

RL-SET-UKS




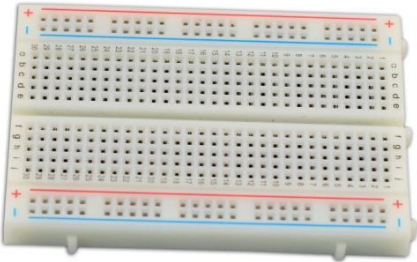

HESTORE no.: 100.371.40




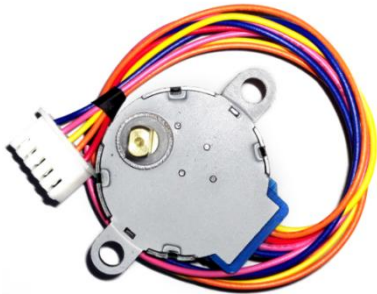
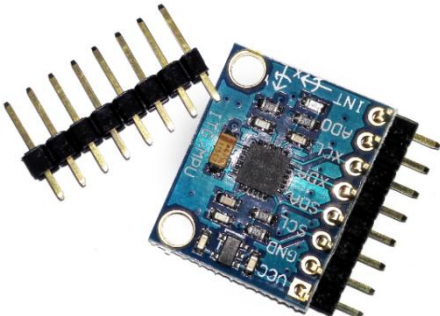
UNO R3 Ultimate LEARNING KIT TUTORIAL

Contents









Preface	2
Components List	4
Lesson 0 Installing IDE.....	23
Lesson 1 Blink.....	23
Lesson 2 Button	30
Lesson 3 Flowing LED Lights	34
Lesson 4 Active Buzzer	37
Lesson 5 Passive Buzzer.....	40
Lesson 6 Photoresistor.....	43
Lesson 7 RGB LED.....	46
Lesson 8 Servo	49
Lesson 9 LCD1602.....	51
Lesson 10 Thermistor	54
Lesson 11 Voltmeter	56
Lesson 12 Ultrasonic	58
Lesson 13 Stopwatch	61
Lesson 14 74HC595 And Segment Display	64
Lesson 15 JOYSTICK	69
Lesson 16 1 CHANNEL RELAY MODULE.....	71
Lesson 17 DC Motors.....	73
Lesson 18 DC Motors Reversing.....	78
Lesson 19 Steeper Motor	83
Lesson 20 Automatically Tracking Light Source	88
Lesson 21 Packing	91
Lesson 22 MPU6050	94
Lesson 23 STEPPER MOTOR AND ULN2003.....	100







Components List

No	Product Name	Quantity	Picture
1	Uno R3 Controller Board	1	
2	USB Cable	1	
3	LCD1602 Module	1	
4	Breadboard	1	
5	9V 1A Power Supply	1	


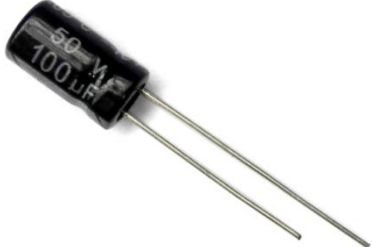



6	Prototype Expansion Board	1	
7	Ultrasonic Sensor	1	
8	ULN2003 Stepper Motor Driver Board	1	
9	Stepper motor	1	
10	MPU 6050	1	






11	Joystick	1	
12	Relay	1	
13	Remote	1	
14	Servo Motor	1	
15	130 Servo Motor	1	







16	Resistor (10Ω)	10	
17	Resistor (100Ω)	10	
18	Resistor (220Ω)	10	
19	Resistor (330Ω)	10	
20	Resistor (1kΩ)	10	
21	Resistor (2kΩ)	10	
22	Resistor (5.1kΩ)	10	
23	Resistor (10kΩ)	10	






24	Resistor (100k Ω)	10	
25	Resistor (1m Ω)	10	
26	Ultrasonic Holder	1	
27	1 digit 7-segment display	1	
28	4 digit 7-segment display	1	
29	Matrix display	1	



30	BC547	5	
31	BC557	5	
32	Pn2222	5	
33	MAX7219	1	
34	L293D	1	

35	74HC595	1	
36	100UF 50V	2	
37	10UF 50V	2	
38	22pf ceramic capacitor	5	
39	104 ceramic capacitor	5	

40	IR receiver	1	
41	Diode Rectifier (1N4007)	5	
42	Photoresistor (Photocell)	2	
43	Rotary Knob (Potentiometer)	2	
44	RGB LED	1	

45	Red LED	5	
46	Yellow LED	5	
47	White LED	5	
48	Green LED	5	
49	Blue LED	5	
50	Active Buzzer	1	

51	Passive Buzzer	1	
52	Thermistor	1	
53	Button (Small)	5	
54	Pin Header	1	
55	65 jumper wire	1	

56	Female-Male cable	4	
57	9v battery with DC	1	

Note:

After unpacking, please check that the number of components is correct and that all components are in good condition.

Lesson 0 Installing IDE

Introduction

In this lesson, you will learn how to setup your computer to use Arduino and how to set about the lessons that follow.

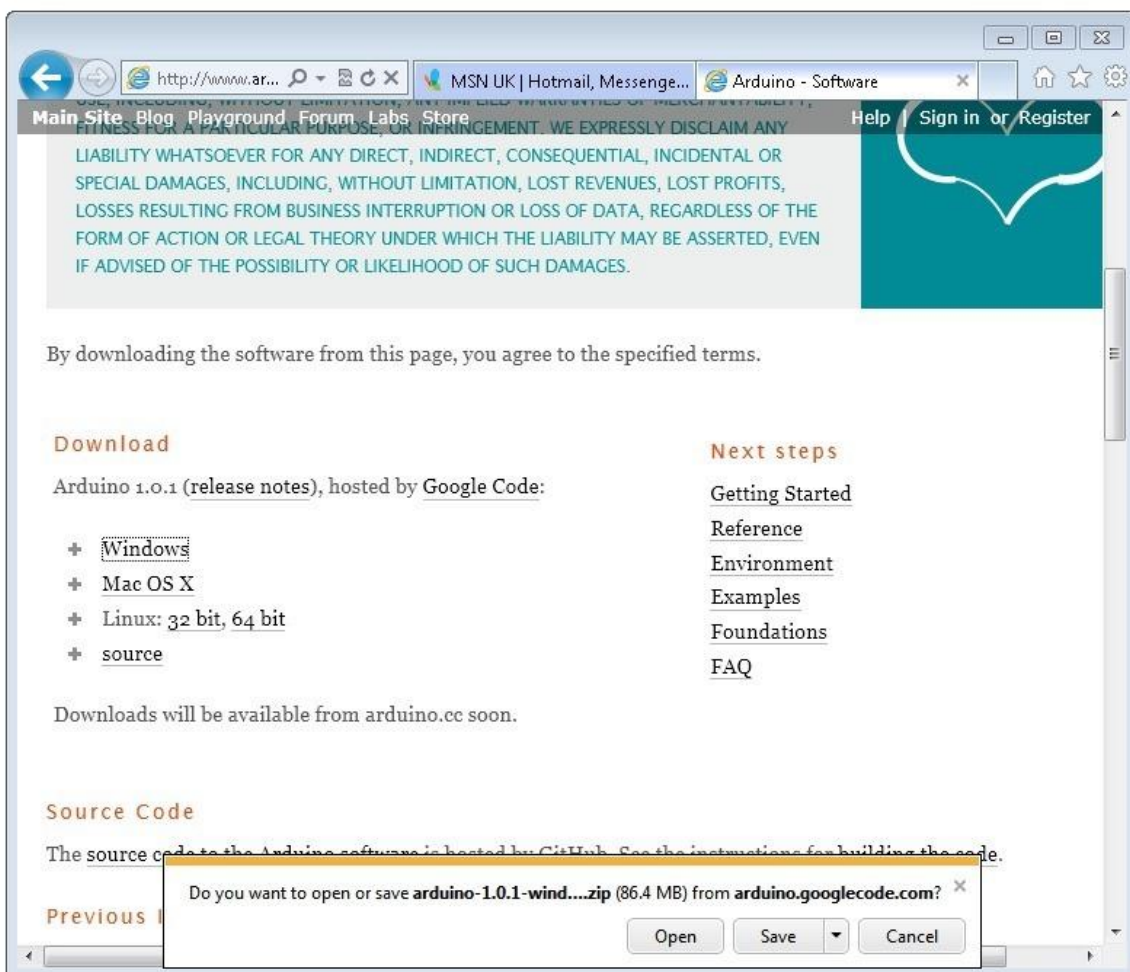
Installing Arduino (Windows)

The Arduino software that you will use to program your Arduino is available for Windows, Mac and Linux. The installation process is different for all three platforms and unfortunately there is a certain amount of manual work to install the software. There is no installer program, but rather you have to unzip a folder which gives you an Arduino folder that contains the Arduino program and a few other items.

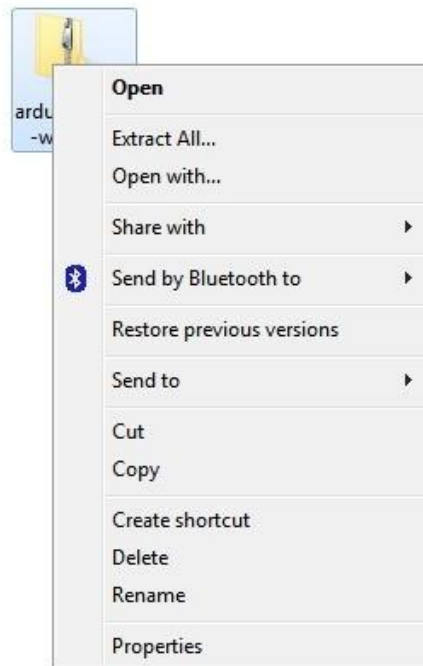
In a separate step, you must then install USB drivers, which is the only bit that is a bit fiddly.

Get started by visiting the [Arduino.cc](http://arduino.cc) website. As of April 2014 we suggest v1.05 as 1.5 is still in beta. If 1.5 is no longer in beta when you read this you can try it out!

Start by downloading the the zip file for Windows. There is only one version of the software, whether you are using Windows XP through to Windows 7.



When the zip file has downloaded, extract the contents onto the Desktop, by right-clicking on the file and selecting 'Extract All...' from the pop-up menu.



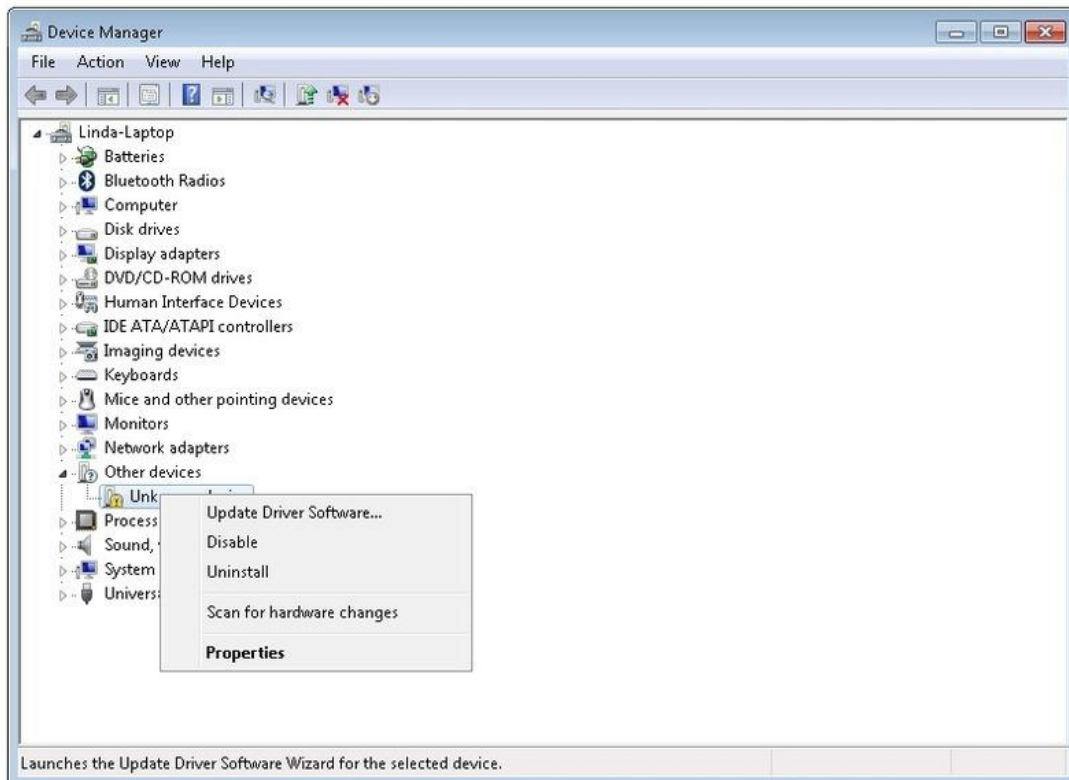
Next select your Desktop and click 'Extract'. You can move it somewhere else onto your computer later, just by moving the folder, but for now, just keep it on the Desktop.

The Arduino folder contains both the Arduino program itself and also the drivers that allow the Arduino to be connected to your computer by a USB cable. Before we launch the Arduino software, you are going to install the USB drivers.

Plug one end of your USB cable into the Arduino and the other into a USB socket on your computer. The power light on the LED will light up and you may get a 'Found New Hardware' message from Windows. Ignore this message and cancel any attempts that Windows makes to try and install drivers automatically for you.

The most reliable method of installing the USB drivers is to use the Device Manager. This is accessed in different ways depending on your version of Windows. In Windows 7, you first have to open the Control Panel, then select the option to view Icons, and you should find the Device Manager in the list.

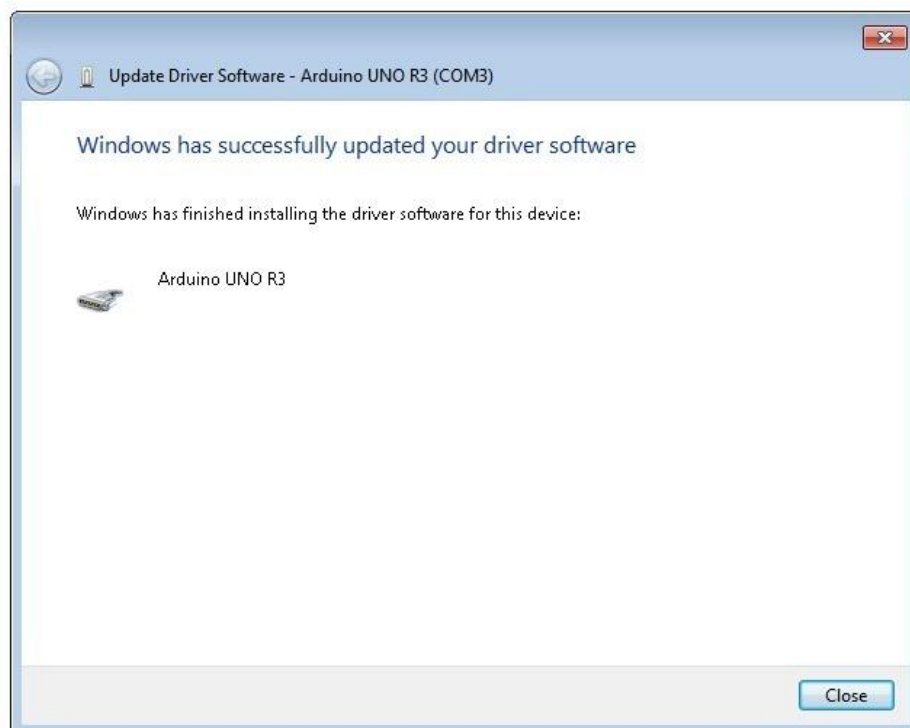
Under the section 'Other Devices' you should see an icon for 'unknown device' with a little yellow warning triangle next to it. This is your Arduino.



Right-click on the device and select the top menu option (Update Driver Software...). You will then be prompted to either 'Search Automatically for updated driver software' or 'Browse my computer for driver software'. Select the option to browse and navigate to the `arduino-1.0.2-windows\arduino1.0.2\drivers`.



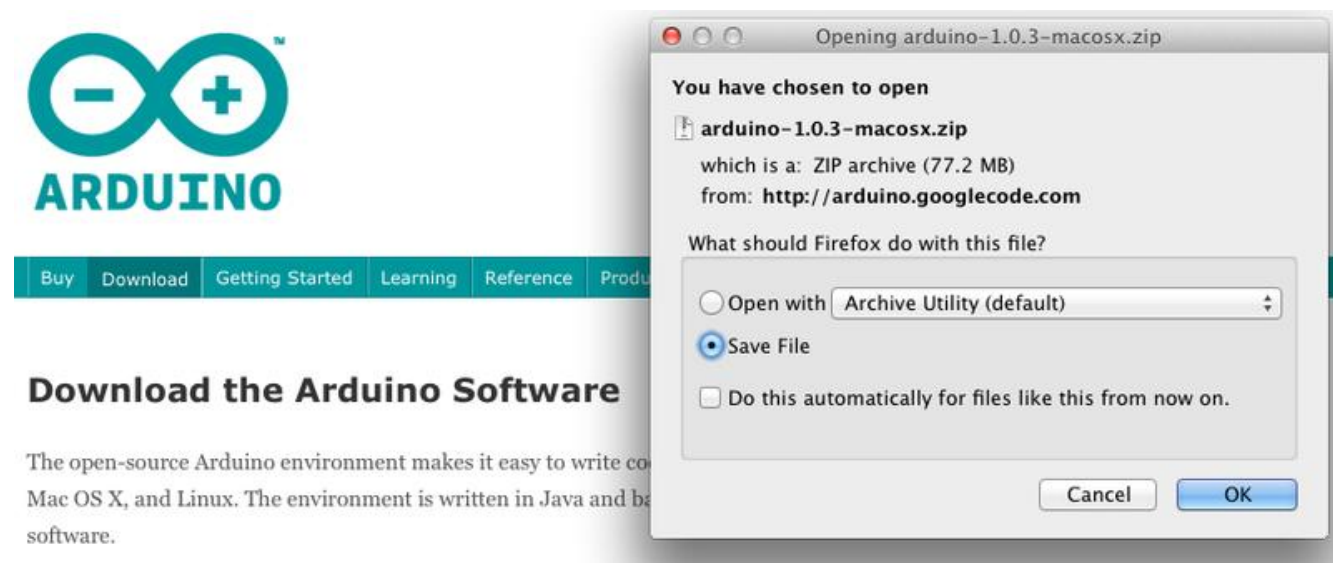
Click 'Next' and you may get a security warning, if so, allow the software to be installed. Once the software has been installed, you will get a confirmation message.



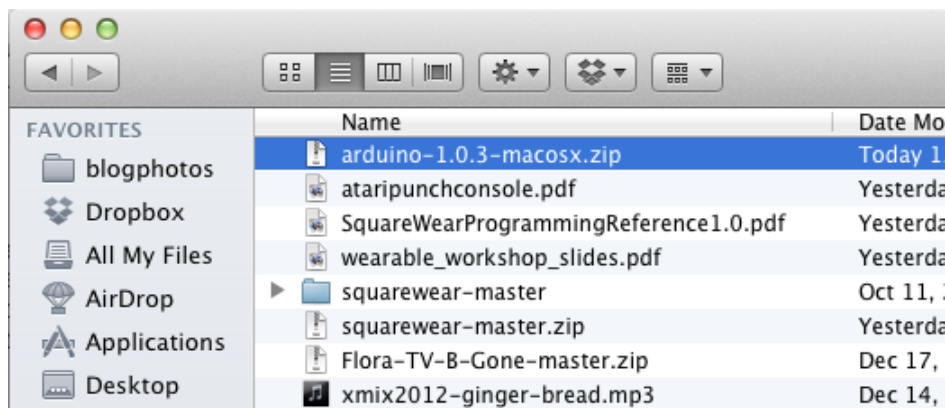
That's it, you are now ready for action, so Skip the next section on installation on Mac and Linux and move straight on to 'Boards and Ports'.

Installing Arduino (Mac and Linux)

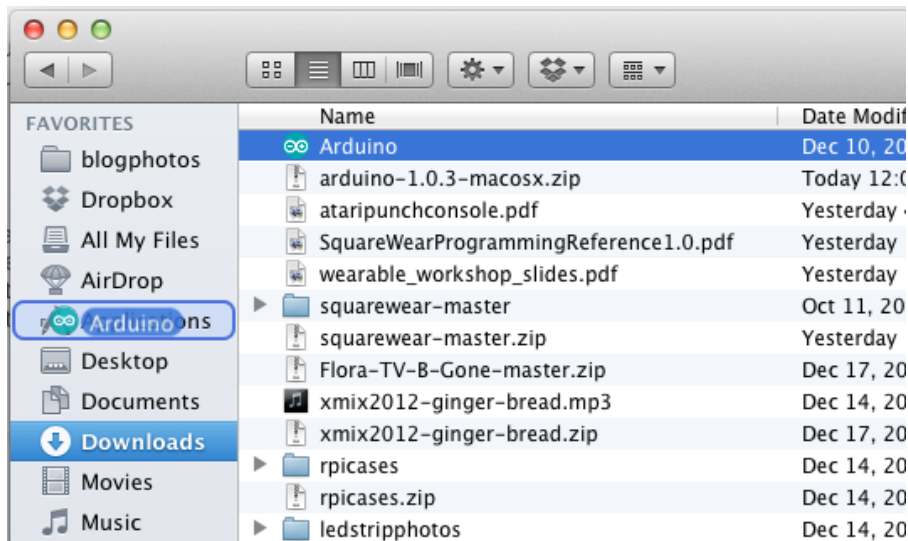
Get started by visiting the Arduino.cc website and downloading the matching IDE for your operating system. As of April 2014 we suggest v1.05 as 1.5 is still in beta. If 1.5 is no longer in beta when you read this you can try it out!



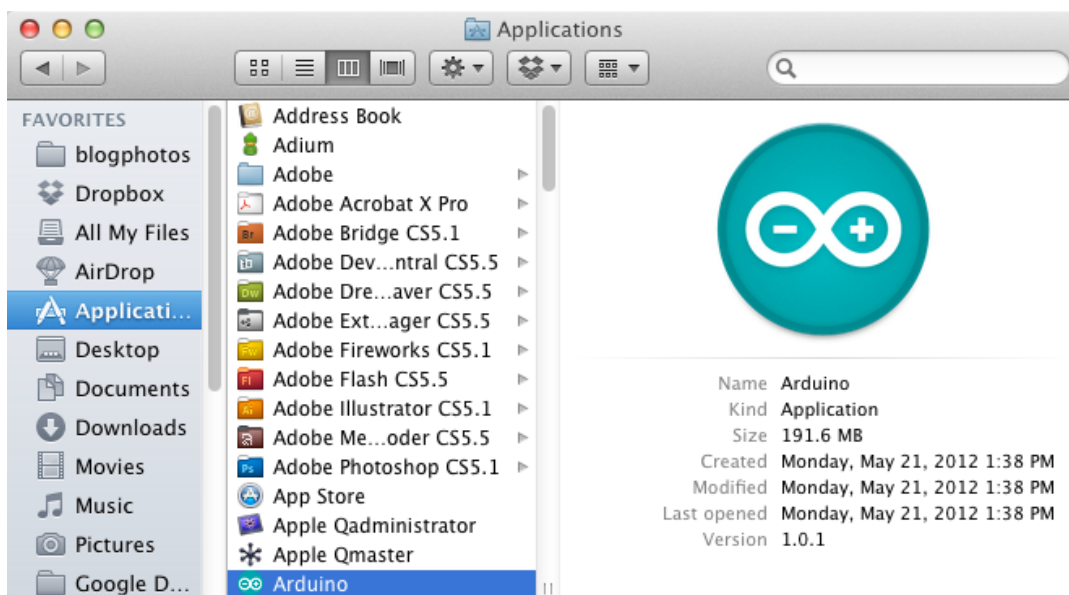
Save the install software to your desktop or wherever

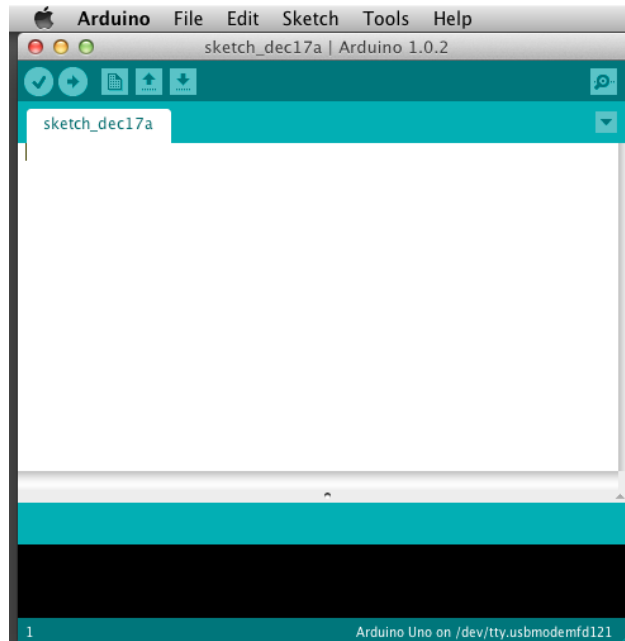


The process for installing the Arduino software on the Mac is a lot easier than on the PC. As before, the first step is to download the file. In the case of the Mac, it is a zip file.



Once downloaded, double-click on the zip file, which will extract a single file called 'Arduino.app'. This is the whole Arduino application, just drag it into your Applications Folder.



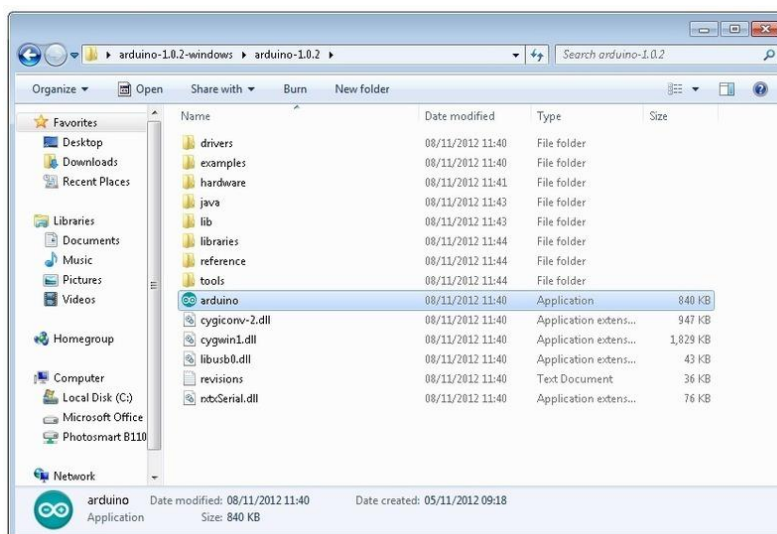


You can now find and launch the Arduino software in your Applications folder. As you are going to use it frequently, you may wish to right-click its icon in the dock and set it to Keep In Dock.

There are many different LINUX distributions and the instructions for each distribution are a little different. The Arduino community has done a great job of putting together sets of instructions for each distribution. So follow the link below and select one of the ten or more distributions on offer.

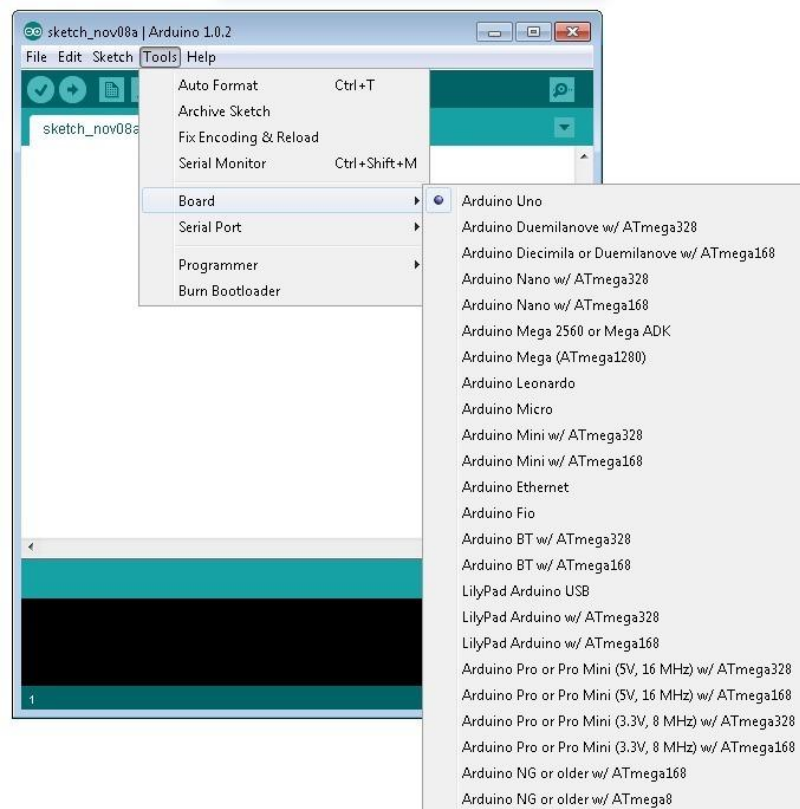
Boards and Ports

You are now ready to start the Arduino Software, so whatever platform you are using, open the Arduino folder and open the Arduino application contained within it.

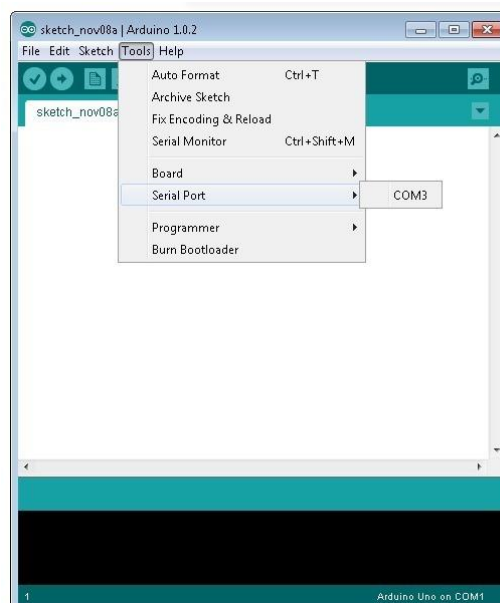


This will start the Arduino IDE, but before you can get programming, you have to tell the Arduino software which type of Arduino board you are using and also select the port it is connecting to.

To tell the Arduino IDE which type of board you are using. From the 'Tools' menu, select Board and then 'Arduino Uno' or 'Leonardo' as appropriate.

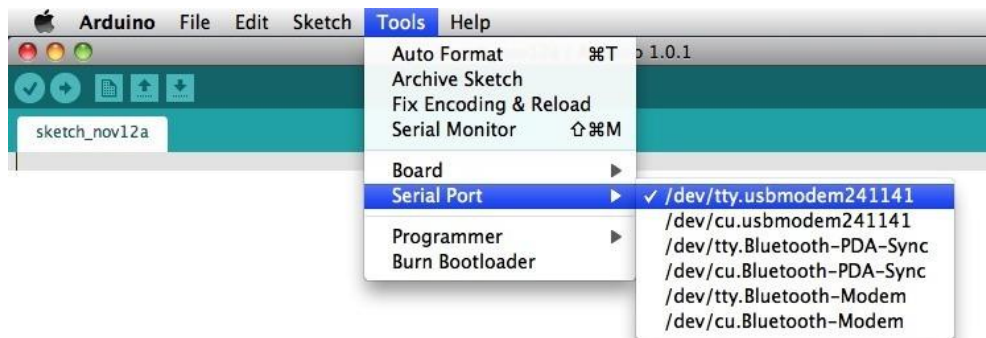


Also on the 'Tools' menu, you will find the 'Serial Port' option. Select this option.



If you are using Windows, there will probably only be one option here and it will either say COM3 or COM4. Even though there is only one option, you will still need to select it.

If you are using a Mac or Linux, there will be more options there, but it will usually be the top option in the list, as this will be the device most recently plugged in. This is useful, as the name of the port may not look like it has anything to do with Arduino. It will probably be called something like **/dev/tty.usbmodemXXXX** or **/dev/ttyUSBn**



In the next lesson, you will start by programming your Arduino board to make its built-in LED blink.

Lesson 1 Blink

Introduction

In this lesson, you will learn how program your Uno R3 controller board to make the Arduino's built-in LED blink.

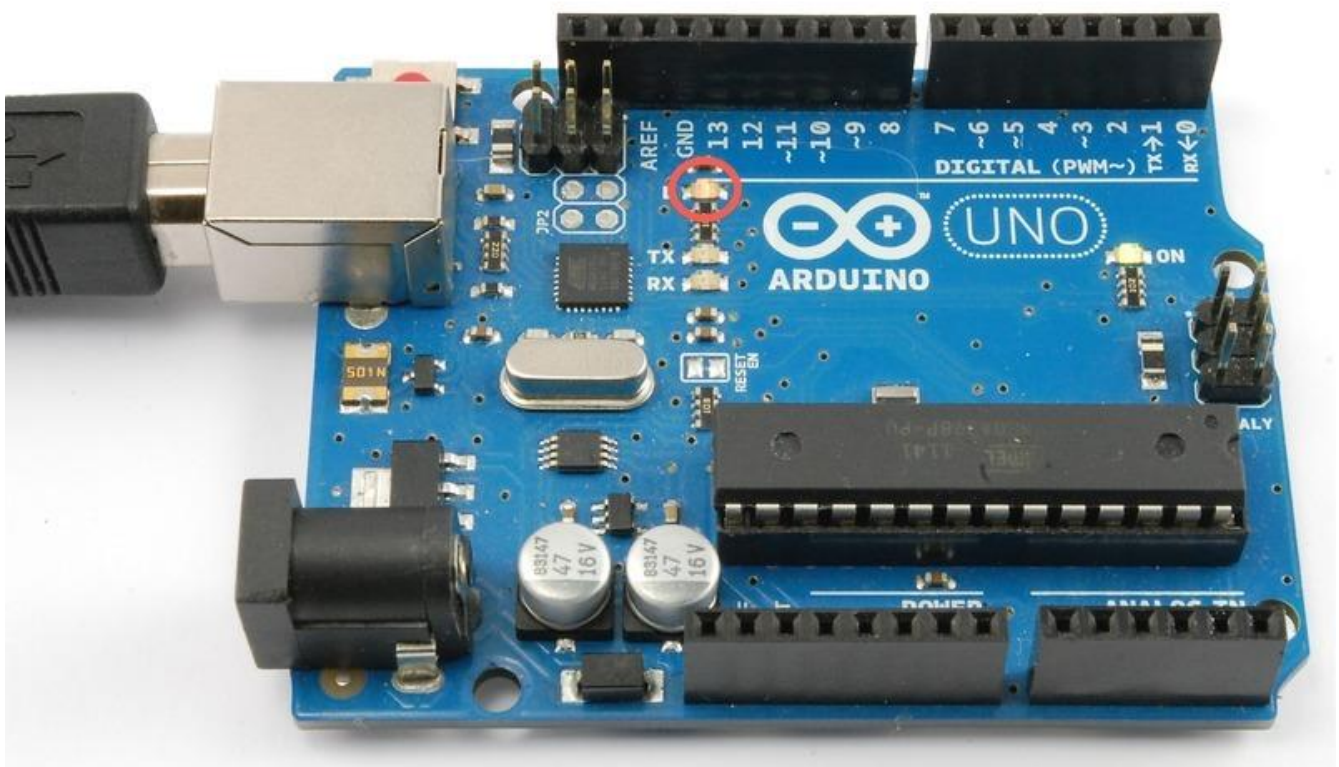
Components

- 1 * Uno board
- 1 * USB cable

Principle

The Arduino has rows of connectors along both sides that are used to connect to electronic devices and plug-in 'shields' that allow the Arduino to do more.

However, the Arduino also has a single LED that you can control from your sketches. This LED is built onto the Arduino board and is often referred to as the 'L' LED as this is how it is labelled on the board.



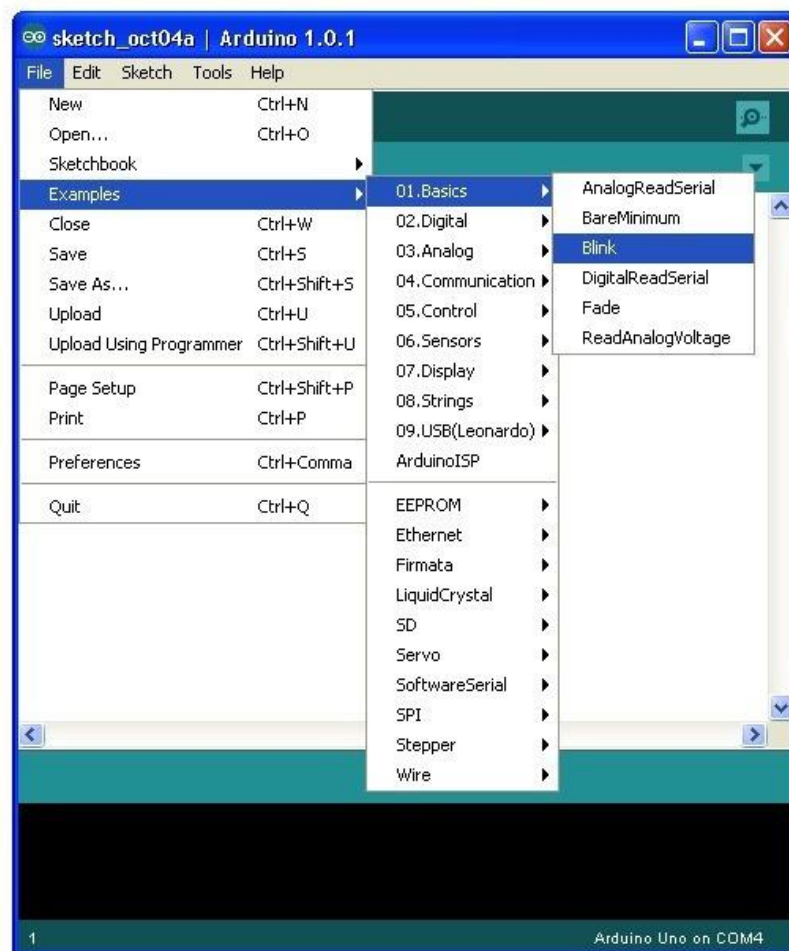
You may find that your Arduino board's 'L' LED already blinks when you connect it to a USB plug. This is because Arduino boards are generally shipped with the 'Blink' sketch pre-installed.

In this lesson, we will reprogram the Arduino with our own Blink sketch and then change the rate at which it blinks.

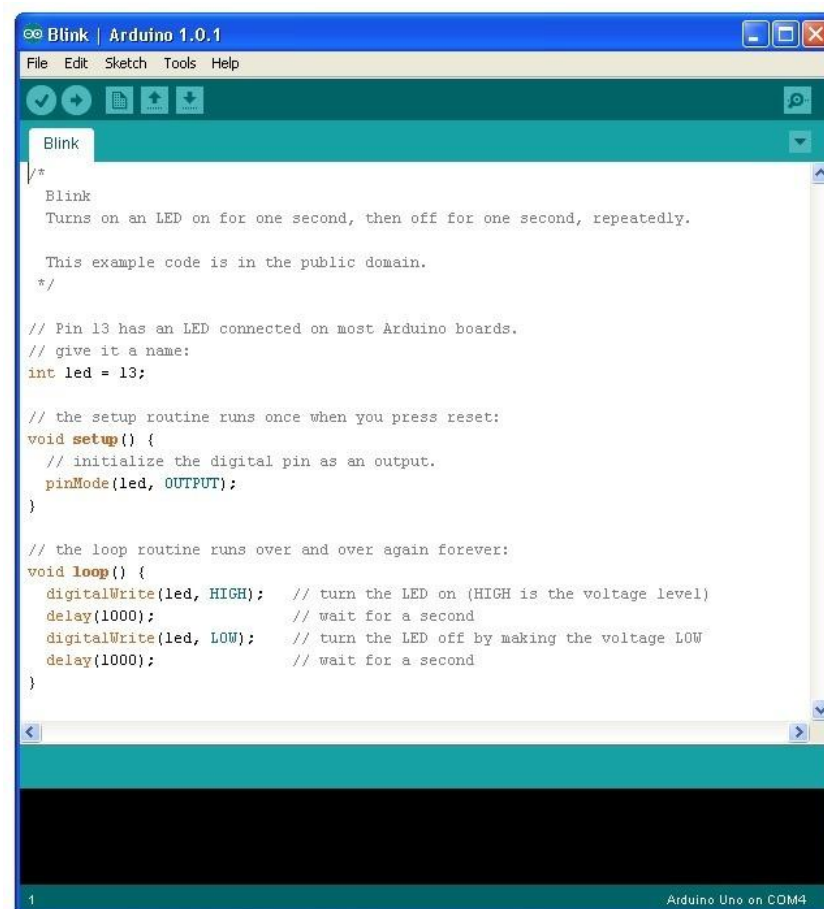
In Lesson 0, you setup your Arduino IDE and made sure that you could find the right serial port for it to connect to your Arduino board. The time has now come to put that connection to the test and program your Arduino board.

The Arduino IDE includes a large collection of example sketches that you can load up and use. This includes an example sketch for making the 'L' LED blink.

Load the 'Blink' sketch that you will find in the IDE's menu system under File → Examples → 01.Basics



When the sketch window opens, enlarge it so that you can see the whole of the sketch in the window.



The example sketches included with the Arduino IDE are 'read-only'. That is, you can upload them to an Arduino board, but if you change them, you cannot save them as the same file.

We are going to change this sketch, so, the first thing you need to do is save your own copy that you can change however you like.

From the File menu on the Arduino IDE select the option 'Save As..' and then save the sketch with the name 'MyBlink'.

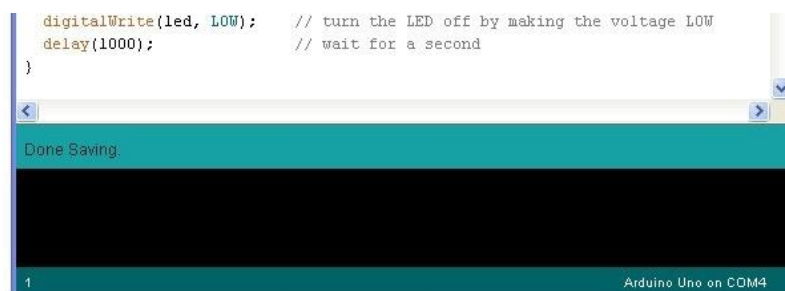


You have saved your copy of 'Blink' in your sketchbook. This means that if you ever want to find it again, you can just open it using the File → Sketchbook menu option.



Attach your Arduino board to your computer with the USB cable and check that the 'Board Type' and 'Serial Port' are set correctly. You may need to refer back to Lesson 0.

The Arduino IDE will show you the current settings for board at the bottom of the window.



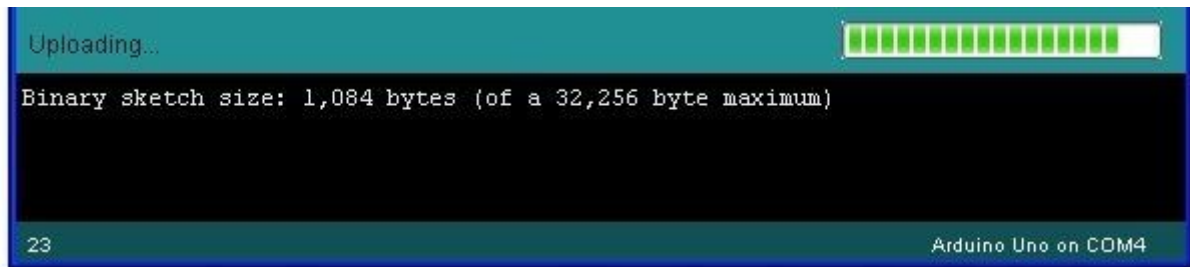
Click on the 'Upload' button. The second button from the left on the toolbar.



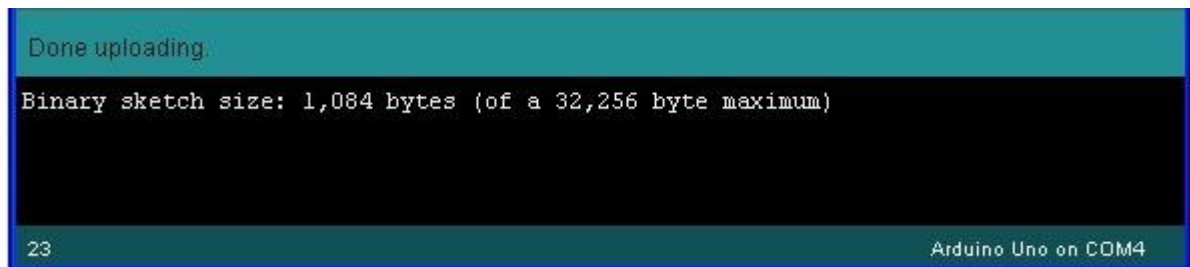
If you watch the status area of the IDE, you will see a progress bar and a series of messages. At first it will say 'Compiling Sketch..'. This converts the sketch into a format suitable for uploading to the board.



Next, the status will change to 'Uploading'. At this point, the LEDs on the Arduino should start to flicker as the sketch is transferred.



Finally, the status will change to 'Done'.



The other message tells us that the sketch is using 1,084 bytes of the 32,256 bytes available. After the 'Compiling Sketch..' stage you could get the following error message:



The clue is at the top here, it probably means that your board is not connected at all, or the drivers have not been installed (if necessary) or that the wrong serial port is selected.

If you get this, go back to Lesson 0 and check your installation.

Once the upload has completed, the board should restart and start blinking.

Open the code

The first thing to note is that quite a lot of this sketch is what is called 'comments'. Comments are not actual program instructions, they are just comments about how the program works. They are there for our benefit, so that there is some explanation to accompany the sketch. Everything between `/*` and `*/` at the top of the sketch is a block comment, that explains what the sketch is for.

There are also single line comments that start with `//` and everything up until the end of the line counts as being a comment.

The first actual line of code is:

Copy Code

```
1. int led = 13;
```

As the comment above explains, this is giving a name to the pin that the LED is attached to. This is 13 on most Arduinos, including the Uno and Leonardo.

Next, we have the 'setup' function. Again, as the comment says, this is run when the reset button is pressed. It is also run whenever the board resets for any reason, such as power first being applied to it, or after a sketch has been uploaded.

Copy Code

```
1. void setup() {  
2.   // initialize the digital pin as an output.  
3.   pinMode(led, OUTPUT);  
4. }
```

Every Arduino sketch must have a 'setup' function, and the part of it where you might want to add instructions of your own is between the `{` and the `}`.

In this case, there is just one command there, which, as the comment states tells the Arduino board that we are going to use the LED pin as an output. It is also mandatory for a sketch to have a 'loop' function. **Unlike the 'setup' function that only runs once, after a reset, the 'loop' function will, after it has finished running its commands, immediately start again.**

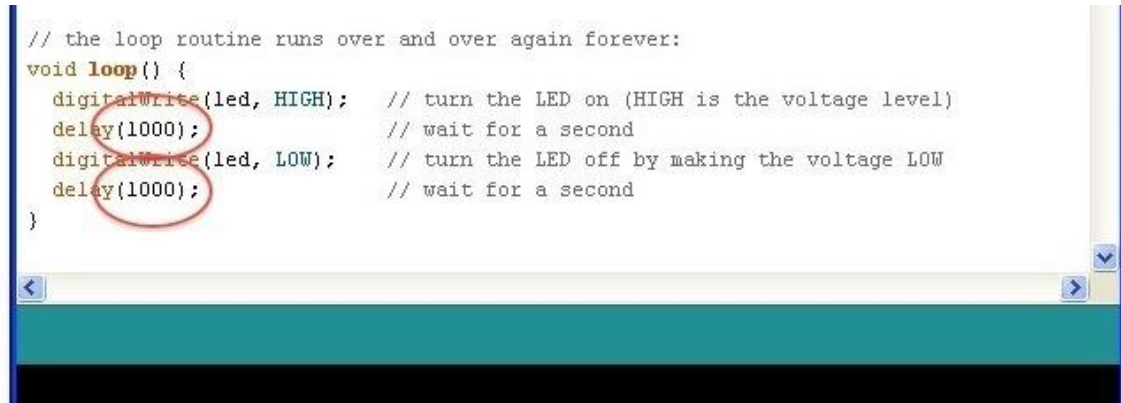
Copy Code

```
1. void loop() {  
2.   digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
3.   delay(1000);           // wait for a second  
4.   digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
```

5. `delay(1000);` `// wait for a second`
6. `}`

Inside the loop function, the commands first of all turn the LED pin on (HIGH), then 'delay' for 1000 milliseconds (1 second), then turn the LED pin off and pause for another second.

You are now going to make your LED blink faster. As you might have guessed, the key to this lies in changing the parameter in () for the 'delay' command.



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

This delay period is in milliseconds, and so if you want the LED to blink twice as fast, change the value of 1000 to 500. This would then pause for half a second each delay rather than a whole second.

Upload the sketch again and you should see the LED start to flash more quickly.

Lesson 2 Button

Introduction

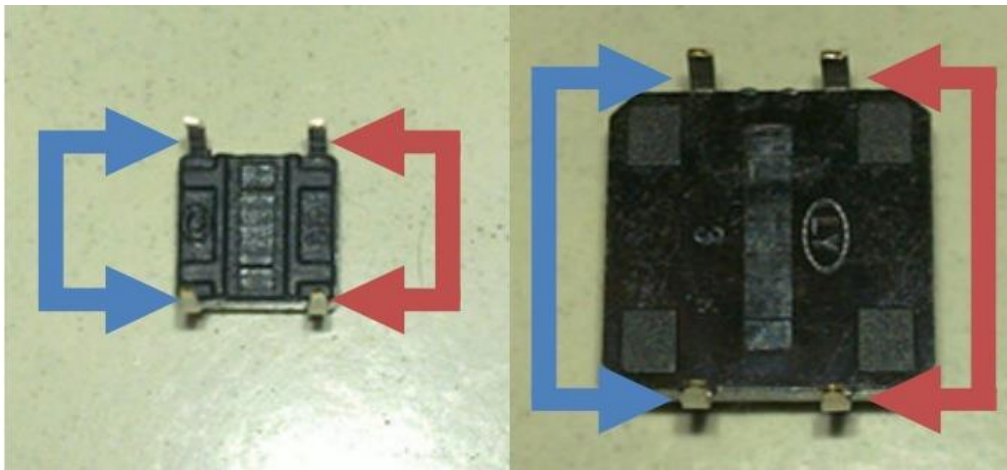
In this experiment, we will learn how to turn a single LED on or off by using an I/O port and button switch. The "I/O port" refers to the INPUT and OUTPUT port. We will use the input function of the Uno I/O port to read the output of an external device. Since the Uno board itself has an LED (connected to Pin 13), we will use the LED to accomplish this experiment for convenience.

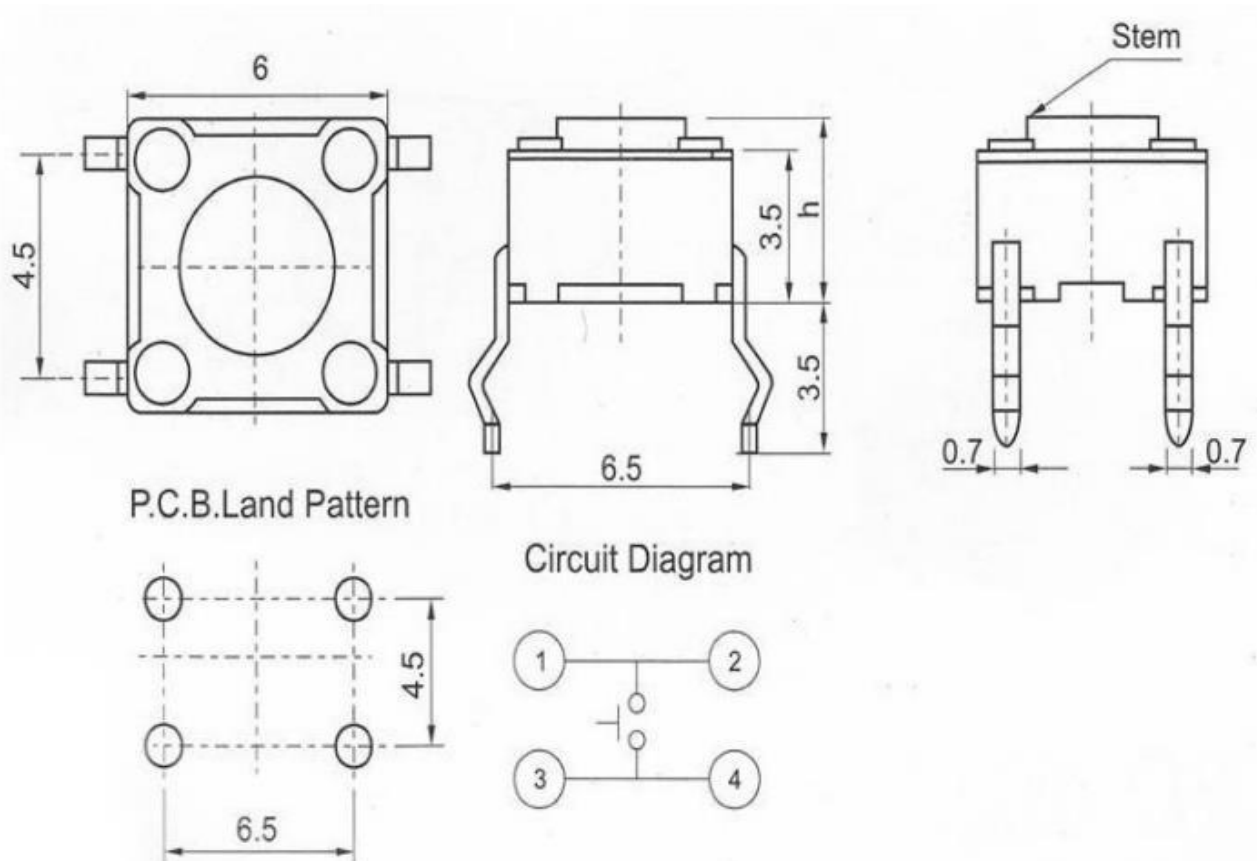
Components

- 1 * Uno board
- 1 * USB cable
- 1 * Button
- 1 * Resistor (10k Ω)
- Jumper wires
- 1 * Breadboard

Principle

Buttons are a common component used to control electronic devices. They are usually used as switches to connect or disconnect circuits. Although buttons come in a variety of sizes and shapes, the one used in this experiment will be a 6mm mini-button as shown in the following pictures. Pins pointed out by the arrows of same color are meant to be connected.





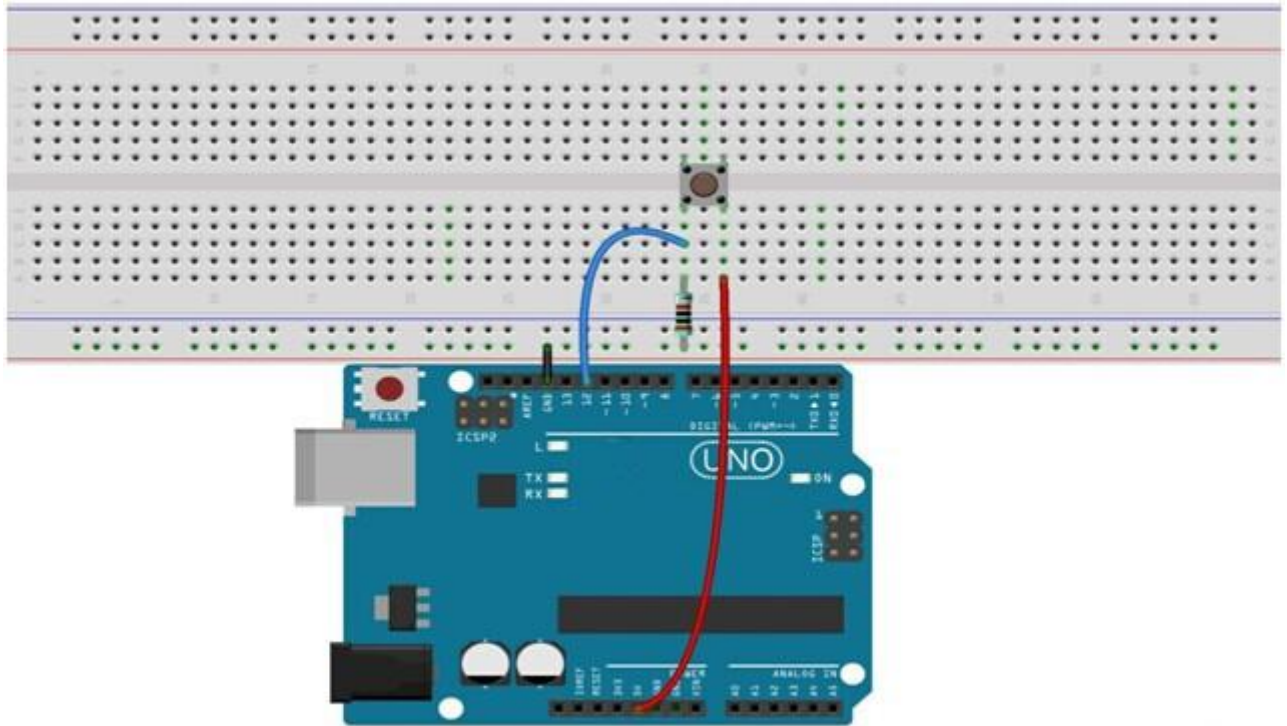
When the button is pressed, the pins pointed by the blue arrows will connect to the pins pointed by the red arrows.

Generally, the button switch is directly connected in an LED circuit in order to turn the LED on or off. This connection is relatively simple. However, sometimes the LED will light up automatically without pressing the button, which is caused by various interferences. In order to avoid these external interferences, we will connect a pull-down resistor, that is, connect a 1K–10K Ω resistor between the button port and the GND. The function of the pull-down resistor is to consume external interferences while connected to the GND for as long as the button switch is turned off.

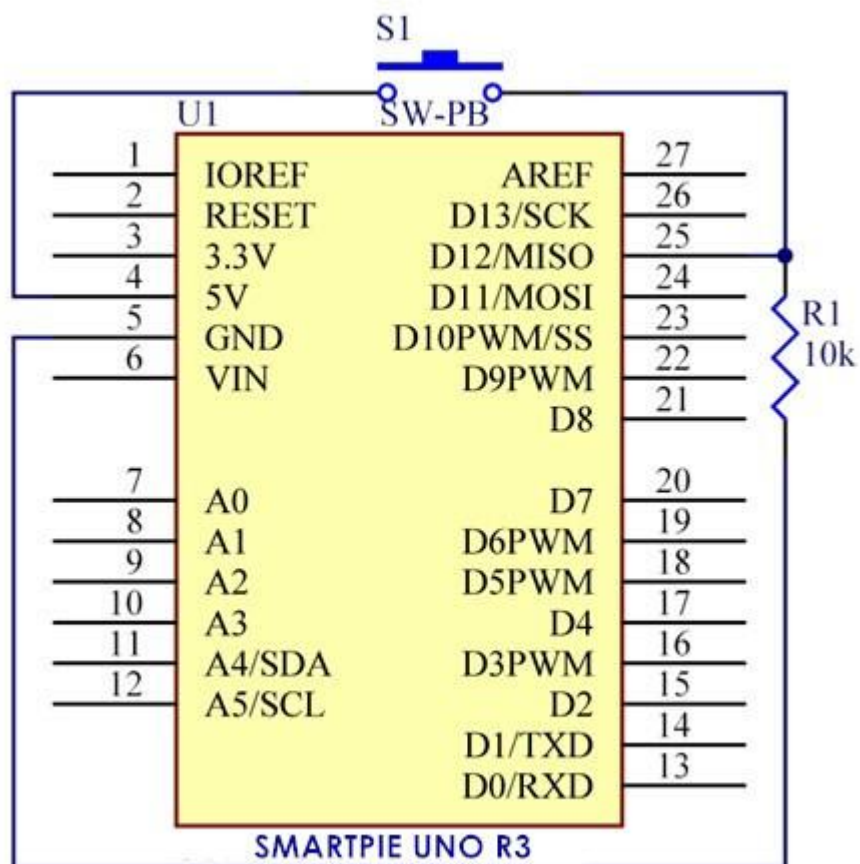
This circuit connection is widely used in numerous circuits and electronic devices. For example, if you press any button on your mobile phone, the backlight will light up.

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram:



The corresponding schematic diagram is as follows:

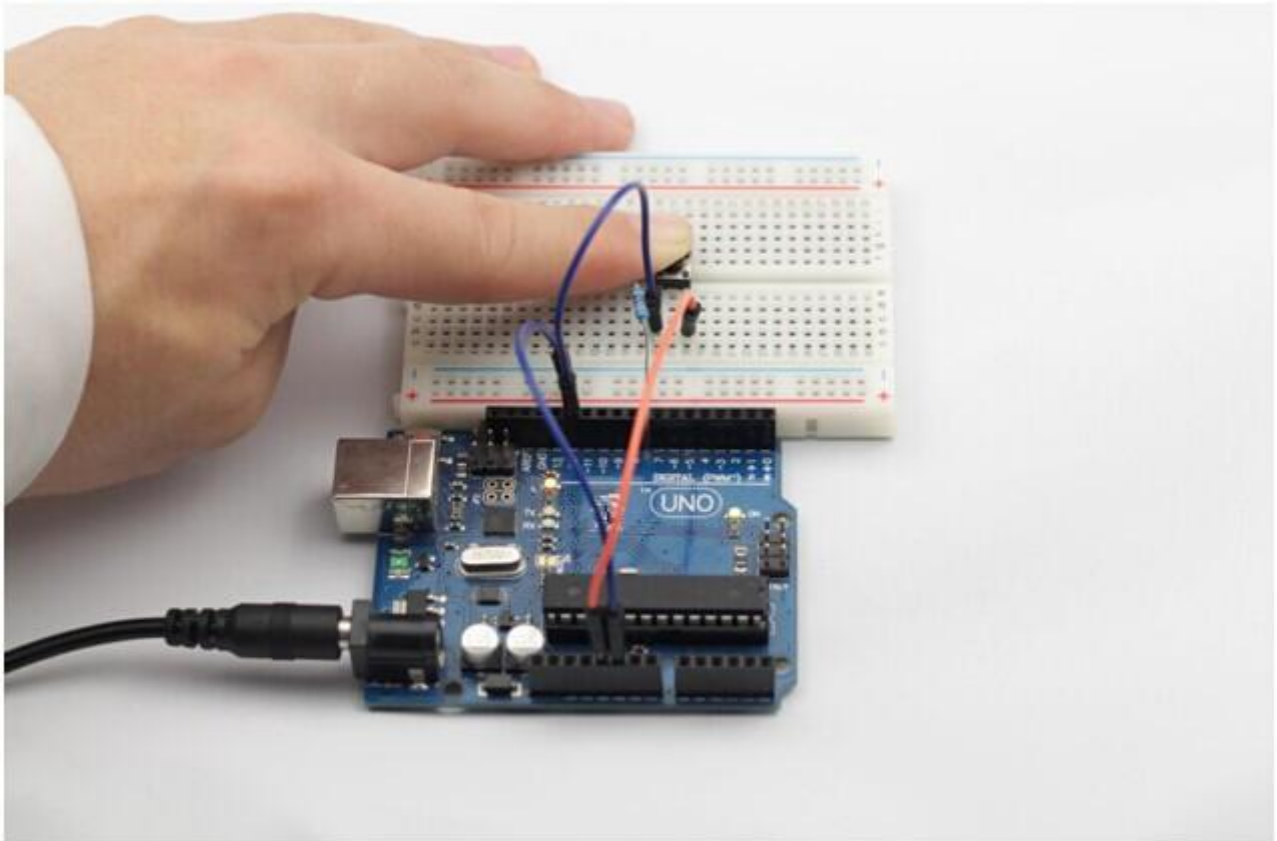


Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

If you press the button, the LED on the Uno board will light up.



Lesson 3 Flowing LED Lights

Introduction

In this lesson, we'll conduct a simple yet interesting experiment – using LEDs to create flowing LED lights. As the name implies, these flowing lights are made up of eight LEDs in a row which successively light up and dim one after another like flowing water.

Components

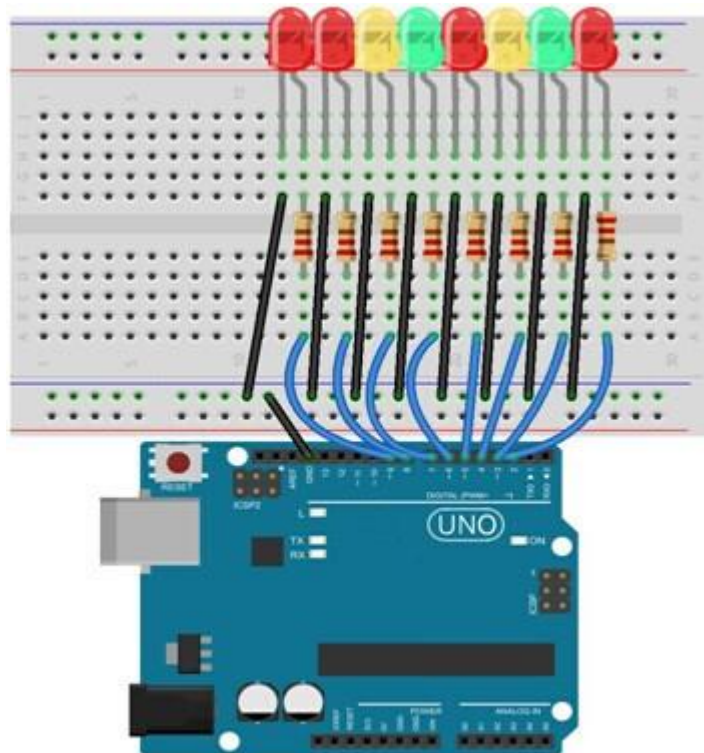
- 1 * Uno board
- 1 * Breadboard
- Jumper wires
- 8 * LED
- 8 * Resistor (220Ω)
- 1 * USB cable

Principle

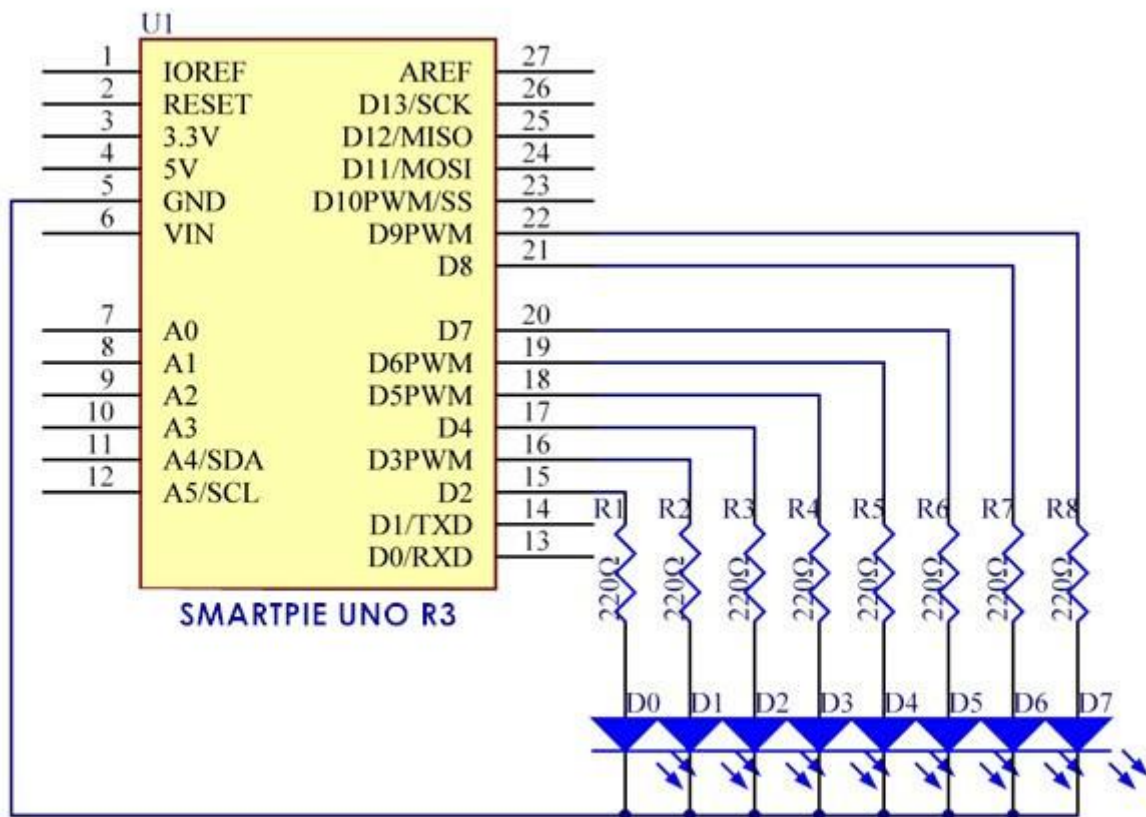
The principle of this experiment is simply to turn eight LEDs on in turn.

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram



The corresponding schematic diagram is as follows:

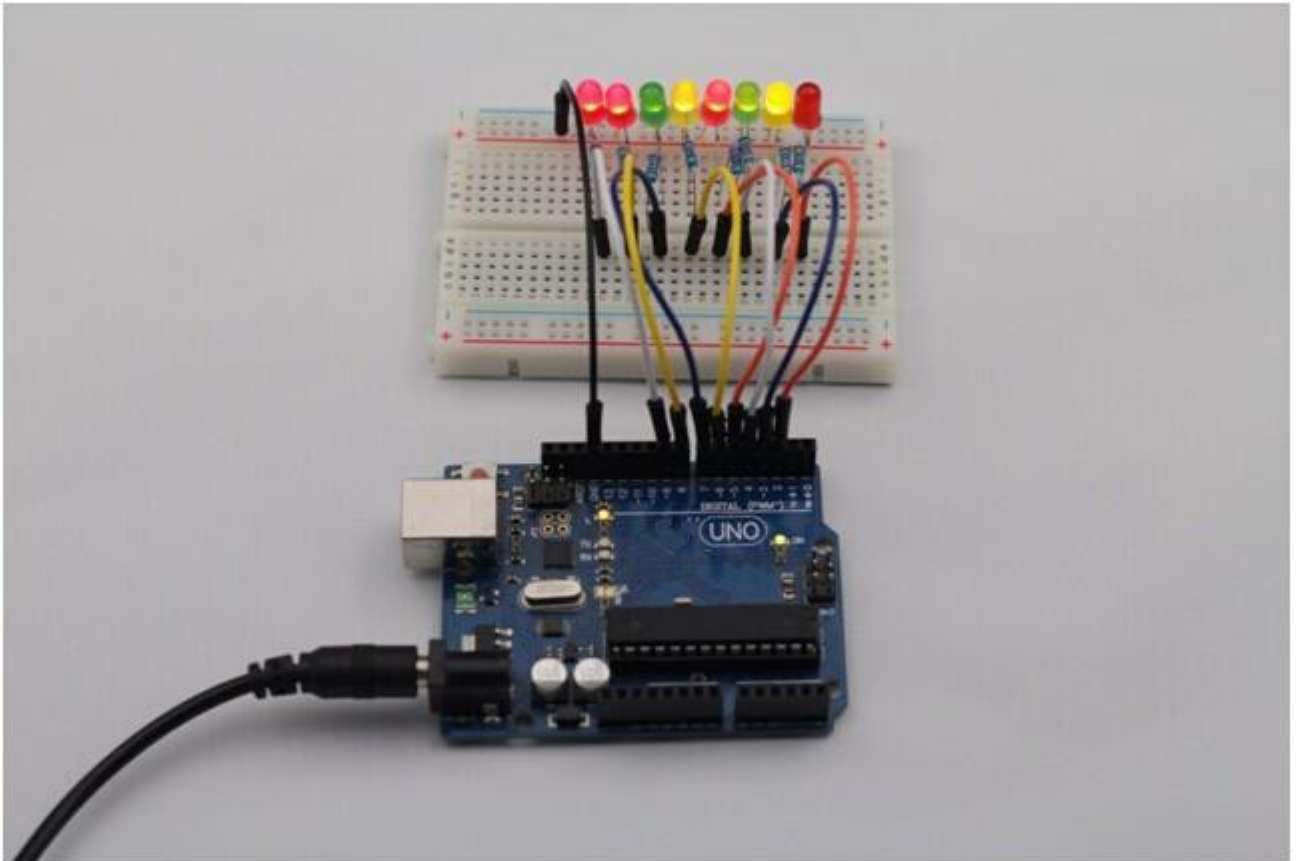


Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Here you should see eight LEDs light up one by one from left to right, and then go out one by one from right to left. After that, the LEDs will light up one by one from right to left, and then go out one by one from left to right. This process will repeat indefinitely.



Experimental Summary

This simple experiment helps to increase proficiency in applying LEDs. Furthermore, you can modify the provided program to create all kinds of fantastic patterns!

Lesson 4 Active Buzzer

Introduction

You can use a buzzer whenever you want to make some noise.

Experimental Conditions

- 1 * Uno board
- 1 * Breadboard
- 1 * USB data cable
- 1 * Buzzer (Active)
- Jumper wires

Principle

As a type of electronic buzzer with 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 for voice devices. Buzzers can be categorized as active and passive ones (see the following picture). Turn the pins of two buzzers face up, and the one with a green circuit board is a passive buzzer, while the other enclosed with a black tape is an active one.



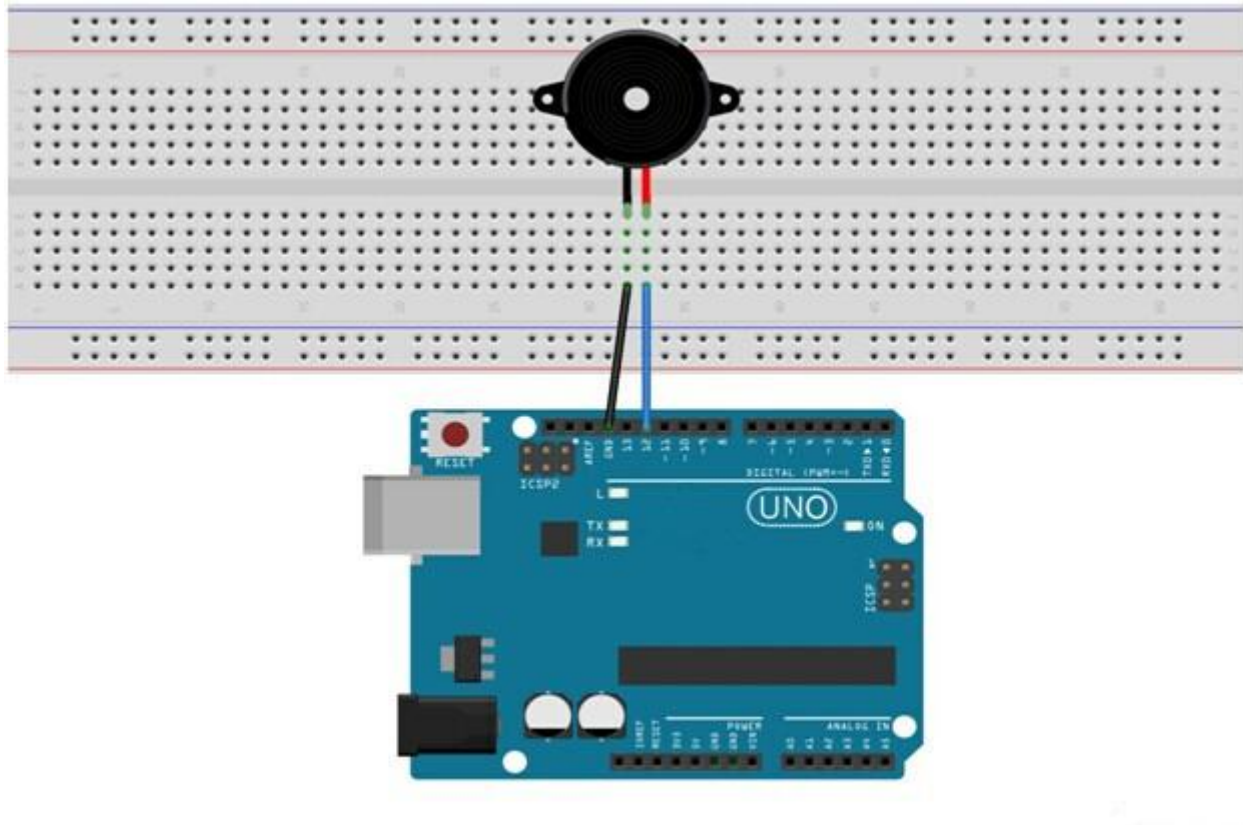
The difference between an active buzzer and a passive buzzer is:

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 tweet 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.

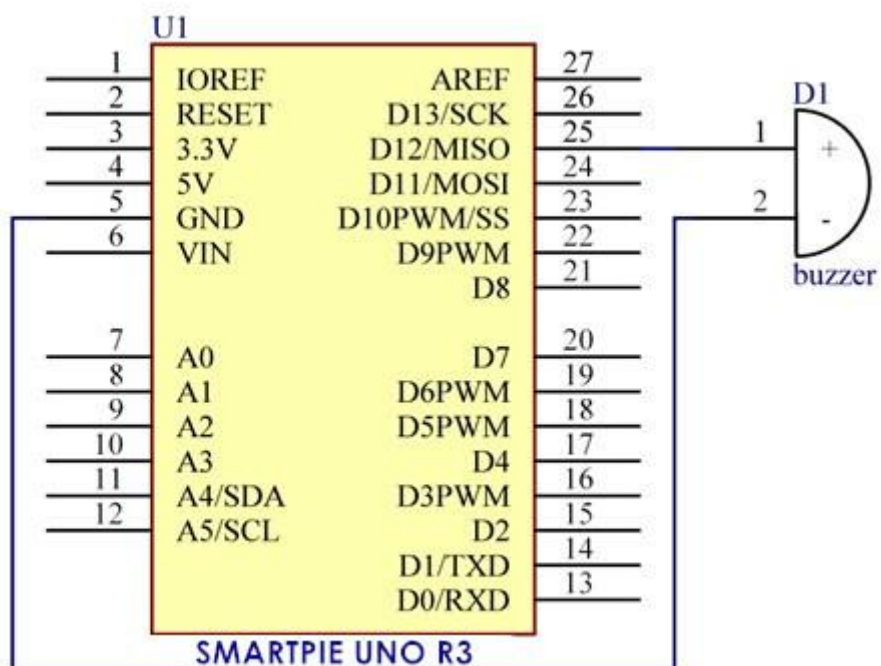
In this experiment, we use the active buzzer.

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram:

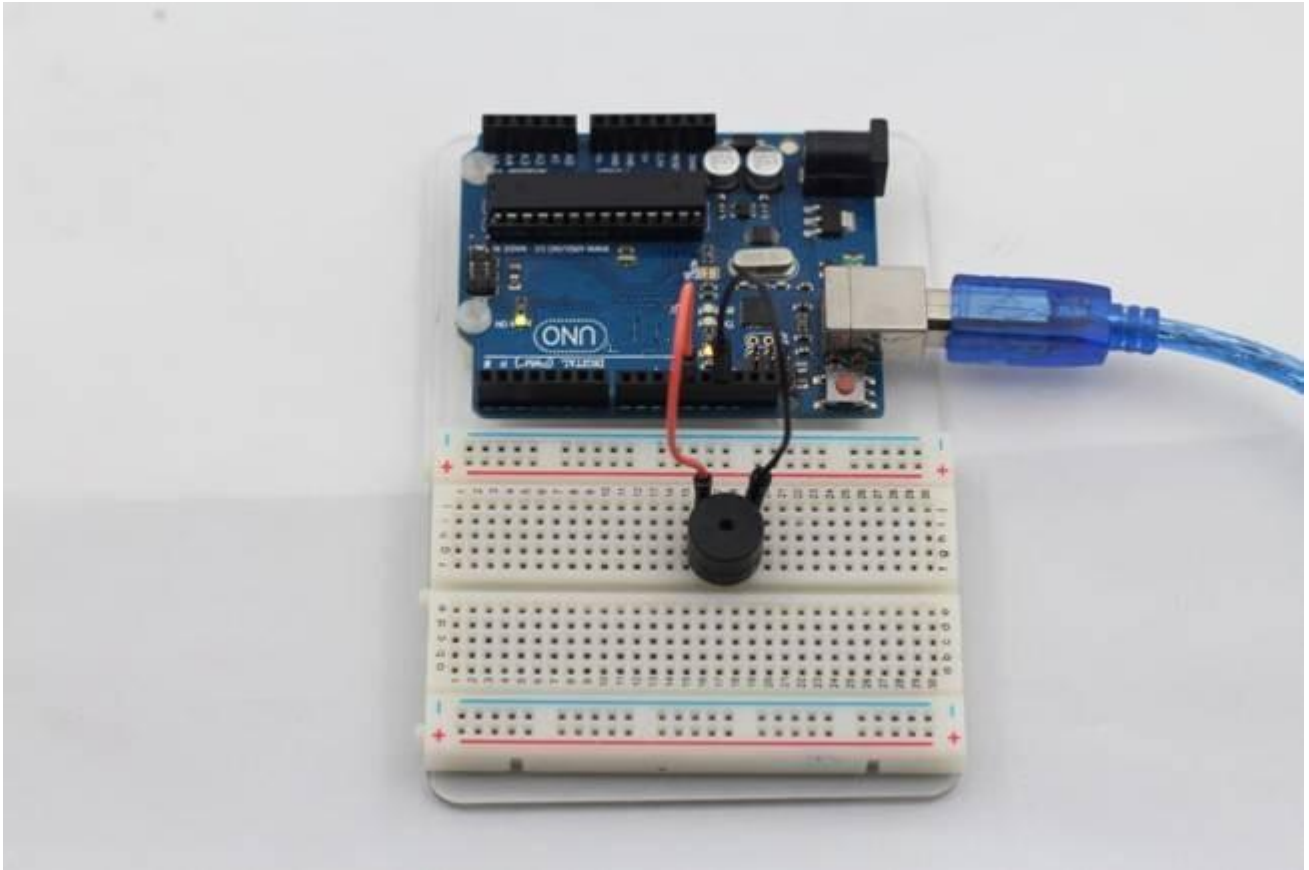


The corresponding schematic diagram is as follows:



Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program



Step 4: Burn the program into Uno board

Now, you should hear the buzzer make sounds.

Lesson 5 Passive Buzzer

Introduction

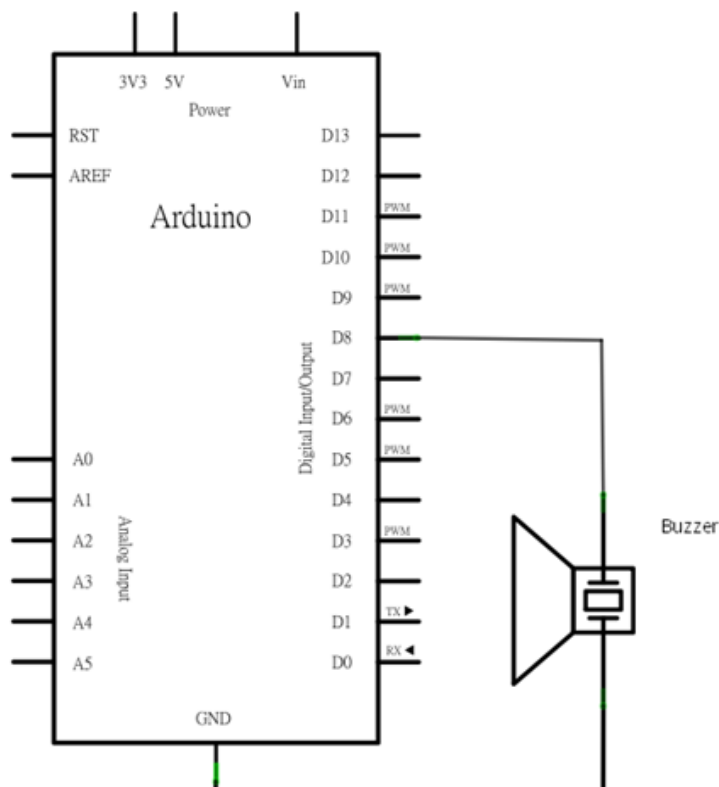
Purpose of the experiment control buzzer, allowing the buzzer Alto Do (523Hz), Re (587Hz), Mi (659Hz), Fa (698Hz), So (784Hz), La (880Hz), Si (988Hz) to Treble Do (1047Hz) This scale of eight different sounds, each sound scale 0.5 seconds.

Components

- 1 * Uno board
- 1 * USB data cable
- 1 * Passive Buzzer
- Several jumper wires
- 1 * Breadboard

Experimental Principle

Principle buzzer, in fact, just use PWM generating audio, drives the buzzer, allowing the air to vibrate, can sound. Appropriately changed as long as the vibration frequency, it can generate different sound scale. For example, sending a pulse wave can be generated 523Hz Alto Do, pulse 587Hz can produce midrange Re, 659Hz can produce midrange Mi. If you then with a different beat, you can play a song. Here be careful not to use the Arduino analogWrite () function to generate a pulse wave, because the frequency analogWrite () is fixed (500Hz), no way to scale the output of different sounds.



Experimental Procedures

Step 1: Connect circuit as shown in the following diagram:

Wiring the buzzer connected to the Arduino board, the red (positive) to the pin8, black wire (negative) to the GND

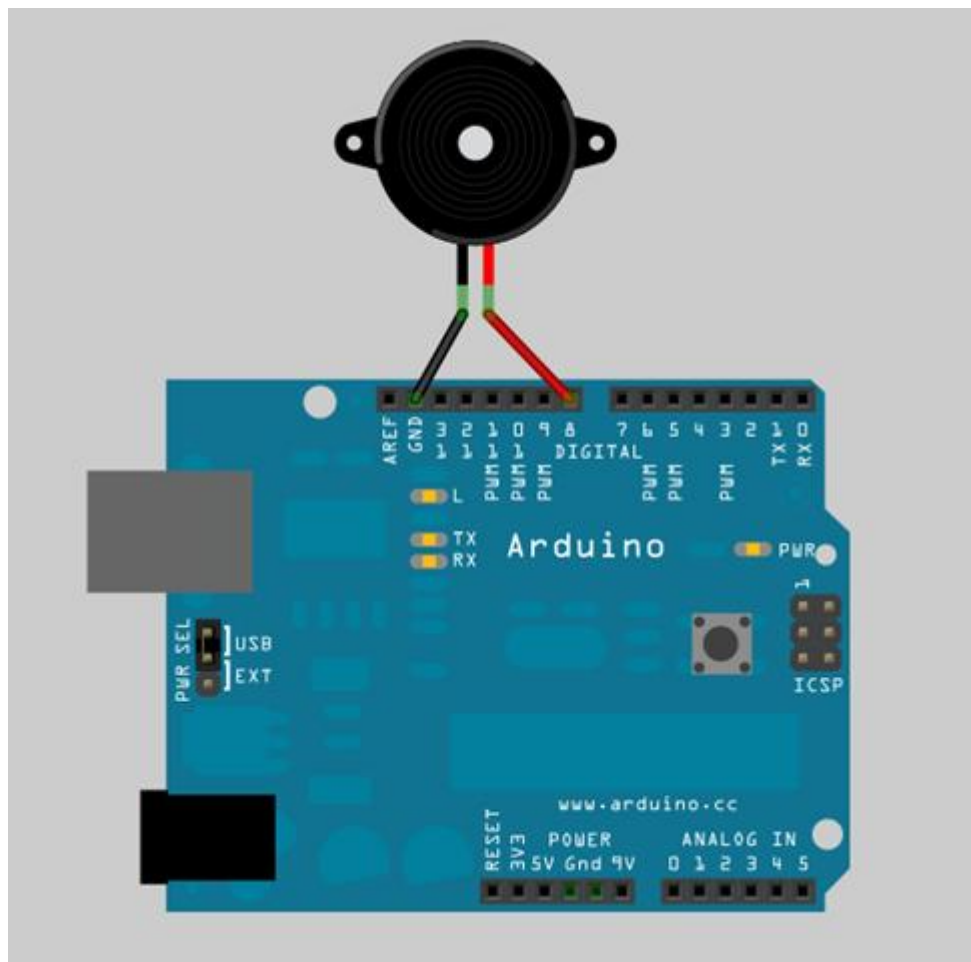
Description:

L04 ~ L05: definition of alto Do, Re, Mi, So, La, Si and treble Do eight octave frequency, the frequency of each scale is already defined in pitches.h file in, so just find the eight Constant scale and stored in the array to melody.

L06: definition of variable duration, representing each scale response time duration, because the scale to make each sound 0.5 seconds, so the duration is set to 500 (in milisecond)

L13 ~ L19: Let the buzzer Alto Do (523Hz), Re (587Hz), Mi (659Hz), Fa (698Hz), So (784Hz), La (880Hz), Si (988Hz) to treble Do (1047Hz) which eight voices of different scales, each scale ring 0.5 seconds

L22: every two seconds, and then replay the content pitches.h stalls:



Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Fixed brains

in this example is based, together with a few LED and modify the program, at the same time to play a sound control LED lights change, so that this paradigm has become a shot in the program. Try to generate an ambulance siren. Tip: Just let the buzzer continuously generate Alto Do (523Hz) and Alto Fa (698Hz), each about 0.8 seconds of sound, you can simulate ambulance siren.

Lesson 6 Photoresistor

Introduction

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 photoconductivity. A photoresistor can be applied in light-sensitive detector circuits, and light- and dark-activated switching circuits.

Experimental Conditions

- 1 * Uno board
- 1 * USB data cable
- 1 * Photoresistor
- 1 * Resistor (10K Ω)
- 8 * LED
- 8 * Resistor (220 Ω)
- Jumper wires
- 1 * Breadboard

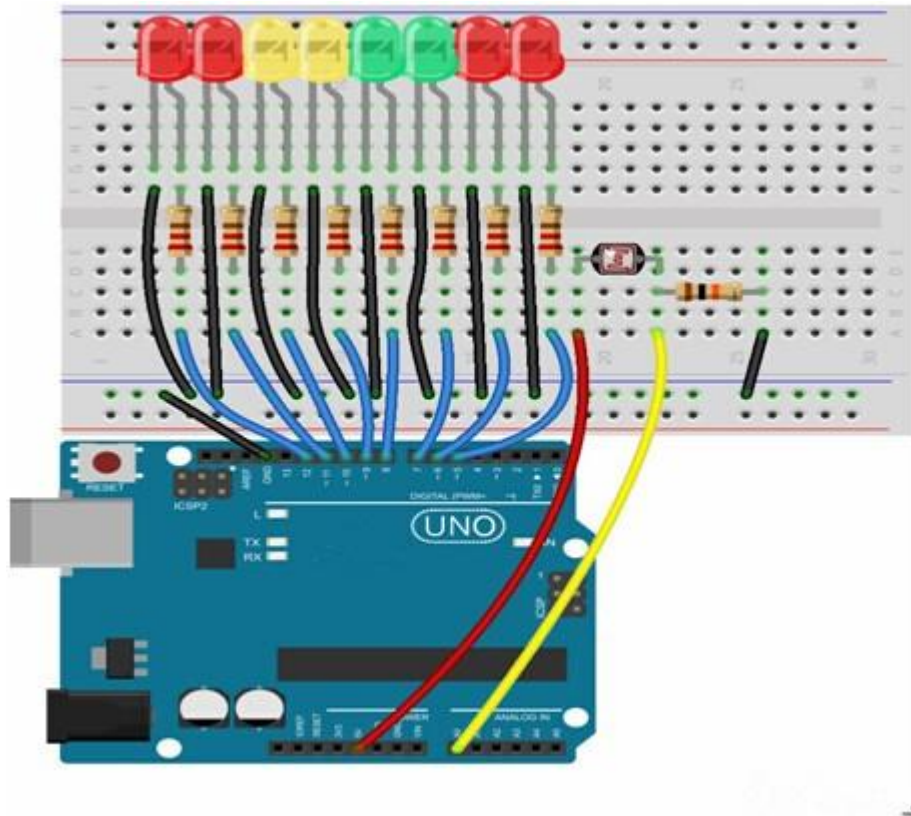
Experimental Principle

The resistance of the photoresistor changes with incident light intensity. If the incident light intensity is high, the resistance reduces; if low, increases.

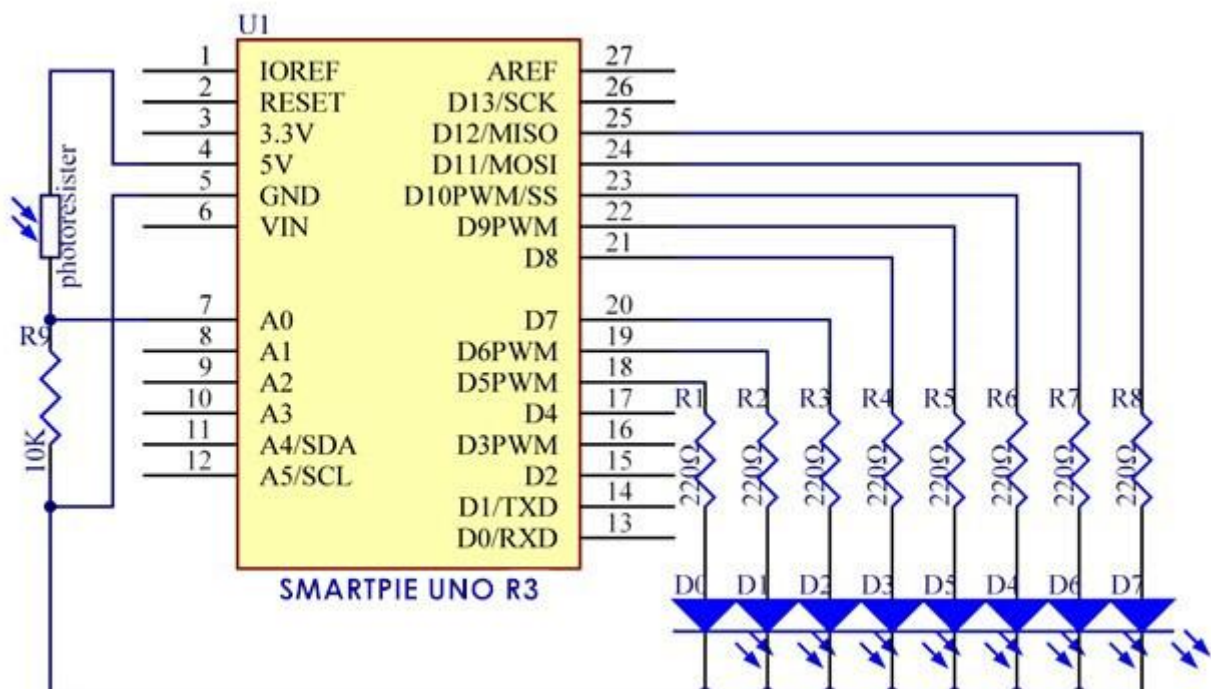
In this experiment, we will use eight LEDs to indicate light intensity. The higher the light intensity is, the more the LED is lit. When the light intensity is high enough, all the LEDs will be lit. When there is no light, all the LEDs will go out.

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram:



The corresponding schematic diagram is as follows:

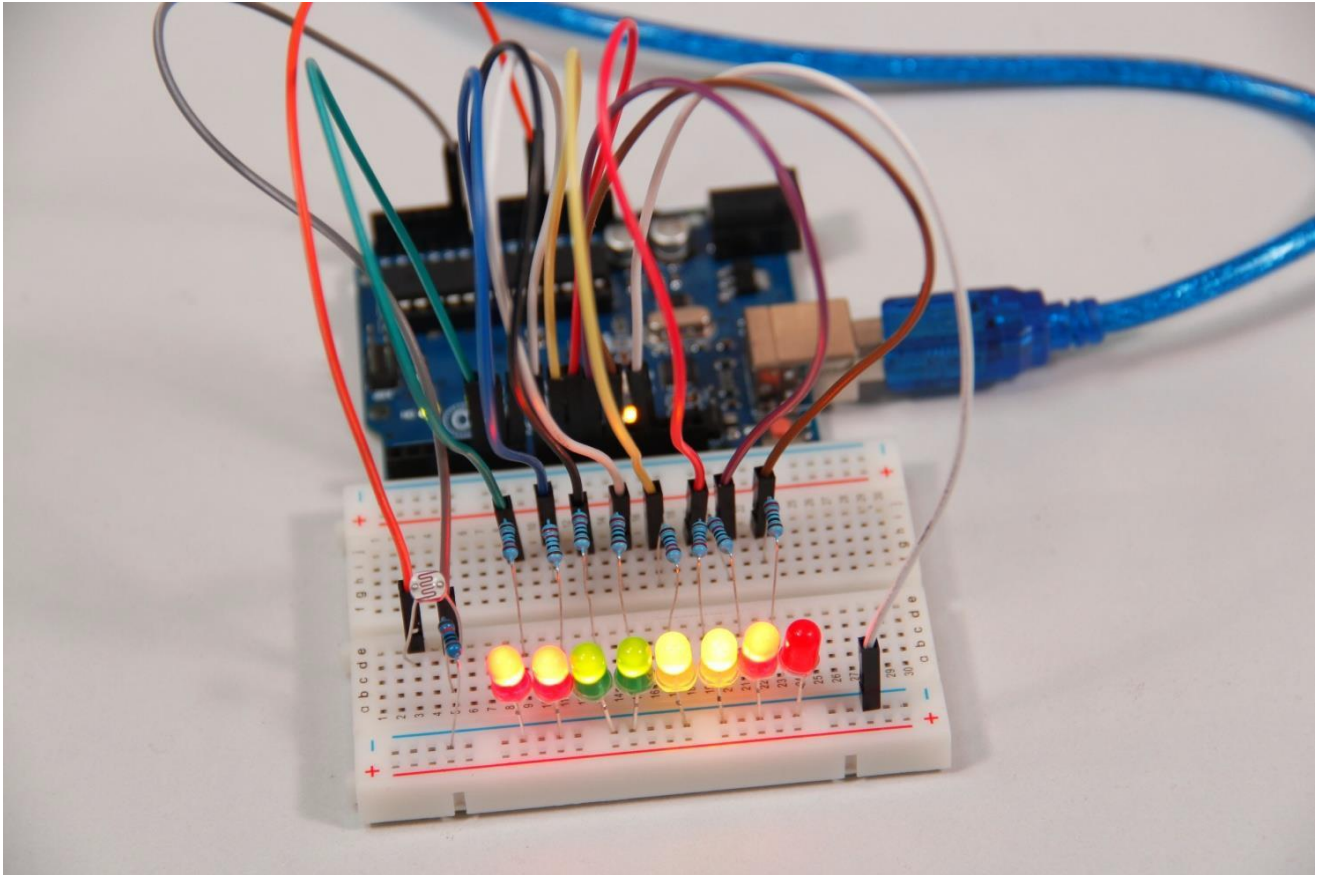


Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Now, if you shine the photoresistor with a certain light intensity, you will see several LEDs light up. If you increase the light intensity, you will see more LEDs light up. When you place it in dark environment, all the LEDs will go out.



Exploration

In addition, you can replace the photoresistor with a microphone to use LEDs to indicate sound intensity. The higher the sound intensity is, the more LEDs are lit. You can realize this effect by yourself.

Lesson 7 RGB LED

Introduction

For this lesson, we will use PWM to control a RGB LED and cause it to display multiple colors.

Components

- 1 * RGB LED
- 3 * Resistor (220Ω)
- 1 * Breadboard
- 1 * Uno board
- Jumper wires
- USB cable

Principle

Color Principle of RGB

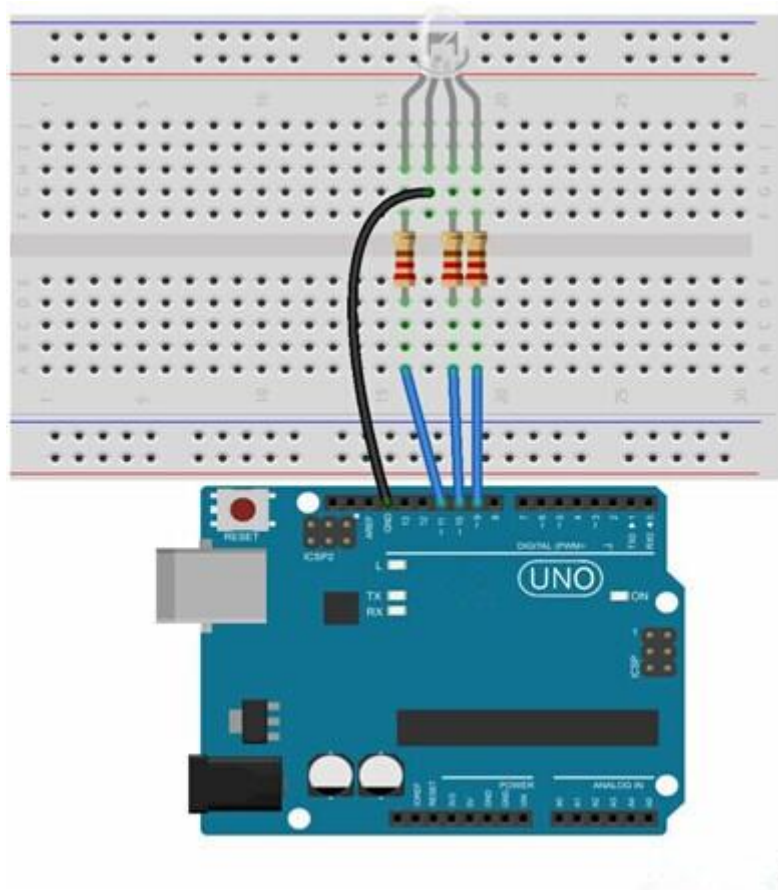
RGB stands for the red, green, and blue color channels and is an industry color standard. RGB displays various new colors by changing the three channels and superimposing them, which, according to statistics, can create 16,777,216 different colors. If you say the color displayed doesn't completely match a natural color, then it almost certainly cannot be differentiated with the naked eye.

Each of the three color channels of red, green, and blue has 255 stages of brightness. When the three primary colors are all 0, "LED light" is the darkest, that is, it turns off. When the three primary colors are all 255, "LED light" is the brightest. When superimposing the light emitted by the three primary colors, the colors will be mixed. However, the brightness is equal to the sum of all brightness, and the more you mix, the brighter the LED is. This process is known as additive mixing.

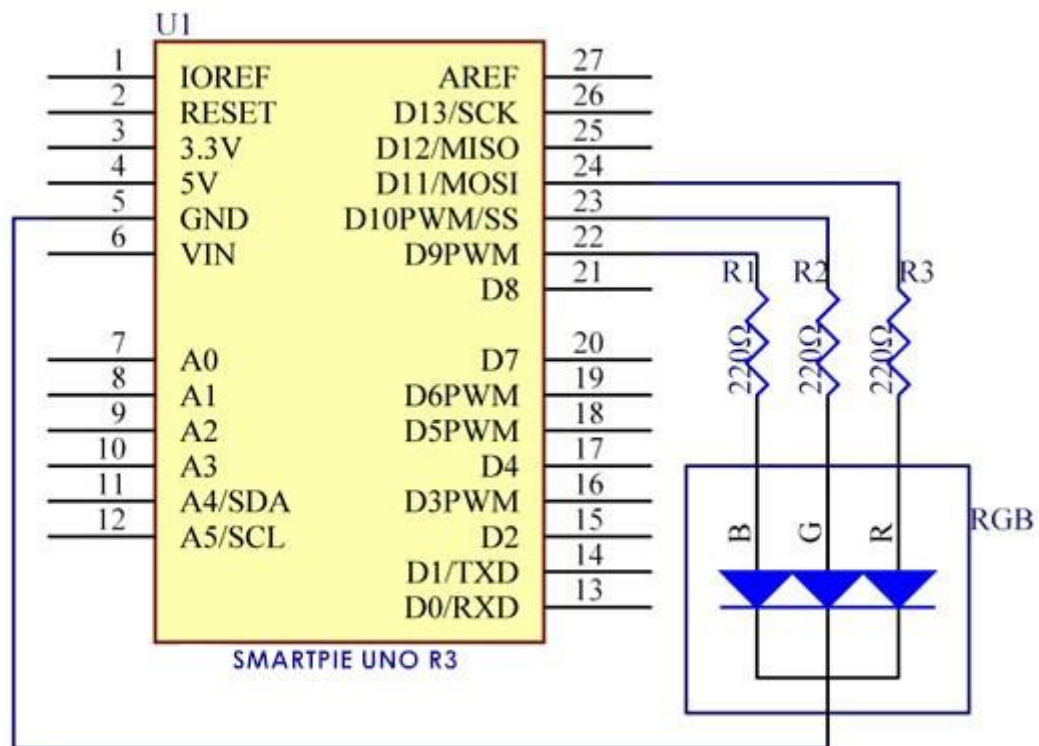
In this experiment, we will also use PWM which you have learnt in super kit. Here we input any value between 0 and 255 to the three pins of the RGB LED to make it display different colors.

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram:



The corresponding schematic diagram is as follows:

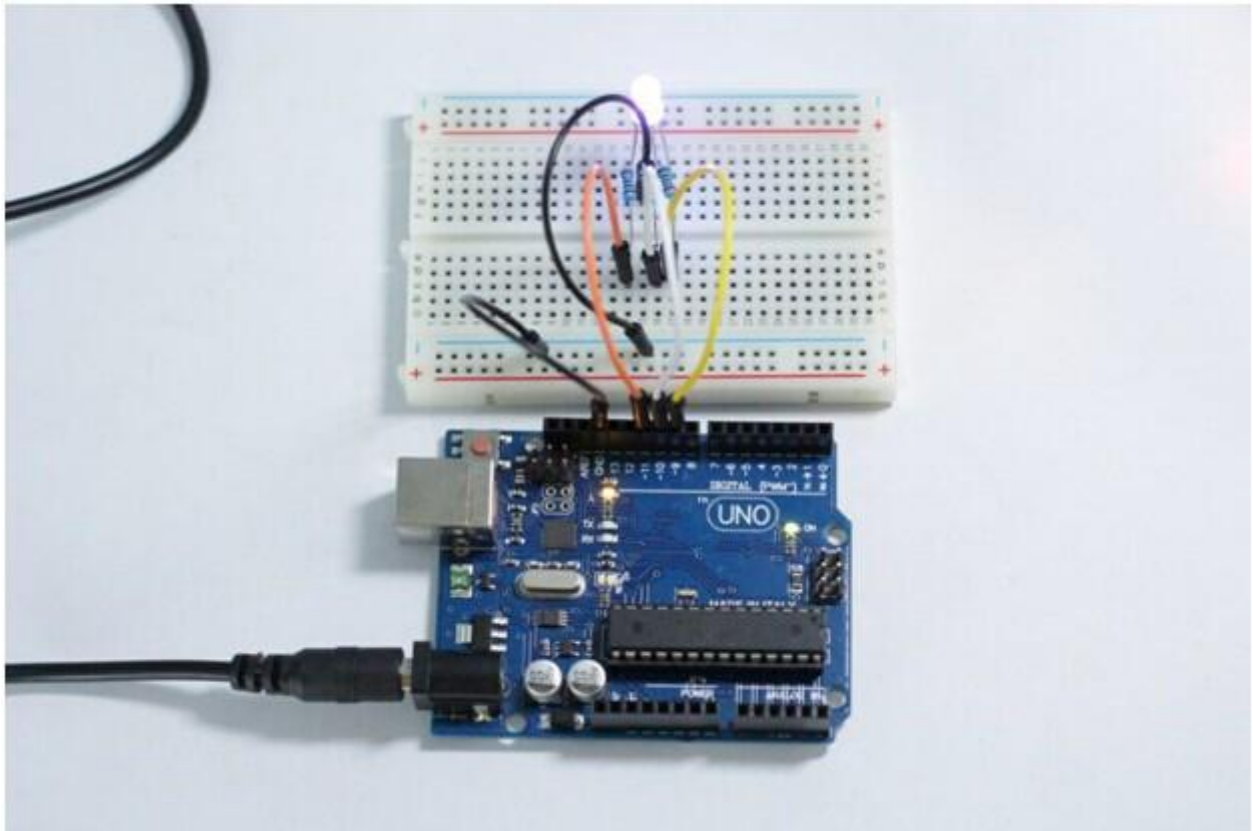


Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

The RGB LED will appear red, green, and blue first, then red, orange, yellow, green, blue, indigo, and purple.



Lesson 8 Servo

Introduction

Servo is a type of geared motor that can only rotate 180 degrees. It is controlled by sending electrical pulses from your Uno board. These pulses tell the servo what position it should move to. A servo has three wires, the brown wire is GND, the red one is VCC, and the orange one is signal line.

Components

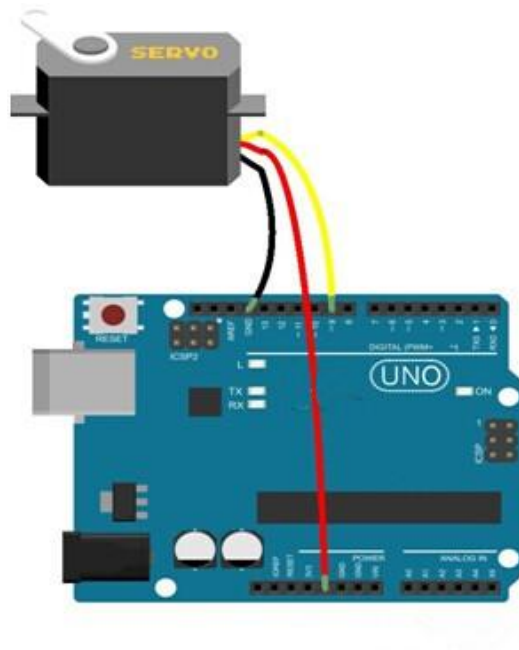
- 1 * Uno board
- 1 * USB data cable
- 1 * Servo
- Several jumper wires

Experimental Principle

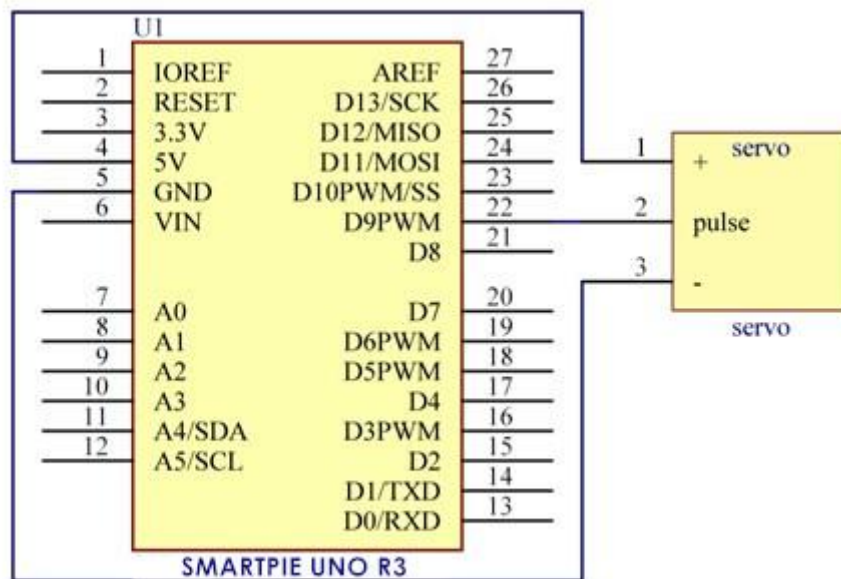
Servo consists of shell, circuit board, non-core motor, gear and location detection. Its working principle is as follows: Uno board sends PWM signal to servo motor, and then this signal is processed by IC on circuit board to calculate rotation direction to drive the motor, and then this driving power is transferred to swing arm by reduction gear. At the same time, position detector returns location signal to judge whether set location is reached or not.

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram:



The corresponding schematic diagram is as follows:

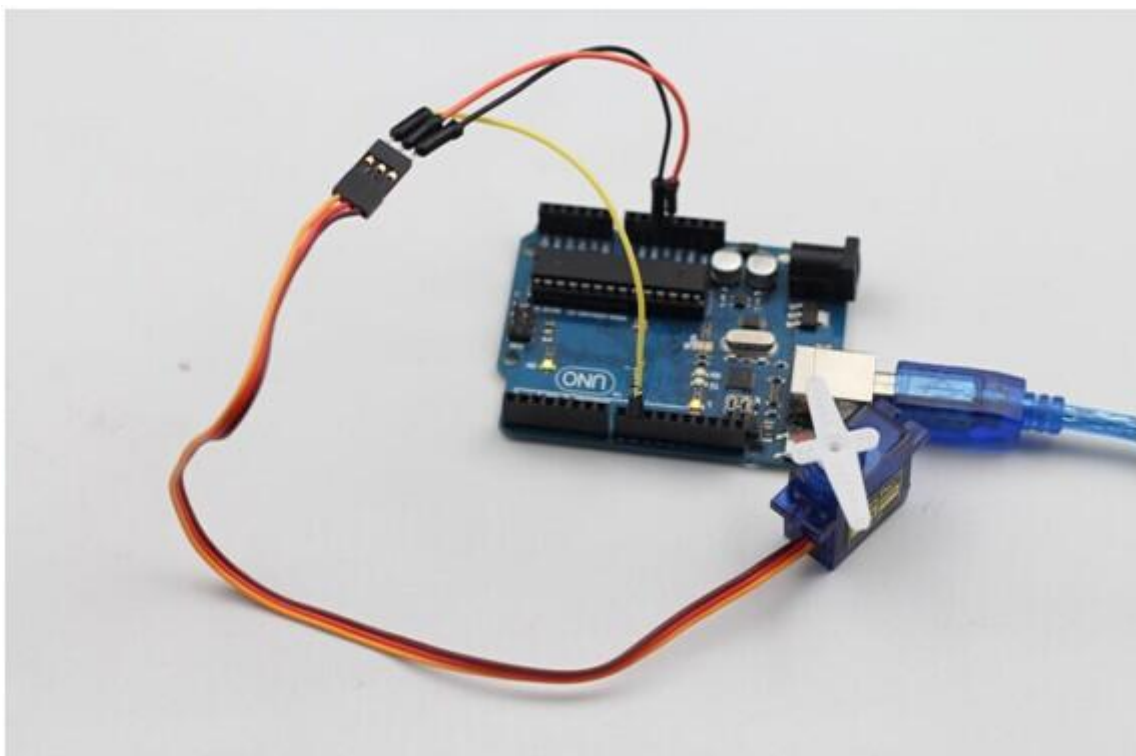


Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Now, you can see the servo motor rotate 90 degrees (rotate once every 15 degrees). And then rotate in opposite direction.



Lesson 9 LCD1602

Introduction

In this experiment, we will use the Uno board to directly drive LCD1602 to display characters.

Components

- 1 * Uno board
- 1 * Breadboard
- 1 * LCD1602
- 1 * Potentiometer (50k Ω)
- Jumper wires
- 1 * USB cable

Principle

Generally speaking, LCD1602 has parallel ports, that is, it needs to control several pins at the same time. LCD1602 can be categorized into an eight-port connection and four-port connection. If the eight-port connection is used, then the digital ports of the Uno board are basically completely occupied. If you want to connect more sensors, there will be no ports available. Therefore, we will use the four-port connection.

Introduction to the pins of LCD1602:

VSS: A pin that connects to ground

VDD: A pin that connects to a +5V power supply

VO: A pin that adjust the contrast of LCD1602

RS: A register select pin that controls where in the LCD's memory you are writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

R/W: A Read/Write pin that selects reading mode or writing mode

E: An enabling pin that, when supplied with low-level energy, causes the LDC module to execute relevant instructions.

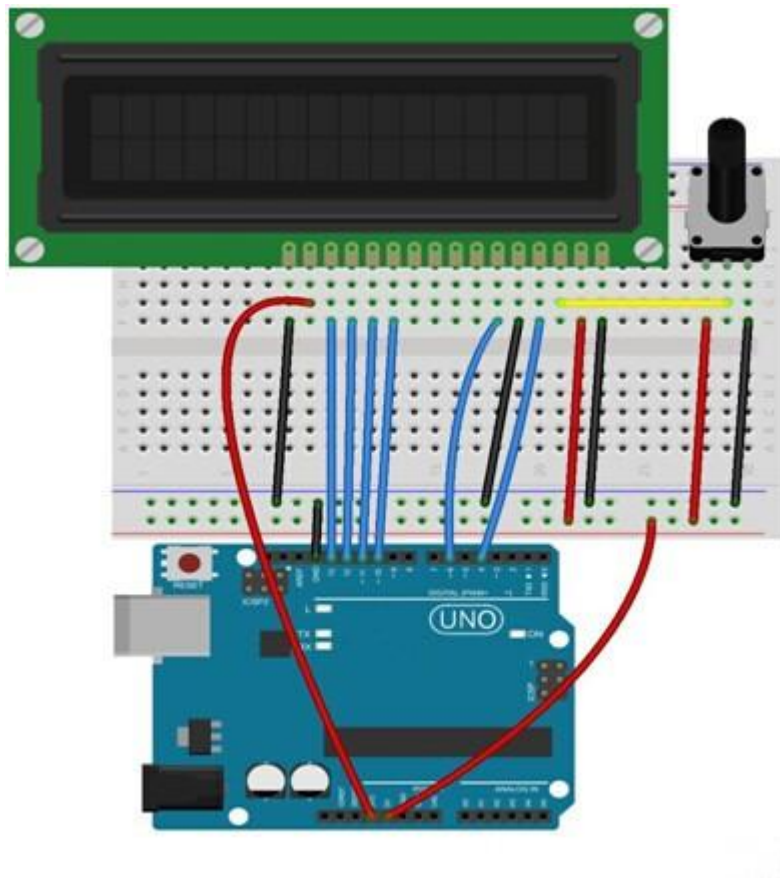
D0-D7 : Pins that read and write data

A and K: Pins that control the LED backlight

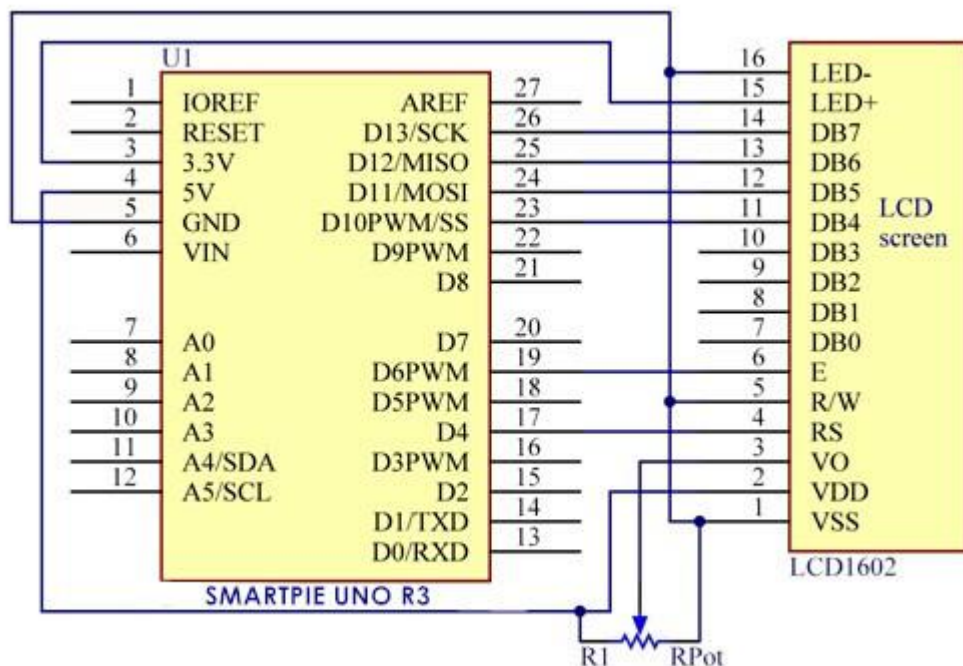
In this experiment, we will use a 50K Ω potentiometer to adjust the contrast of LCD1602 to display characters or figures however you want. For programming, we will optimize it by calling function libraries.

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram (please make sure pins are connected correctly or characters will not display properly):



The corresponding schematic diagram is as follows:

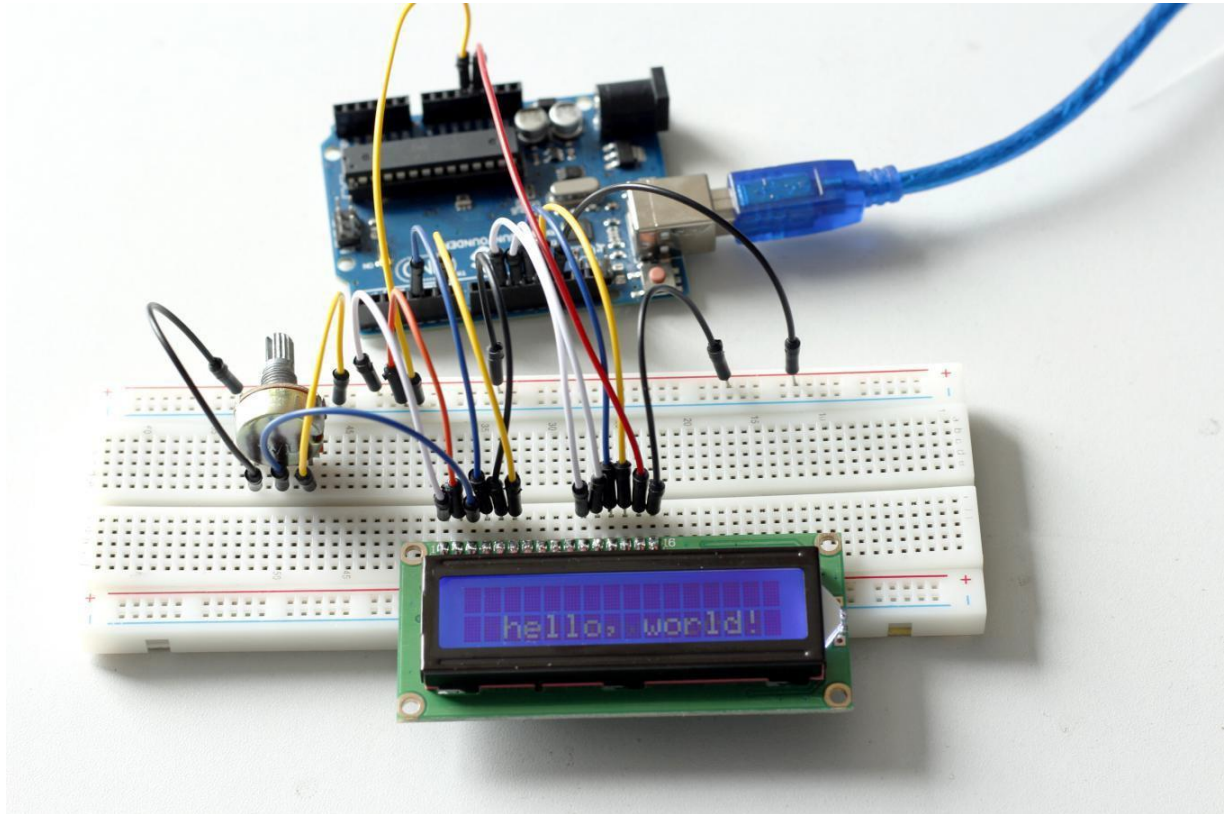


Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

You should now see your LCD1602 display the flowing characters "HSTORE" and "hello, world!".



Experimental Summary

Through this experiment, you've learned how to drive LCD1602. Now you can create your own messages to display! You can also try letting your LCD1602 display numbers.

Lesson 10 Thermistor

Introduction

A thermistor is a type of resistor whose resistance varies significantly with temperature.

Components

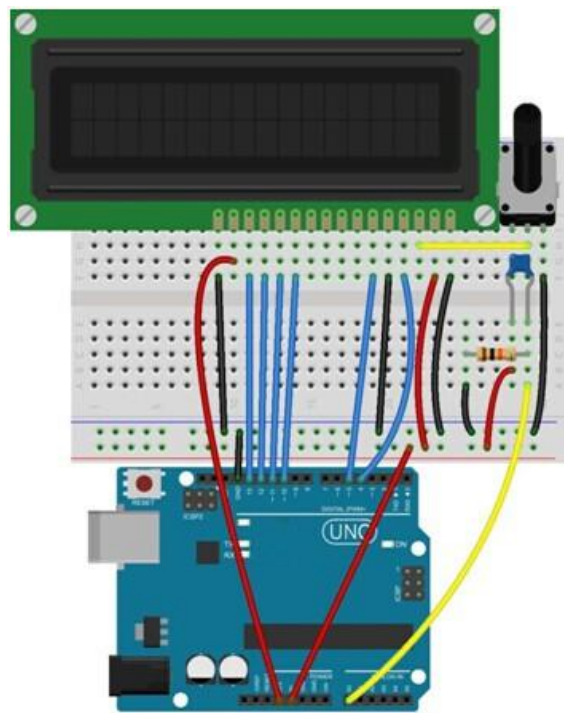
- 1 * Uno board
- 1 * USB data cable
- 1 * Breadboard
- 1 * Thermistor
- Several jumper wires
- 1 * Potentiometer (50K Ω)
- 1 * Resistor (10K Ω)
- 1 * LCD1602

Experimental Principle

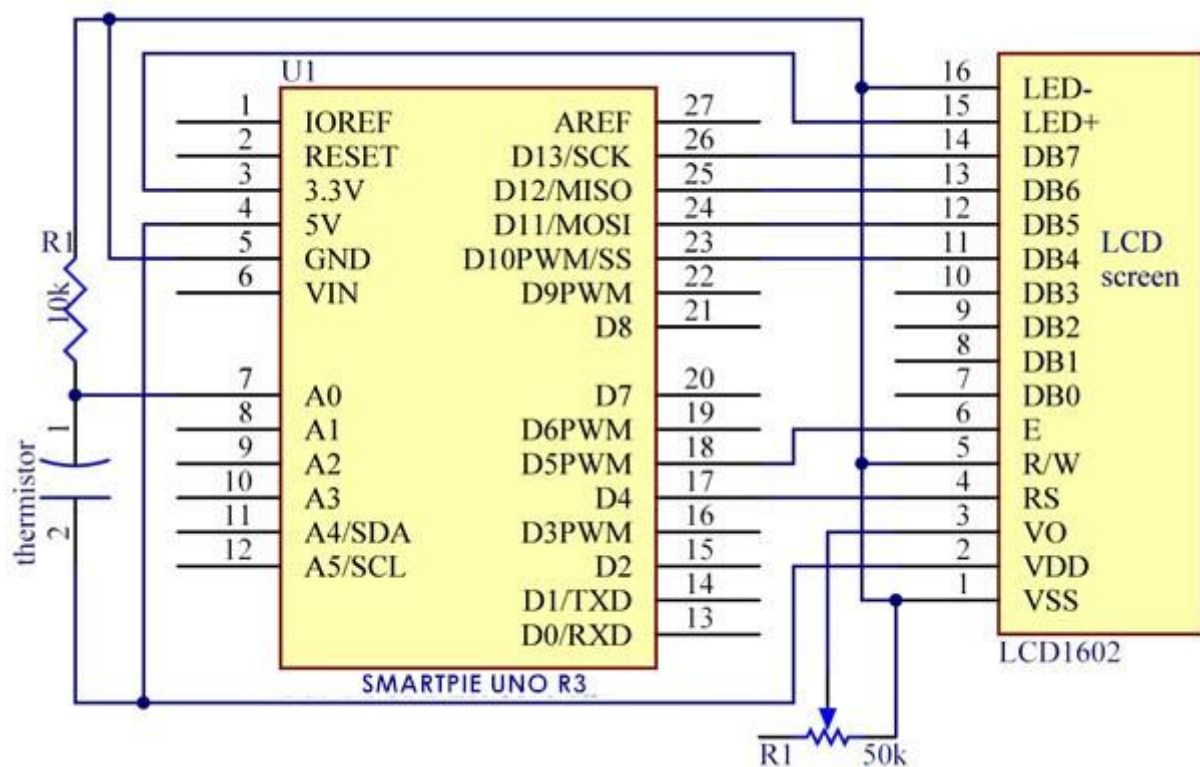
The resistance of the thermistor varies significantly with ambient temperature. It can detect surrounding temperature changes in real time. Send the temperature data to analog I/O port of Uno board. Next we only need to convert sensor output to Celsius temperature by simple programming and display it on the LCD1602.

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram:



The corresponding schematic diagram is as follows:

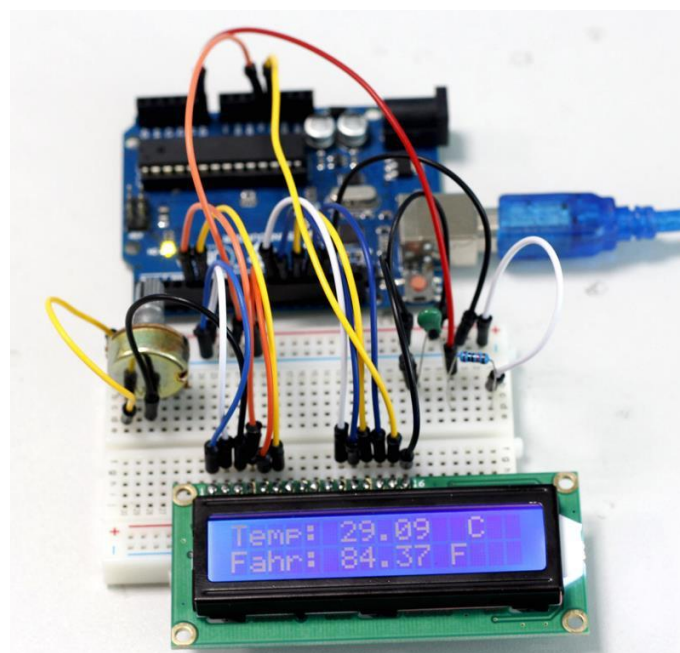


Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Now, you can see current temperature displayed on LCD1602 both in Celcius and Fahrenheit degrees.



Lesson 11 Voltmeter

Introduction

In this lesson, we will use two potentiometers and a LCD1602 to make a voltmeter.

Components

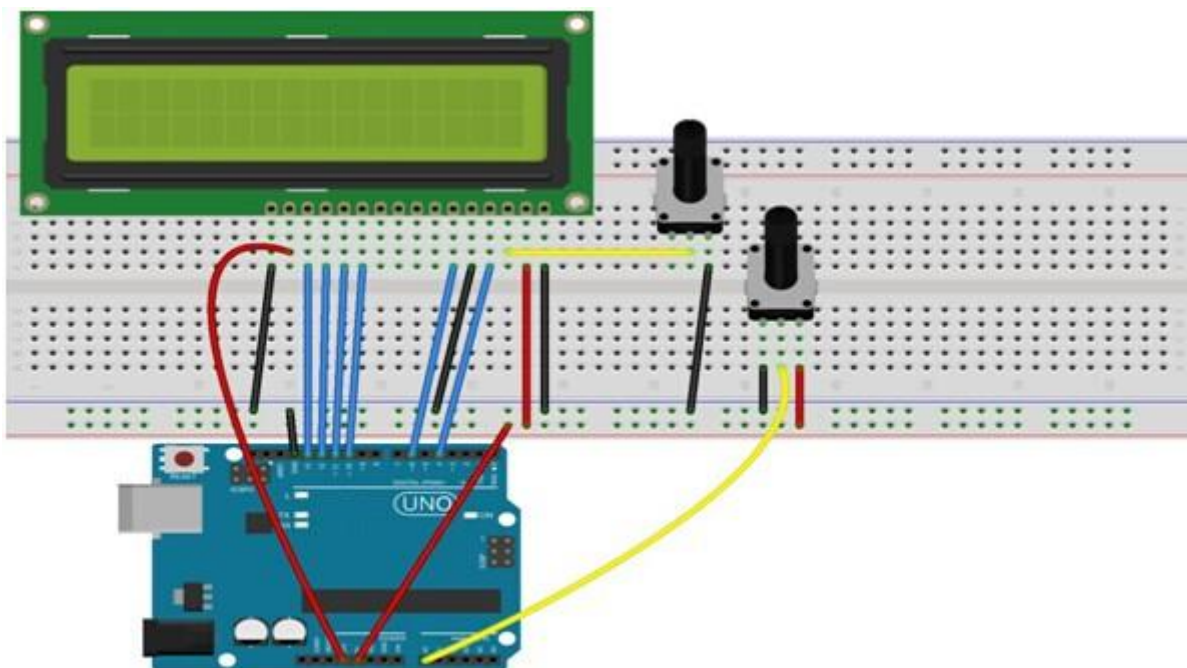
- 1 * Uno board
- 1 * USB data cable
- 2 * Potentiometer
- 1 * LCD1602
- Several jumper wires
- 1 * Breadboard

Experimental Principle

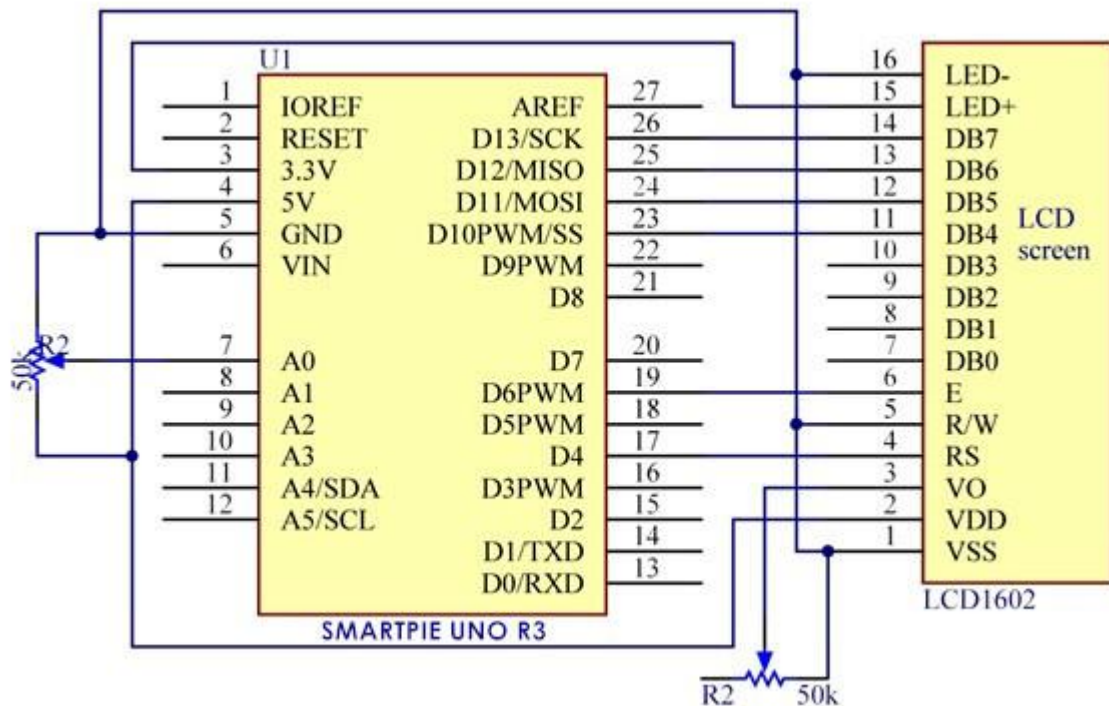
One potentiometer is used to adjust the contrast of the LCD1602. And the other is used to divide voltage.

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram:



The corresponding schematic diagram is as follows:

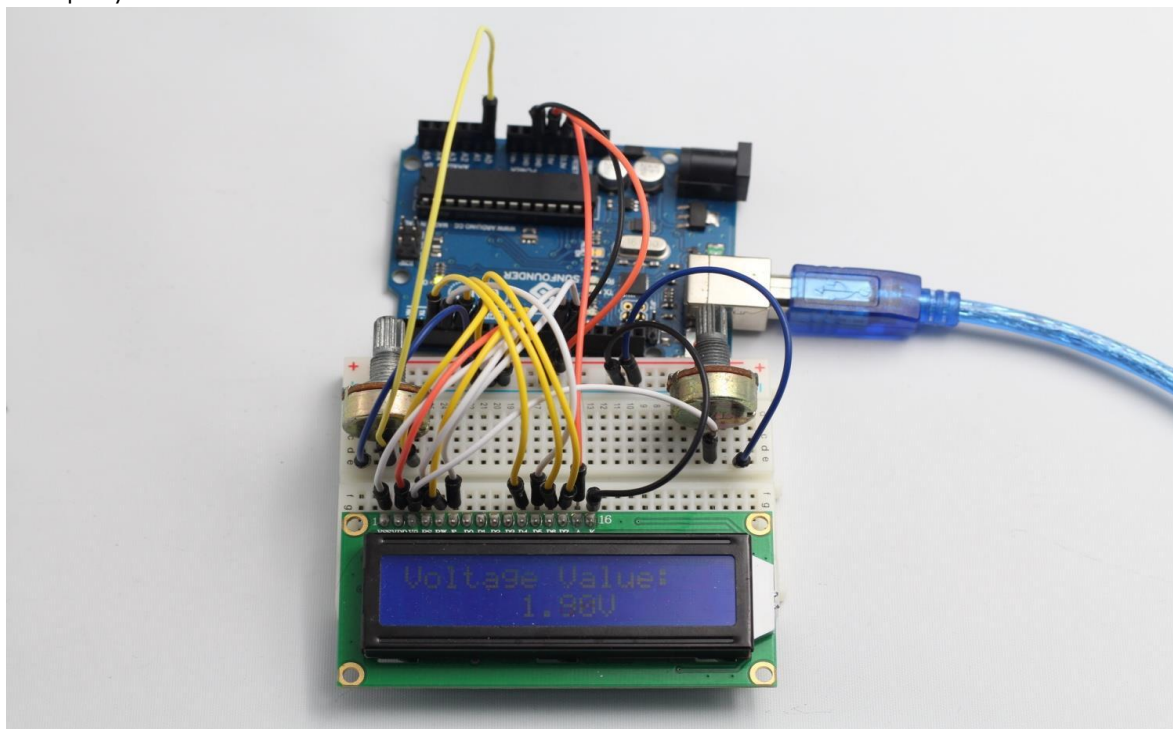


Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Now, if you adjust the potentiometer which is used to divide voltage, you will see the voltage value displayed on the LCD1602 varies.



Lesson 12 Ultrasonic

Introduction

The ultrasonic sensor is used to sense distance from objects.

Components

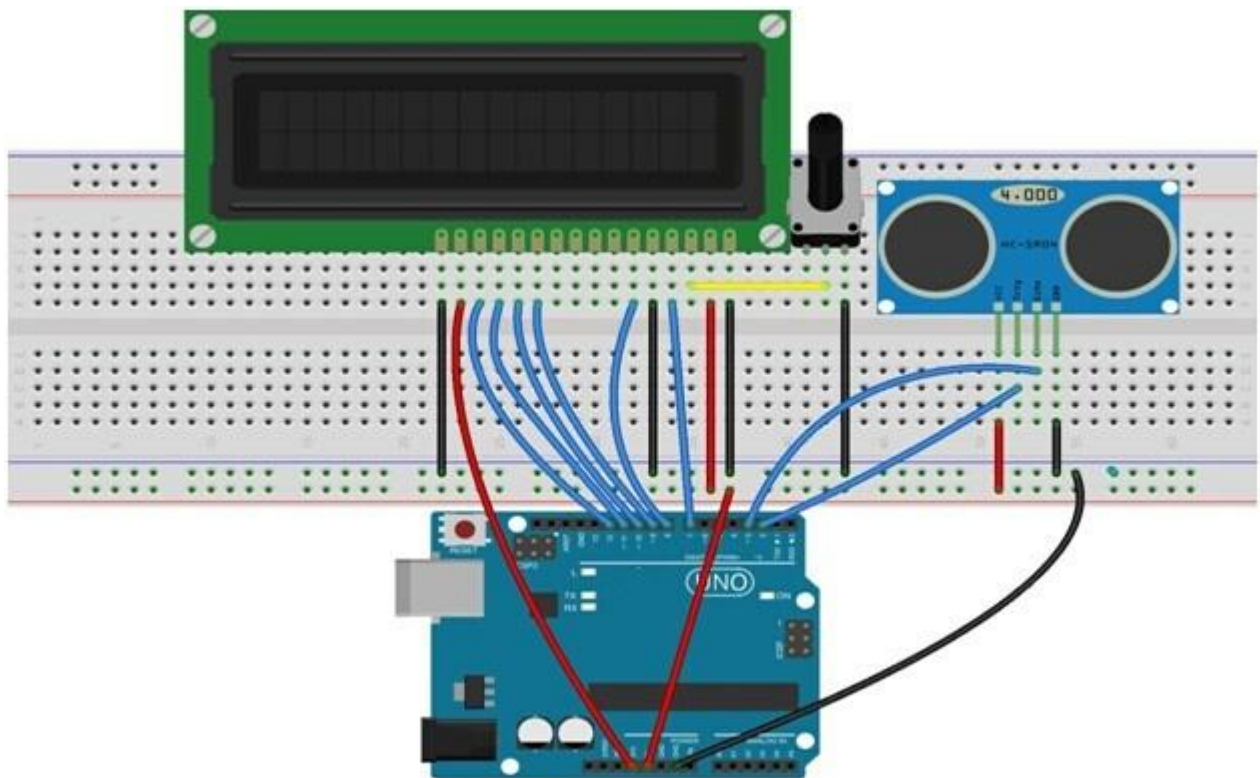
- 1 * Uno board
- 1 * USB data cable
- 1 * Breadboard
- 1 * Ultrasonic sensor
- 1 * LCD1602
- 1 * Potentiometer
- Several jumper wires

Experimental Principle

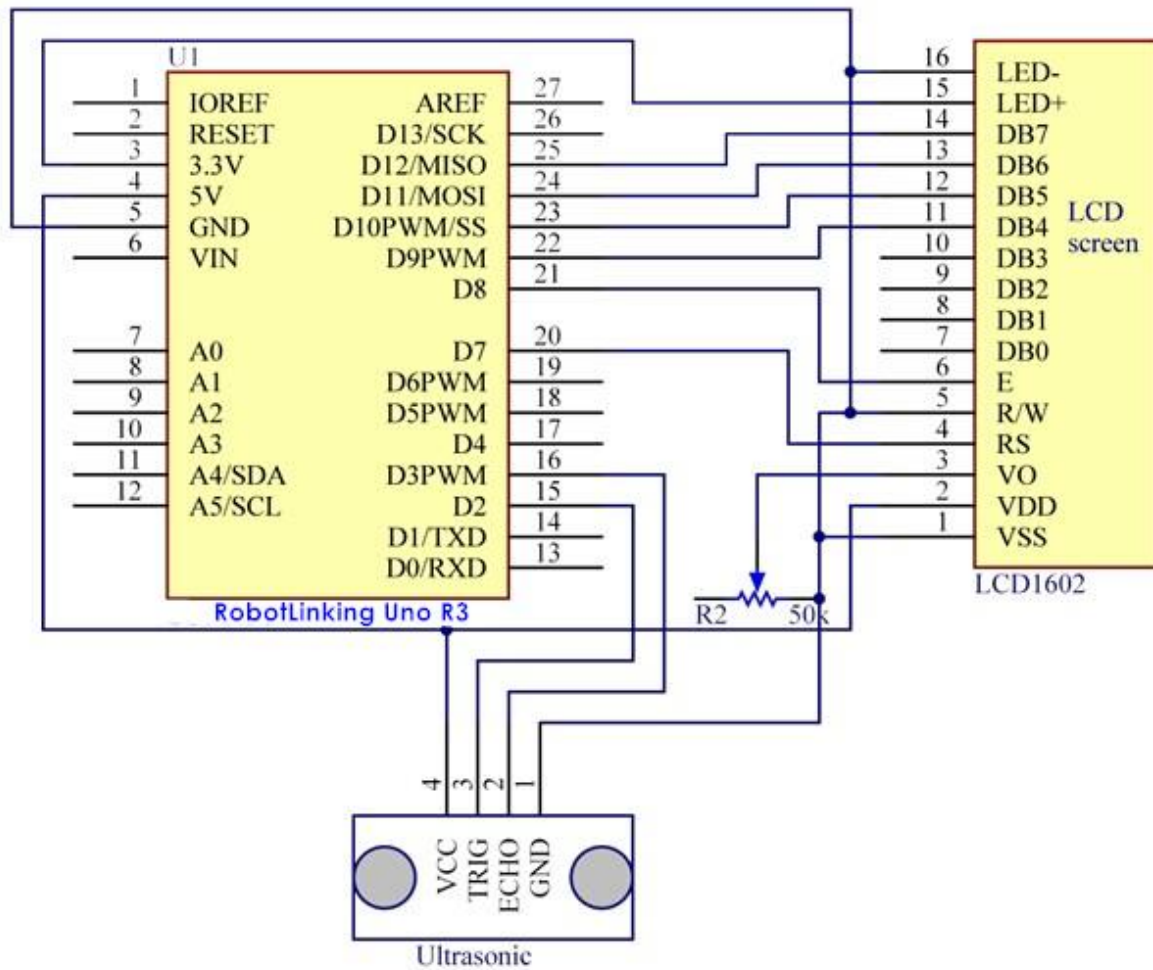
This sensor works by sending a sound wave out and calculating the time it takes for the sound wave to get back to the ultrasonic sensor. By doing this, it can tell us how far away objects are relative to the ultrasonic sensor.

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram:



The corresponding schematic diagram is as follows:

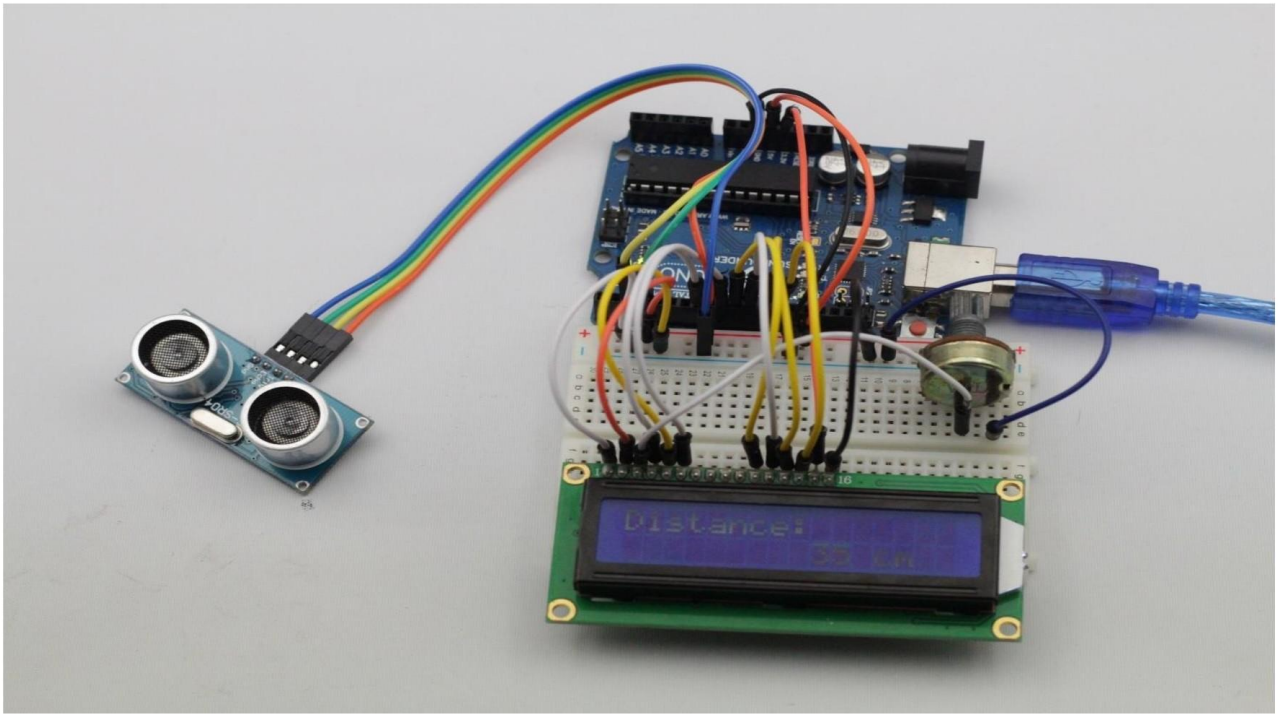


Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Now, if you use a piece of paper to approach or keep it far away from the sensor. You will see the value displayed on the LCD varies, which indicates the distance between the paper and the ultrasonic sensor.



Lesson 13 Stopwatch

Introduction

In this lesson, we will use a four digit 7-segment display to make a stopwatch.

Components

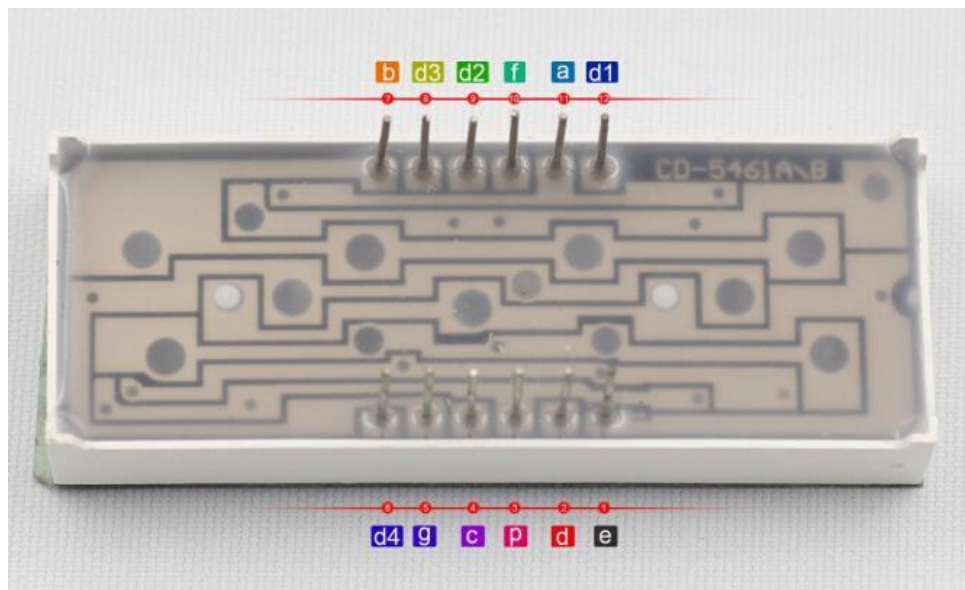
- 1 * Uno board
- 1 * USB data cable
- 1 * Four digit 7-segment display
- Several jumper wires
- 1 * Breadboard
- 8 * Resistor (220Ω)

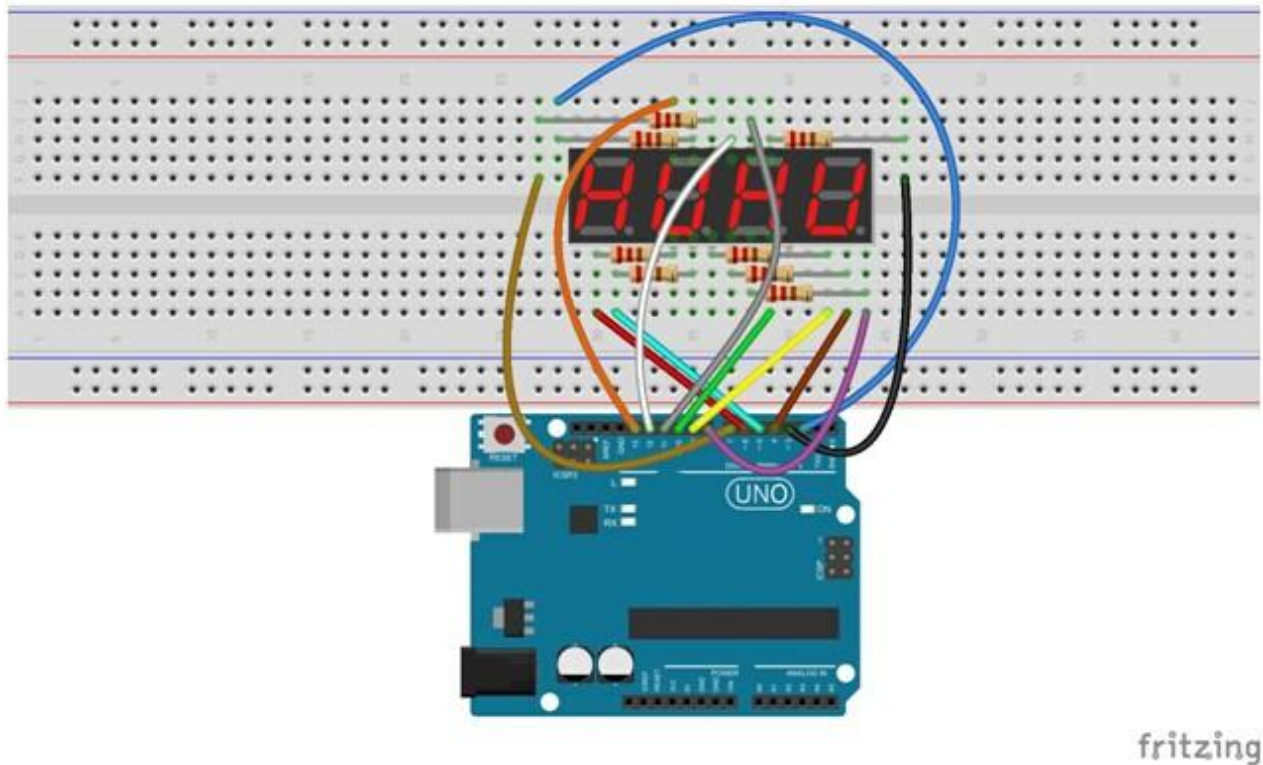
Experimental Principle

When using one digit 7-segment display, if it is common anode, we will connect common anode pin to power source; if it is common cathode, we will connect common cathode pin to GND. When using four digit 7-segment display, the common anode or common cathode pin are used to control which digit is displayed. There is only one digit working. However, based on the principle of Persistence of Vision, we can see four 7-segment display is all displaying numbers. This is because electronic scanning speed is fast and we cannot notice it.

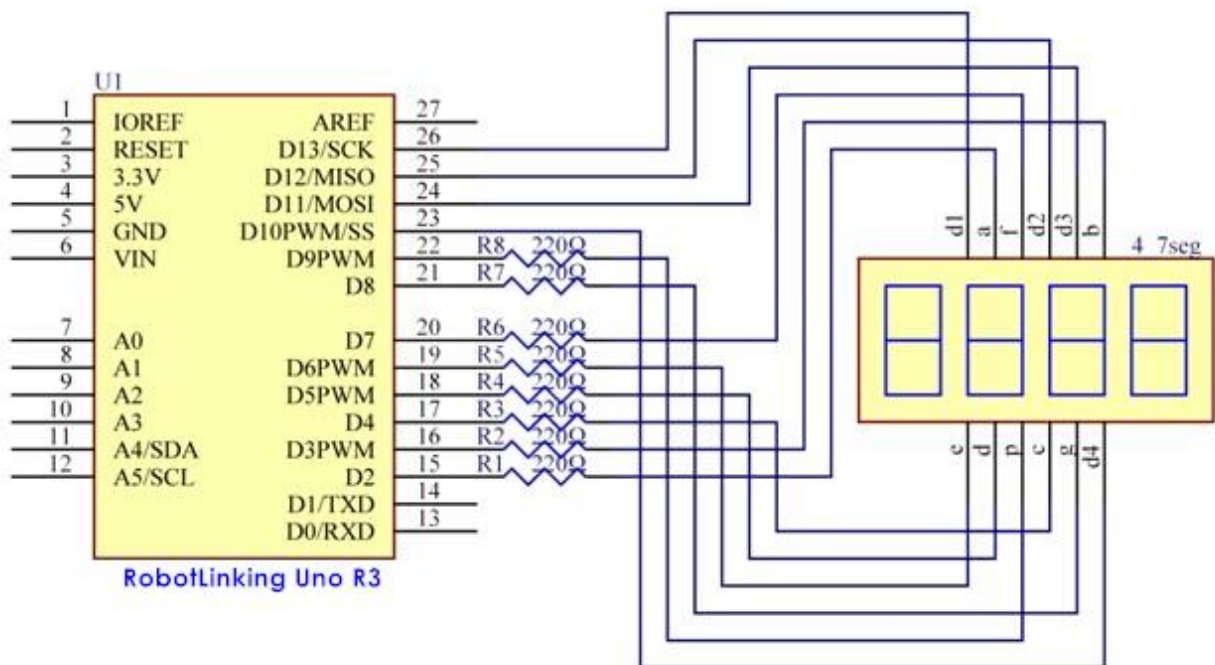
Experimental Procedures

Step 1: Connect the circuit





The corresponding schematic diagram is as follows:

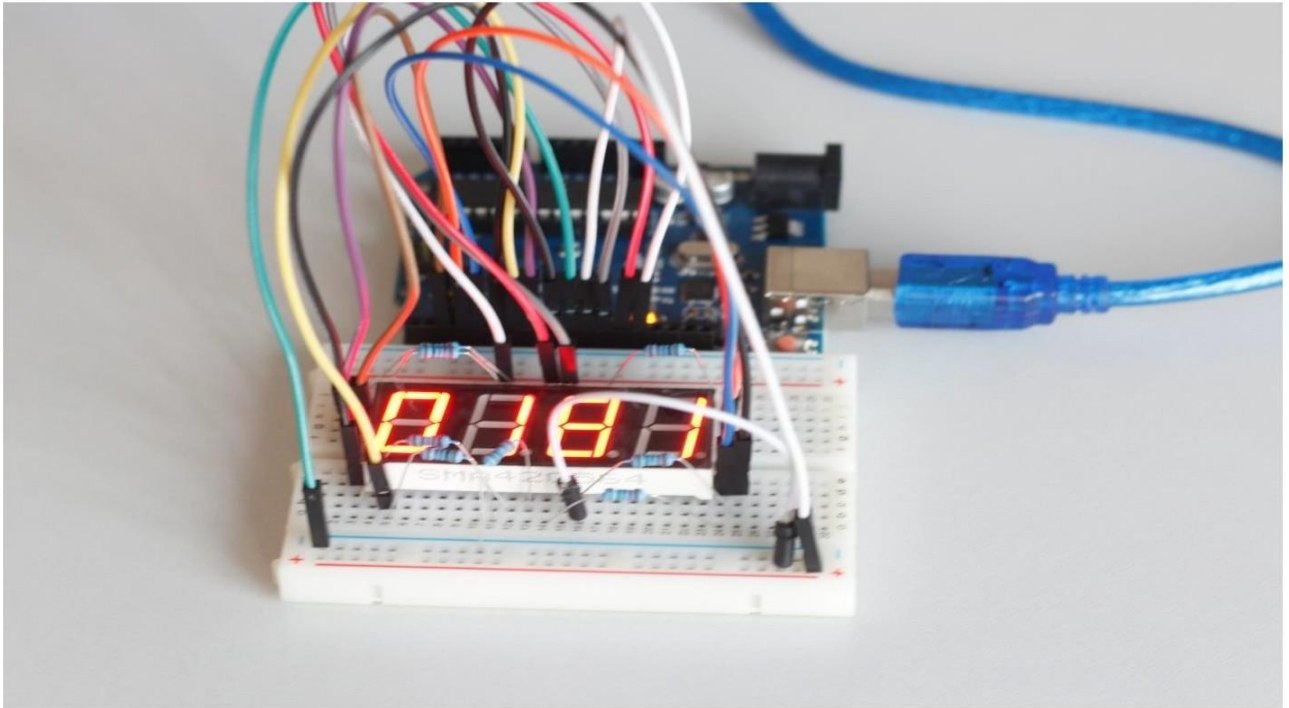


Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board.

Now you can see number plus one per second on segment display.



Lesson 14 74HC595 And Segment Display

Introduction

Rewrite " Lab7 make use of seven-segment display countdown function "to 74HC595 shift register control a seven-segment display sequentially displays the number from 9-0, making the effect of the digital countdown on the seven-segment display.

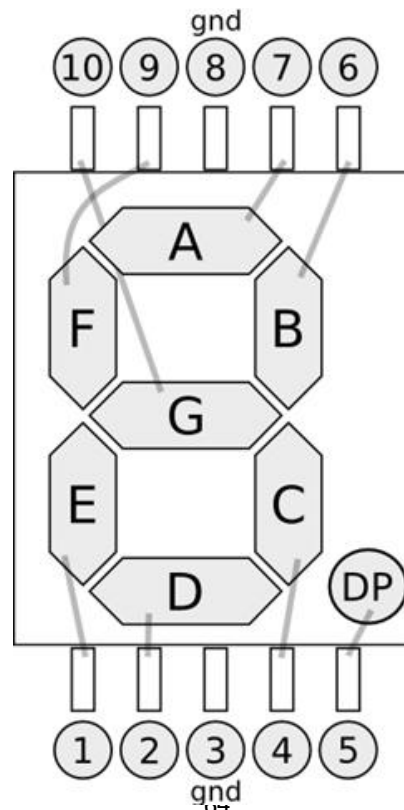
Note: This test assumes that you have done, " Lab7 make reciprocal use of seven-segment display function "and"Lab11 use 74HC595 and three pins to control 8 LED "principle has been known seven-segment display and a 74HC595 using separate ways.

Components

- 1 * Uno board
- 1 * USB data cable
- 1 * 74HC595
- 1 * 1 Segment Display
- 8 * 220R Resistors
- Several jumper wires
- 1 * Breadboard

Experimental Principle

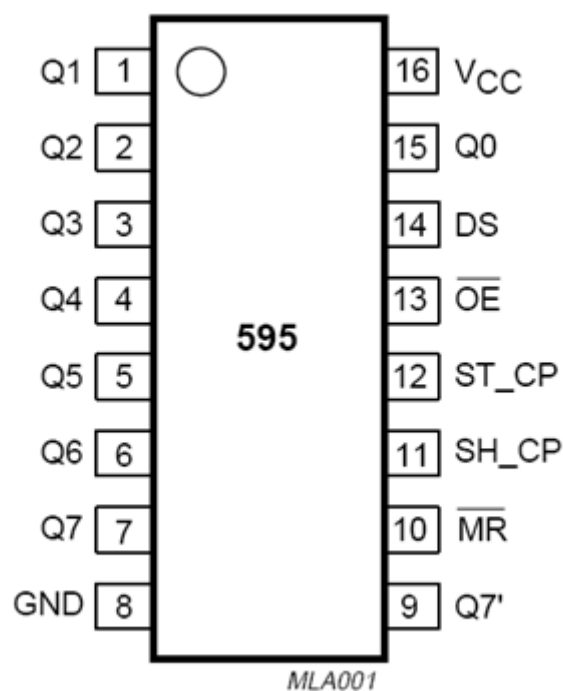
Seven segment display pin diagram below (the picture shows common cathode seven-segment display):



0-9 ten digits correspond with each segment are as follows (the following table applies common cathode seven segment display device, if you are using a common anode, the table should be replaced every 1 0 0 should all replaced by 1):

Display digital	dp	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
1	0	0	1	1	0	0	0	0
2	0	1	1	0	1	1	0	1
3	0	1	1	1	1	0	0	1
4	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1
6	0	1	0	1	1	1	1	1
7	0	1	1	1	0	0	0	0
8	0	1	1	1	1	1	1	1
9	0	1	1	1	1	0	1	1

74HC595 shift register pin diagram is as follows:



Pin Number	Name	Explanation
1-7, 15	Q0 ~ Q7	Output pins
8	GND	Ground
7	Q7'	Sequence output (Serial Out)
10	MR	Master Reset, clear all data, active low (Active low)
11	SH_CP	SHift register clock pin (Clock Pin)
12	ST_CP	STorage register clock pin (Latch Pin)
13	OE	Output Enable, allowing the output, active low (Active low)
14	DS	Sequence data input (Serial data input)
16	Vcc	Supply voltage

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram:

The following table shows the seven-segment display 74HC595 pin correspondence table:

74HC595 pin	Seven shows remarkable control pin (stroke)
Q0	7 (A)
Q1	6 (B)
Q2	4 (C)
Q3	2 (D)
Q4	1 (E)
Q5	9 (F)
Q6	10 (G)
Q7	5 (DP)

Step one: Connect 74HC595

First, the wiring is connected to power and ground:

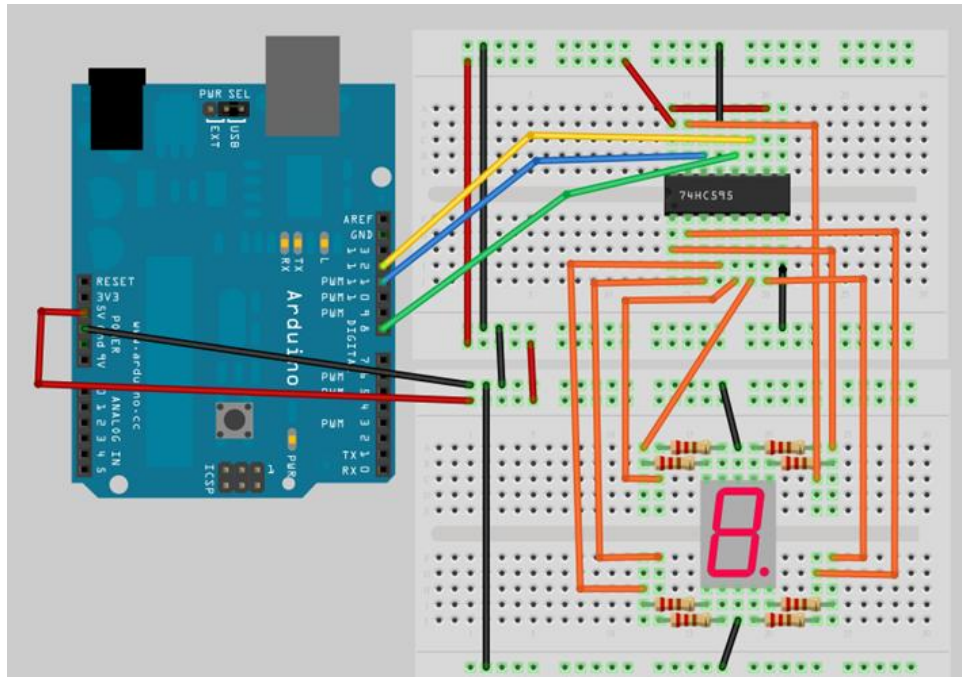
- Vcc (pin 16) and MR (pin 10) connected to 5V
- GND (pin 8) and OE (pin 13) to ground

Connection DS, ST_CP and SH_CP pin:

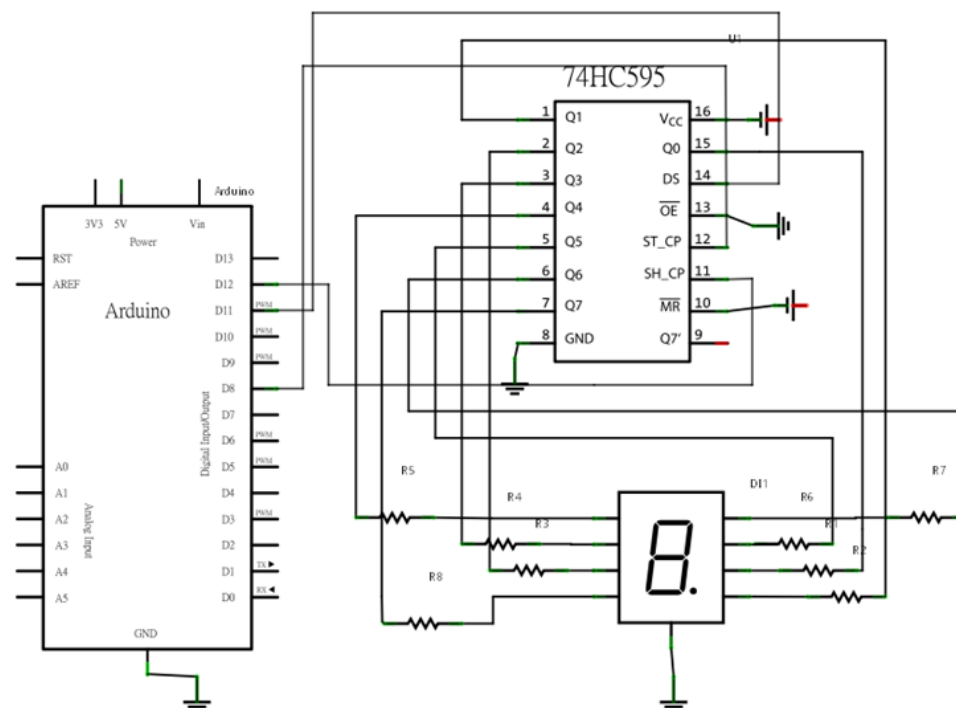
- DS (pin 14) connected to Arduino pin 11 (the figure below the blue line)
- ST_CP (pin 12, latch pin) connected to Arduino pin 8 (FIG green line below)
- SH_CP (pin 11, clock pin) connected to Arduino pin 12 (the figure below the yellow line)

Step two: Connect the seven segment display

- The seven-segment display 3, 8 pin to GND (This example uses the common cathode, anode set if the total of 3, 8 pin to + 5V)
- According to the table of the 74HC595 Q0 ~ Q7 received a seven-segment display corresponding pin (A ~ G and DP), and then each foot in a 220 ohm resistor in series



Circuit diagram



Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Lesson 15 JOYSTICK

Introduction

In this lesson we will learn how to use a joystick module.

Components

- 1 * Uno board
- 1 * Breadboard
- 1 * Joystick
- Jumper wires
- 1 * USB cable

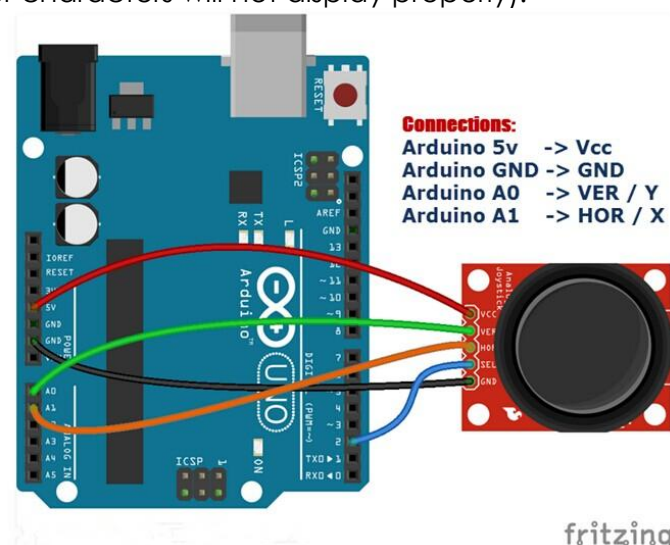
Principle

The module has 5 pins: Vcc, Ground, X, Y, Key. Note that the labels on yours may be slightly different, depending on where you got the module from. The thumbstick is analog and should provide more accurate readings than simple 'directional' joysticks that use some forms of buttons, or mechanical switches. Additionally, you can press the joystick down (rather hard on mine) to activate a 'press to select' push-button.

We have to use analog Arduino pins to read the data from the X/Y pins, and a digital pin to read the button. The Key pin is connected to ground, when the joystick is pressed down, and is floating otherwise. To get stable readings from the Key /Select pin, it needs to be connected to Vcc via a pull-up resistor. The built in resistors on the Arduino digital pins can be used.

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram (please make sure pins are connected correctly or characters will not display properly):



Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Step 5: Open "tool-monitor", to see the data.

Lesson 16 1 CHANNEL RELAY MODULE

Introduction

In this experiment, we will learn how to use the 1 channel relay module.

Relay is a kind of component when the change of the input variables (incentive) to specified requirements, the output electric circuits of the charged amount occurs due to the step change of a kind of electrical appliances. This company produces the relay module can meet in 28 v to 240 v ac or dc power to control all kinds of other electric parts. MCU can be used to achieve the goal of timing control switch. Can be applied to guard against theft and alarm, toys, construction and other fields. Relay is an electrical control device. It has a control system (also called input circuit) and control system (also called the output circuit), the interaction between. Usually used in automatic control circuit, it is actually with a small current to control large current operation of a kind of "automatic s

Components

- 1 * Uno board
- 1 * USB cable
- 1 * 1 channel relay module
- Dupont wires (Female to Male)

Principle



Experimental Procedures

Step 1: Connect circuit as :

Uno R3 GND --> Module pin -

Uno R3 +5V --> Module pin +5V

Uno R3 Digital 2 --> Resistor or Not-->Module SW

Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Lesson 17 DC Motors

Introduction

In this lesson, you will learn how to control a small DC motor using an UNO R3 and a transistor.

Components

- 1 * Uno board
- 1 * Breadboard
- 1 * Small 6V DC Motor
- 1 * PN2222 Transistor
- 1 * 1N4007 diode
- 1 * 220 Ω Resistor
- Jumper wires
- 1 * USB cable

Principle

You will use an Arduino analog output (PWM) to control the speed of the motor by sending a number between 0 and 255 from the Serial Monitor.

When you put together the breadboard, there are two things to look out for.

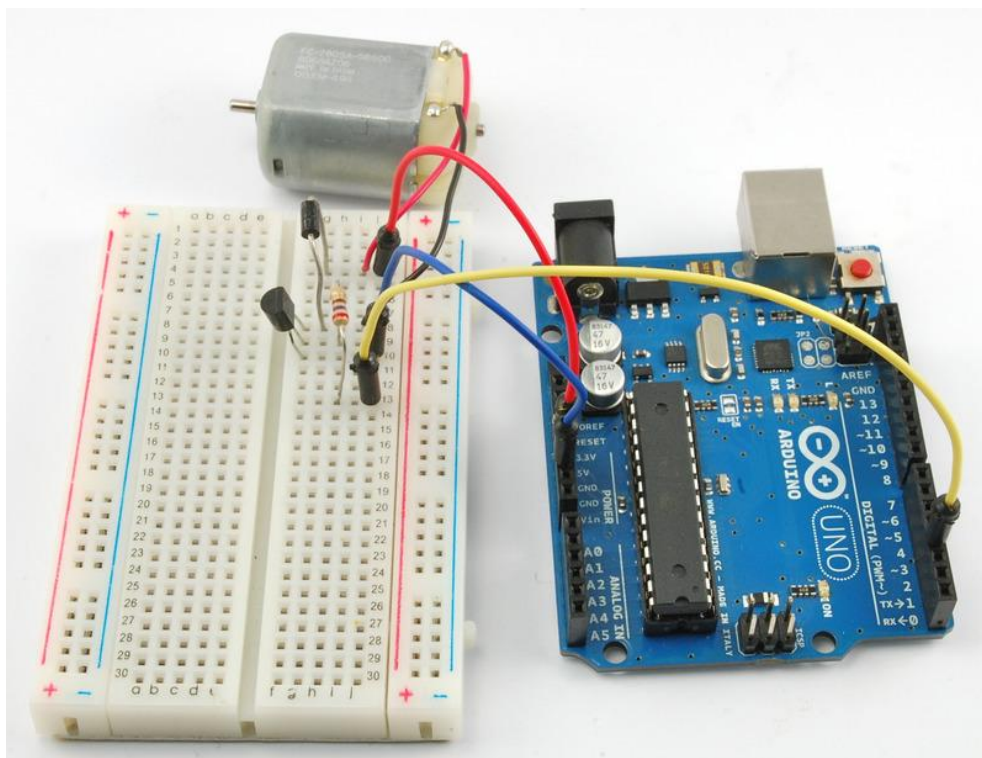
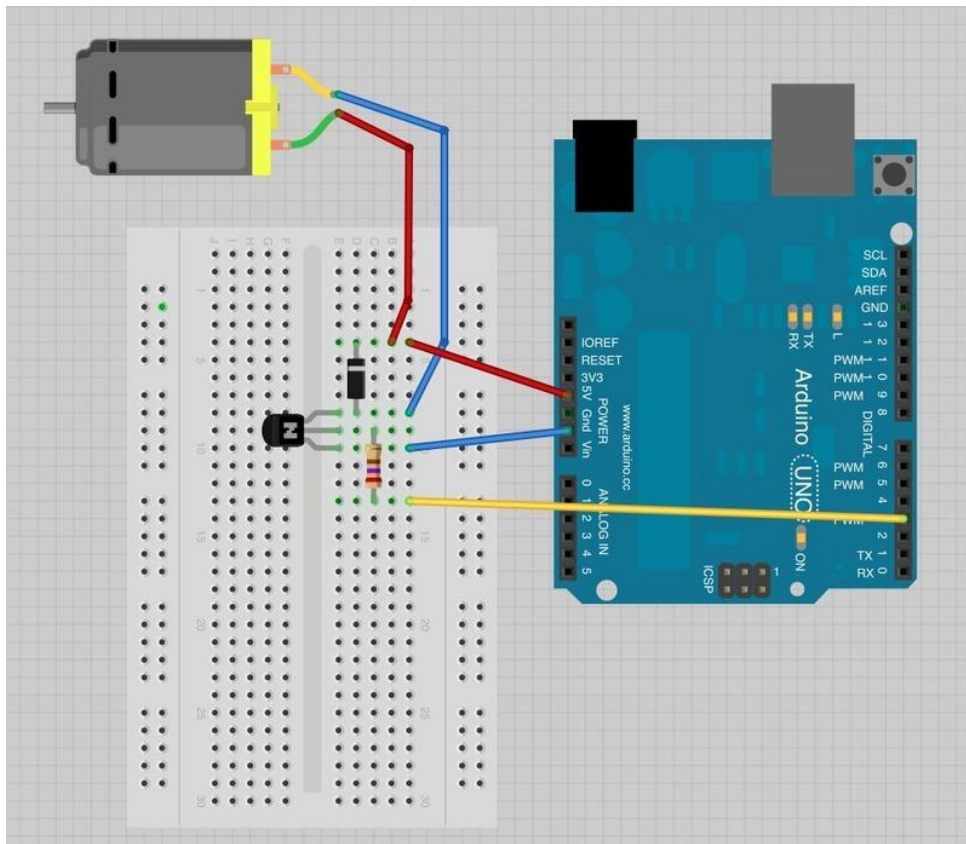
Firstly, make sure that the transistor is the right way around. The flat side of the transistor should be on the right-hand side of the breadboard.

Secondly the striped end of the diode should be towards the +5V power line - see the image below!

The motor that comes with Adafruit Arduino kits does not draw more than 250mA but if you have a different motor, it could easily draw 1000mA, more than a USB port can handle! If you aren't sure of a motor's current draw, power the Arduino from a wall adapter, not just USB

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram (please make sure pins are connected correctly or characters will not display properly):



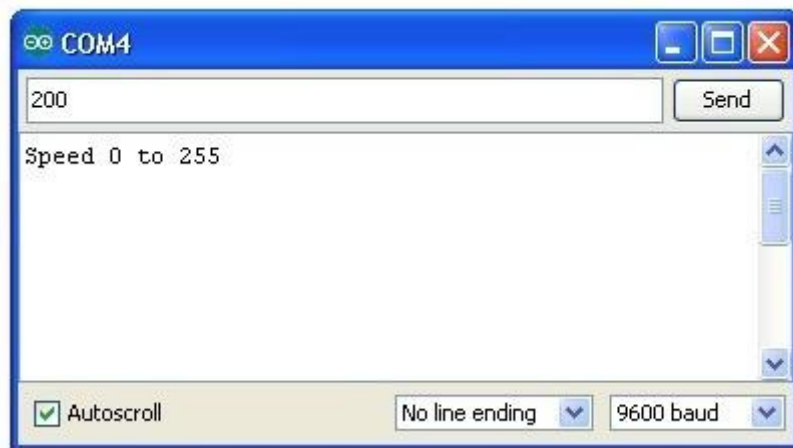
Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

The transistor acts like a switch, controlling the power to the motor, Arduino pin 3 is used to turn the transistor on and off and is given the name 'motorPin' in the sketch.

When the sketch starts, it prompts you, to remind you that to control the speed of the motor you need to enter a value between 0 and 255 in the Serial Monitor.



In the 'loop' function, the command 'Serial.parseInt' is used to read the number entered as text in the Serial Monitor and convert it into an 'int'.

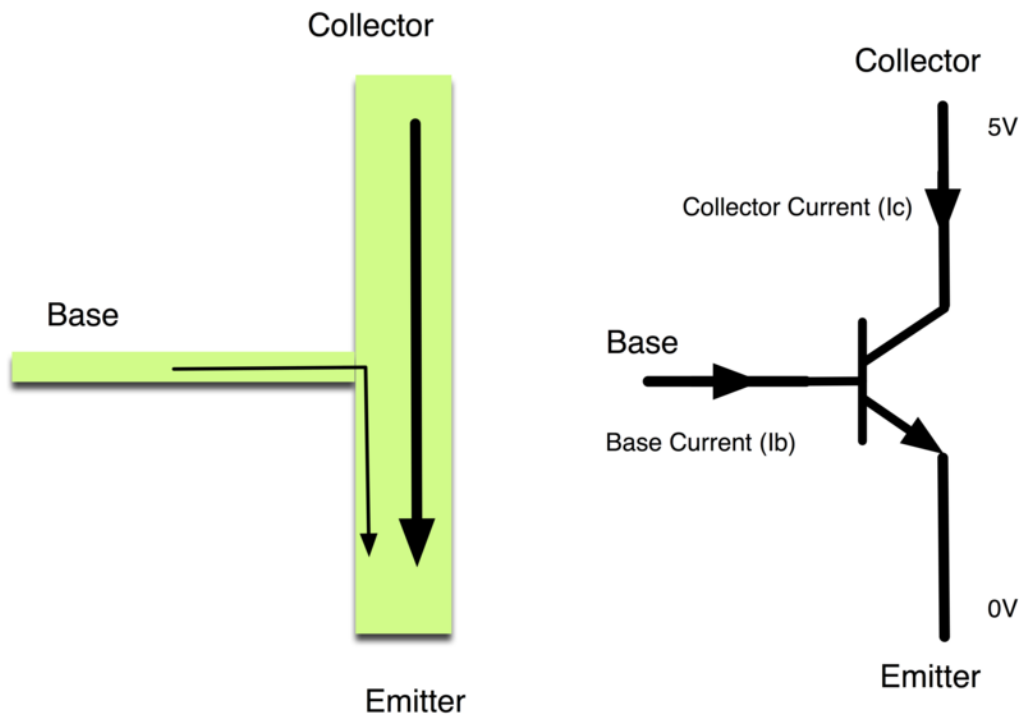
You could type any number here, so the 'if' statement on the next line only does an analog write with this number if the number is between 0 and 255.

Experimental Summary

A. Transistors

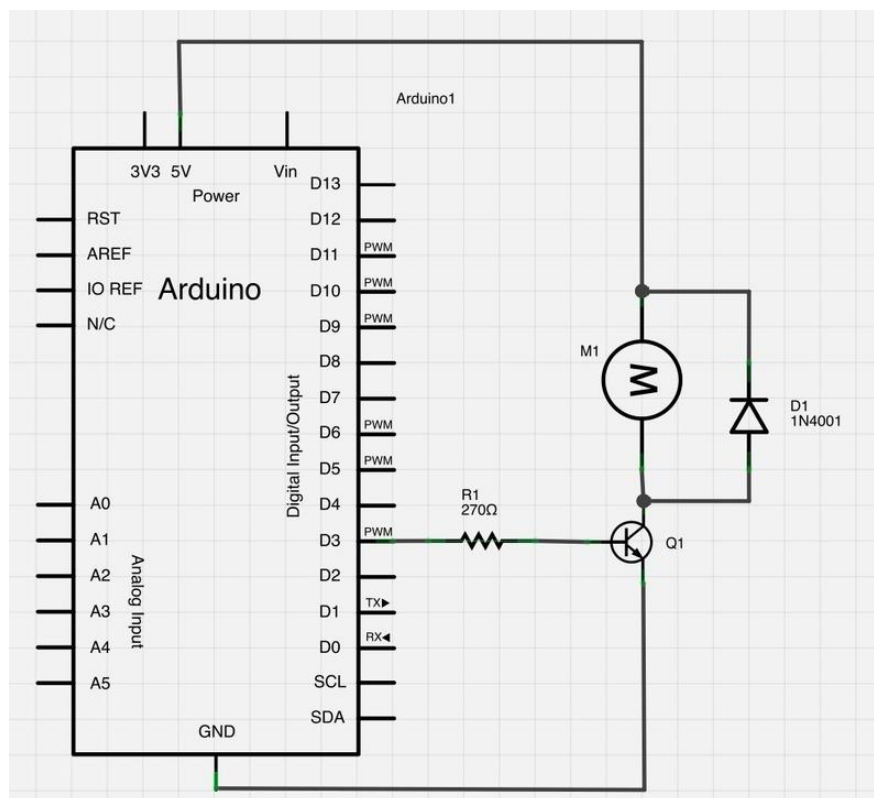
The small DC motor, is likely to use more power than an Arduino digital output can handle directly. If we tried to connect the motor straight to an Arduino pin, there is a good chance that it could damage the Arduino.

A small transistor like the PN2222 can be used as a switch that uses just a little current from the Arduino digital output to control the much bigger current of the motor.



The transistor has three leads. Most of the electricity flows from the Collector to the Emitter, but this will only happen if a small amount is flowing into the Base connection. This small current is supplied by the Arduino digital output.

The diagram below is called a schematic diagram. Like a breadboard layout, it is a way of showing how the parts of an electronic project are connected together.



The pin D3 of the Arduino is connected to the resistor. Just like when using an LED, this limits the current flowing into the transistor through the base.

There is a diode connected across the connections of the motor. Diodes only allow electricity to flow in one direction (the direction of their arrow).

When you turn the power off to a motor, you get a negative spike of voltage, that can damage your Arduino or the transistor. The diode protects against this, by shorting out any such reverse current from the motor.

Through this experiment, you've learned how to drive LCD1602. Now you can create your own messages to display! You can also try letting your LCD1602 display numbers.

B. Other Things to Do

Try reversing the connections to the motor. What happens?

Try entering different values (starting at 0) into the Serial Monitor and notice at what value the motor starts to actually turn. You will find that the motor starts to 'sing' as you increase the analog output.

Try pinching the drive shaft between your fingers. Don't hold it like that for too long, or you may cook the transistor, but you should find that it is fairly easy to stop the motor. It is spinning fast, but it does not have much torque.

Lesson 18 DC Motors Reversing

Introduction

In this lesson, you will learn how to control both the direction and speed of a small DC motor using an Arduino and the L293D motor driver chip.

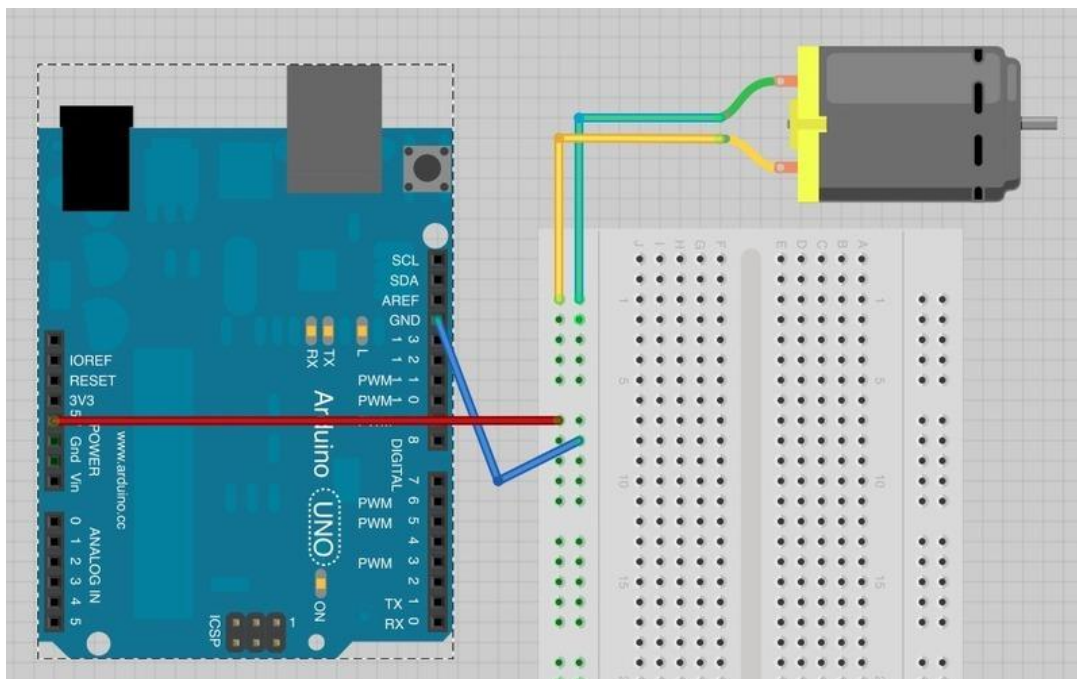
Components

- 1 * Uno board
- 1 * Breadboard
- 1 * Small 6V DC Motor
- 1 * 10 kΩ variable resistor (pot)
- 1 * L293D IC
- 1 * Tactile push switch
- Jumper wires
- 1 * USB cable

Principle

Before we get the UNO R3 board to control the motor, we should experiment with the L293D motor control chip to get an idea how it works.

We can start by just using the UNO R3 to supply 5V to the motor.

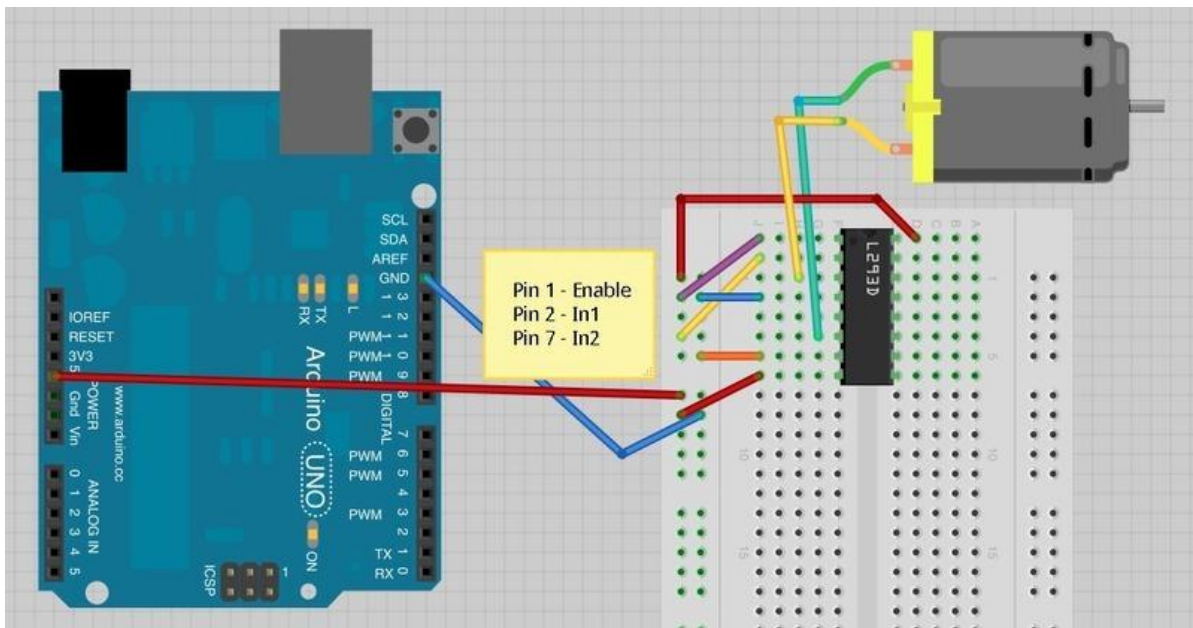


Note which way the motor is spinning. You can do this by pinching the motor shaft between

your fingers. Swap over the motor leads so that the motor lead that was going to +5V now goes to GND and vice-versa. The motor will turn in the opposite direction.

This gives us a clue as to how the L293D chip works. Its control pins allow us to do the equivalent of swapping over the motor terminals to reverse the direction of the motor.

Build up the breadboard as below. The Arduino is still just supplying power, but we can experiment manually with the control pins before we let the Arduino take over.



The three pins of L293D that we are interested in are Pin 1 (Enable), Pin 2 (In1) and Pin 7 (In2). These are attached to either 5V or GND using the purple, yellow and orange jumper wires.

As shown above, the motor should be turning on one direction, let's call that direction A.

If you move Pin 1 (Enable) to GND the motor will stop, no matter what you do with the control pins In1 and In2. Enable turns everything on and off. This makes it useful for using a PWM output to control the motor speed. Reconnect Pin 1 to 5V so that the motor starts again.

Now try moving In1 (pin 2, yellow) from 5V to GND. In1 and In2 are both now connected to GND, so again the motor will stop.

Moving In2 from GND to 5V will cause the motor to turn in the opposite direction (direction B).

Finally, moving In1 back to 5V so that both In1 and In2 are at 5V will again cause the motor to stop.

The effect of the pins In1 and In2 on the motor are summarized in the table below:

In1

Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Pins are defined and their modes set in the 'setup' function as normal.

In the loop function, a value for the motor speed is found by dividing the analog reading from the pot by 4.

The factor is 4 because the analog reading will be between 0 and 1023 and the analog output needs to be between 0 and 255.

If the button is pressed, the motor will run in forward, otherwise it will run in reverse. The value of the 'reverse' variable is just set to the value read from the switch pin. So, if the button is pressed, this will be False, otherwise it will be True.

The speed and reverse values are passed to a function called 'setMotor' that will set the appropriate pins on the driver chip to control the motor.

```
void setMotor(int speed, boolean reverse)
{
  analogWrite(enablePin, speed);
  digitalWrite(in1Pin, ! reverse);
  digitalWrite(in2Pin, reverse);
}
```

Firstly, the speed is set, by using an analog Write to the enable pin. The enable pin of the L293 just turns the motor on or off irrespective of what the in1 and in2 pins of the L293 are set to.

To control the direction of the motor, the pins in1 and in2 must be set to opposite values.

If in1 is HIGH and in2 is LOW, the motor will spin one way, if on the other hand in1 is HIGH and in2 LOW then the motor will spin in the opposite direction.

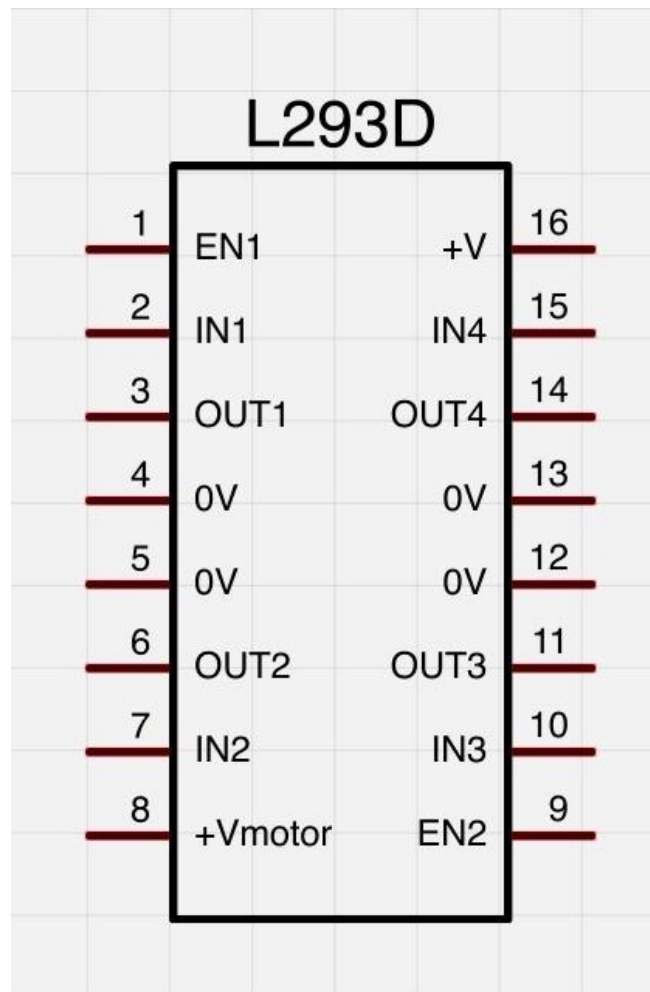
The '!' command means 'not'. So the first digitalWrite command for in1 sets it to the opposite of whatever the value of 'reverse' is, so if reverse is HIGH it sets it to LOW and vice versa.

The second digitalWrite for 'in2' sets the pin to whatever the value of 'reverse' is. This means that it will always be the opposite of whatever in1 is.

Experimental Summary

A.L293D

This is a very useful chip. It can actually control two motors independently. We are just using half the chip in this lesson, most of the pins on the right hand side of the chip are for controlling a second motor.



A second motor would be attached between OUT3 and OUT4. You will also need three more control pins.

EN2 is connected to a PWM enabled output pin on the Arduino

IN3 and IN4 are connected to digital outputs on the Arduino

The L293D has two +V pins (8 and 16). The pin '+Vmotor' (8) provides the power for the motors, and +V (16) for the chip's logic. We have connected both of these to the Arduino 5V pin. However, if you were using a more powerful motor, or a higher voltage motor, you would provide the motor with a separate power supply using pin 8 connected to the positive power supply and the ground of the second power supply is connected to the ground of the Arduino.

B.Other Things to Do

You could try changing the sketch to control the motor without using the pot or switch. It could start slow in the forward direction, gradually get faster, slow down and then go into reverse, repeating the pattern.

Lesson 19 Steeper Motor

Introduction

Stepper motors fall somewhere in between a regular DC motor and a servo motor. They have the advantage that they can be positioned accurately, moved forward or backwards one 'step' at a time, but they can also rotate continuously.

In this lesson you will learn how to control a stepper motor using your Arduino and the same L293D motor control chip that you used with the DC motor in this lesson

Components

- 1 * Uno board
- 1 * Breadboard
- 1 * 5V Stepper Motor
- 1 * L293D IC
- Jumper wires
- 1 * USB cable

Principle

The stepper motor has five leads, and we will be using both halves of the L293D this time. This means that there are a lot of connections to make on the breadboard.

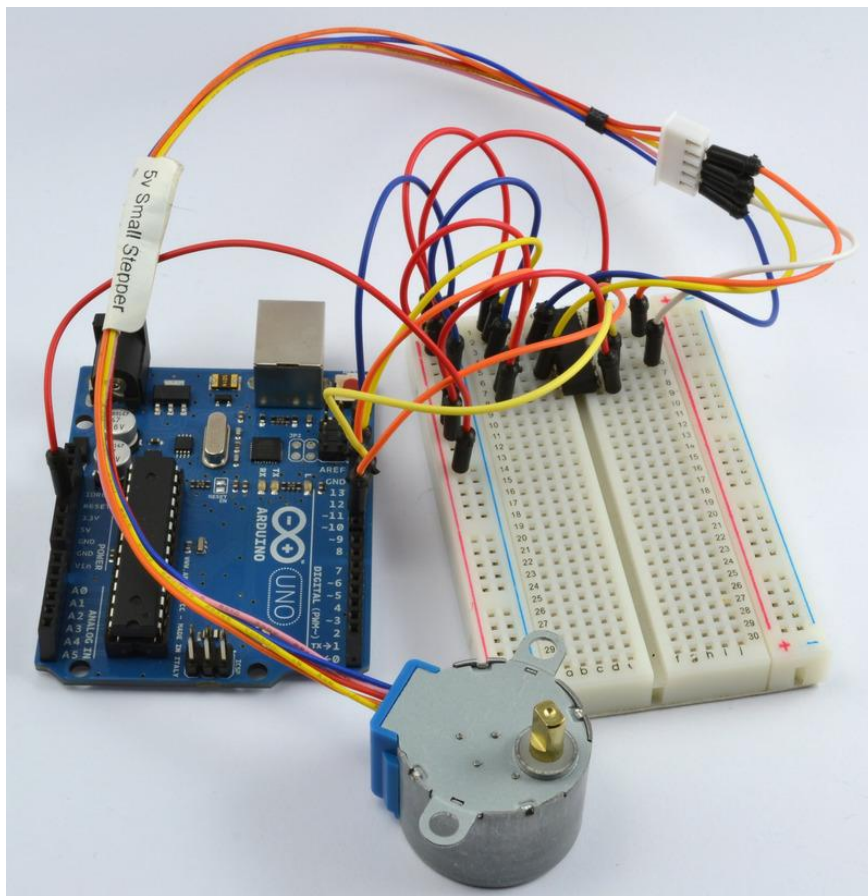
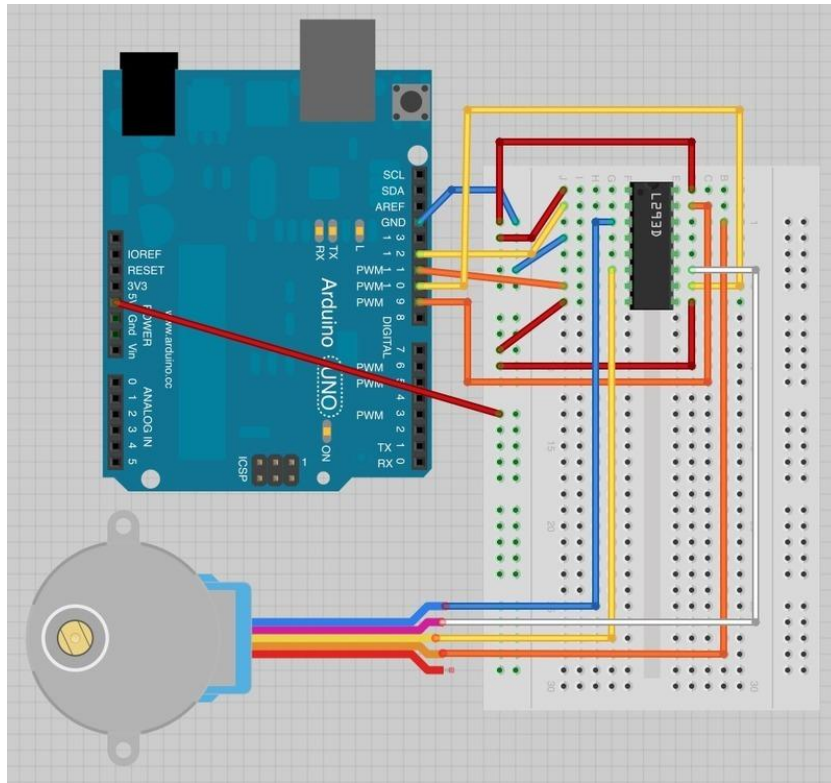
The motor has a 5-way socket on the end. Push jumper wires into the sockets to allow the motor to be connected to the breadboard.

The following sketch uses the Serial Monitor, so once the sketch is installed and running, open the Serial Monitor and enter a number of 'steps'. Try a value of about 500, this should cause the motor to turn through about 360 degrees. Enter -500 and it will turn back in the reverse direction.

The Stepper library is included in newer distributions of the Arduino IDE - you may need to upgrade.

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram (please make sure pins are connected correctly or characters will not display properly):



Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

As you might expect, there is an Arduino library to support stepper motors. This makes the process of using a motor very easy.

After including the 'Stepper' library, the four control pins 'in1' to 'in4' are defined.

To tell the Arduino Stepper library which pins are connected to the motor controller, the following command is used:

```
Stepper motor(768, in1Pin, in2Pin, in3Pin, in4Pin);
```

The first parameter is the number of 'steps' that the motor will take to complete one revolution. The motor can be moved by one step at a time, for very fine positioning.

Serial communications is then started, so that the Arduino is ready to receive commands from the Serial Monitor.

Finally the following command sets the speed that we wish the stepper motor to move, when we subsequently tell it how many steps to rotate.

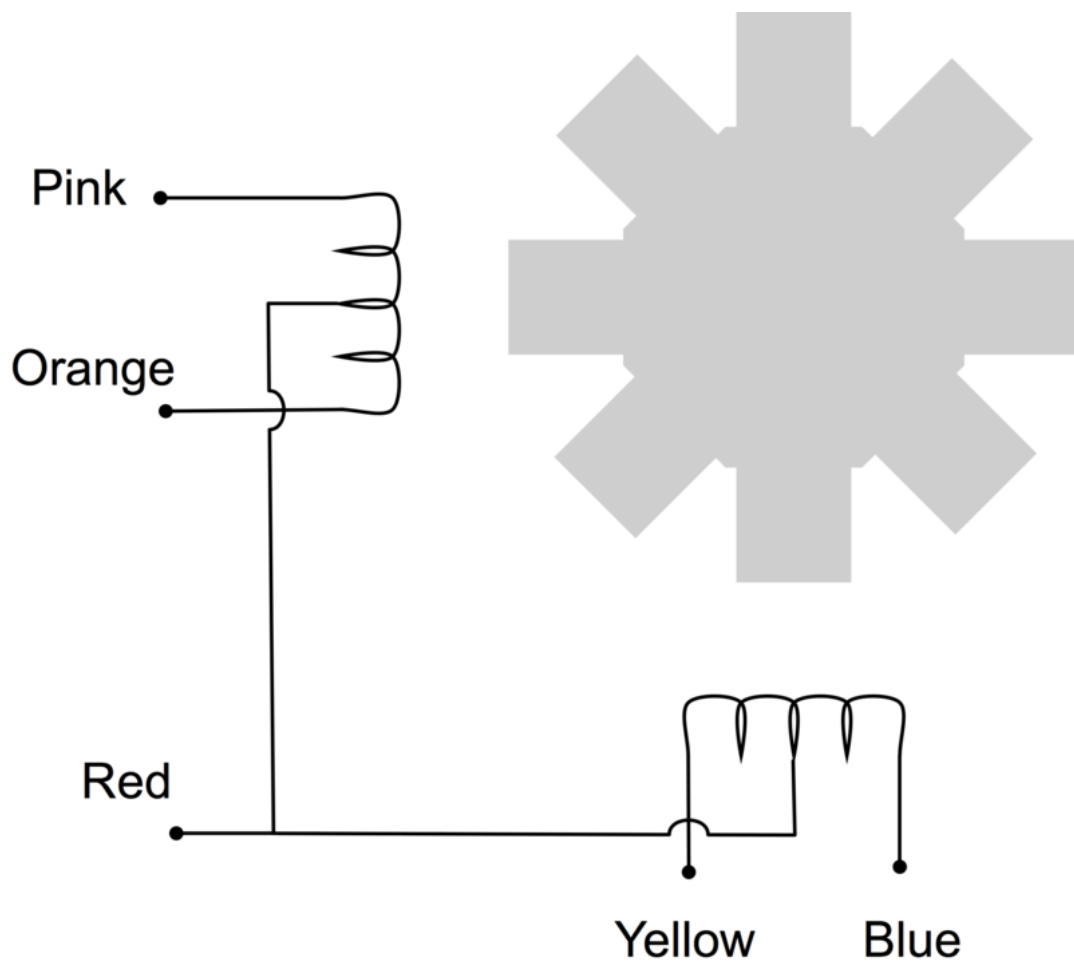
```
motor.setSpeed(10);
```

The 'loop' function is very simple. It waits for a command to come in from the Serial Monitor and converts the text of the number sent into an int using 'parseInt'. It then instructs the motor to turn that number of steps.

Experimental Summary

A. Stepper Motors

Stepper motors use a cogged wheel and electro magnets to nudge the wheel round a 'step' at a time.



By energizing the coils in the right order, the motor is driven round. The number of steps that the stepper motor has in a 360 degree rotation is actually the number of teeth on the cog. The motor we are using has 48 steps, but then the motor also incorporates a reduction gearbox of 1:16 that means that it needs $16 \times 48 = 768$ steps.

In this lesson, we do not use the common Red connection. This connection is only provided if you are using a different type of drive circuit that does not allow the current in each coil to be reversed. Having a center connection to each coil means that you can either energise the left or right side of the coil, and get the effect of reversing the current flow without having to use a circuit that can reverse the current.

Since we are using a L293D that is very good at reversing the current, we do not need this common connection, we can supply current in either direction to the whole of each of the coils.

B.Other Things to Do

Try changing the command that sets the speed of the stepper motor:

```
motor.setSpeed(20);
```

to a lower value (say 5) upload the sketch and notice that the stepper turns more slowly. Now try and find the maximum speed for the stepper by increasing the speed above 20. After a certain point, you will find that the motor does not move at all. This is because it just cannot keep up with the stream of pulses asking it to step.

Try disconnecting the orange and pink leads of the stepper. It should still turn, but you will notice that it is weaker, as it does not have both coils working to push the motor around.

Lesson 20 Automatically Tracking Light Source

Introduction

In this lesson, we will use a servo motor, a photoresistor and a pull-down resistor to assemble an automatically tracking light source system.

Components

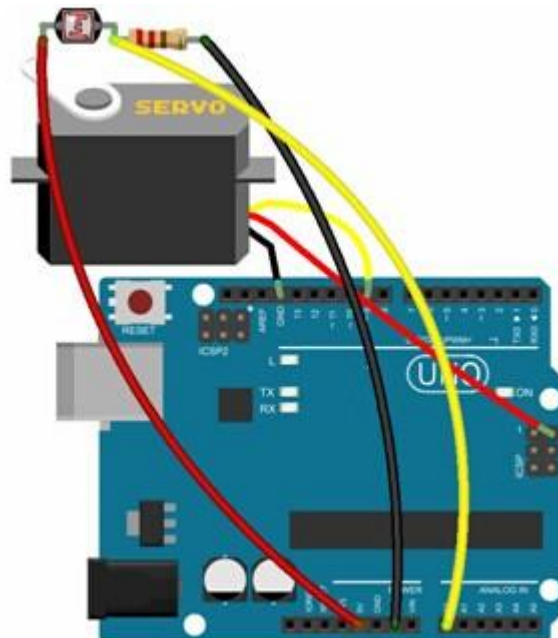
- 1 * Uno board
- 1 * Servo motor
- 1 * Photoresistor
- 1 * Resistor (10K Ω)
- Several jumper wires
- 1 * USB data cable

Experimental Principle

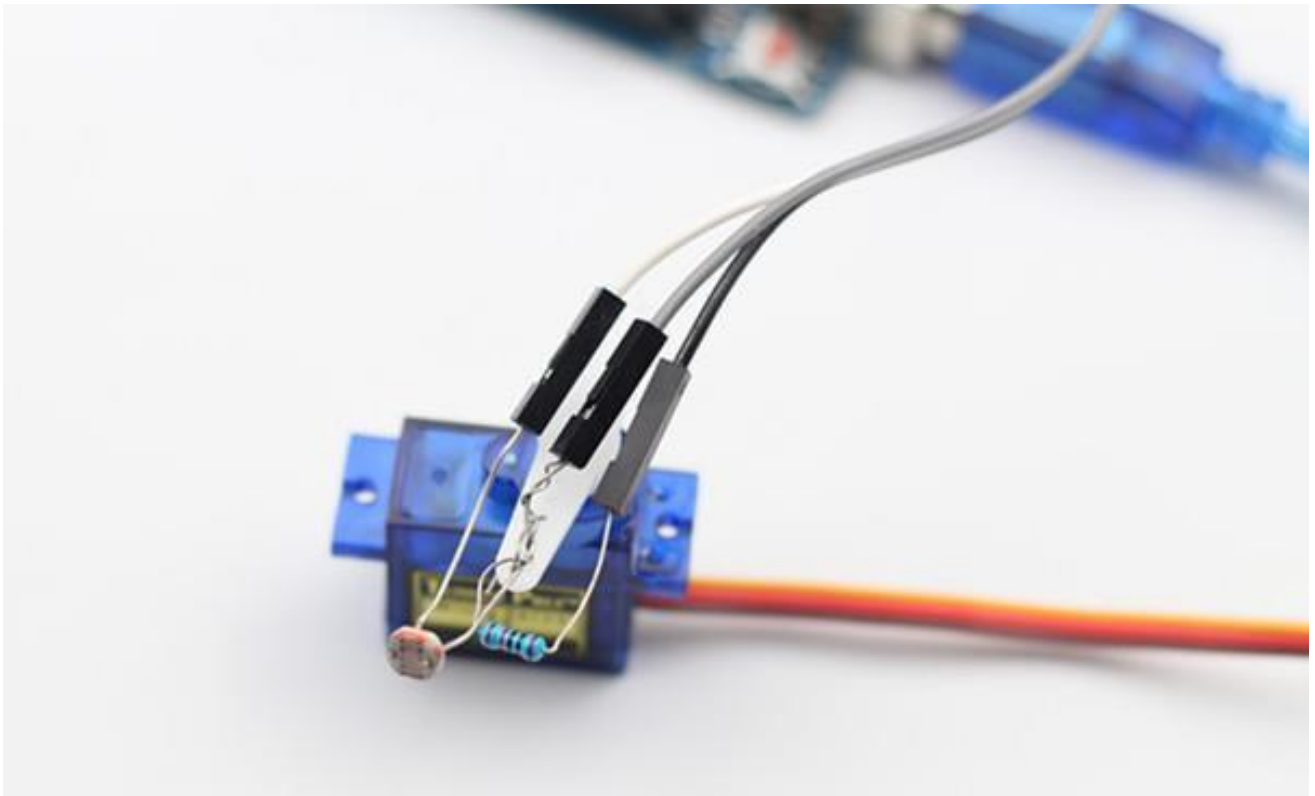
The servo motor and the photoresistor scan and look for light source in 180 degree and record the location of light source. After finishing scanning, the servo motor and the photoresistor stop at the direction of light source.

Experimental Procedures

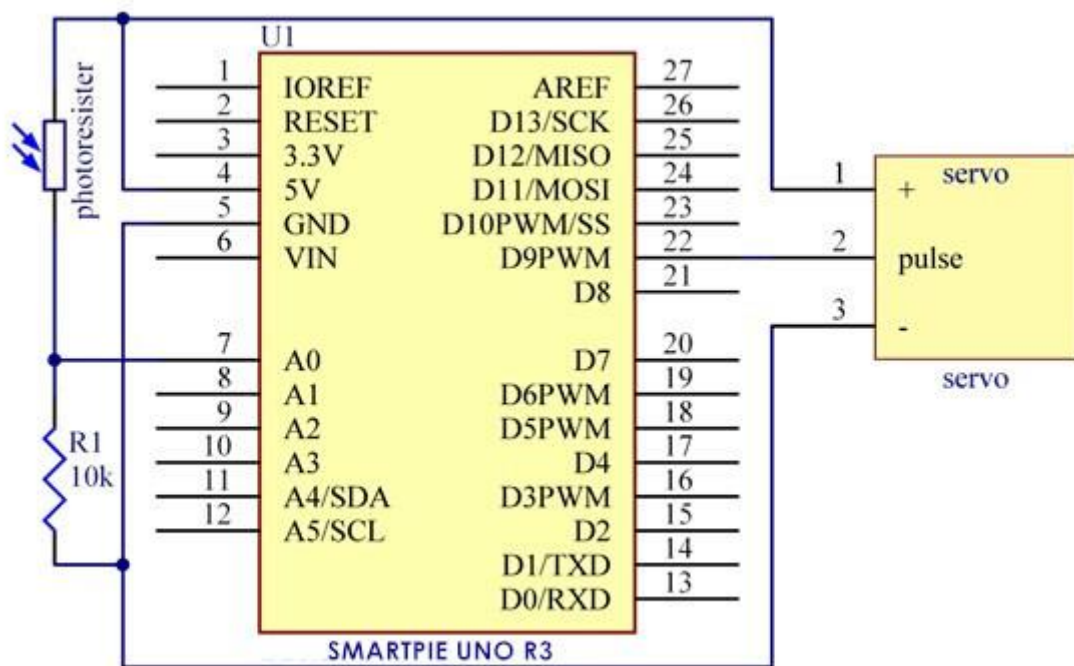
Step 1: Connect circuit as shown in the following diagram:



Note: you need to bind one end of the resistor and photoresistor to the wing of the servo, as shown below:



The corresponding schematic diagram is as follows:

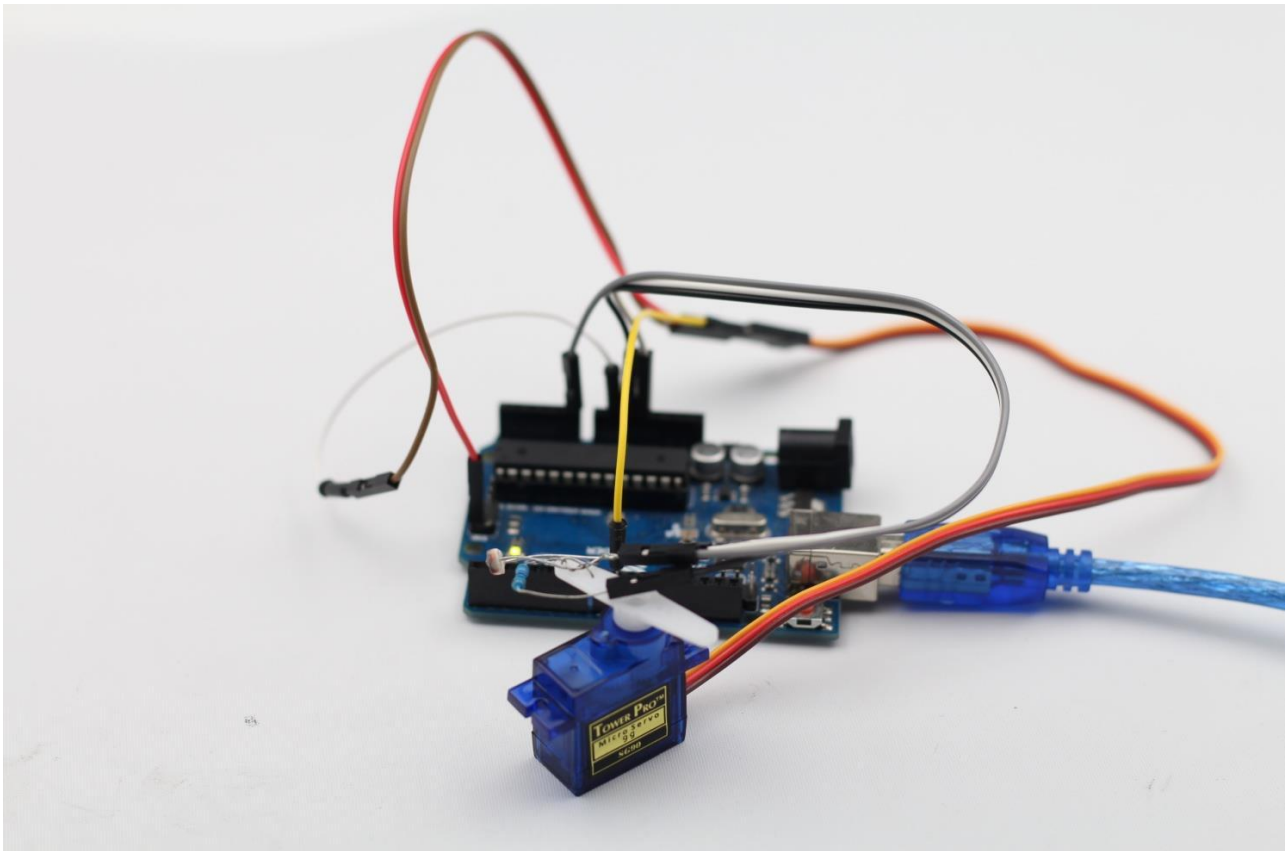


Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Now, if you use a flashlight to shine the photoresistor, you will see the servo motor and the photoresistor rotate, finally stop at the direction of light source.



Lesson 21 Packing

Introduction

In this lesson, we will use an ultrasonic sensor and some other components to simulate how to back a car.

Components

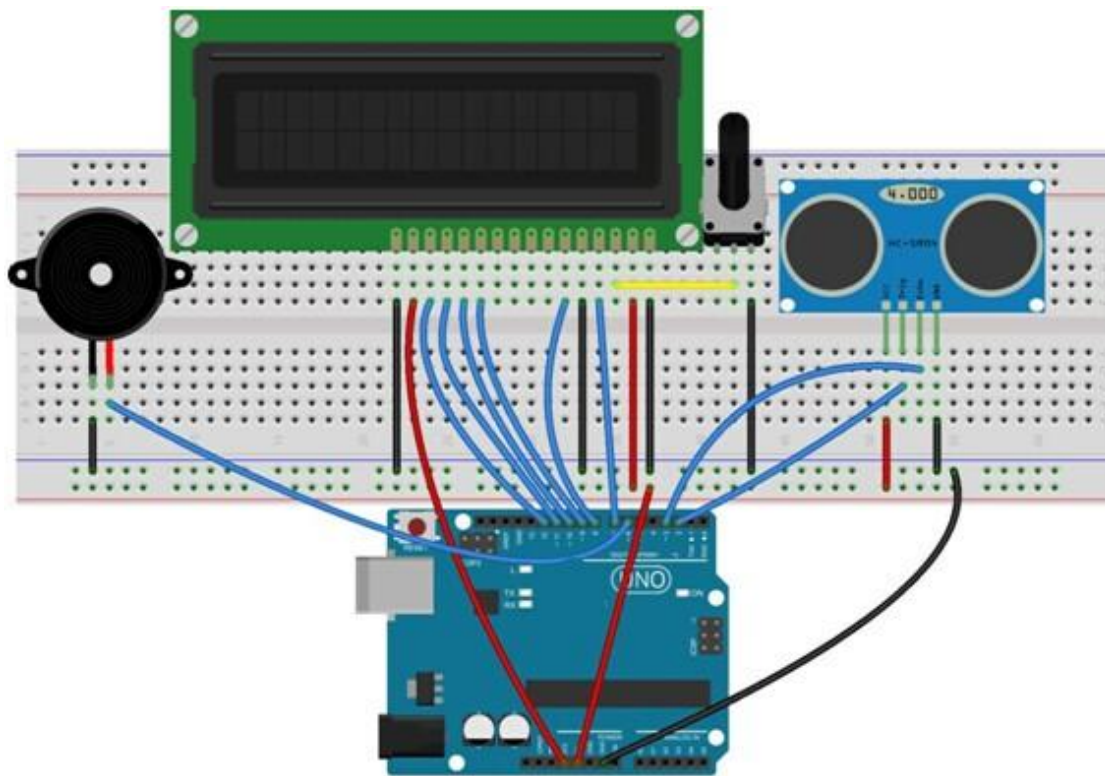
- 1 * Uno board
- 1 * USB data cable
- 1 * Ultrasonic sensor
- 1 * Buzzer
- 1 * LCD1602
- Several jumper wires
- 1 * Breadboard
- 1 * Potentiometer

Experimental Principle

When the distance between the ultrasonic and the obstacle is greater than 5cm and less than 15cm, the buzzer will make sounds in a low frequency, which means backing a car is in progress. When the distance is less than 5cm, the buzzer will alarm and make sounds in a high frequency.

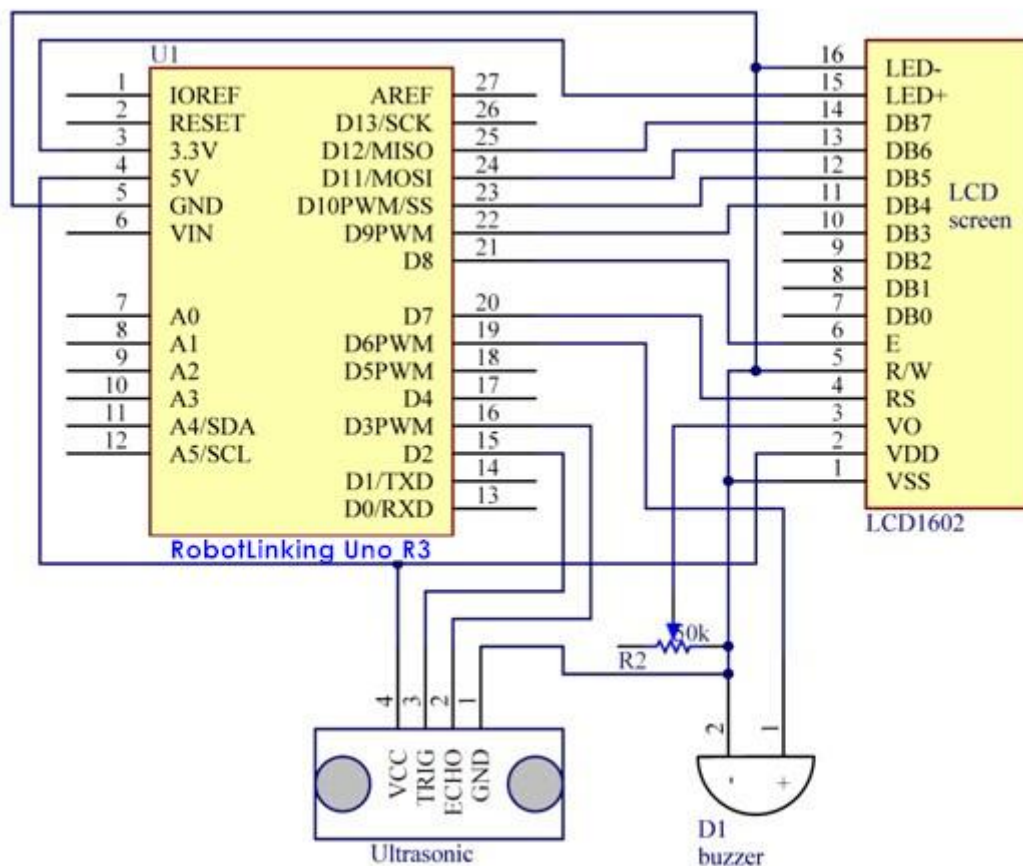
Experimental Procedures

Step 1: Connect circuit as shown in the following diagram:



fritzing

The corresponding schematic diagram is as follows:

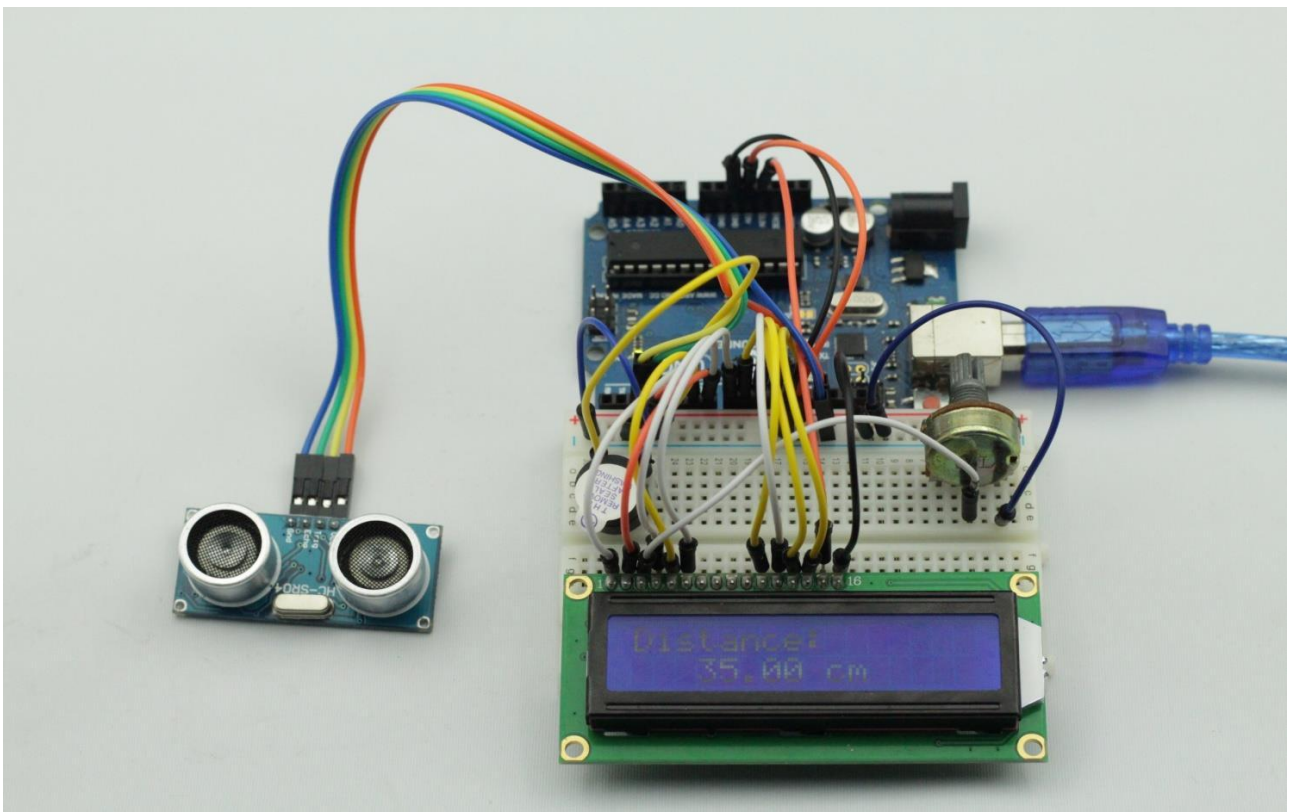


Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Now, if the distance between the ultrasonic and the obstacle is greater than 5cm and less than 15cm, the buzzer will make sounds in a low frequency; if the distance is less than 5cm, the buzzer will alarm and make sounds in a high frequency. At the same time, the current distance between the ultrasonic and the obstacle will display on LCD1602.



Lesson 22 MPU6050

Introduction

In this lesson we will learn how to use a MPU6050 module.

Components

- 1 * Uno board
- 1 * Breadboard
- 1 * MPU6050 module
- Jumper wires
- 1 * USB cable

Principle

According to the InvenSense MPU-6050 datasheet, this chip contains a 3-axis gyroscope and a 3-axis accelerometer. This makes it a "6 degrees of freedom inertial measurement unit" or 6DOF IMU, for short. Other features include a built in 16-bit analog to digital conversion on each channel and a proprietary Digital Motion Processor (DMP) unit.

The DMP combines the raw sensor data and performs some complex calculations onboard to minimize the errors in each sensor. Accelerometers and gyros have different inherent limitations, when used on their own. By combining the data from the two types of sensors and using some math wizardry (a process referred to as sensor fusion), you apparently can get a much more accurate and robust estimate of the heading. The DMP on the MPU6050 does exactly that and returns the result in "quaternions". These can then be converted to yaw-pitch-roll, or to Euler angles for us humans to read and understand. The DMP also has a built in auto-calibration function that definitely comes in handy, as we will see later.

The biggest advantage of the DMP is that it eliminates the need to perform complex and resource intensive calculations on the Arduino side. The main downside is that it seems that the manufacturer did not provide much information on the proprietary inner workings of the DMP. Nevertheless, smart and creative folks figured out how to use its main features, and were nice enough to share the results with the rest of us. You can still pull the raw, accelerometer and gyro data as well, disabling the DMP, if this works better for your application, or you want to apply your own filtering and sensor fusion algorithms.

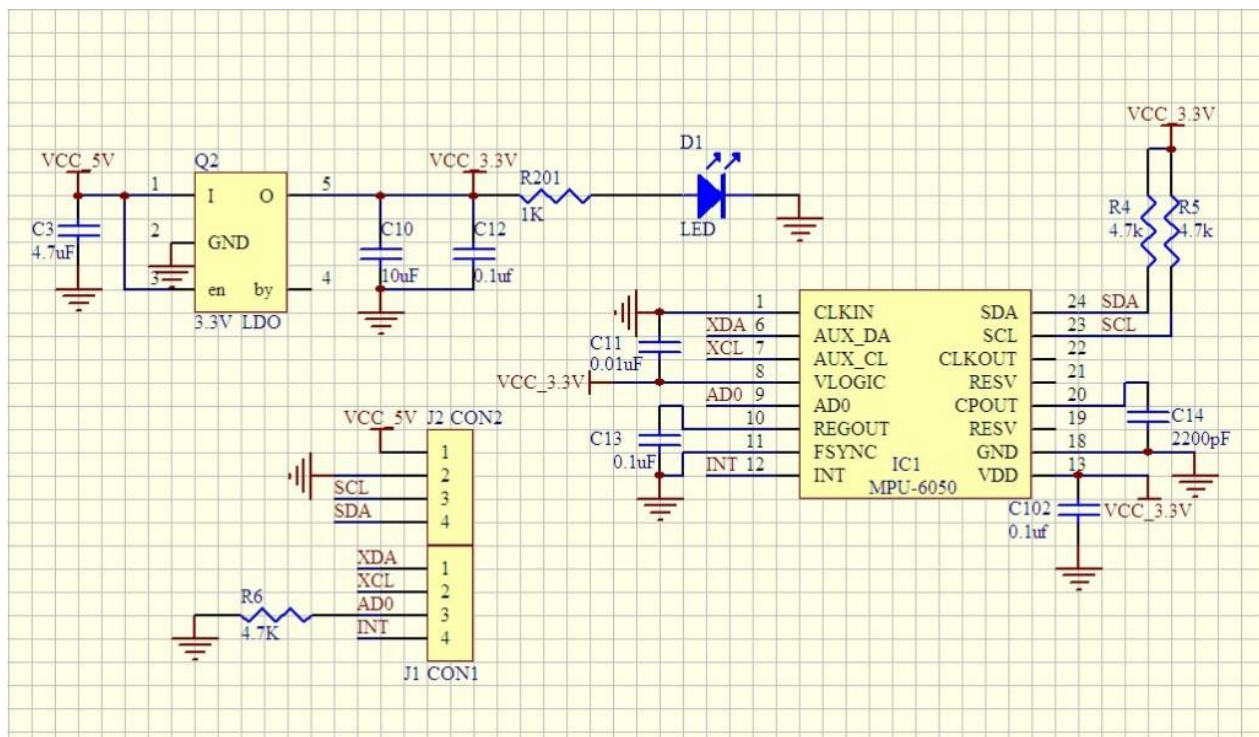
The MPU-6050 communicates with a microcontroller through an I2C interface. It even has a built in an additional I2C controller, that allows it to act as a master on a second I2C bus. The

intention is for the IMU to read data from say, an external magnetometer (hooked up via those XDA / XCL pins you see on the breakout board) and send it to the DMP for processing. I have not found much detail on how to make the DMP use external magnetometer data yet, but fortunately that is not needed for my self-balancing robot at this point.

Lastly, the MPU-6050 has a FIFO buffer, together with a built-in interrupt signal. It can be instructed to place the sensor data in the buffer and the interrupt pin will tell the Arduino, when data is ready to be read.

MPU-6050 / GY-521 break-out board schematics

Below is the schematic of the GY-521 break-out board for the MPU6050 chip.



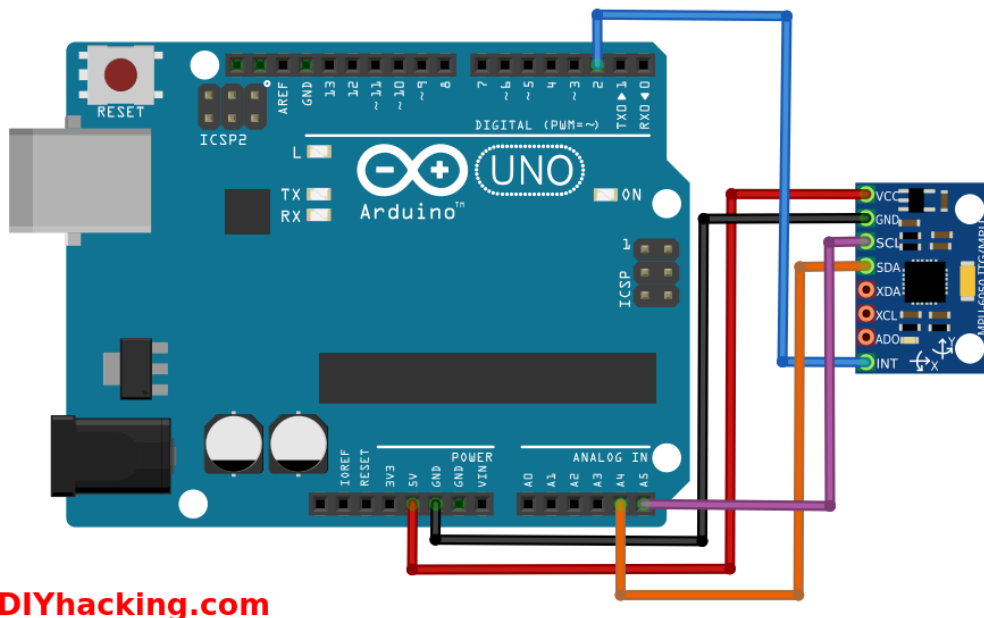
The InvenSense MPU6050 chip is a 3.3V IC, with a working voltage range of 2.375V-3.46V, according to its datasheet. As you can see from the schematics above, the GY-521 breakout board has a built in low drop-out voltage regulator, so it is safe to power the chip through the Arduino 5V rail. This is recommended, as due to the voltage drop-out of the regulator on the VCC line, using the Arduino 3.3V rail may not provide enough voltage. I tested powering the chip both with 3.3V and 5V from the Arduino successfully, but in my final set-up opted for the 5V input.

Based on what I have read online and what I saw on my tests, the 3.3V SDA / SCL lines of the IMU work fine connected directly to the corresponding Arduino 5V I2C pins. If you want to be absolutely safe, you could use a level shifter, voltage divider, or an inline 10k resistor to protect the MPU6050 I2C lines. I opted for simplicity over safety, in my set-up and (so far) all is well.

MPU6050 / GY-521	Arduino UNO Pin
VCC	5V (the GY-521 has a voltage regulator)
GND	GND
SDA	A4 (I2C SDA)
SCL	A5 (I2C SCL)
INT	D2 (interrupt #0)

Experimental Procedures

Step 1: Connect circuit as shown in the following diagram (please make sure pins are connected correctly or characters will not display properly):



Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the program

Step 4: Burn the program into Uno board

Step 5: Open "tool-monitor", to see the data.

Here is where credit and a big thanks is due to Jeff Rowberg for his I2Cdev library and sample code for interfacing with the InvenSense MPU6050 chip and partially reverse-engineering the DMP functions. Also, you might check out the "teapot demo" post from Debra at "Geek Mom Projects" that pointed me to the i2cdev library in the first place.

To get started you need to follow these simple steps:

Download the I2C Device Library (i2cdevlib) master zip file and extract the contents to a convenient location on your hard-drive.

You should see a folder called "i2cdevlib-master" that will contain an "Arduino" subfolder in it and a couple of other items, that we do not need.

Open the Arduino folder and locate the "I2Cdev" and "MPU6050" sub-folders. Those are the two Arduino libraries we will need to copy to your Arduino libraries folder (here is how to install an Arduino library, in case you have not done this before).

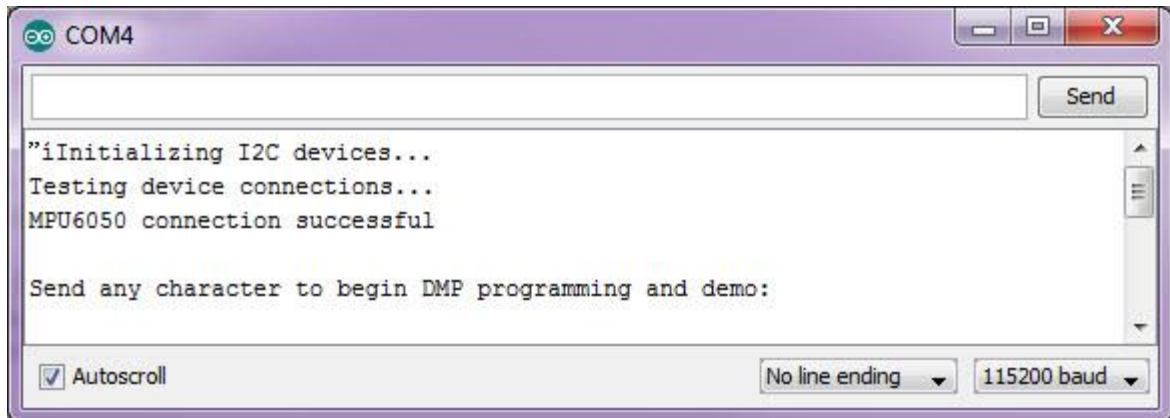
Under the Sketch->Examples menu, you should now see a menu item for the newly installed MPU6050 library. Load the MPU6050_DMP6 example and follow the instructions in the sketch comments to define what type of serial output you would like to see from the sensor. I suggest using yaw-pitch-roll or Euler while you are testing things out, as the data is a bit more intuitive.

Find the section in the code (around line 100) in the MPU6050_DMP6 example sketch and uncomment the `#define OUTPUT_READABLE_YAWPITCHROLL` line. Then make sure that all other output options are commented out. Here is the section for the yaw/pitch/roll output for reference:

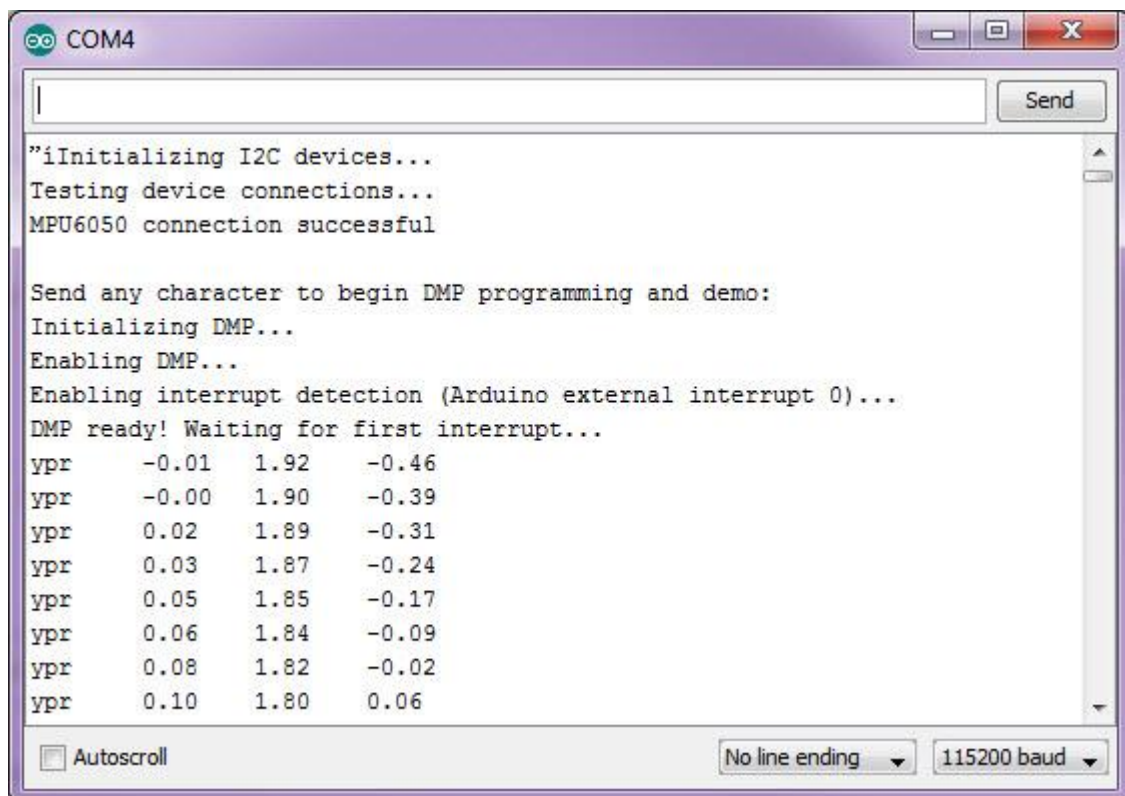
```
// uncomment "OUTPUT_READABLE_YAWPITCHROLL" if you want to see the yaw/  
// pitch/roll angles (in degrees) calculated from the quaternions coming  
// from the FIFO. Note this also requires gravity vector calculations.  
// Also note that yaw/pitch/roll angles suffer from gimbal lock (for  
// more info, see: http://en.wikipedia.org/wiki/Gimbal\_lock)
```

```
#define OUTPUT_READABLE_YAWPITCHROLL
```

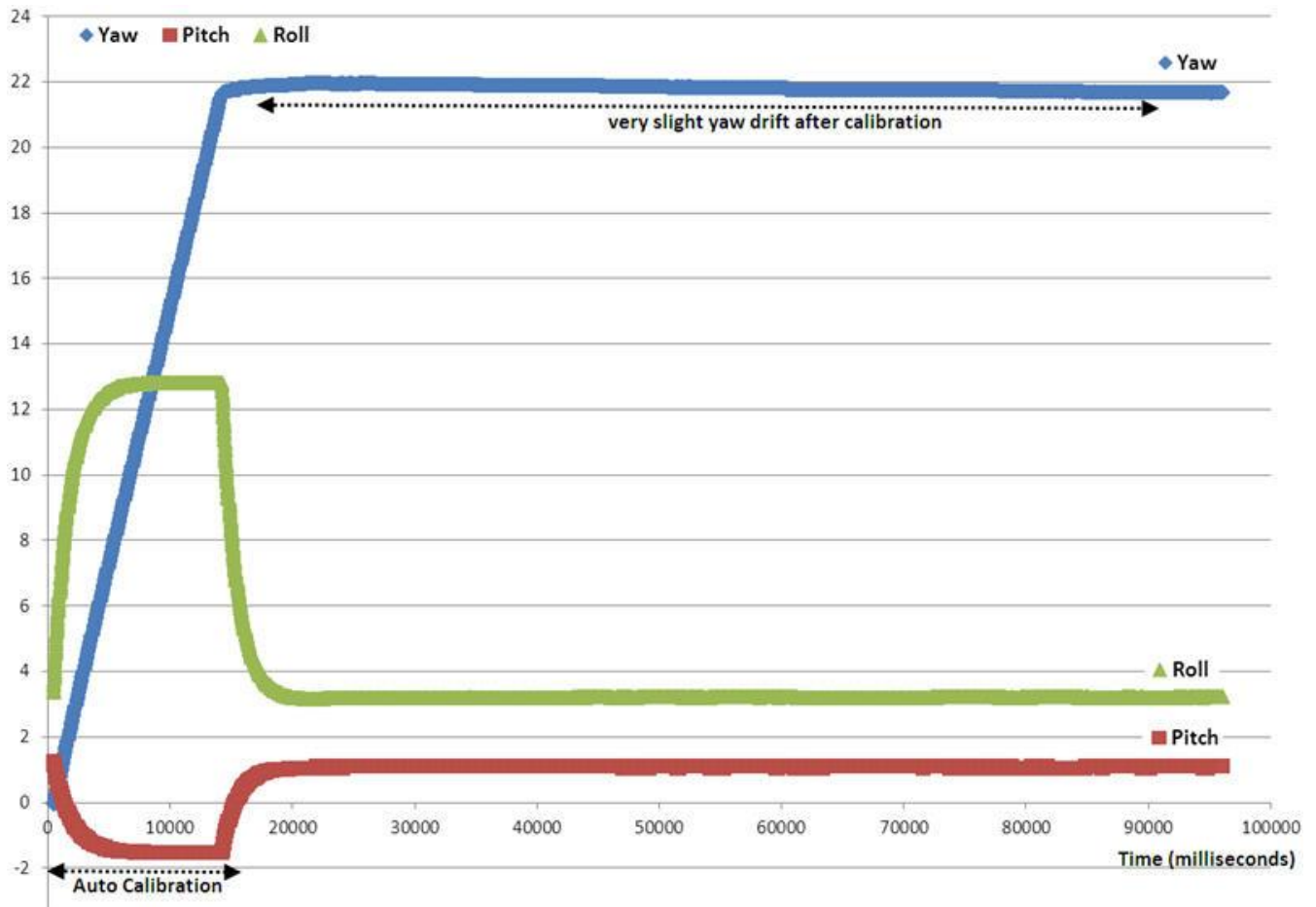
If all is well, when you upload the sketch to the Arduino and open your serial monitor (remember to set it at the correct baud rate), you will be prompted to send a random character via serial.



Once you type any character in the serial input field and hit "Send", a few status messages will appear and (if all is well) a steady stream of data from the IMU will follow!



I charted the DMP output of the sensor (the yaw-pitch-roll data), while the breakout board was set still on a “relatively” flat service. As you can see, initially some sort of calibration algorithm is run by the DMP. Once that completes, the sensor values are very stable.



With 6 axis IMUs you can expect yaw drift. After the initial calibration, the DMP compensates for most of that and only a very slight yaw drift remains (the light blue line).

Likely, with some additional fine-tuning and calibration, that can also be eliminated. It should also be possible to compensate for that in the Arduino code, as a last resort.

Lesson 23 STEPPER MOTOR AND ULN2003

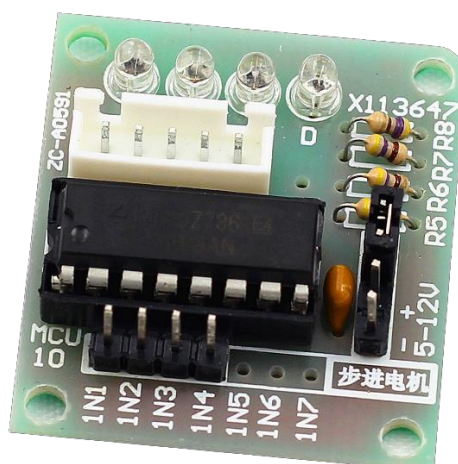
Introduction

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".



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 UNO board or other MCUs cannot directly drive stepper motors. A driver circuit is necessary, so we use a stepper motor driver board (as shown in the following picture) to drive the stepper motor. There are four LEDs on the top. The white booth in the middle is connected to the stepper motor. The bottom is four IOs used to connect with MCUs. When an IO is high, the corresponding LED will light up. The black jump hat on the right is power source input end. The driving method for stepper motor can be categorized as four-beat and eight-beat. In this experiment, we take four-beat for example, for it is simple. You can drive the motor as long as you input HIGH to the four ports A, B, C and D in turn.

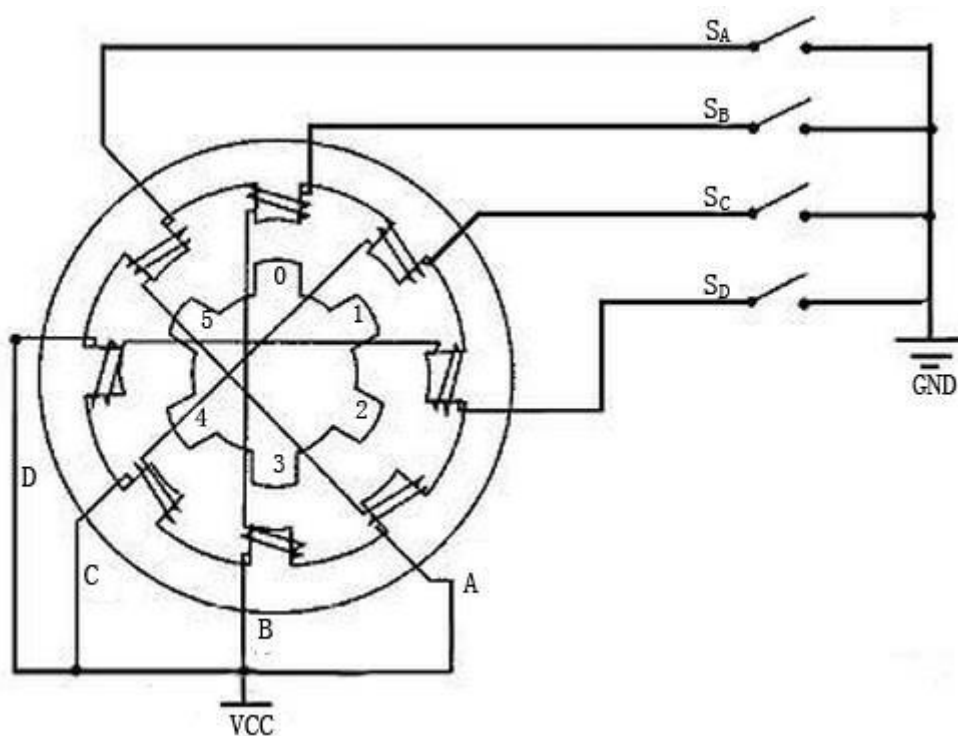


Components

- 1* Uno board
- 1*USB data cable
- 1*Potentiometer
- 1*Stepper Motor
- 1*ULN2003 Driver
- Several jumper wires
- 1*Breadboard

Principle

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 stepper motor by an appropriate timing sequence, you can make the motor rotate step by step. The schematic diagram of a four-phase reactive stepper motor is as shown below:



As shown 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 S_A , S_B , S_C , and S_D . Therefore, the four phases are in parallel in the circuit, and the two magnetic poles in one phase are in series.

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

At the beginning, switch S_B is power on, switch S_A , S_C , and S_D 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 S_C is power on, switch S_B , S_A , and S_D is power off, the rotor rotates under magnetic field of C-phase winding and that between tooth 1 and 1. 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.

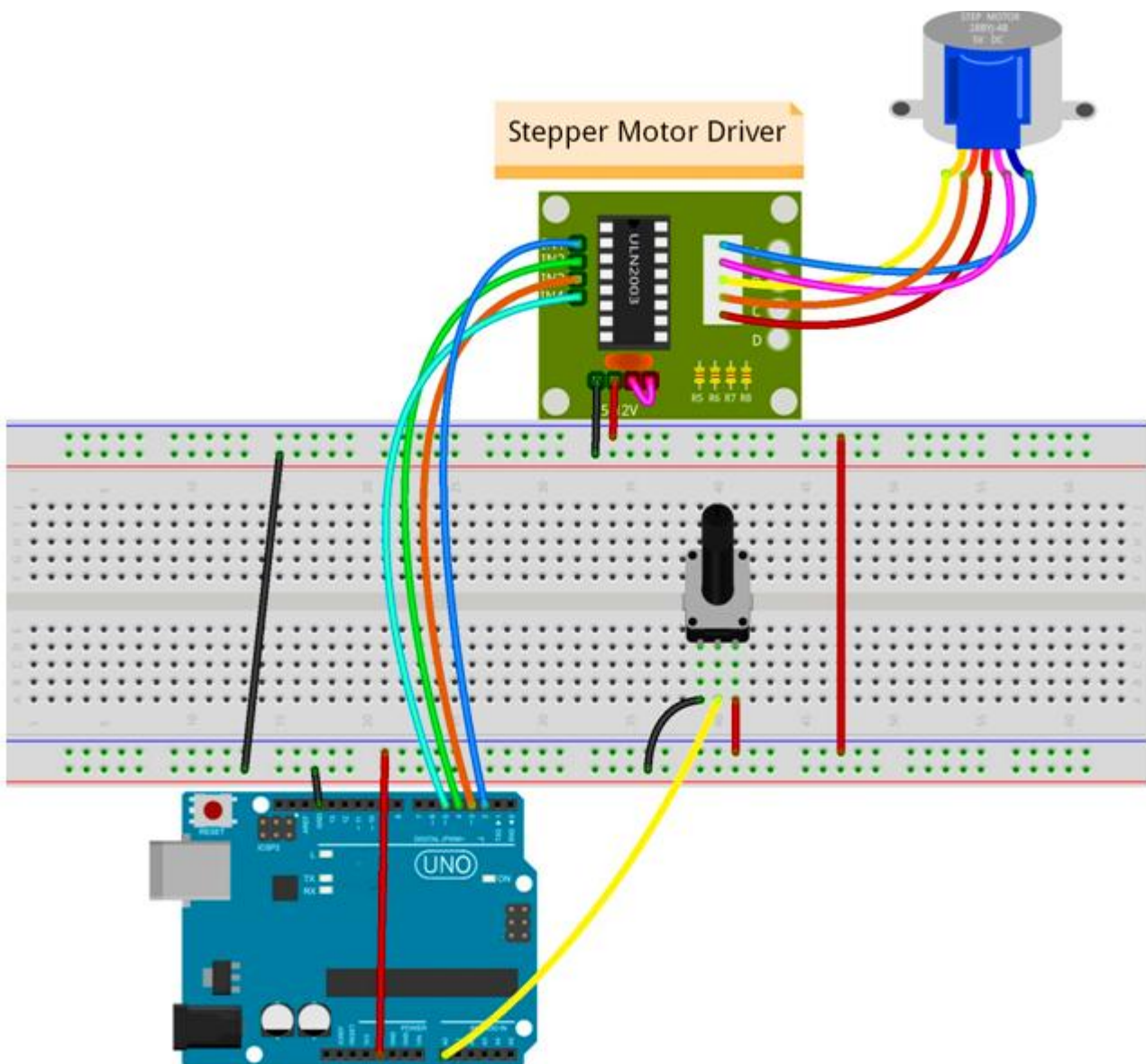
To apply the motor in the circuit, a driver board needs to be used. Stepper Motor Driver-ULN2003 is a 7-channel inverter circuit. That is, when the input end is at high level, the output end of ULN2003 is at low level, and vice versa. If we supply high level to IN1, and low level to IN2, IN3 and IN4, then the output end OUT1 is at low level, and all the other output ends are at high level. So D1 lights up, switch S_A 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.

Experimental Procedures

Step 1: Connect the circuit

ULN2003	Uno
IN1	2
IN2	4

IN3	3
IN4	5
GND	GND
VCC	5v



Step 2: Program (please refer to the example code in the ZIP folder or official website)

Step 3: Compile the code

Step 4: Upload the sketch to the Uno board

Now, adjust the potentiometer, the stepper motor will spin corresponding degrees.

