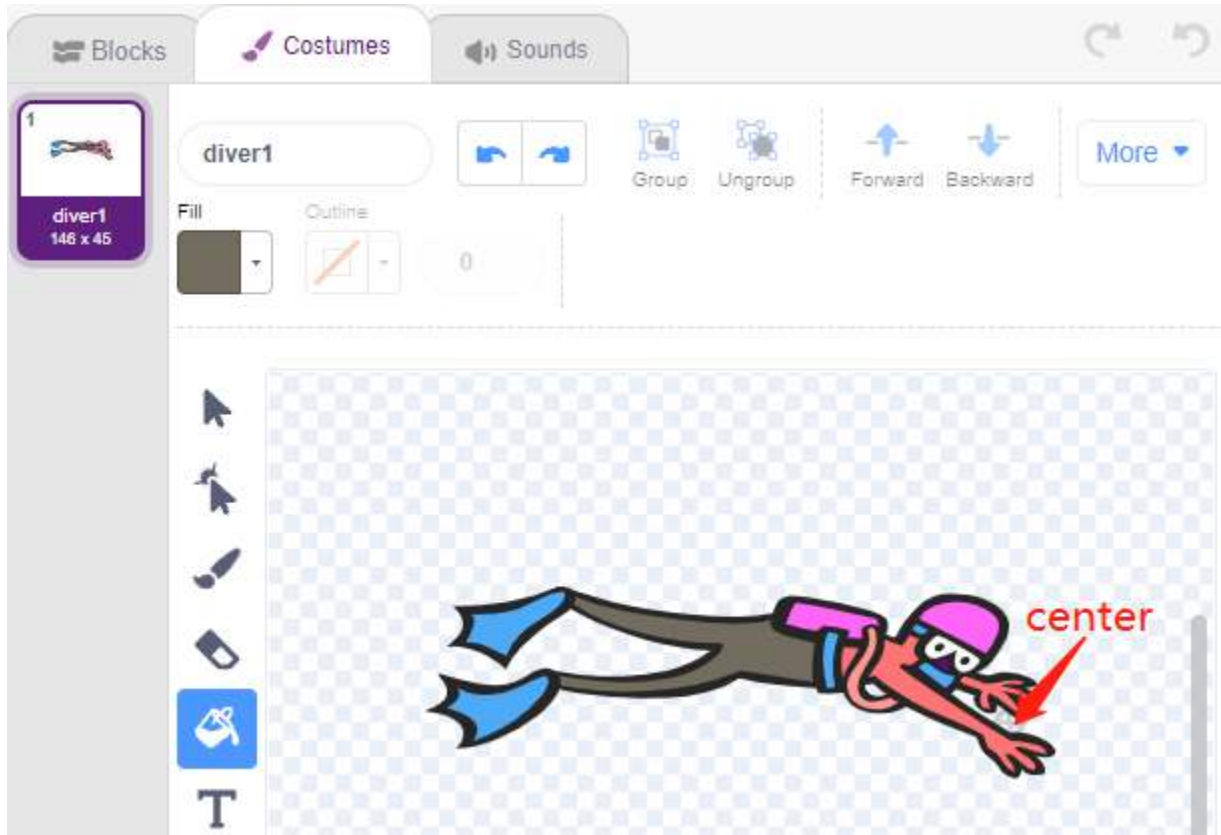
1. Select sprites and backdrop

Delete the default sprite, select the **Diver1** and **Starfish** sprite.

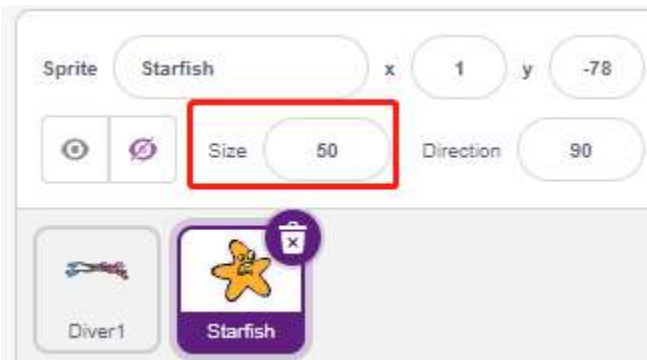


Go to **Diver1's** Costumes page and use the **Fill** tool to fill in the colors you like.

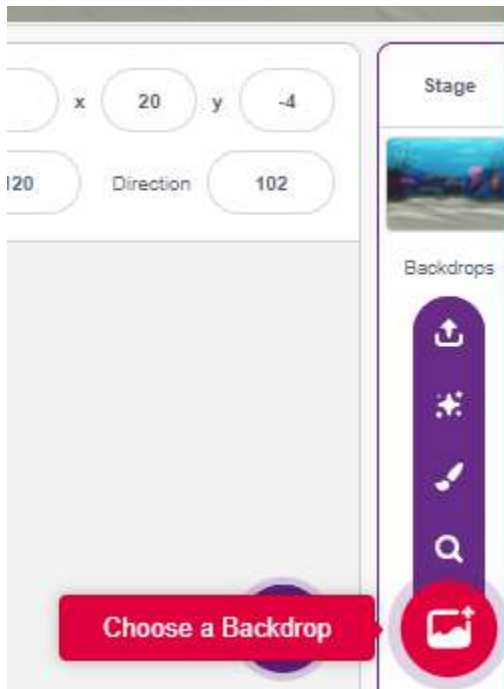
Note: You will need to fill both arms of Diver1 with a unique color that will not find the same on the stage.



Reduce the size of the **Starfish** sprite, you can also modify its color as you like.

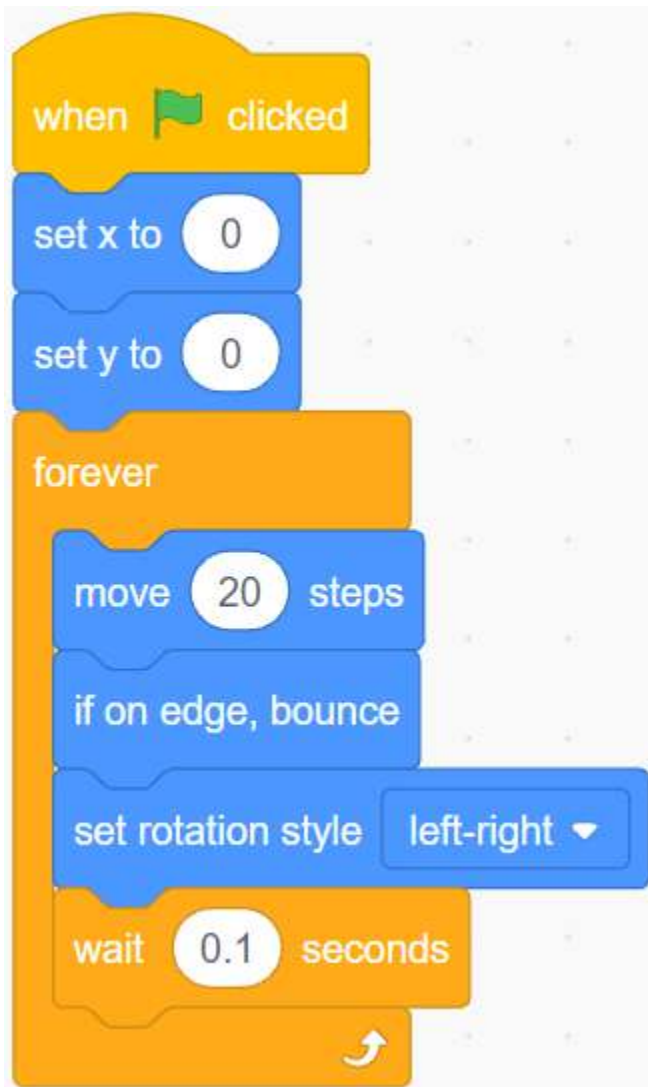


Select an **Underwater1** backdrop.



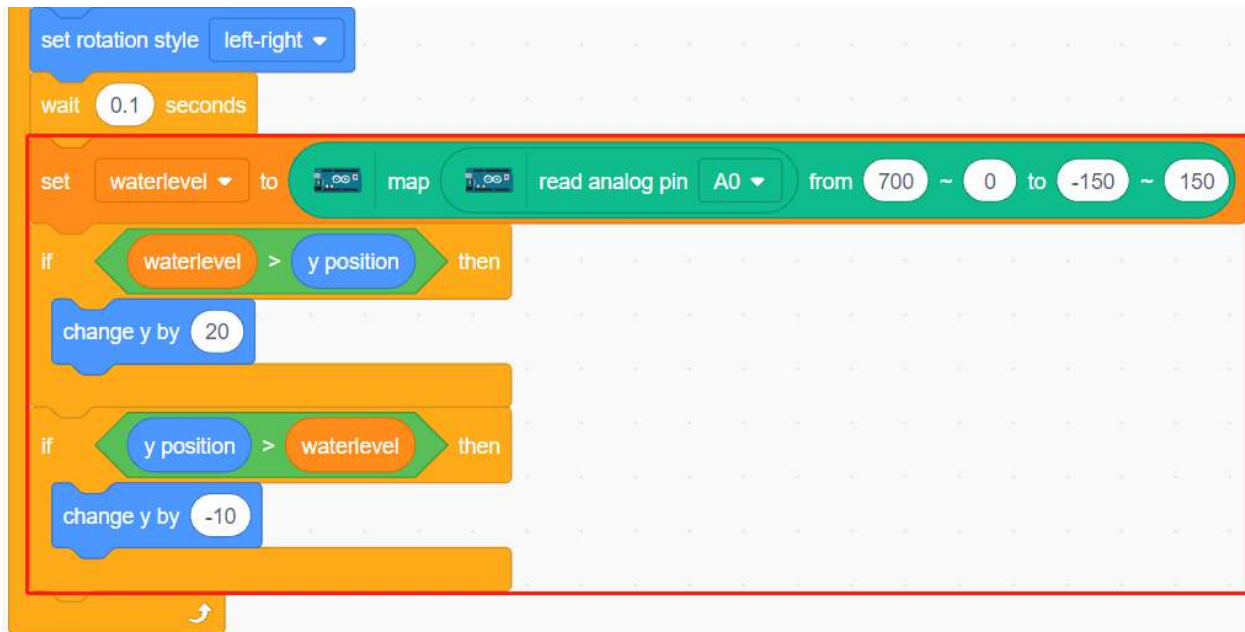
2. Scripting the Diver1 sprite

Set the initial position of the **Diver1** sprite and have it swim back and forth across the stage.



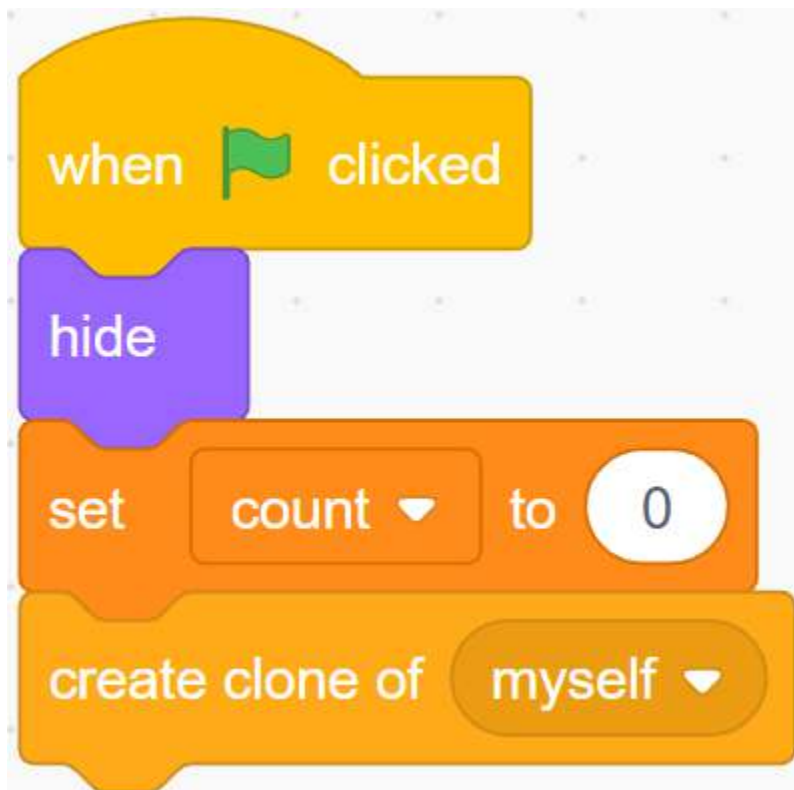
The deeper you put the water level sensor in, the greater the value you get as a way to determine the depth (y coordinate) that the **Diver1** sprite swims.

- Read A0 (the value of the water level sensor) and map its range to the y-coordinate of the stage as a way to get a new y-coordinate.
- If **new_y** is greater than the current Y coordinate, let it move up to the **new_y** position.
- If **new_y** is less than the current Y coordinate, let it move down to the **new_y** position

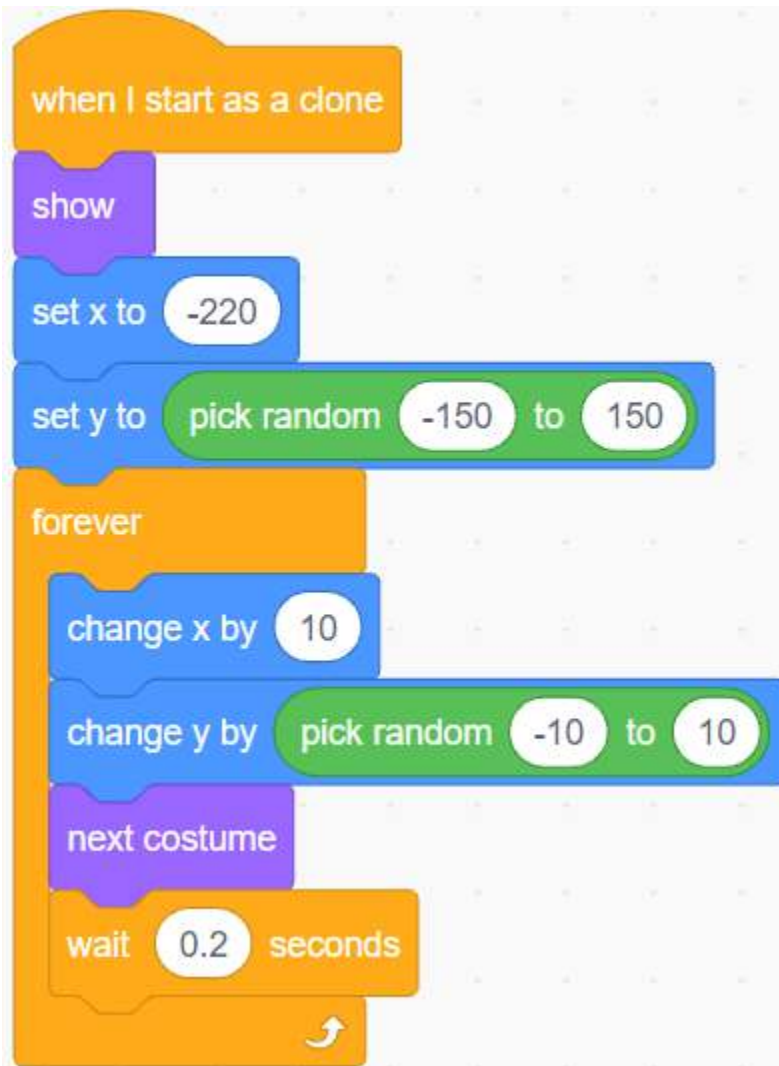


3. Scripting the Starfish sprite

When the script starts, first hide **Starfish** sprite and then clone it.



When appearing as a clone, set its movement effect.

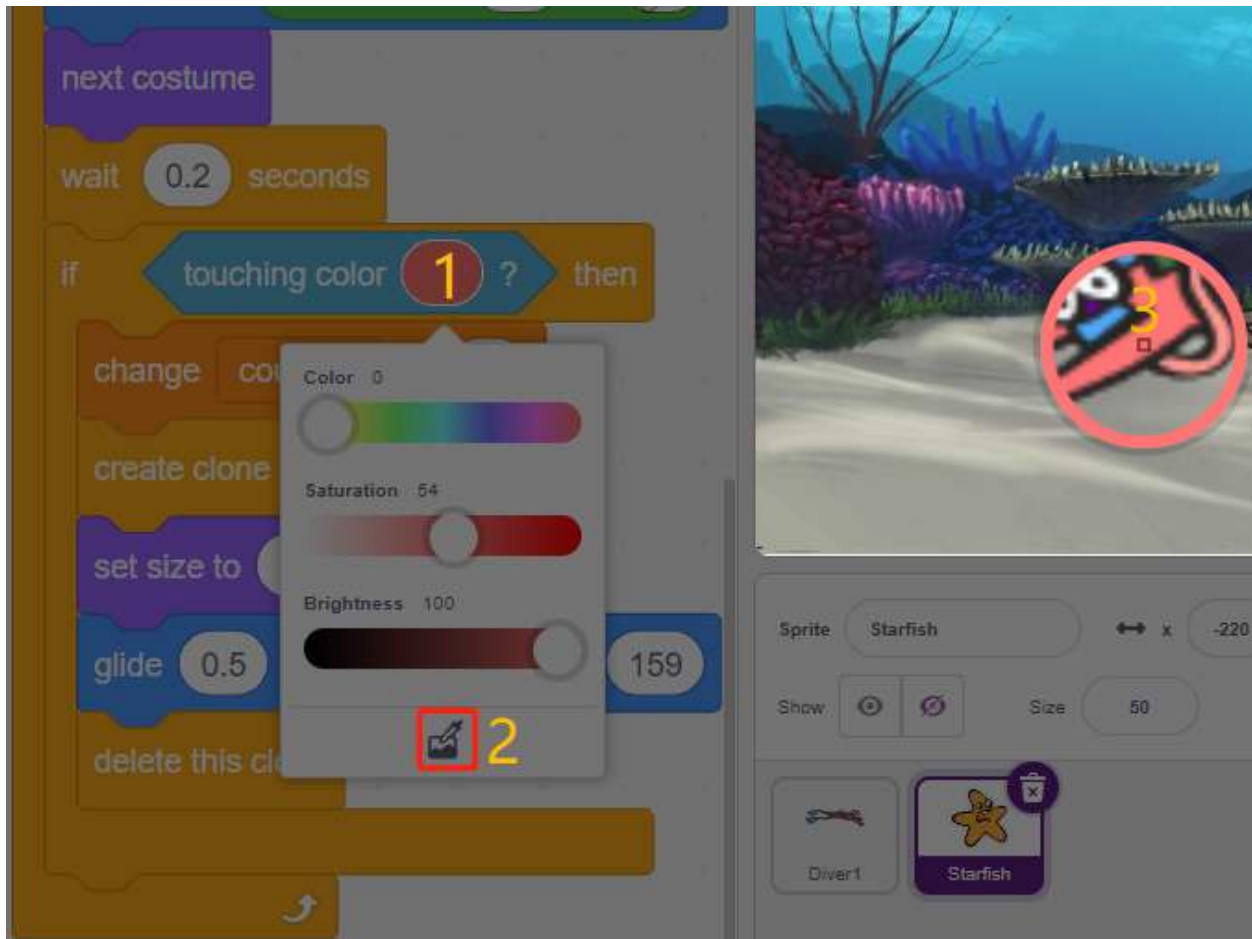


When it touches red (the color of **Diver1**'s two arms), which means it is caught by **Diver1** sprite, the following effect is made.

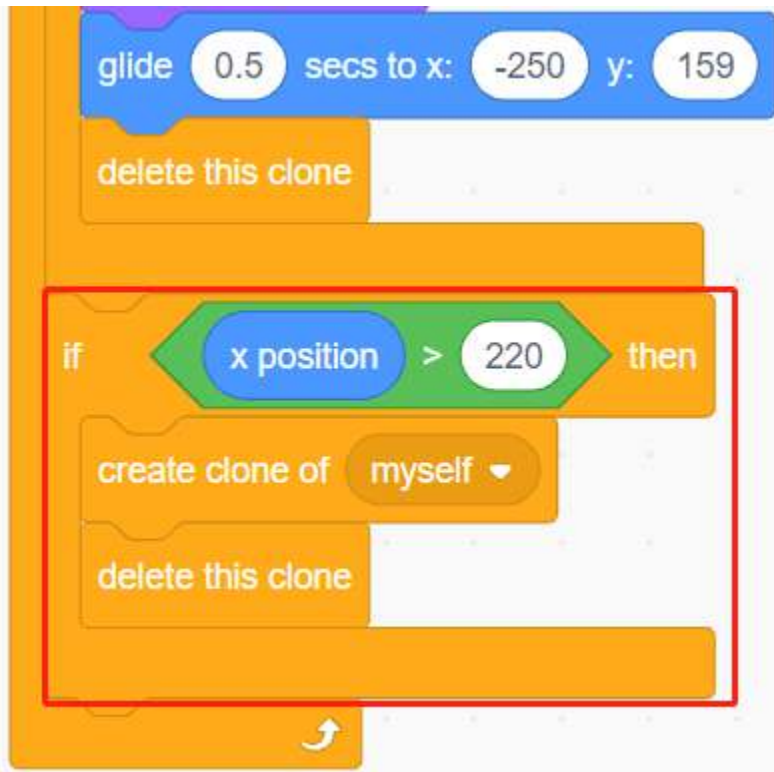
- Adds 1 to the value of the variable **count**.
- A score animation is displayed (shrinks by 20%, moves quickly to the position of the scoreboard and disappears).
- At the same time, a new starfish is cloned and the game continues.



Note: You need to click on the color area in the [Touch color] block, and then select the eyedropper tool to pick up the red color of the **Diver1** sprite on the stage. If you choose a color arbitrarily, this [Touch color] block will not work.



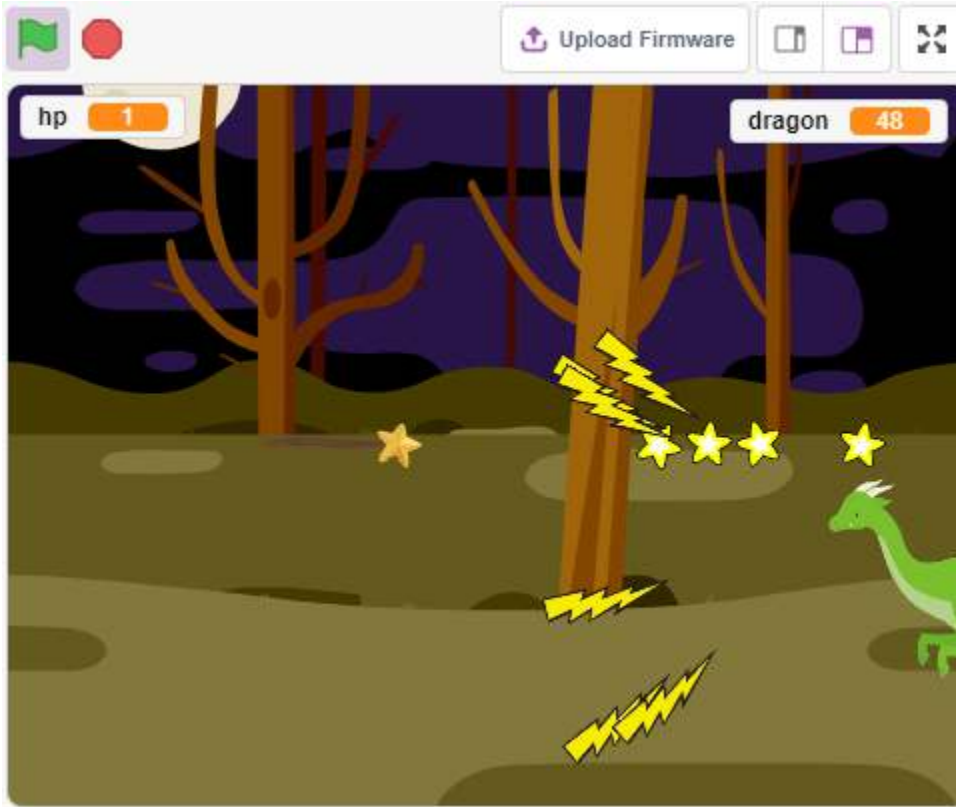
When the Starfish's clone swims to the far right, delete the clone and clone it again.



3.25 2.22 GAME - Kill Dragon

Here, we use the joystick to play a game of dragon killing.

When clicking on green, the dragon will float up and down on the right side and blow fire intermittently. You need to use the joystick to control the movement of the magic wand and launch star attacks at the dragon, while avoiding the flames it shoots, and finally defeat it.



3.25.1 Build the Circuit

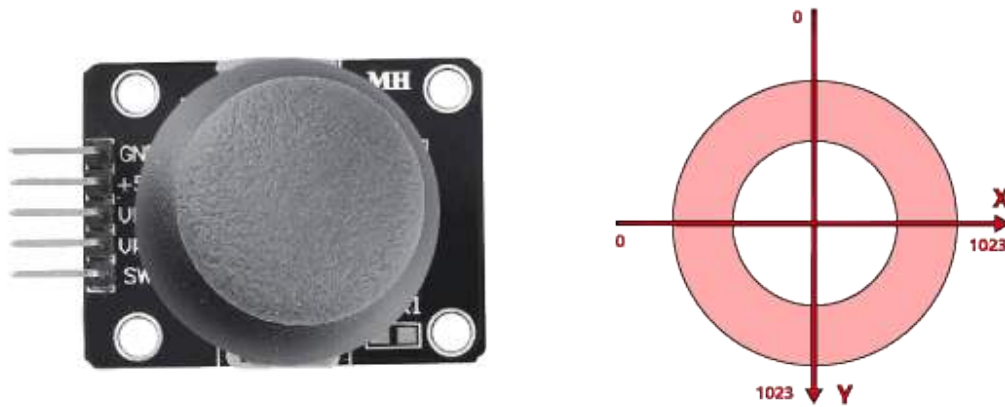
A joystick is an input device consisting of a stick that pivots on a base and reports its angle or direction to the device it is controlling. Joysticks are often used to control video games and robots.

In order to communicate a full range of motion to the computer, a joystick needs to measure the stick's position on two axes – the X-axis (left to right) and the Y-axis (up and down).

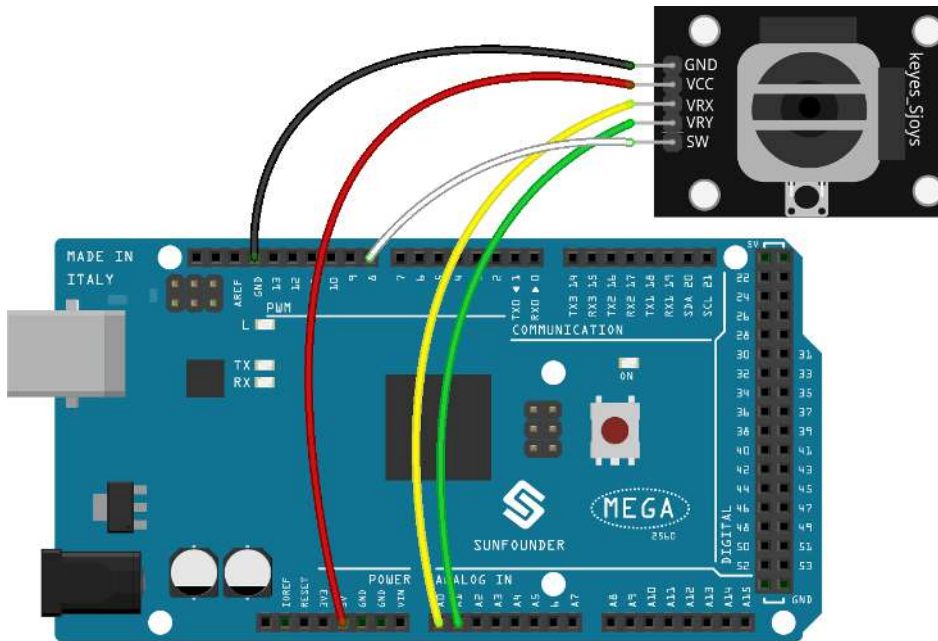
The motion coordinates of the joystick are shown in the following figure.

Note:

- The x coordinate is from left to right, the range is 0-1023.
 - y coordinate is from top to bottom, range is 0-1023.
-



Now build the circuit according to the following diagram.

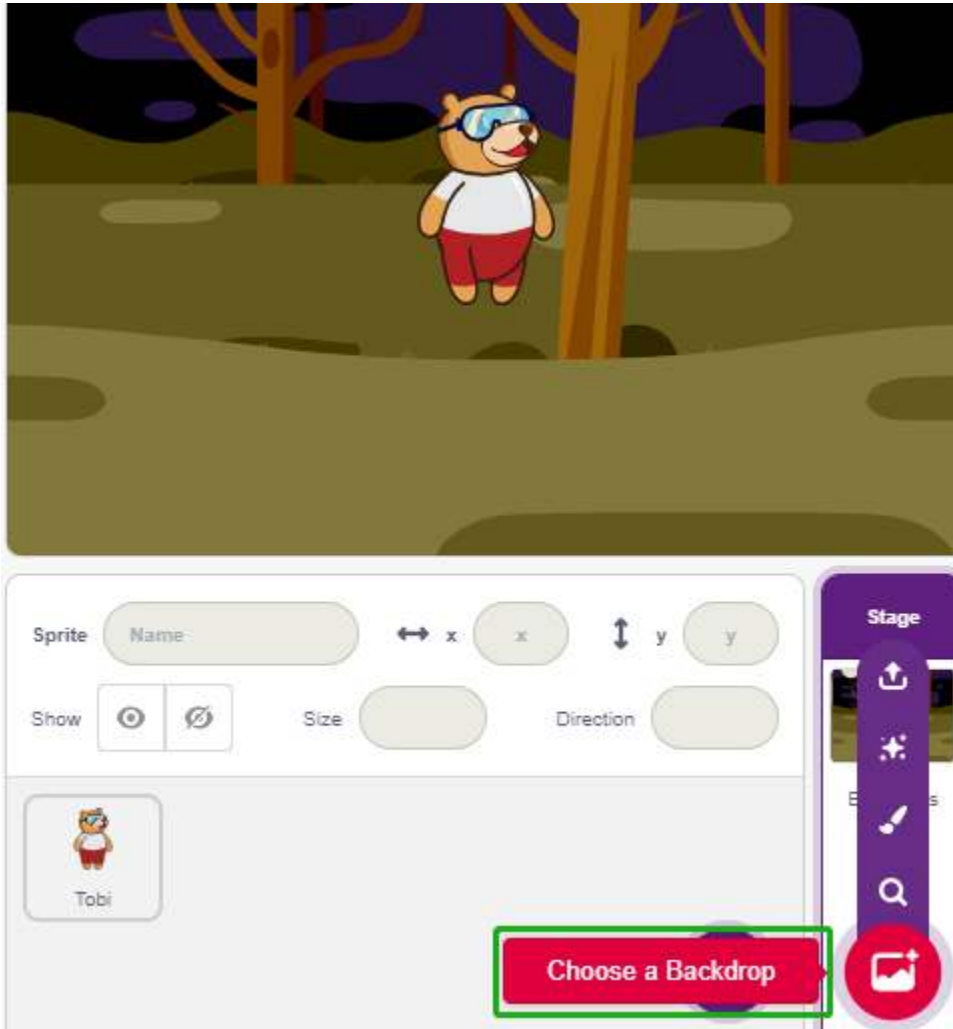


- Breadboard
- Joystick Module

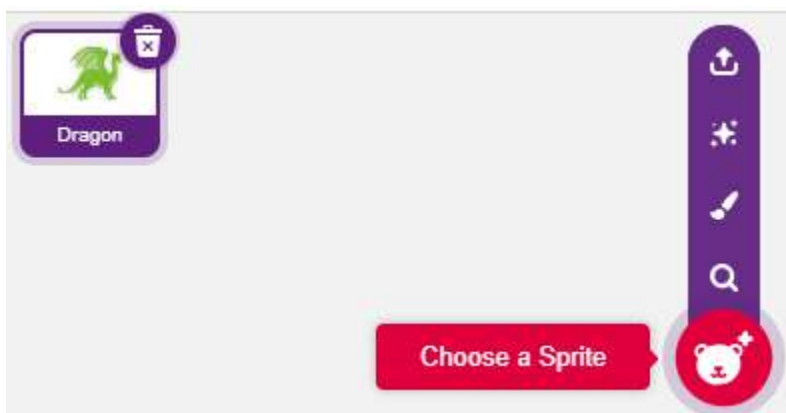
3.25.2 Programming

1. Dragon

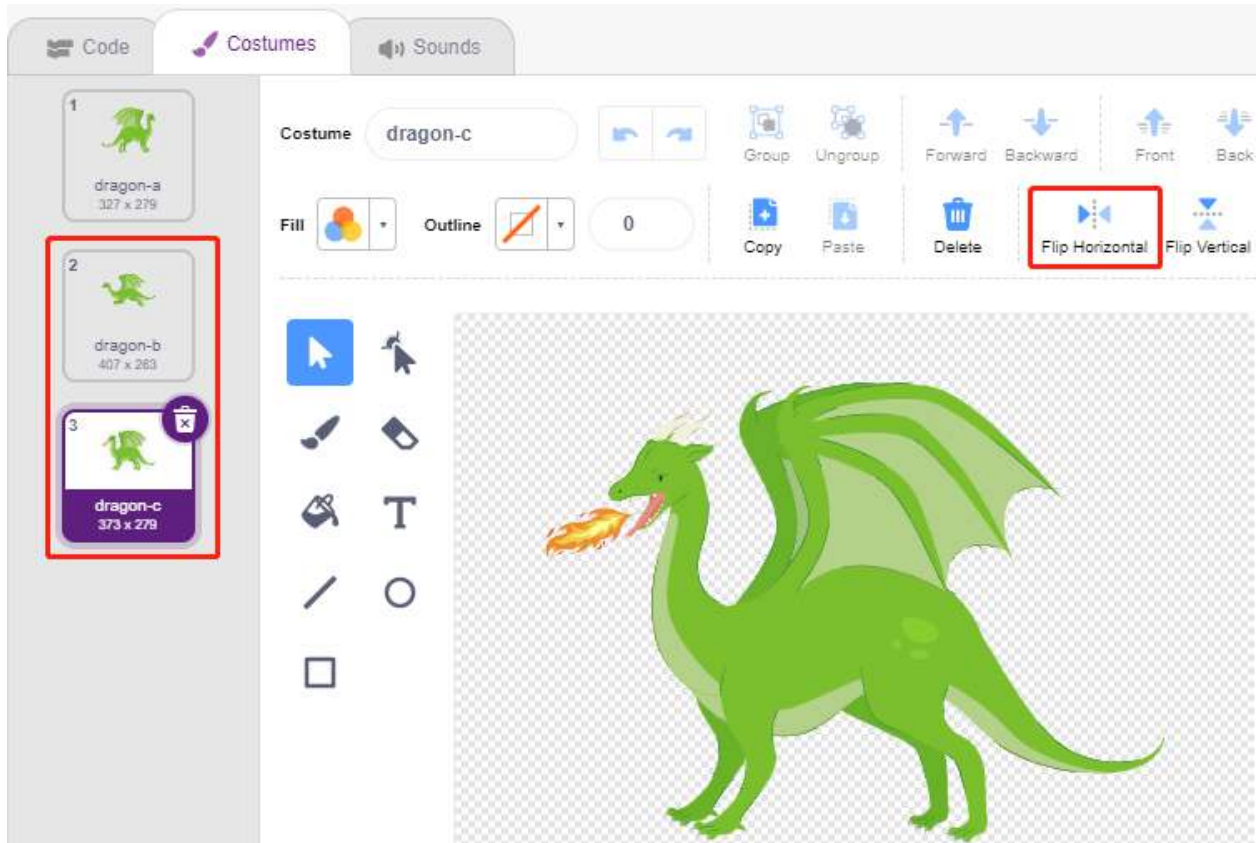
Woods backdrop added via the **Choose a Backdrop** button.



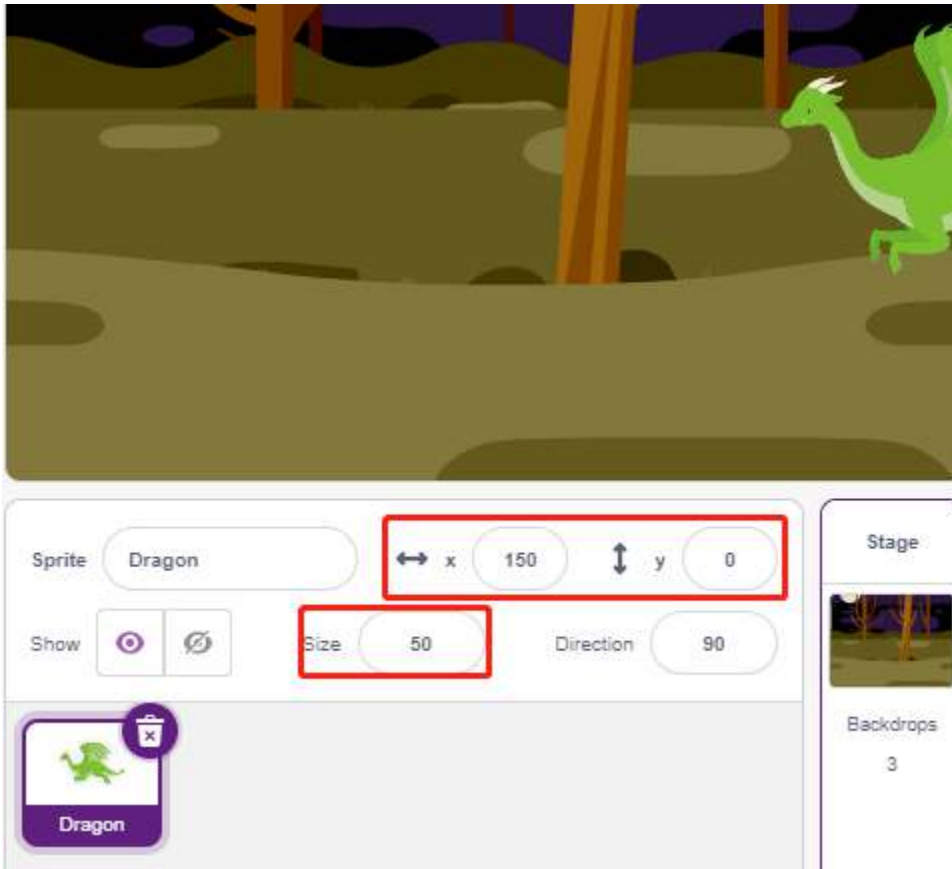
- Delete the default sprite and add the **Dragon** sprite.



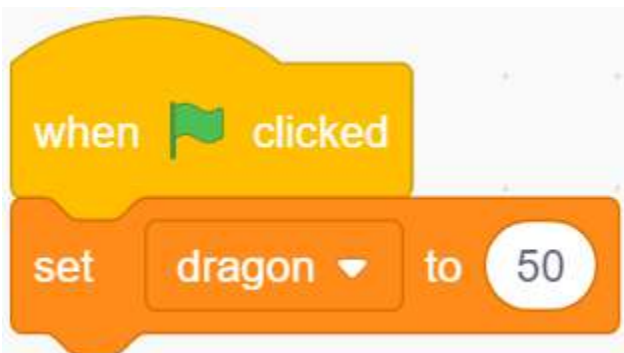
- Go to the **Costumes** page and flip the dragon-b and dragon-c horizontally.



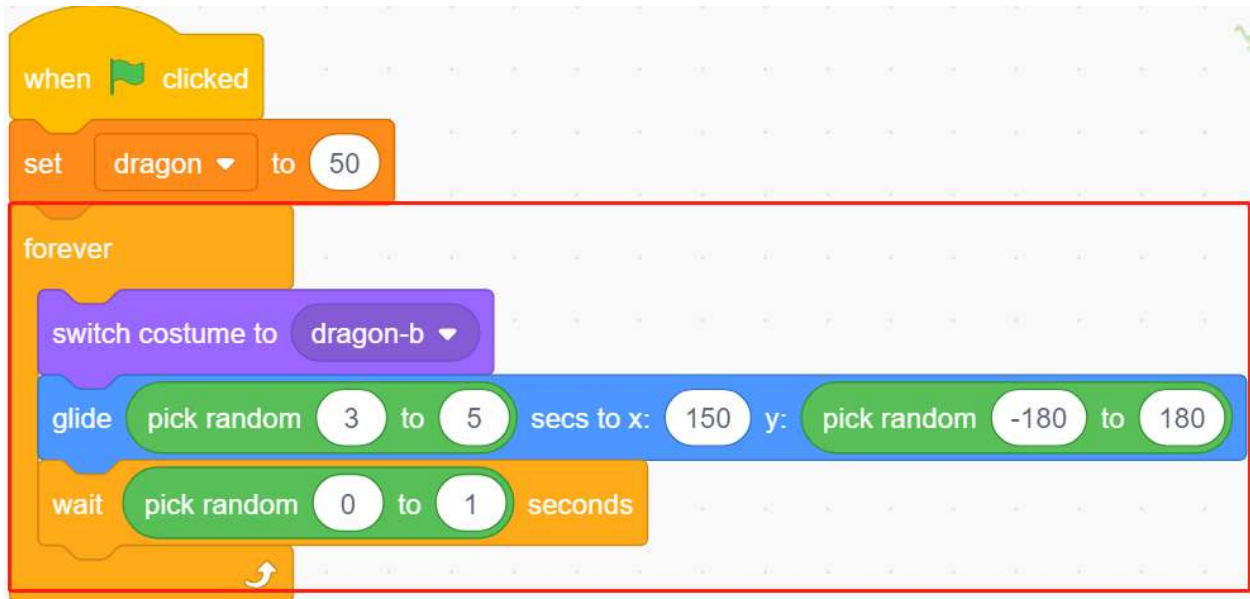
- Set the size to 50%.



- Now create a variable - **dragon** to record the dragon's life points, and set the initial value to 50.

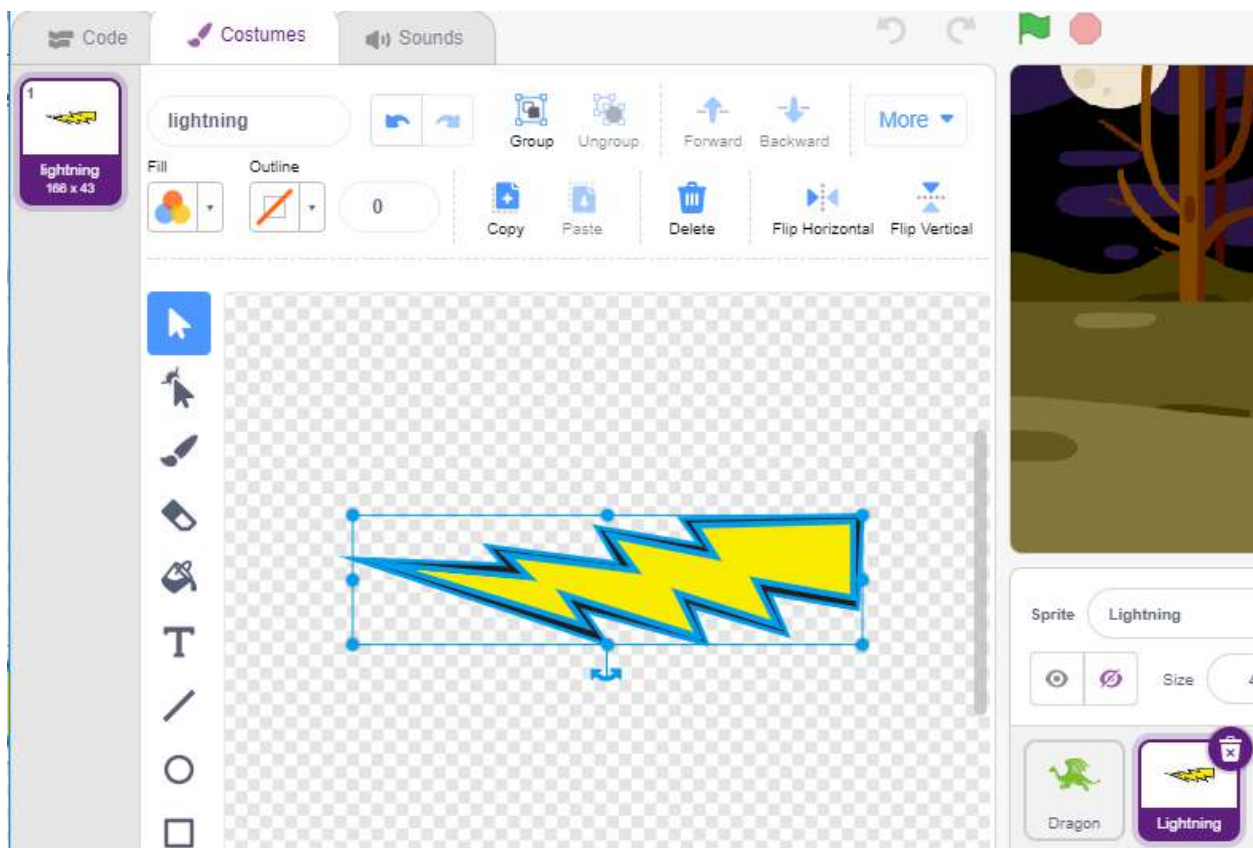


- Next, switch the sprite costume to **dragon-b** and have the **Dragon** sprite up and down in a range.

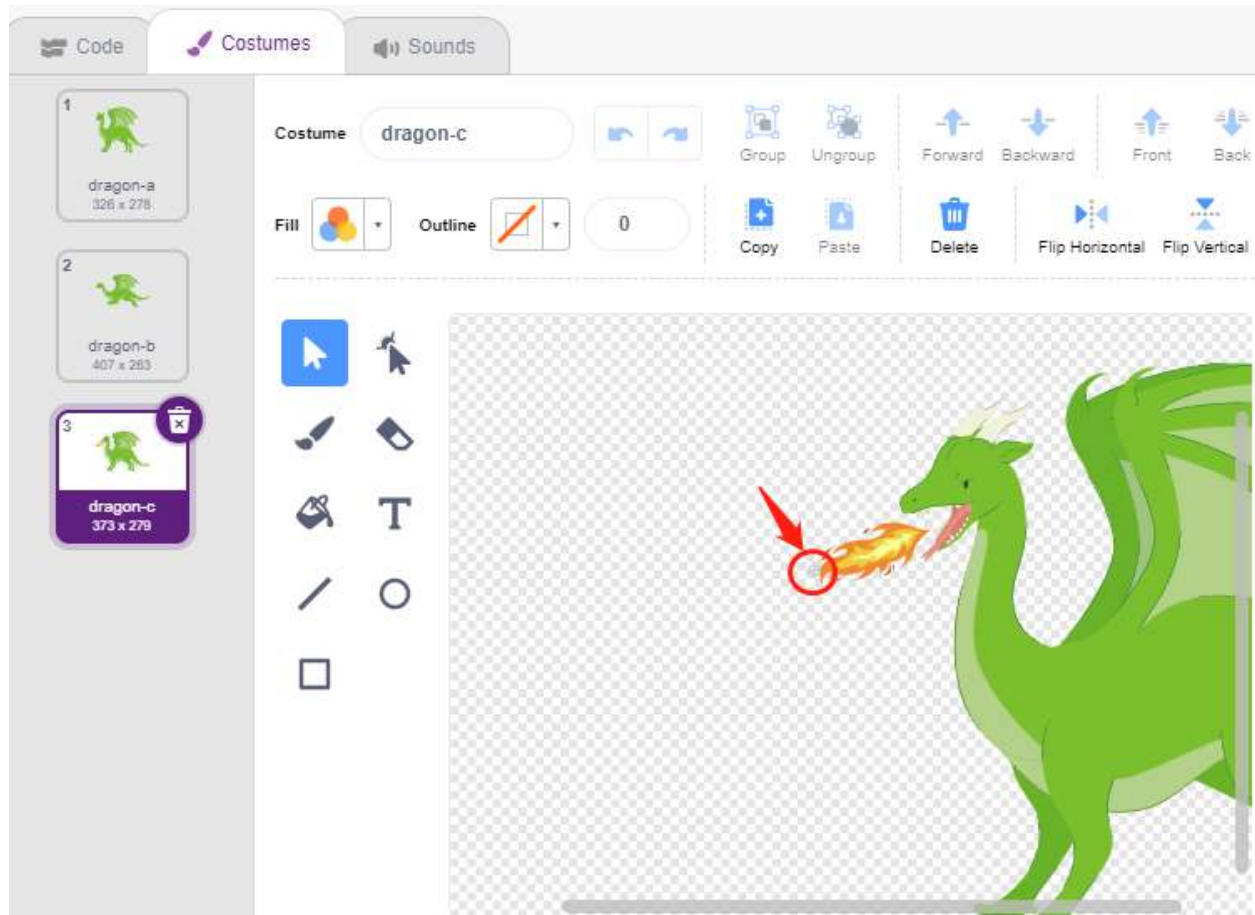


- Add a **Lightning** sprite as the fire blown by the **Dragon** sprite. You need to rotate it 90° clockwise in the Costumes page, this is to make the **Lightning** sprite move in the right direction.

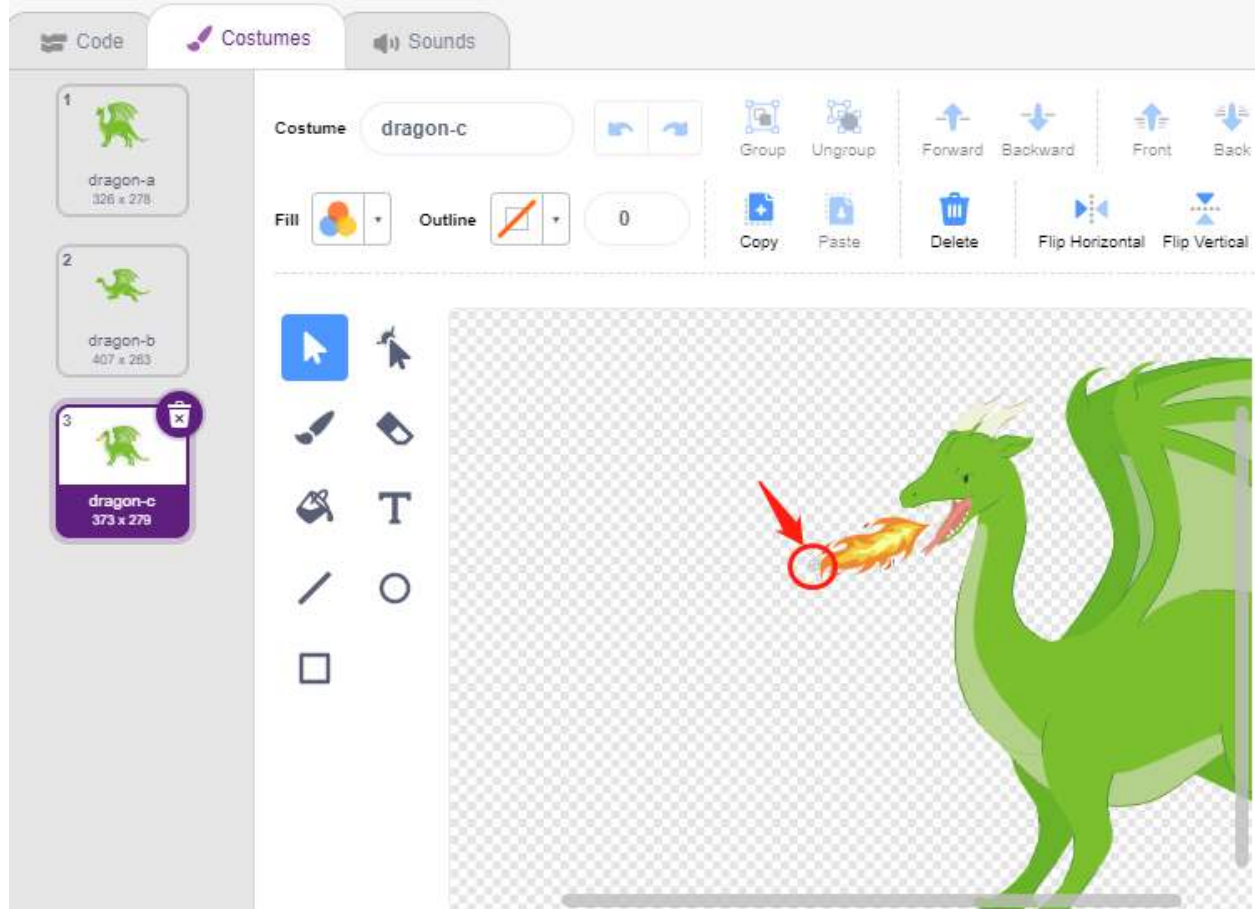
Note: When adjusting the **Lightning** sprite's costume, you may move it off-center, which must be avoided! The center point must be right in the middle of the sprite!



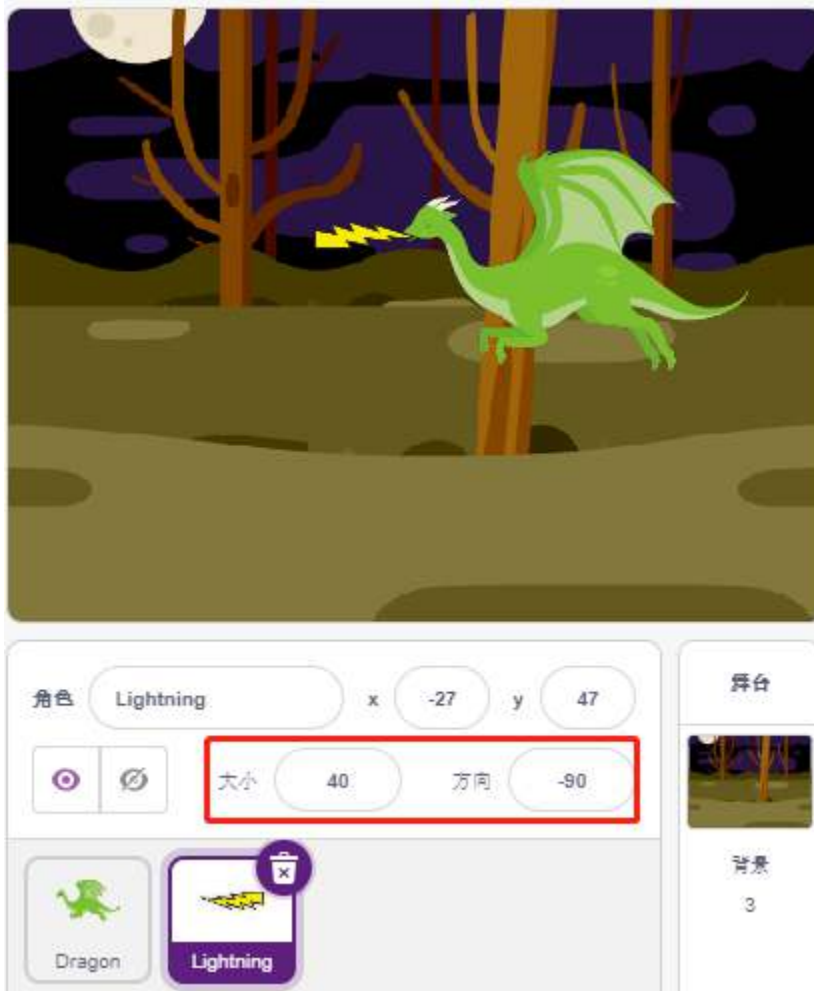
- Then adjust the **dragon-c** costume of the **Dragon** sprite so that its center point should be at the tail of the fire. This will make the positions of the **Dragon** sprite and the **Lightning** sprite correct, and prevent **Lightning** from launching from the dragon's feet.



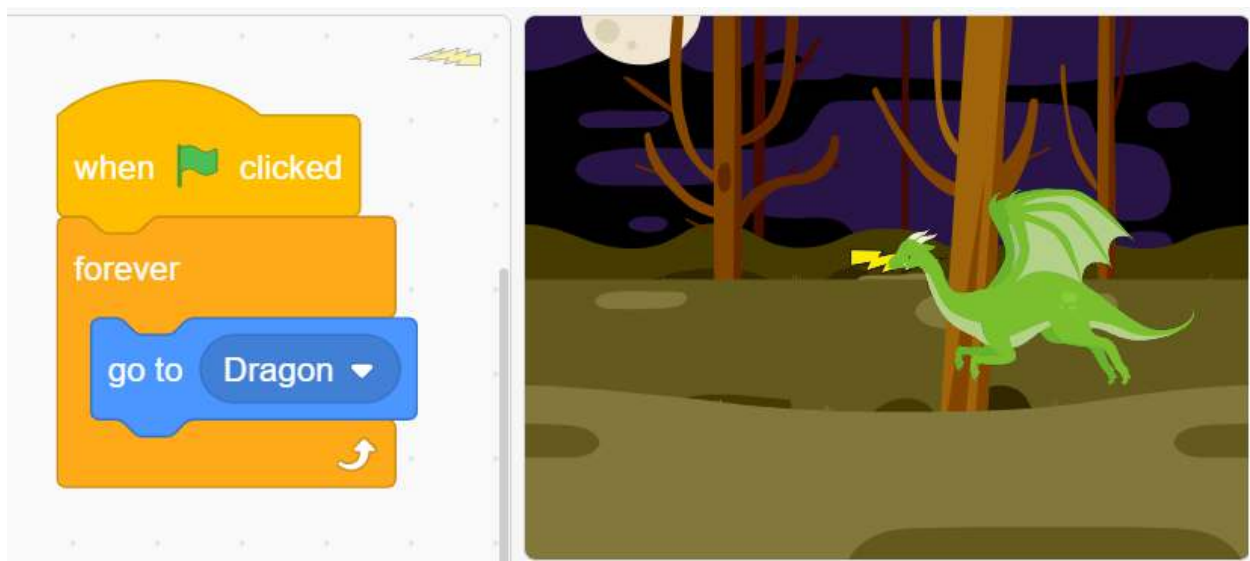
- Correspondingly, **dragon-b** needs to make the head of the dragon coincide with the center point.



- Adjust the size and orientation of the **Lightning** sprite to make the image look more harmonious.

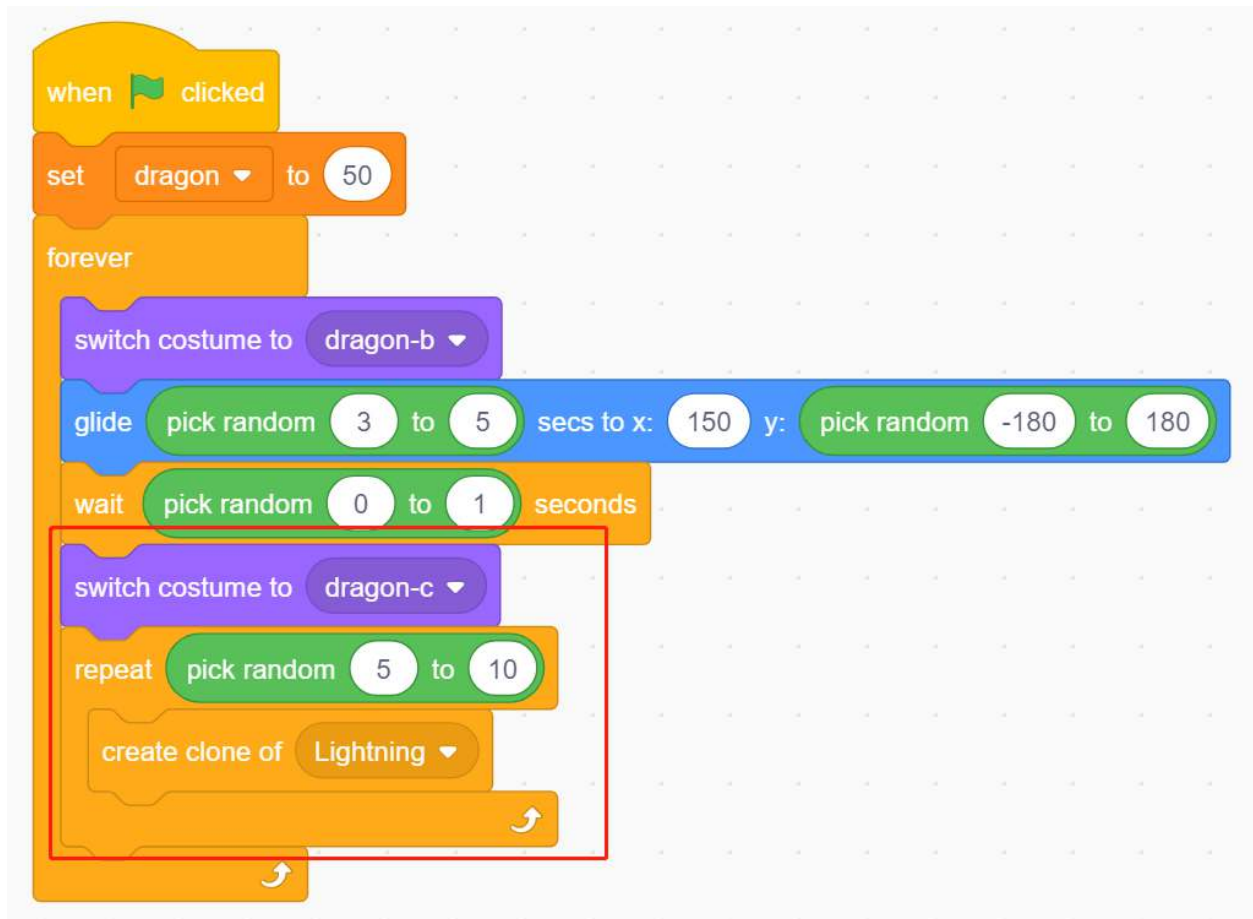


- Now script the **Lightning** sprite. This is easy, just have it follow the **Dragon** sprite all the time. At this point, click on the green flag and you will see **Dragon** moving around with lightning in its mouth.

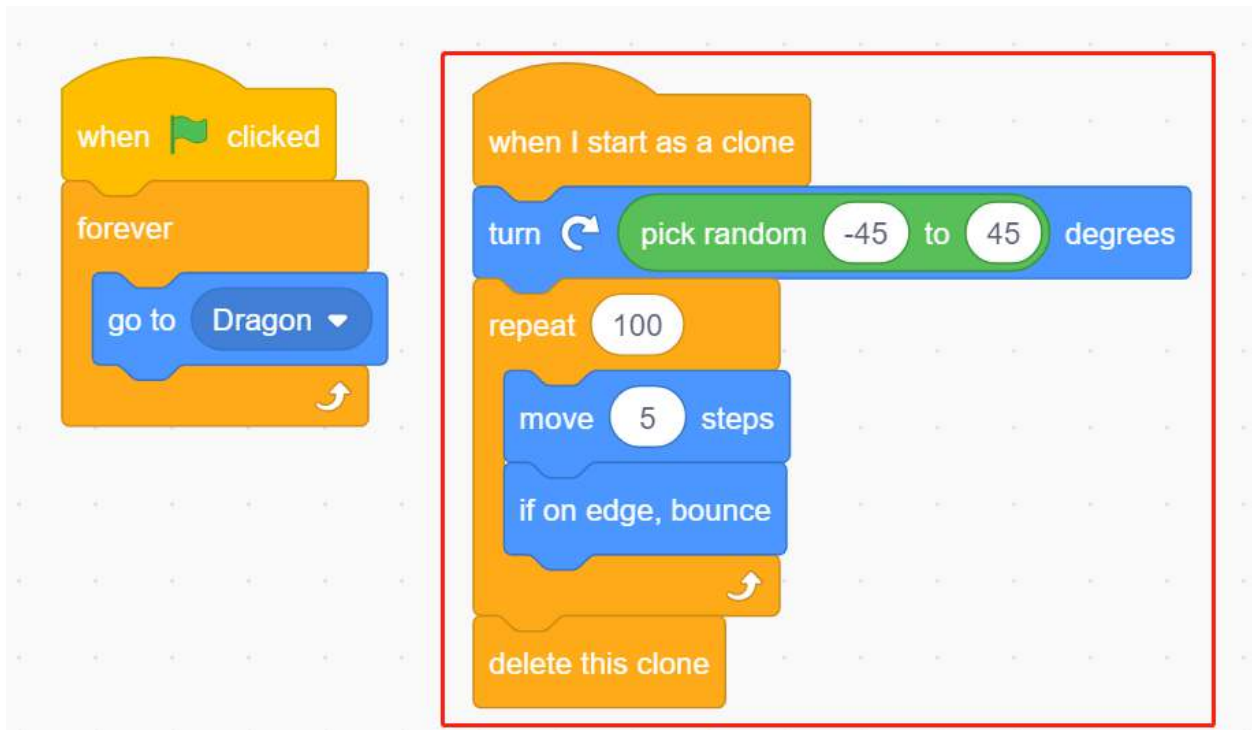


- Back to the **Dragon** sprite, now have it blow out fire, being careful not to let the fire in its mouth shoot out, but

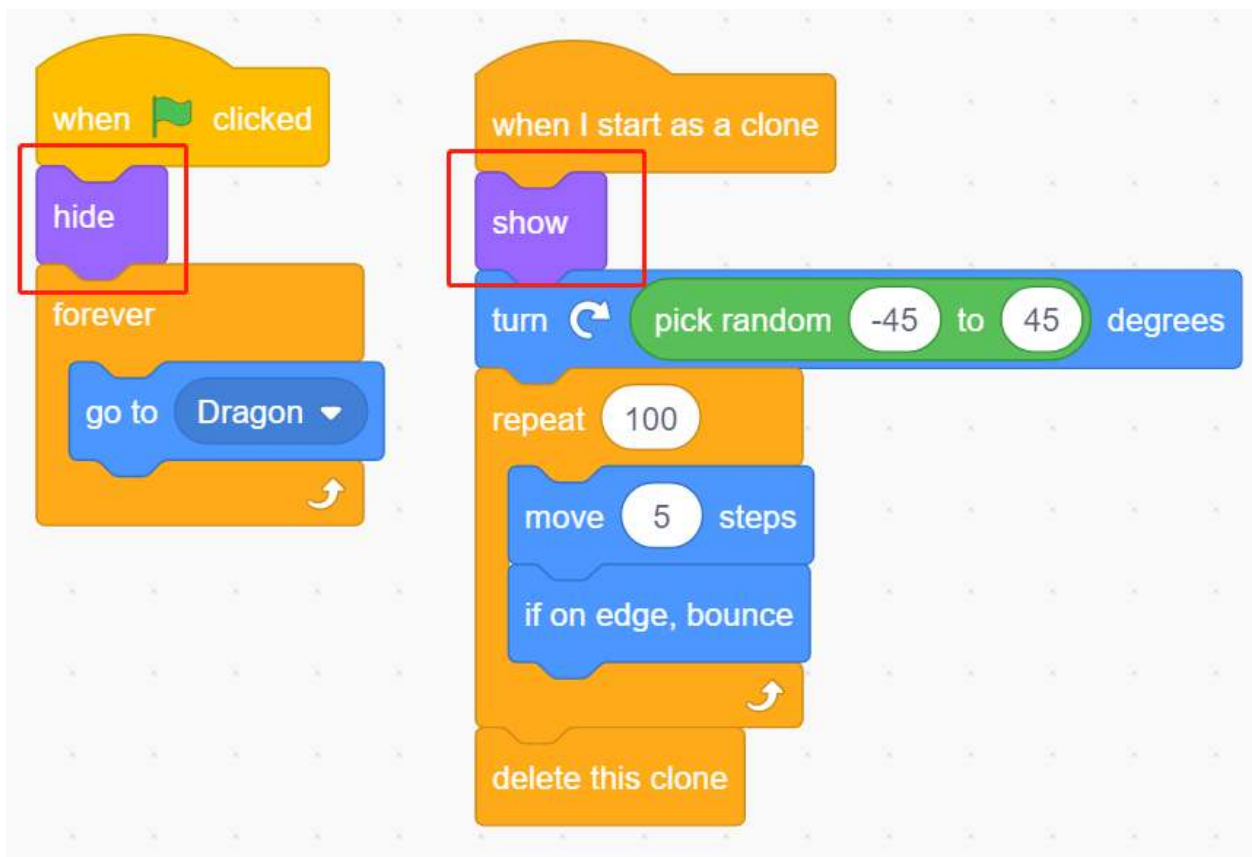
to create a clone for the **Lightning** sprite.



- Click on the **Lightning** sprite and let the **Lightning** clone shoot out at a random angle, it will bounce off the wall and disappear after a certain amount of time.



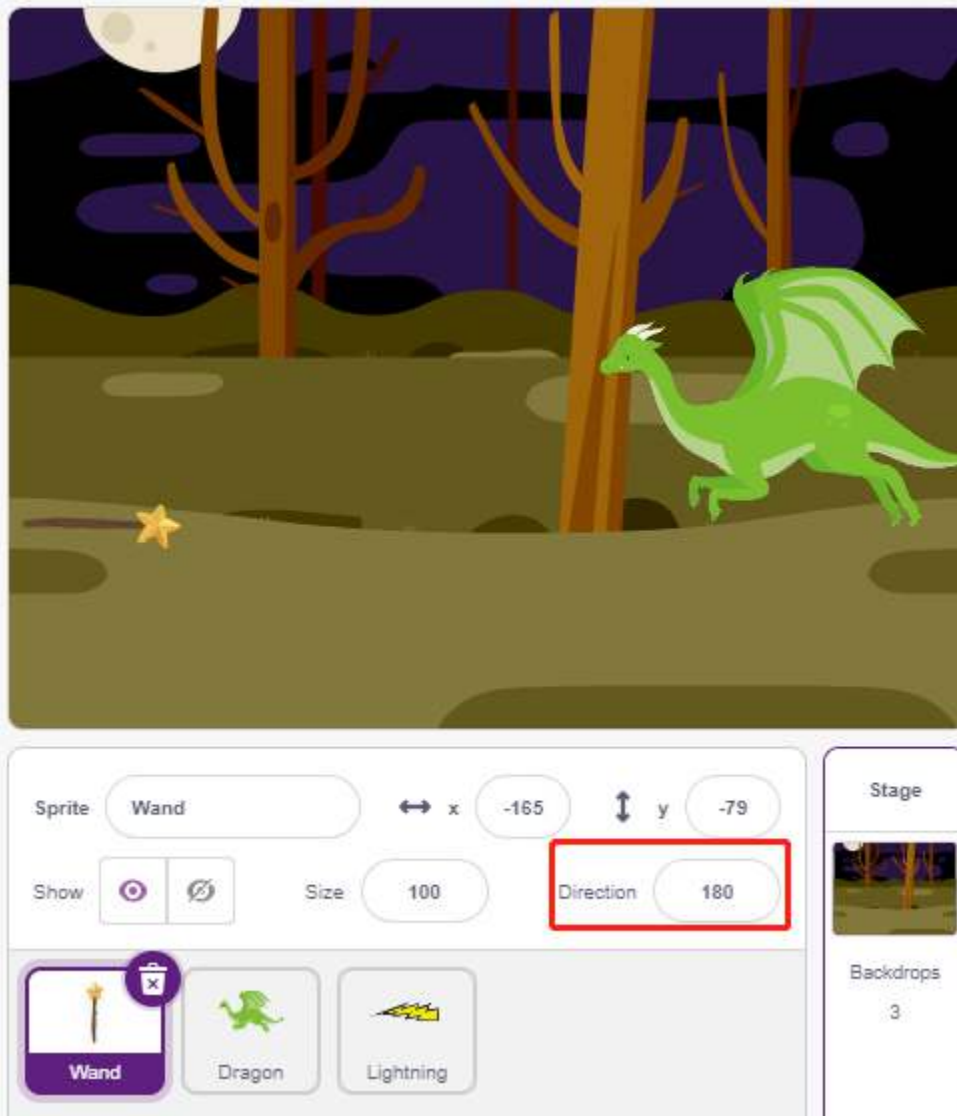
- In the **Lightning** sprite, hide its body and show the clone.



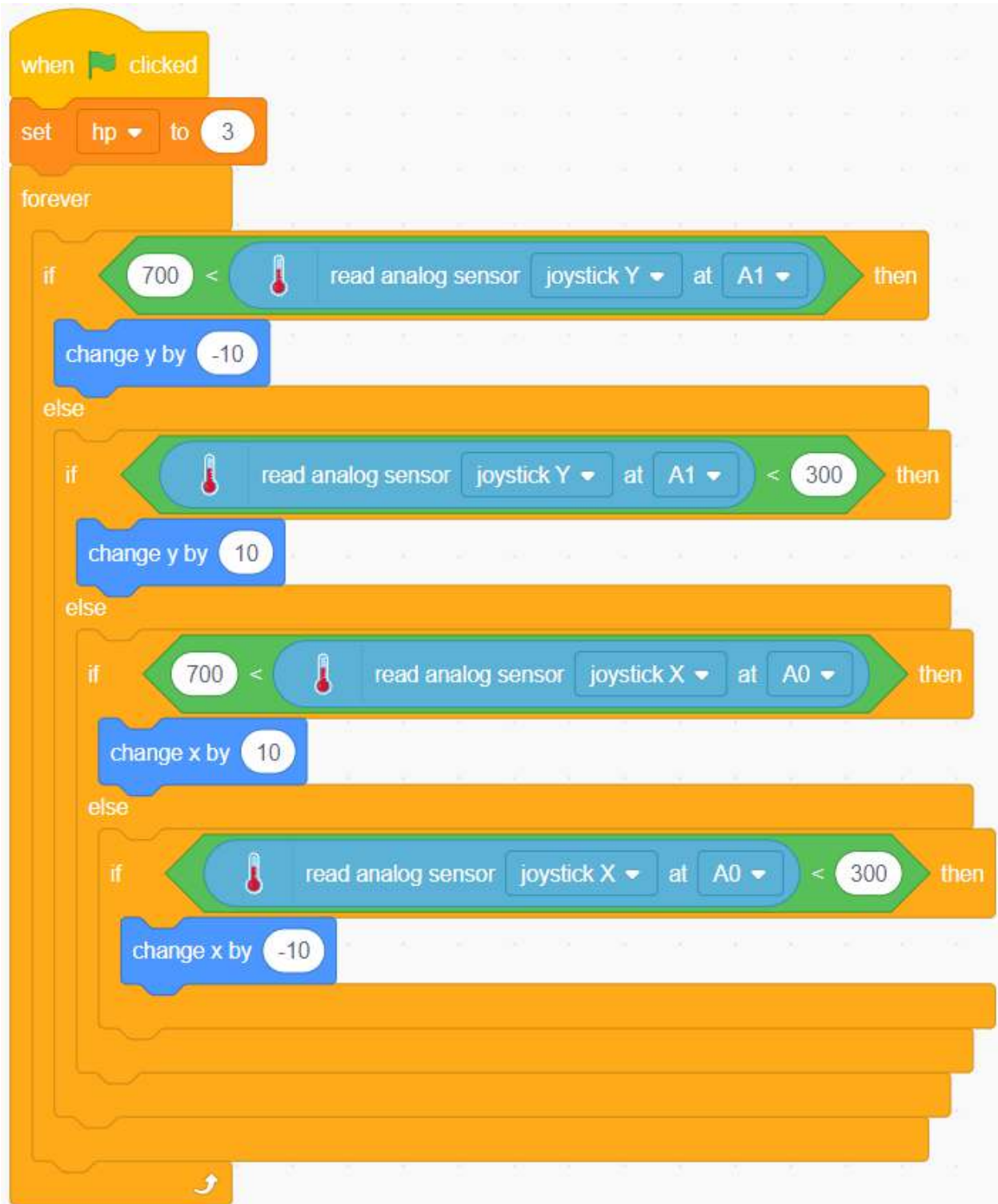
Now the dragon can move up and down and blow out fire.

2.Wand

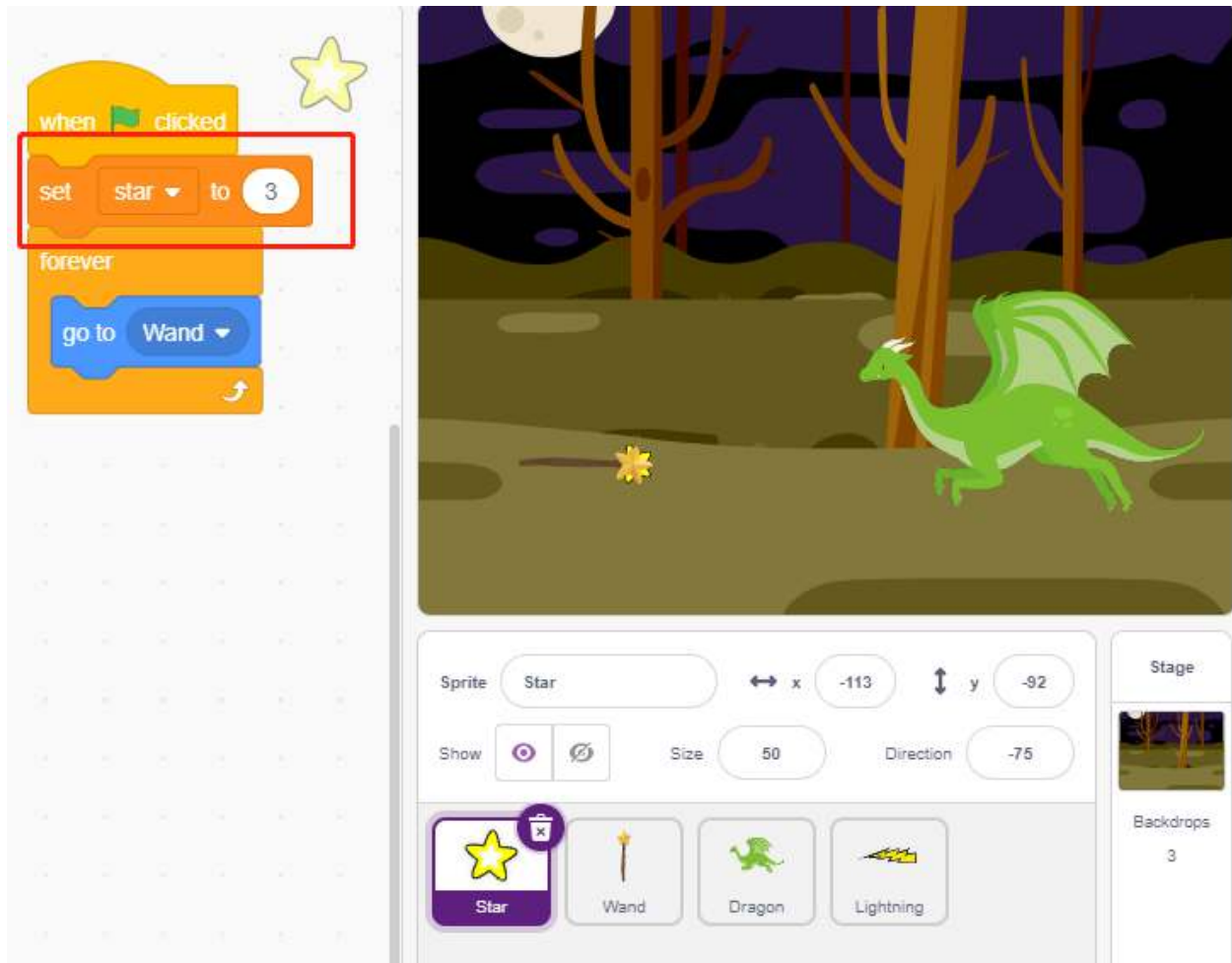
- Create a **Wand** sprite and rotate its direction to 180 to point to the right.



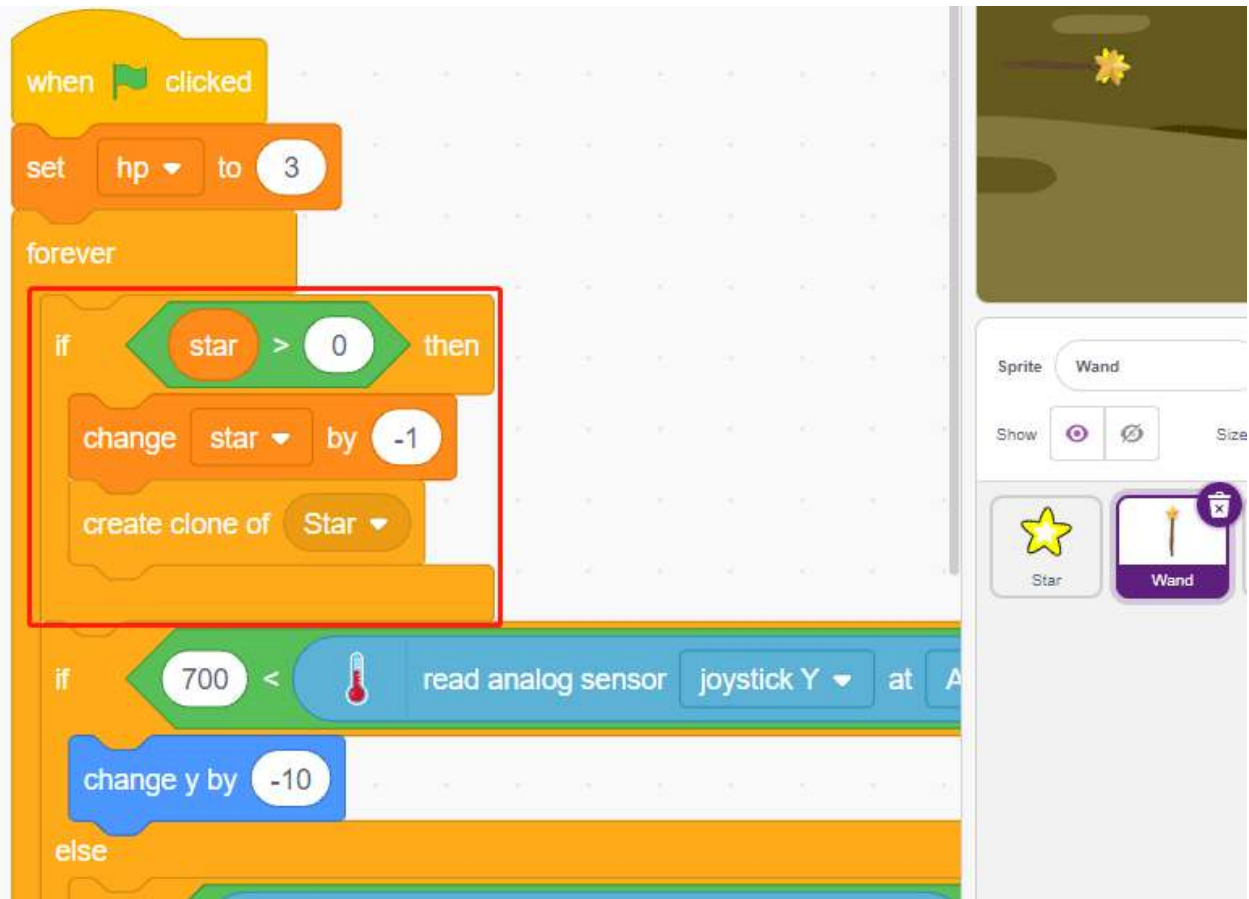
- Now create a variable **hp** to record its life value, initially set to 3. Then read the Joystick's value, which is used to control the wand's movement.



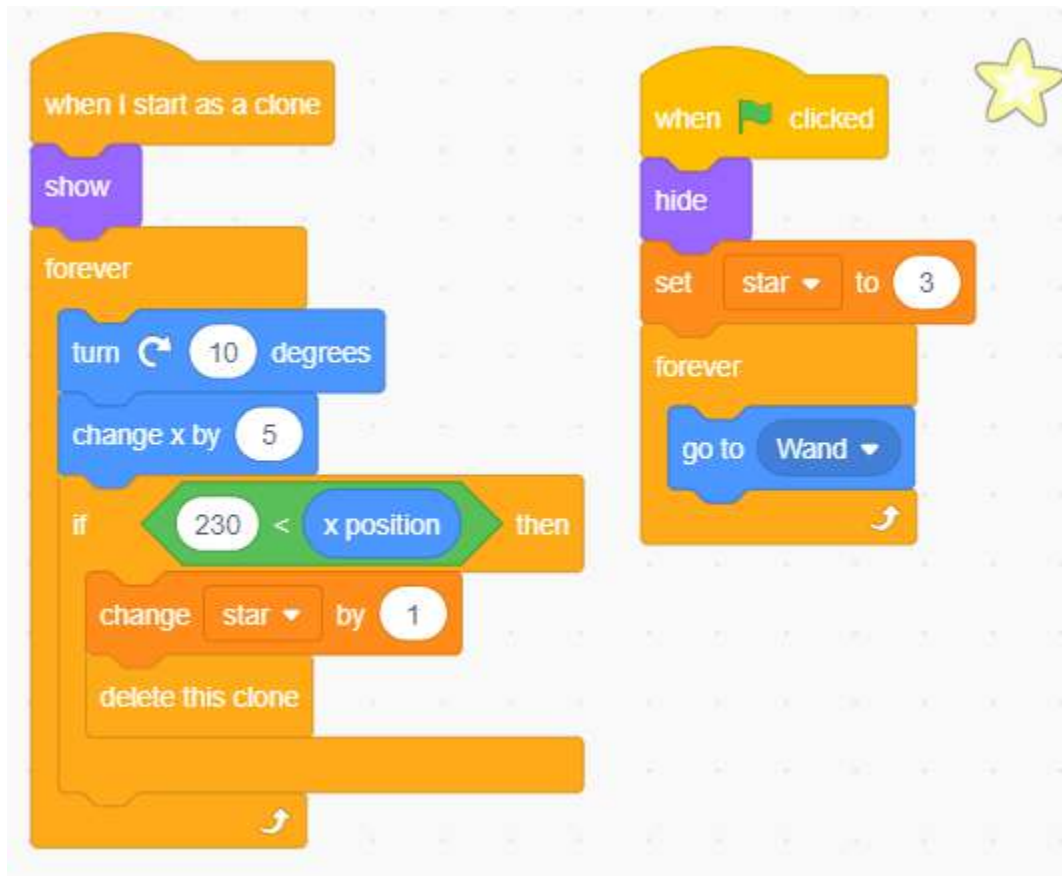
- The dragon has lightning, and the wand that crushes it has its “magic bullet”! Create a **Star** sprite, resize it, and script it to always follow the **Wand** sprite, and limit the number of stars to three.



- Make the **Wand** sprite shoot stars automatically. The **Wand** sprite shoots stars the same way the dragon blows fire – by creating clones.



- Go back to the **Star** sprite and script its clone to spin and shoot to the right, disappear after going beyond the stage and restoring the number of stars. Same as **Lightning** sprite, hide the body and show the clone.



Now we have a wand that shoots star bullets.

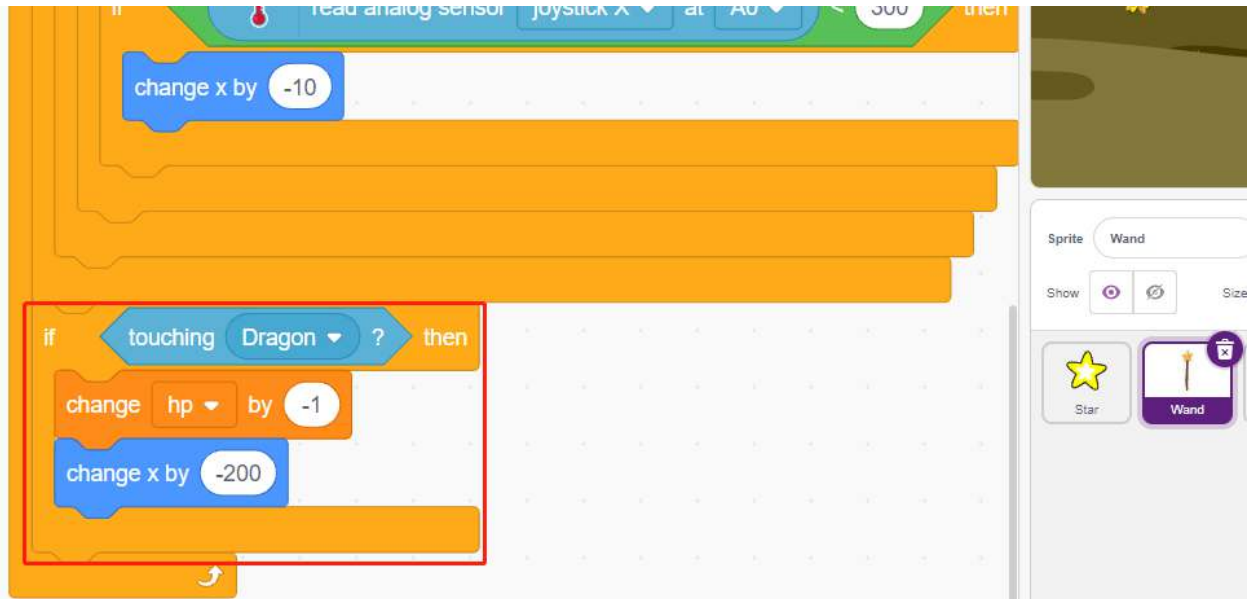
3. Fight!

The wand and the dragon are currently still at odds with each other, and we're going to make them fight. The dragon is strong, and the wand is the brave man who crusades against the dragon. The interaction between them consists of the following parts.

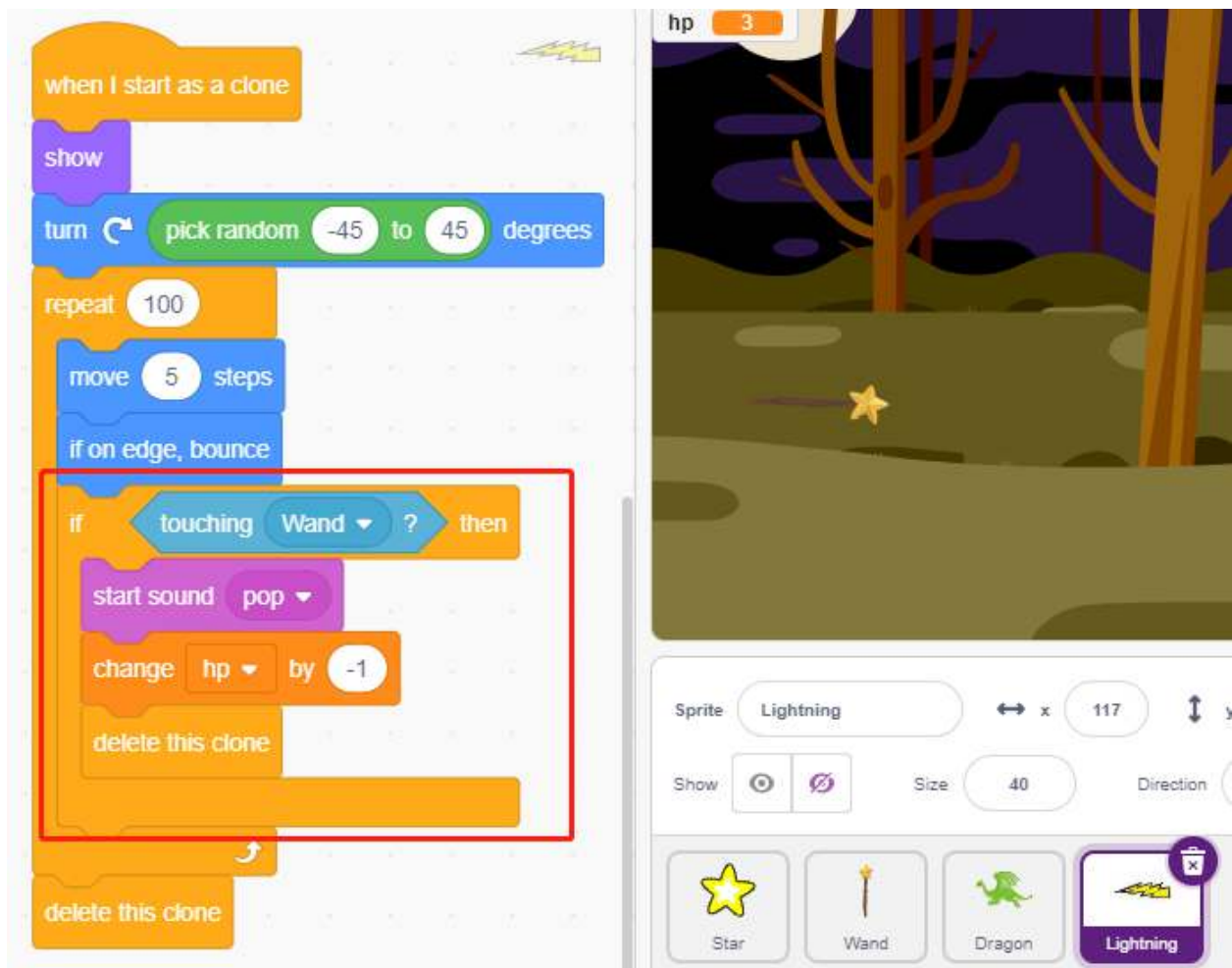
1. if the wand touches the dragon, it will be knocked back and lose life points.
2. if lightning strikes the wand, the wand will lose life points.
3. if the star bullet hits the dragon, the dragon will lose life points.

Once that's sorted out, let's move on to changing the scripts for each sprite.

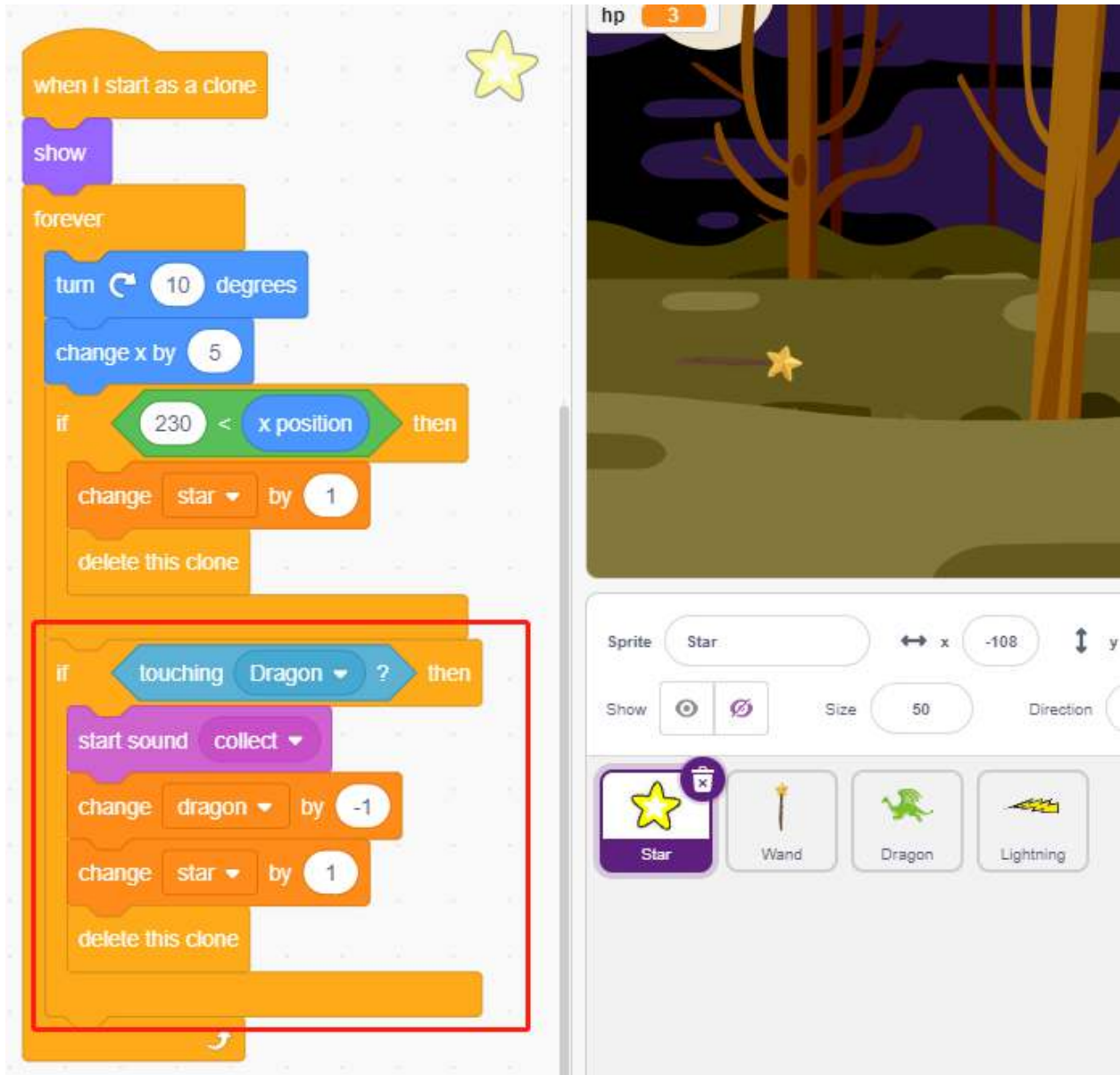
- If the **Wand** hits the **Dragon**, it will be knocked back and lose life points.



- If **Lightning** (a **Lightning** sprite clone) hits the **Wand** sprite, it will make a pop sound and disappear, and the **Wand** will lose life points.



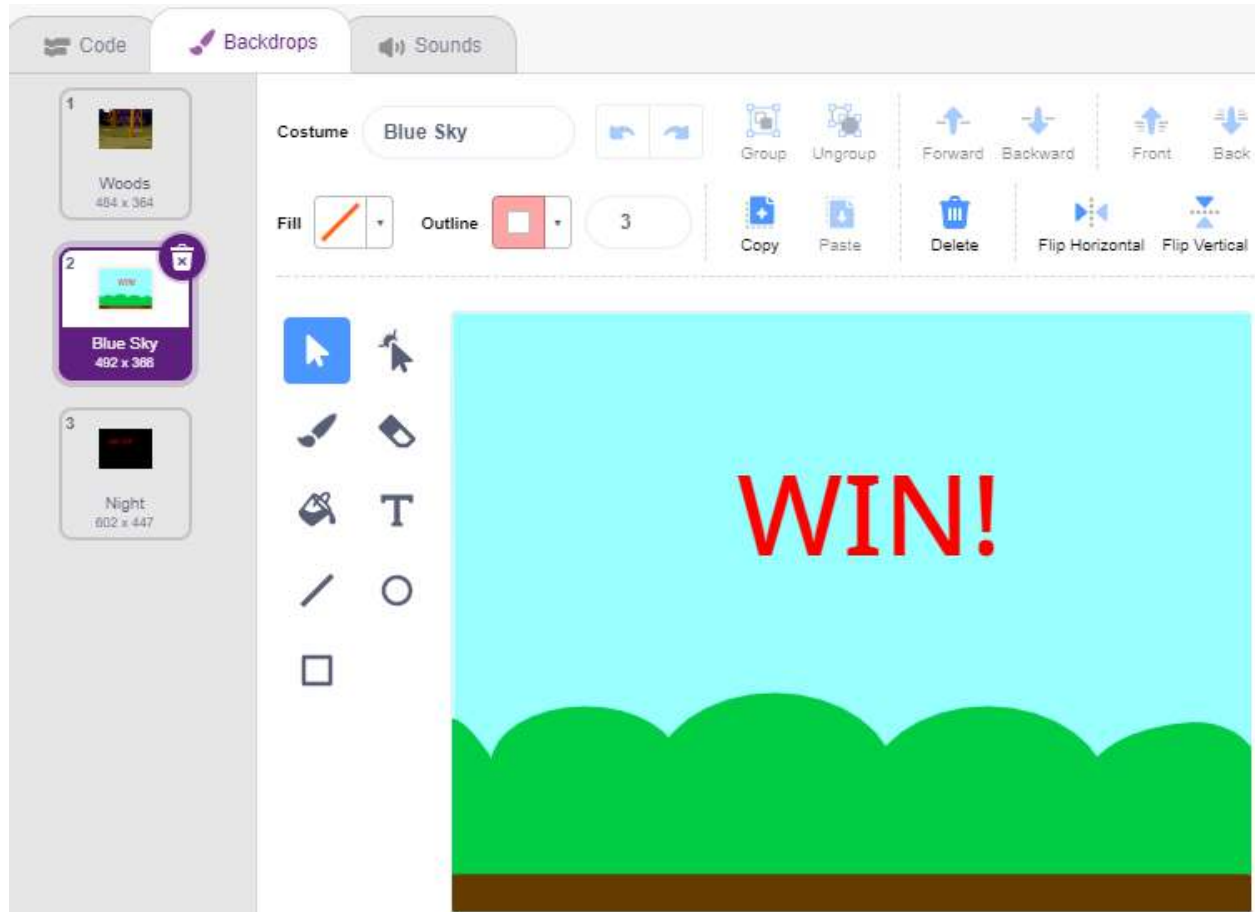
- If a **Star** (clone of the **Star** sprite) hits the **Dragon**, it will emit a collect sound and disappear, while restoring the **Star** count, and the **Dragon** will lose life points.



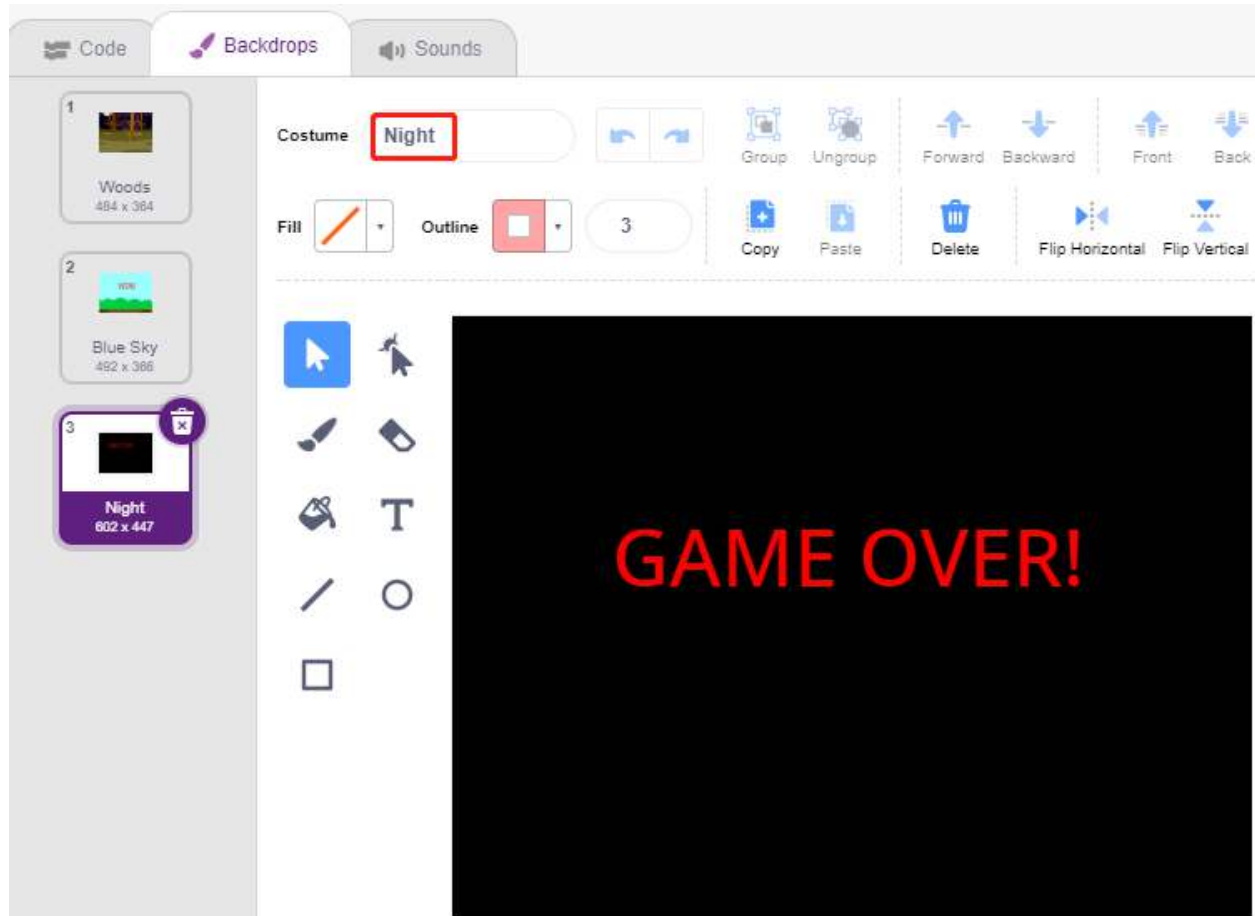
4. stage

The battle between the **Wand** and the **Dragon** will eventually be divided into winners and losers, which we represent with the stage.

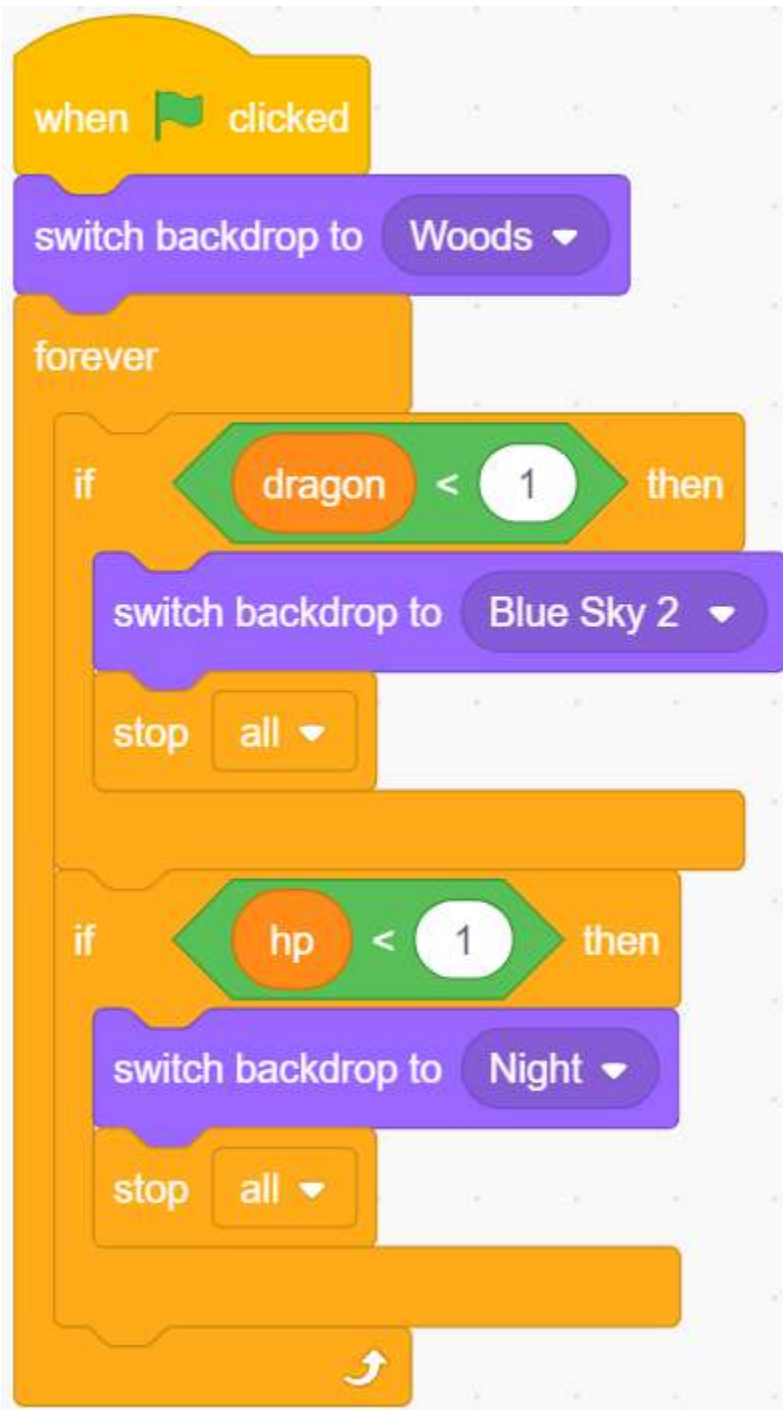
- Add **Blue Sky** backdrop, and write the character “WIN!” on it to represent that the dragon has been defeated and the dawn has come.



- And modify the blank backdrop as follows, to represent that the game has failed and everything will be in darkness.



- Now write a script to switch these backdrops, when the green flag is clicked, switch to **Woods** backdrop; if the dragon's life point is less than 1, then the game succeeds and switch the backdrop to **Blue Sky**; if the life value point of the **Wand** is less than 1, then switch to **Night** backdrop and the game fails.



THANK YOU

Thanks to the evaluators who evaluated our products, the veterans who provided suggestions for the tutorial, and the users who have been following and supporting us. Your valuable suggestions to us are our motivation to provide better products!

Particular Thanks

- Len Davisson
- Kalen Daniel
- Juan Delacosta

Now, could you spare a little time to fill out this questionnaire?

Note: After submitting the questionnaire, please go back to the top to view the results.

GERMAN ONLINE-TUTORIALS

- [German Online-Tutorials](#)

COPYRIGHT NOTICE

All contents including but not limited to texts, images, and code in this manual are owned by the SunFounder Company. You should only use it for personal study, investigation, enjoyment, or other non-commercial or nonprofit purposes, under the related regulations and copyrights laws, without infringing the legal rights of the author and relevant right holders. For any individual or organization that uses these for commercial profit without permission, the Company reserves the right to take legal action.