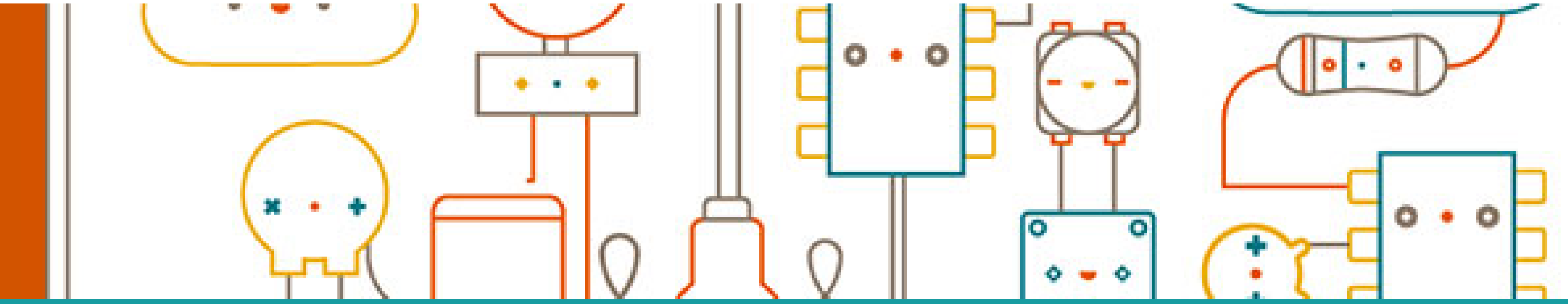
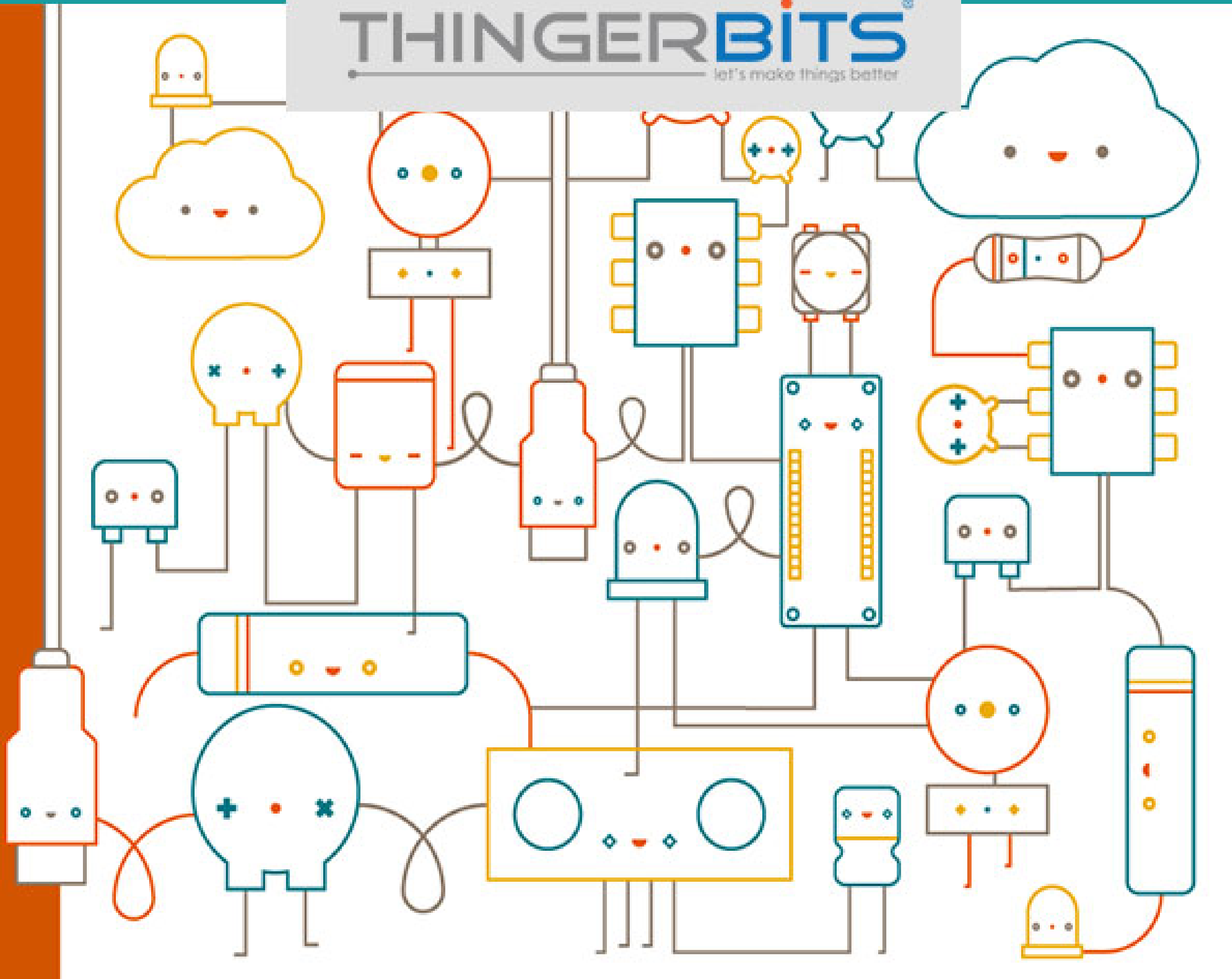


ARDUINO FOR BEGINNERS

THINGERBITS®

let's make things better



Preface

About Thingerbits

Thingerbits.lk is a technology company focused on 3D Printer, Raspberry Pi and Arduino open source community development. Committed to the promotion of open source culture, we strive to bring the fun of electronics making to people all around the world and enable everyone to be a maker. Our products include learning kits, development boards, robots, sensor modules, development tools and printer machine. In addition to high quality products, Thingerbits.lk also offers video tutorials to help your own project. If you have interest in open source or making something cool, welcome to join us! Visit www.thingerbits.lk for more!

In this book, we will show you circuits with both realistic illustrations and schematic diagrams. You can go to our official website www.thingerbits.com to download related code.

Arduino is committed to helping customer quickly realize the creative idea and product prototypes, making it easy to get started for enthusiasts of programing and electronics and launching innovative open source products. Our services include:

- Electronic components and modules
- Learning kits for Arduino
- Learning kits for Raspberry Pi
- Learning kits for Technology
- Robot kits
- Auxiliary tools for creations

Our code and circuit are open source. You can obtain the details and the latest information through visiting the following web sites:

- https://www.thingerbits.com
- https://github.com/thingerbits

Your comments and suggestions are warmly welcomed, If you have any questions, please send them to the following email address : support@thingerbits.lk

You can also leave a message and share your projects on our forum.

Like Us : https://www.facebook.com/thingerbits/
Join with Us : https://www.facebook.com/groups/thingerbits/

Contents

Introduction

In 2005, in Ivrea, Italy, a project was initiated to make a device for controlling student-built interactive design projects that was less expensive than other prototyping systems available at the time. One of the cofounders, Massimo Banzi, named this piece of hardware Arduino in honor of Bar di Re Arduino (In 1002, King Arduin became the ruler of the Italy. Today, the Bar di Re Arduino, a pub on a cobblestoned street in town, honors his memory), and began producing boards in a small factory located in the same region as the computer company Olivetti.
The Arduino project is a fork of the open source Wiring platform and is programmed using a Wiring-based language (syntax and libraries), similar to C++ with some slight simplifications and modifications, and a Processing-based integrated development environment (IDE).

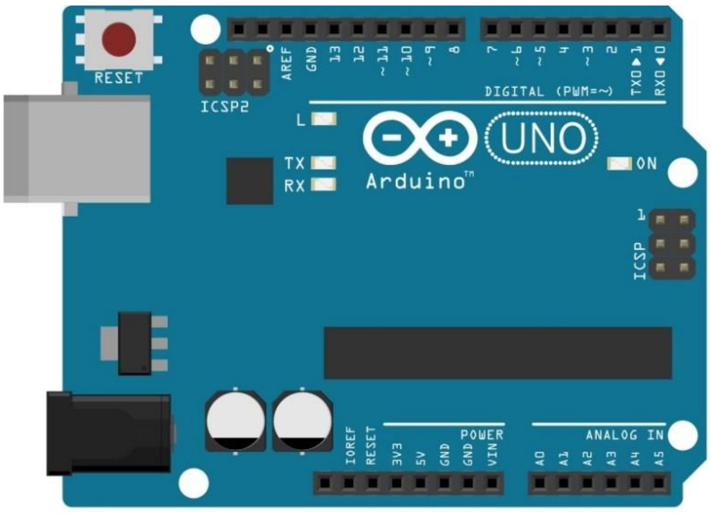
What Does it Do?

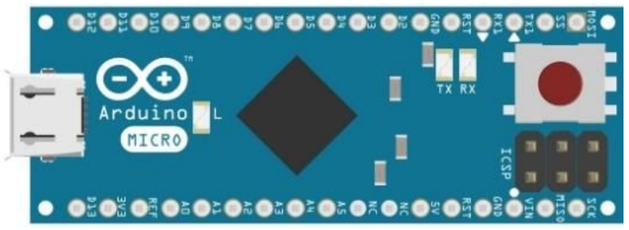
The Arduino hardware and software was designed for artists, designers, hobbyists, hackers, newbies, and anyone interested in creating interactive objects or environments. Arduino can interact with buttons, LEDs, motors, speakers, GPS units, cameras, the internet, and even your smart-phone or your TV! This flexibility combined with the fact that the Arduino software is free, the hardware boards are pretty cheap, and both the software and hardware are easy to learn has led to a large community of users who have contributed code and released instructions for a huge variety of Arduino-based projects.

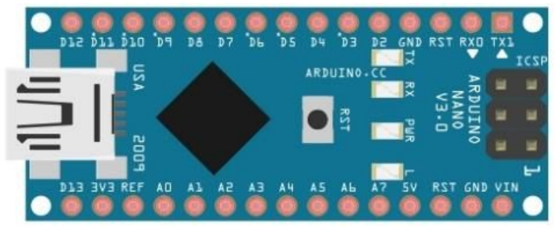
Arduino Board

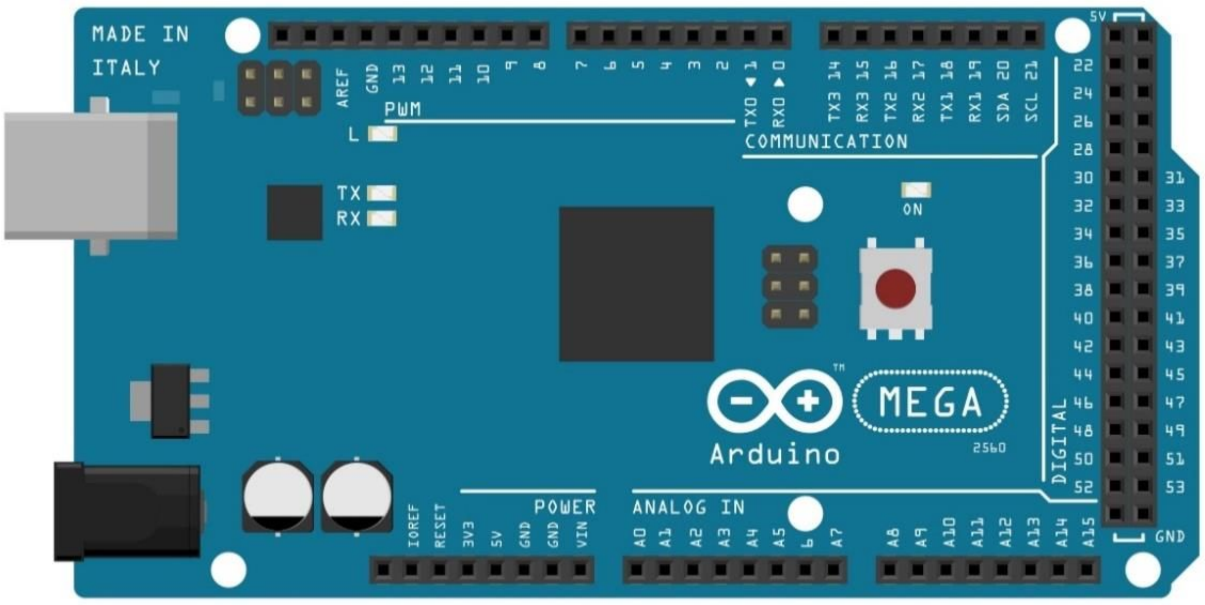
Arduino Board is a circuit board, which integrates micro controller, input, output interface and etc. Arduino Board can use the sensor to sense the environment and receive user's operation to control LED, motor rotation, etc. We just need to assembly circuit and write the code.

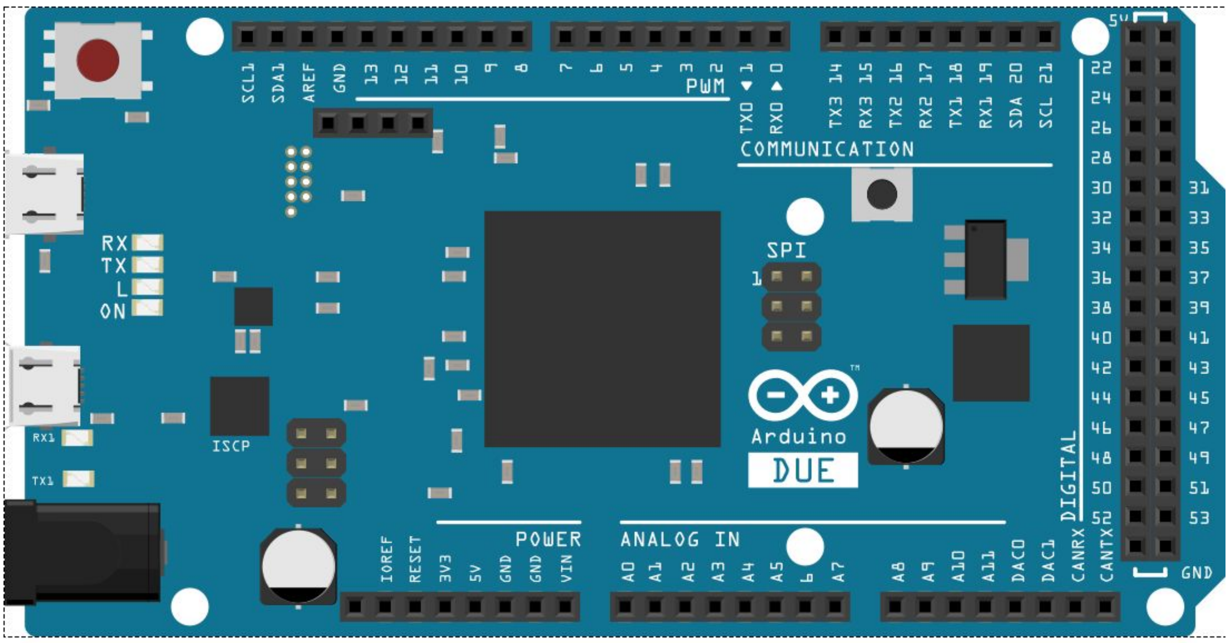
Currently, Arduino Board has several models, and the code between boards of different types is universal (some boards may not be completely compatible because of the differences in hardware). Popular boards include:


ARDUINO UNO

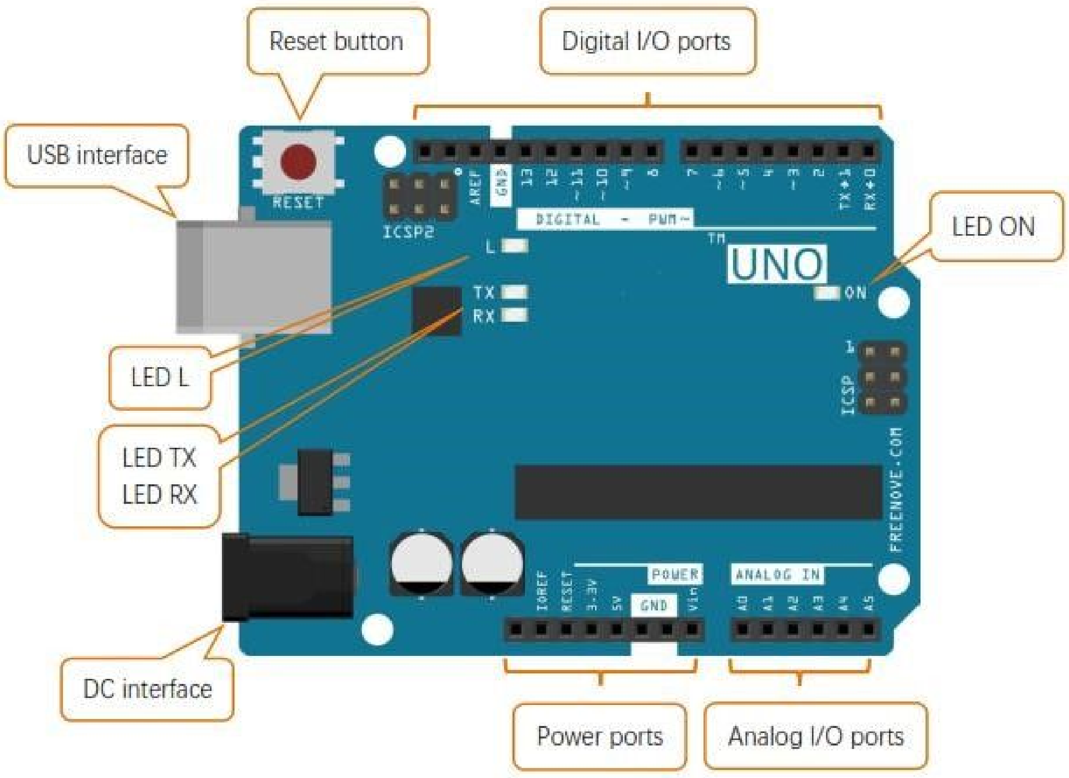

ARDUINO MICRO


ARDUINO NANO


ARDUINO MEGA


ARDUINO DUE

The board used in this Course is Arduino UNO compatible Board , and it is fully compatible to Arduino UNO. Diagram of Arduino UNO board is shown below:



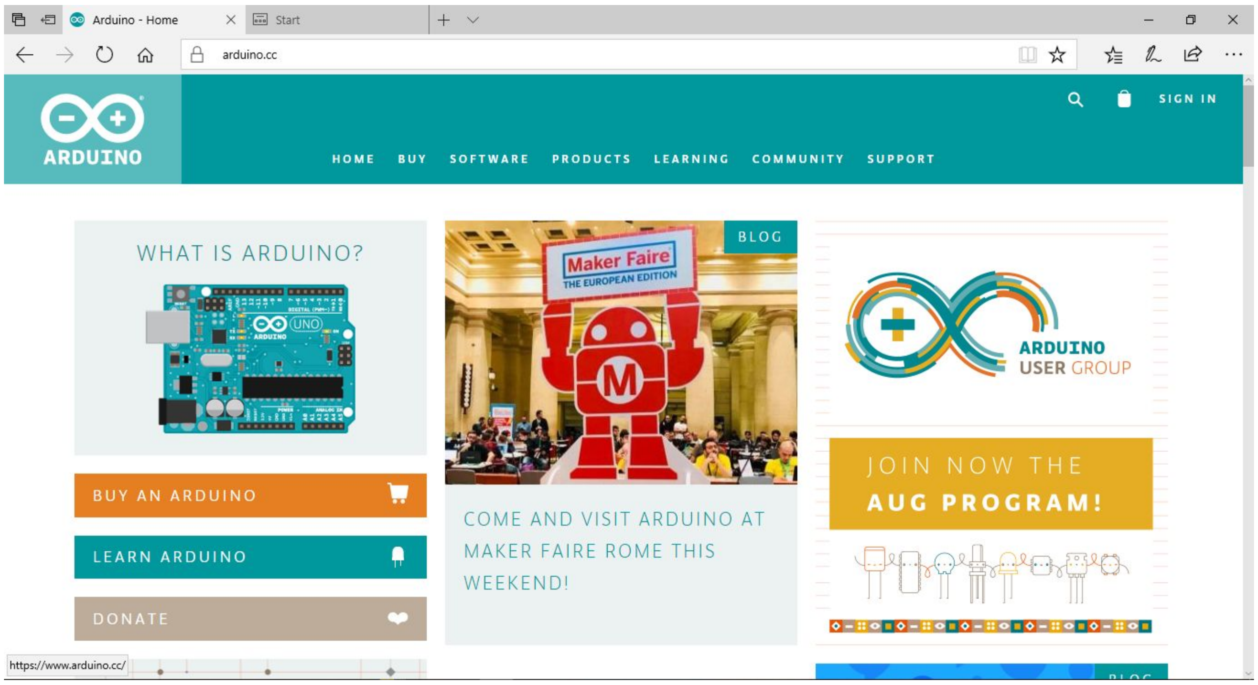
- Digital I/O ports is used to connect to other components or modules, to receive an input signal, or to send a control signal. Usually, we name it by adding a "D" in front of the number, such as D13.
- USB interface is used to provide power, upload code or communicate with PC.
- LED L is connected to digital I/O port 13 (D13).

- ○ LED TX, RX is used to indicate the state of the serial communication.
- ○ DC interface is connected DC power to provide power for the board.
- ○ Power ports can provide power for electronic components and modules.
- ○ Analog I/O ports can be used to measure analog signals.
- ○ LED ON is used to indicate the power state.

Arduino UNO is the most suitable board to complete the experiments of this book. You can also choose to use Arduino UNO, Arduino MICRO, Arduino NANO, Arduino MEGA (or other boards compatible to them).

Arduino Software

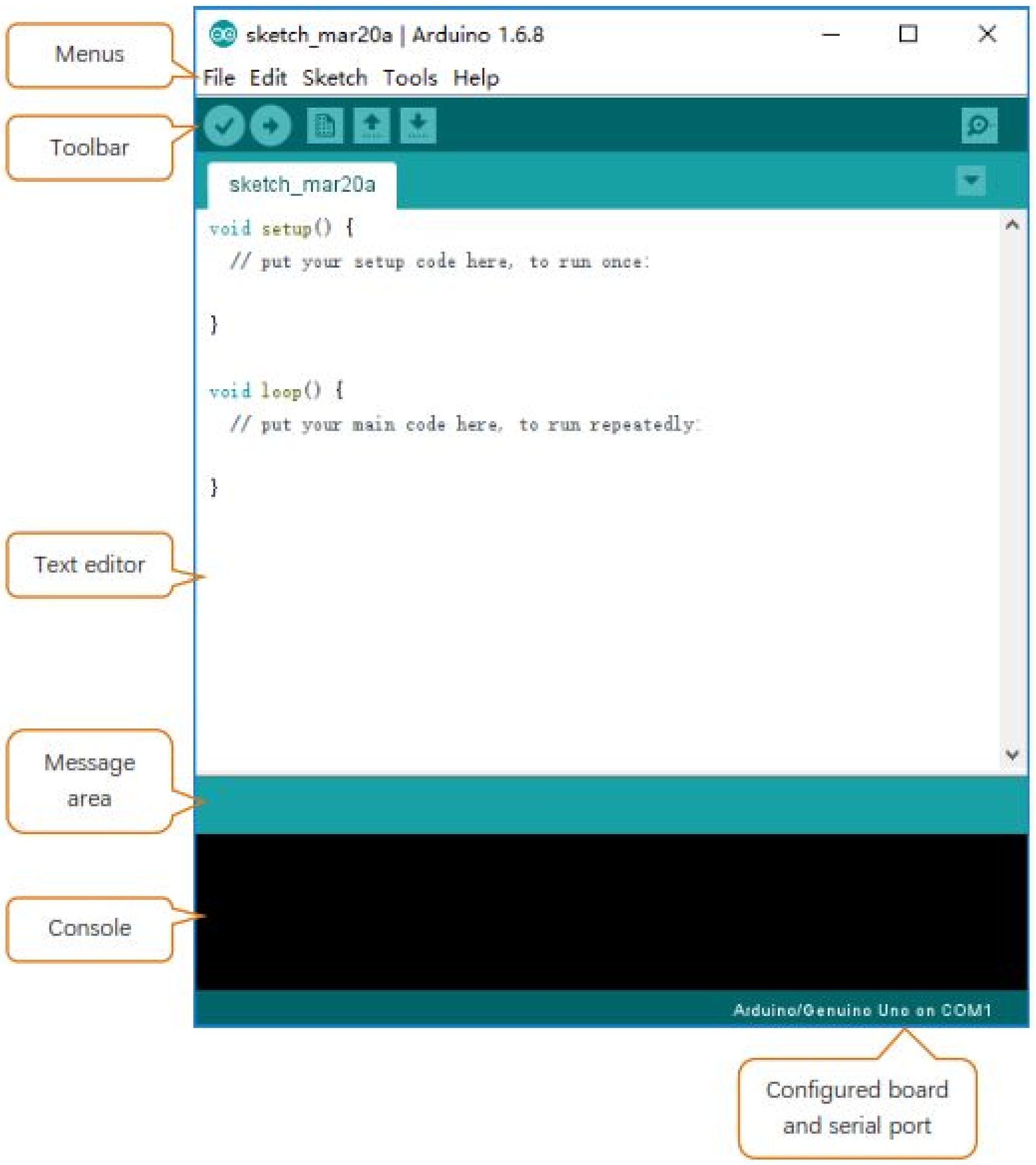
Arduino Software (IDE) is used to write and upload the code for Arduino Board.
First, install Arduino Software (IDE): visit https://www.arduino.cc, click "Download" to enter the download page.



You can see two method of use Arduino IDE

1. Online Web-base Arduino IDE
2. Offline Arduino IDE

The interface of Arduino Software is as follows:


sketch_mar20a | Arduino 1.6.8
File Edit Sketch Tools Help
sketch_mar20a
Arduino/Genuino Uno on COM1
Menus
Toolbar
Text editor
Message area
Console
Configured board and serial port

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

✓	**Verify** Checks your code for errors compiling it.
➜	**Upload** Compiles your code and uploads it to the configured board.
🗎	**New** Creates a new sketch.

⬆	**Open** Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.
⬇	**Save** Saves your sketch.
🔍	**Serial Monitor** Opens the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

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 IDE (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.

Installing Arduino IDE (MacOS)

This part will show you how to install and test the Arduino software on a Mac computer running OSX.

- Go to the Arduino download page and download the latest version of the Arduino software for Mac.
- When the download is finished, un-zip it and open up the Arduino folder to confirm that yes, there are indeed some files and sub-folders inside. The file structure is important so don't be moving any files around unless you really know what you're doing.

- Power up your Arduino by connecting your Arduino board to your computer with a USB cable (or FTDI connector if you're using an Arduino pro). You should see the an LED labed 'ON' light up. (this diagram shows the placement of the power LED on the UNO).
- Move the Arduino application into your Applications folder.

FTDI Drivers

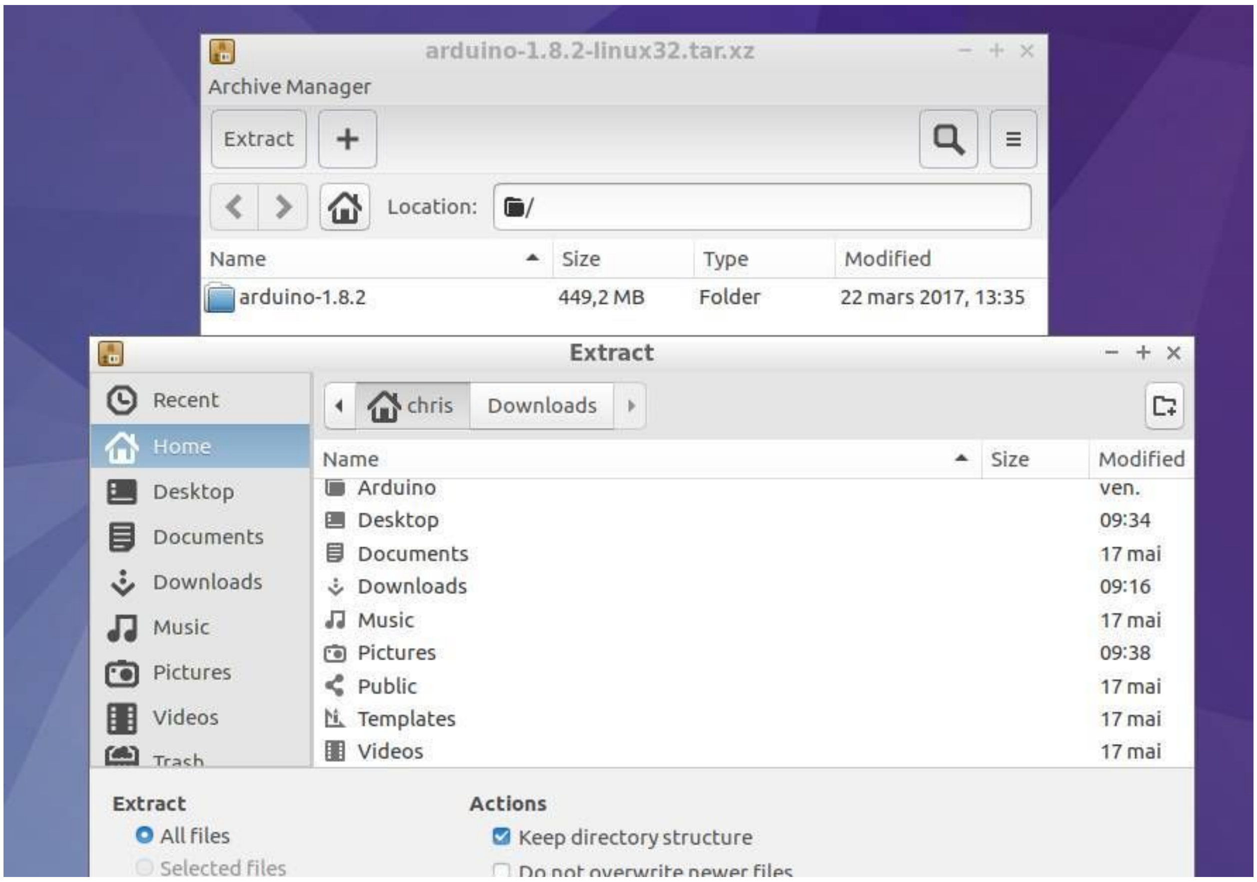
If you have an UNO, Mega2560, or Redboard, you shouldn't need this step, so skip it!

- For other boards, you will need to install drivers for the FTDI chip on your Arduino.
- Go to the FTDI website and download the latest version of the drivers.
- Once you're done downloading, double click the package and follow the instructions from the installer.
- Restart your computer after installing the drivers.

Installing Arduino IDE (Linux)

To install the IDE Arduino on Ubuntu, simply go to the official site on this page and download the archive corresponding to the version of your system.

Ubuntu has an archive decompression utility (Archive Manager). At the end of the download, open the file manager and double click on the tar archive. The Archive Manager opens. Click the Extract button to unpack the archive. Unzip the folder to the destination directory directly. For example, in your user folder.



Name	Size	Type	Modified
arduino-1.8.2	449,2 MB	Folder	22 mars 2017, 13:35

Name	Size	Modified
Arduino		ven.
Desktop		09:34
Documents		17 mai
Downloads		09:16
Music		17 mai
Pictures		09:38
Public		17 mai
Templates		17 mai
Videos		17 mai

Extract
- [✓ All files]
- Selected files

Actions
- [✓ Keep directory structure]
- Do not overwrite newer files

The installation script will simply create symbolic links and add an icon on the desktop as well as in the programming menu.

Open a Terminal and place it in the IDE installation folder, for example

```
cd /home/user/arduino-1.8.2
```

Then run the installation script

```
./install.sh
```

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.

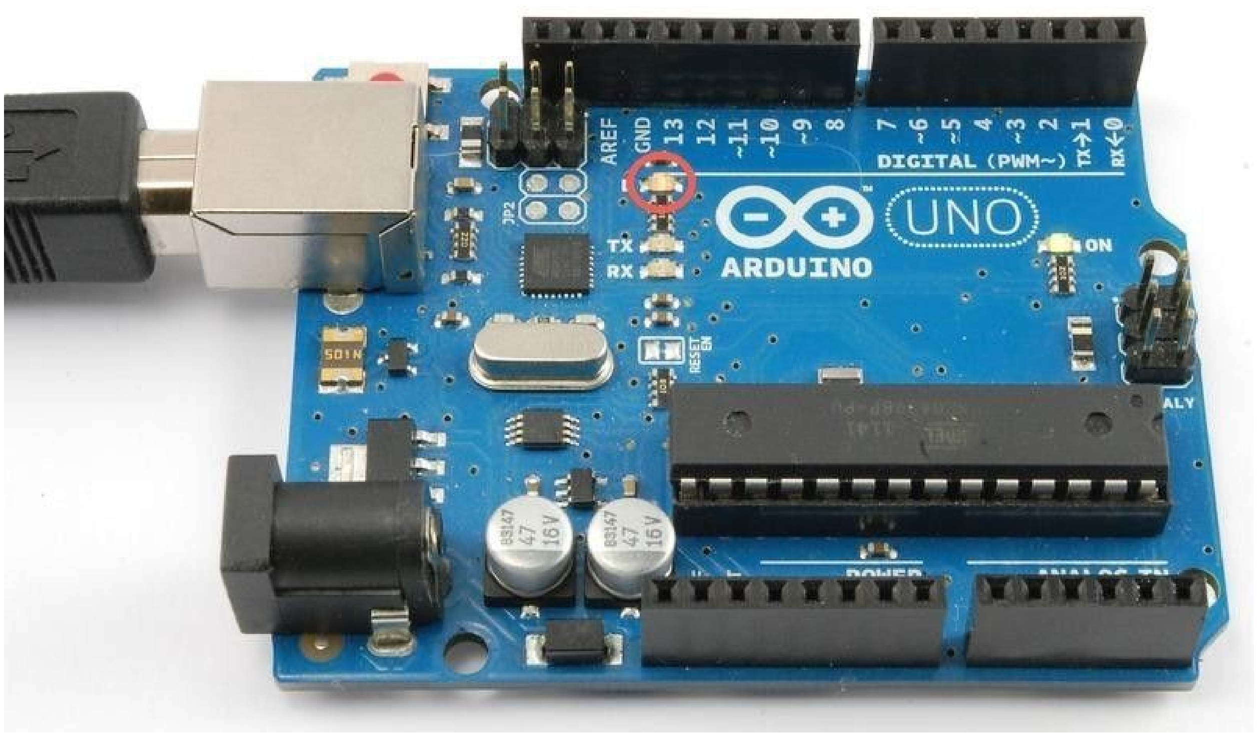
Components

- 1 * Arduino 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.



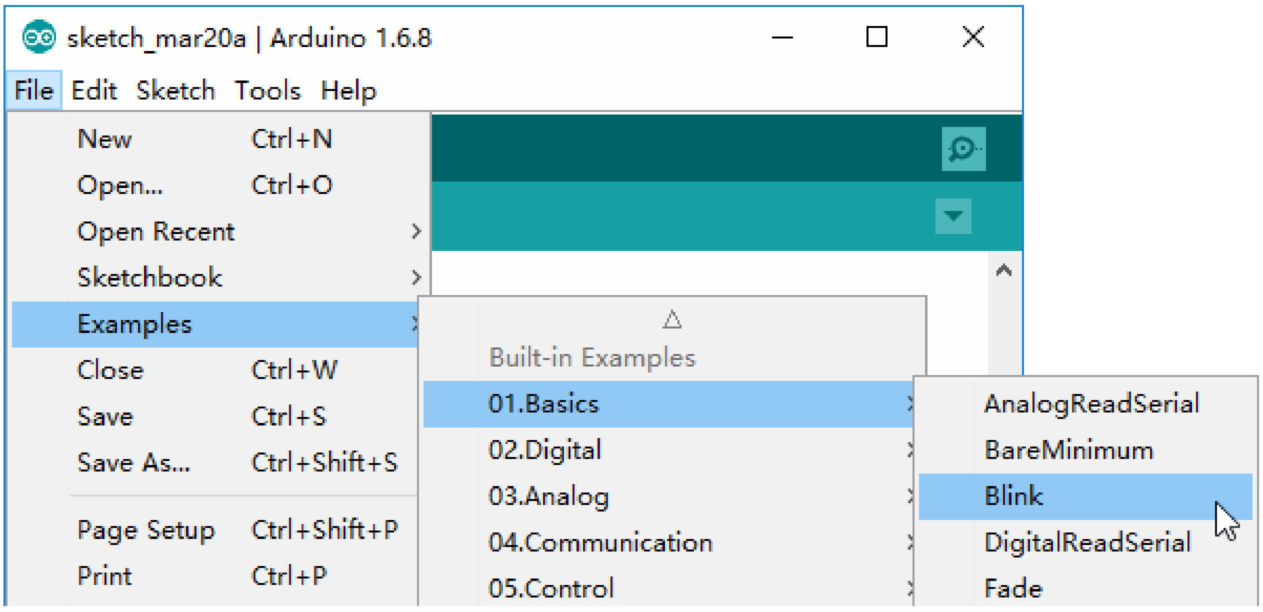
You may find that your Arduino board's 'L' LED already blinks when you connect it to a USB plug. This is because Arduino boards are generally shipped with the 'Blink' sketch pre- installed.

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

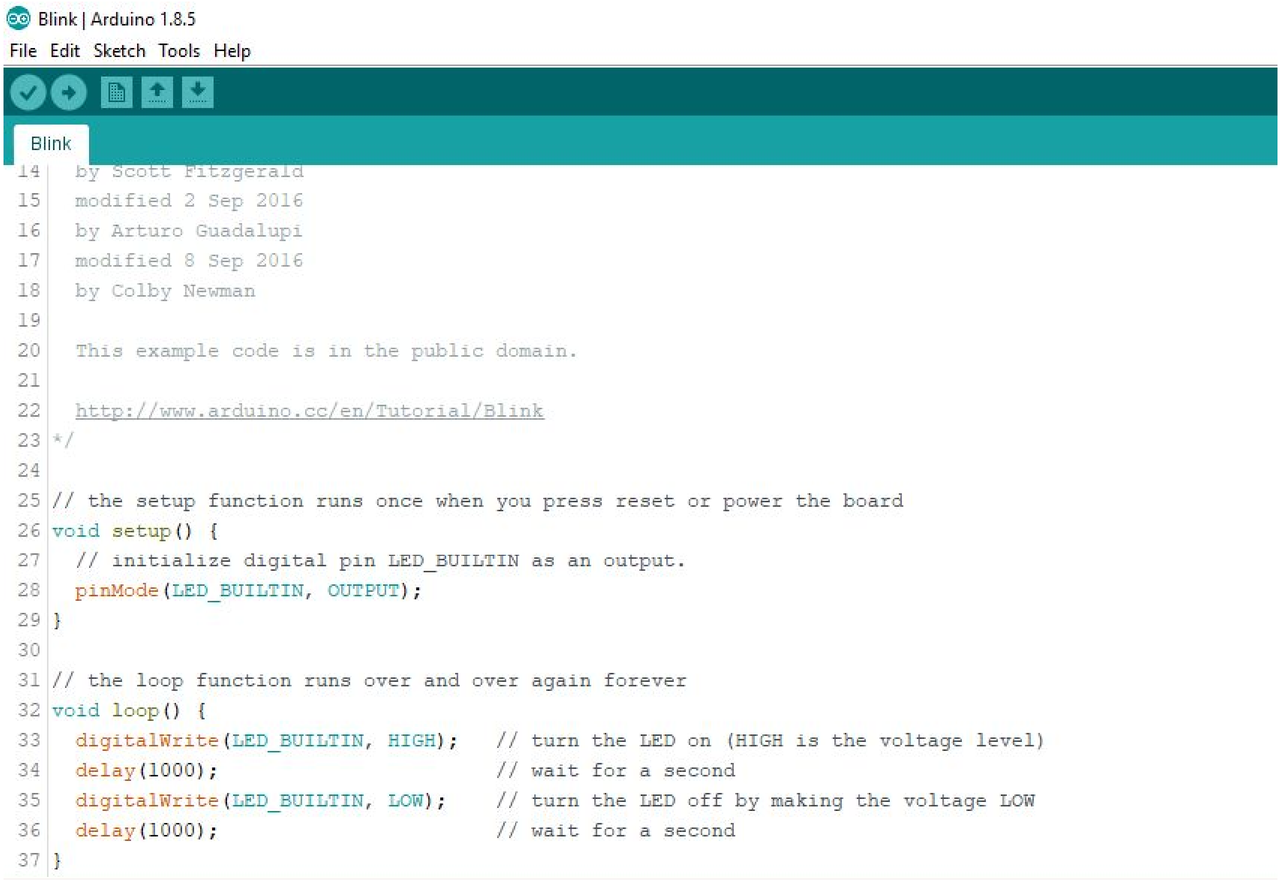
In lesson 0, you setup your Arduino IDE and made sure that you could find the right serial port for it to connect to your Arduino board. The time has now come to put that connection to the test and program your Arduino board.

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. (http://bit.ly/tb1dayl1)

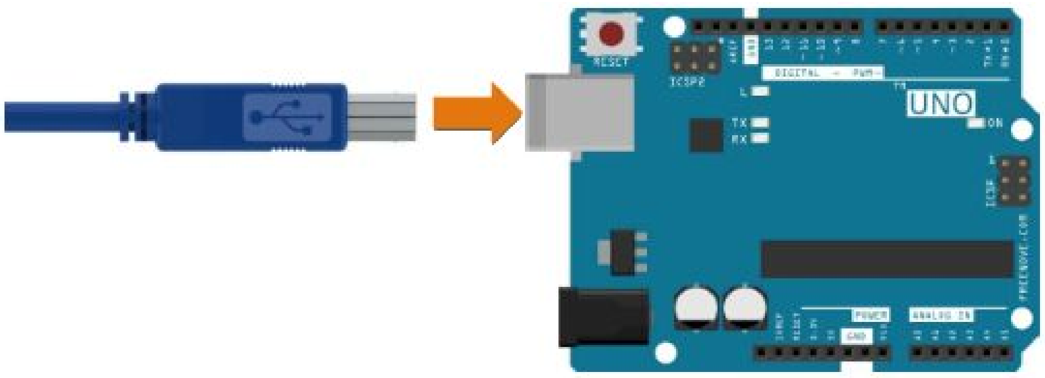


```
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

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

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

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.

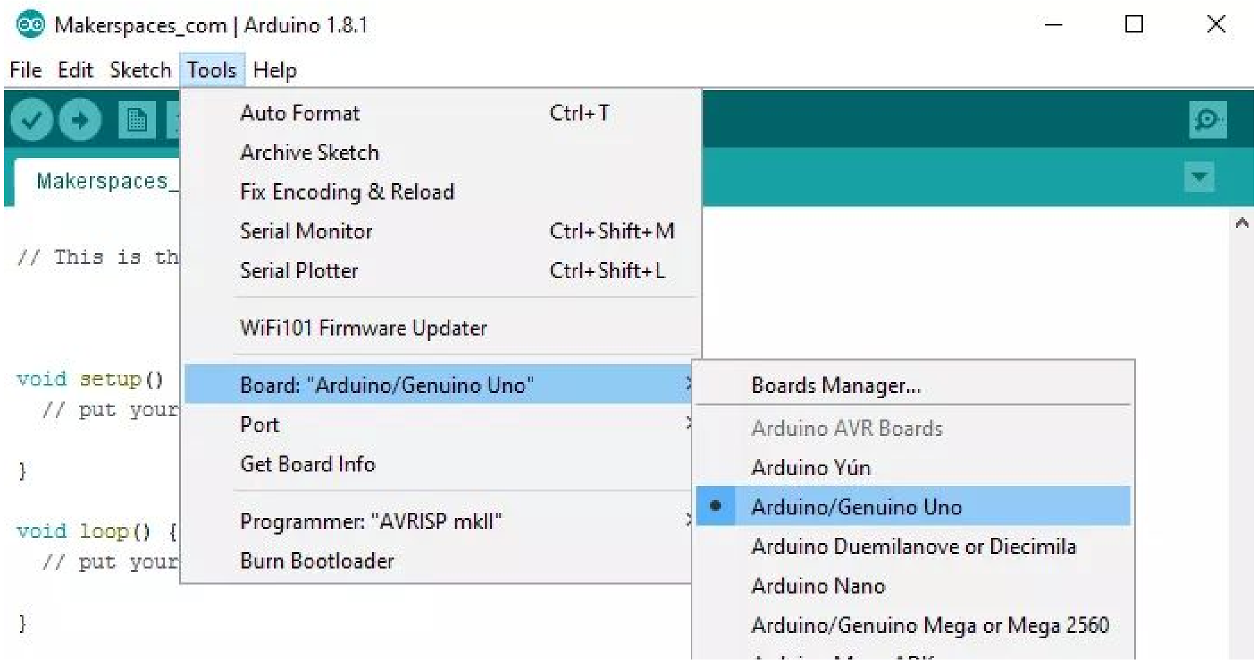
We are going to change this sketch, so, the first thing you need to do is save your own copy that you can change however you like.

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



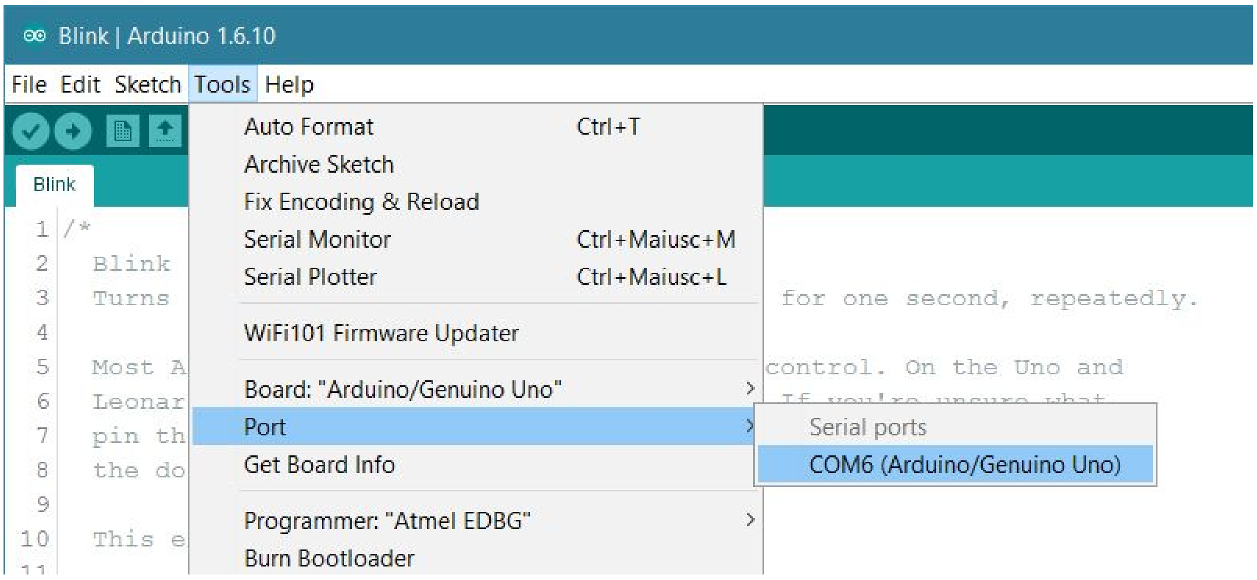
Attach your Arduino board to your computer with the USB cable and check that the 'Board Type' and 'Serial Port' are set correctly.

Select board "Arduino/Genuino Uno".

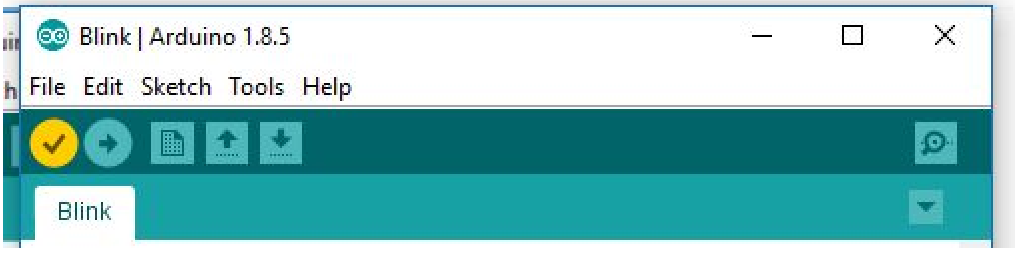

Makerspaces_com | Arduino 1.8.1
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino/Genuino Mega or Mega 2560

```
// This is th

void setup()
  // put your

}

void loop() {
  // put your

}
```

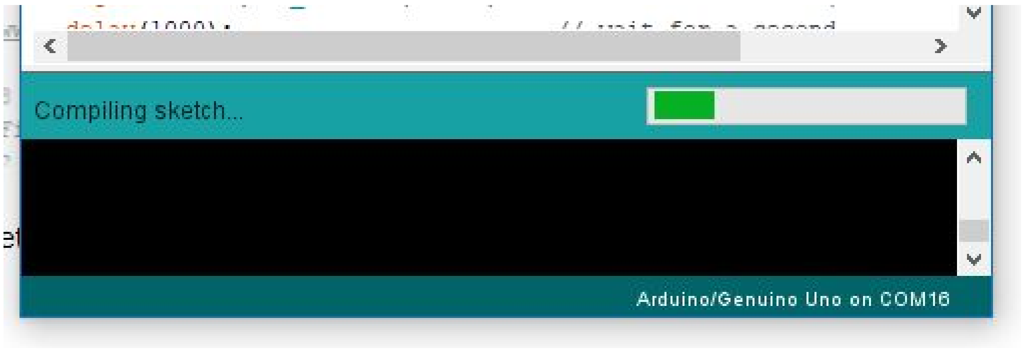
Select the serial port. Your serial number may be different from the following figure. If it is not detected immediately, please wait for a while, then click "Tools" to check again.



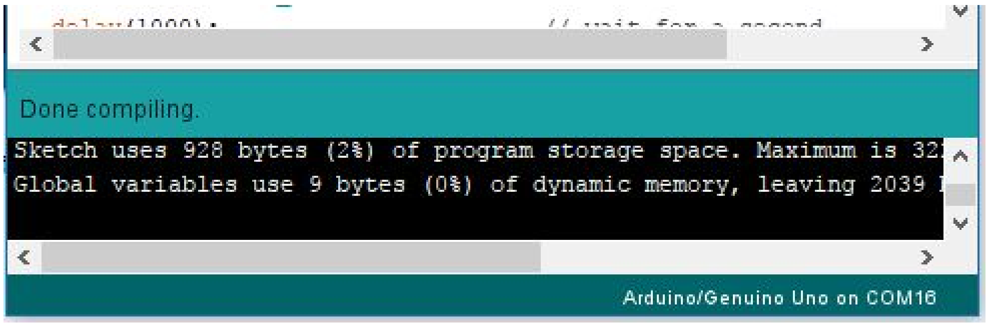
Click "Verify" button.



The following figure shows the code is being compiled.



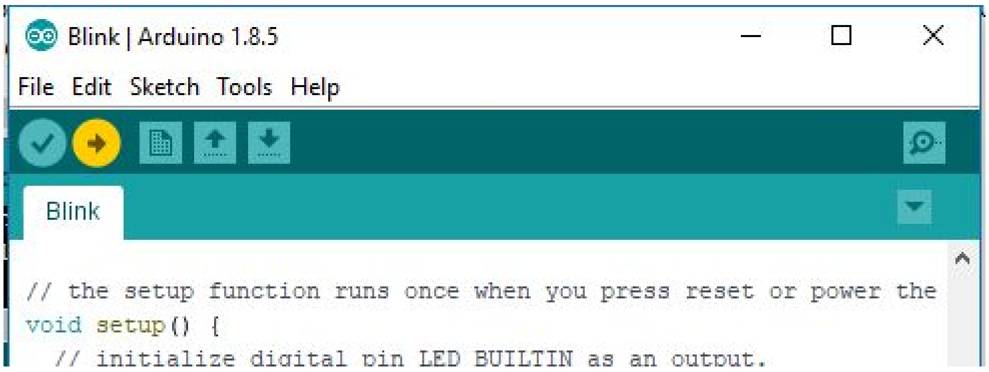
Wait a moment for the compiling to be completed. Figure below shows the code size and percentage of space occupation.



```
Sketch uses 928 bytes (2%) of program storage space. Maximum is 32
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 
```

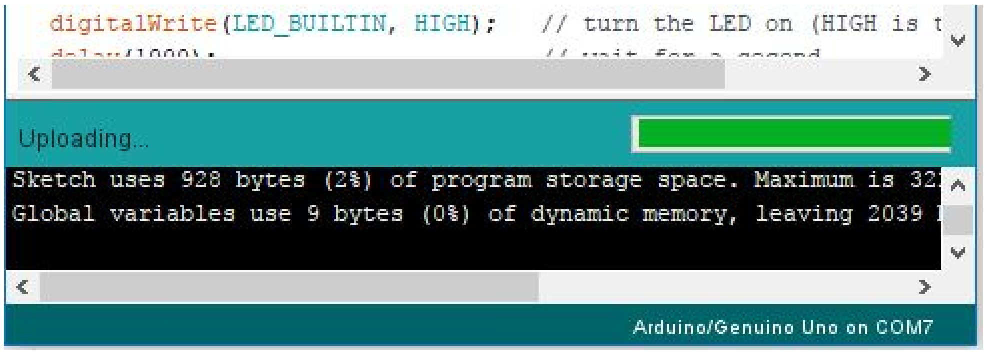
Usually, when we write code, if it has a syntax error, the interface will prompt the error message. Then the compiling can't be completed.

Click "Upload" button.



```
// the setup function runs once when you press reset or power the
void setup() {
  // initialize digital pin LED BUILTIN as an output.
```

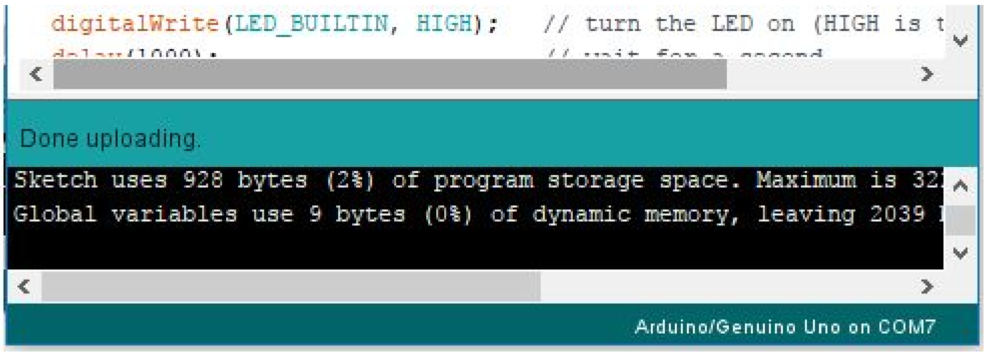
Figure below shows code are uploading.



```
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is t
```

```
Sketch uses 928 bytes (2%) of program storage space. Maximum is 32
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 
```

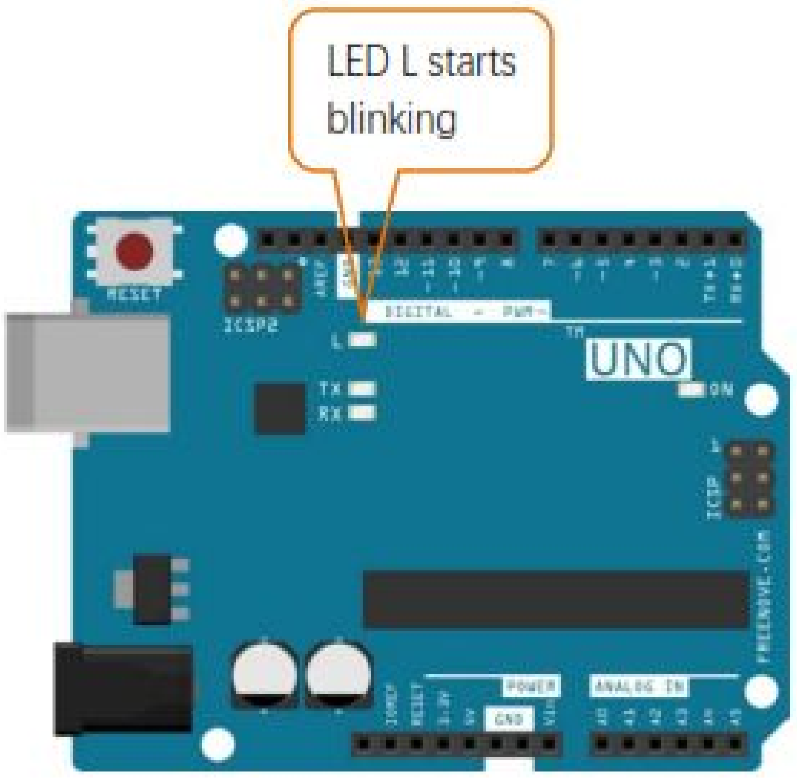
Wait a moment, then the uploading is completed.



```
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is t
```

```
Sketch uses 928 bytes (2%) of program storage space. Maximum is 32
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 
```

After that, we will see the LED marked with "L" on Arduino UNO starts blinking. It indicates that the code is running now!



So far, we have completed the first use. I believe you have felt the joy of it. Next, we will carry out a series of projects, from easy to difficult, taking you to learn the Arduino programming and the building of electronic circuit.

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 out 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 intil the end of the line counts as being a comment.

The first actual line of code is:

Copy Code

```
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

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

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.

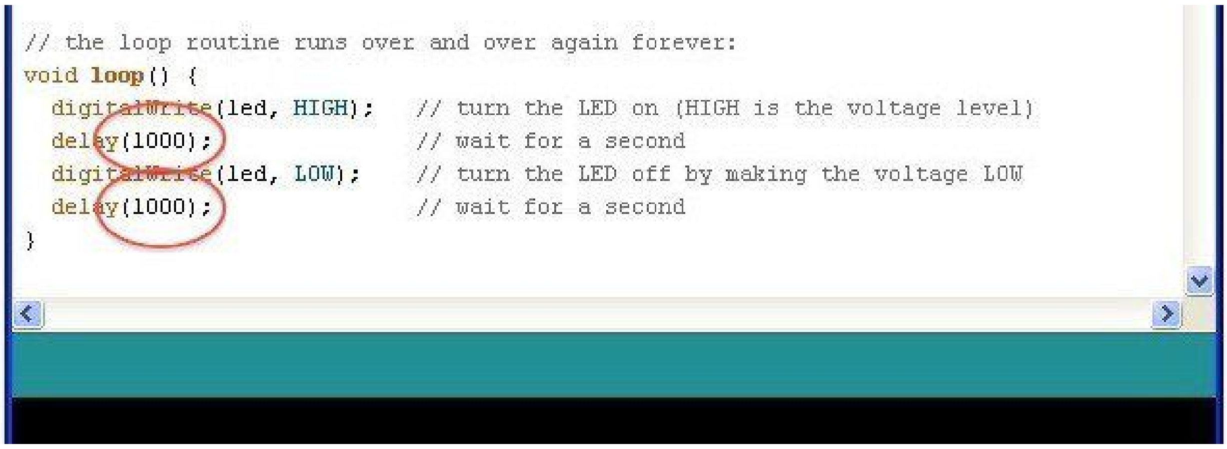
Copy Code

```
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 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.

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.



```
// 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
}
```

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

Login to LMS : https://lms.thingerbits.com/login

Lesson 2 - Button

Introduction

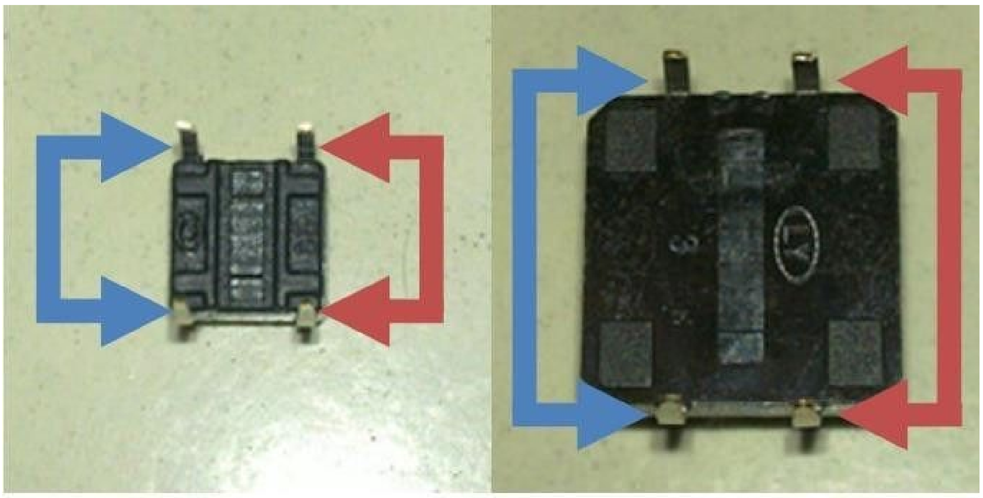
In this experiment, we will learn how to turn a single LED on or off by using an I/O port and button switch. The "I/O port" refers to the INPUT and OUTPUT port. We will use the input function of the Arduino Uno I/O port to read the output of an external device. Since the Arduino Uno board itself has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

Components

- 1 * Arduino Uno/Mega/Nano board
- 1 * USB cable
- 1 * Button
- 1 * Resistor (10kΩ)
- 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.



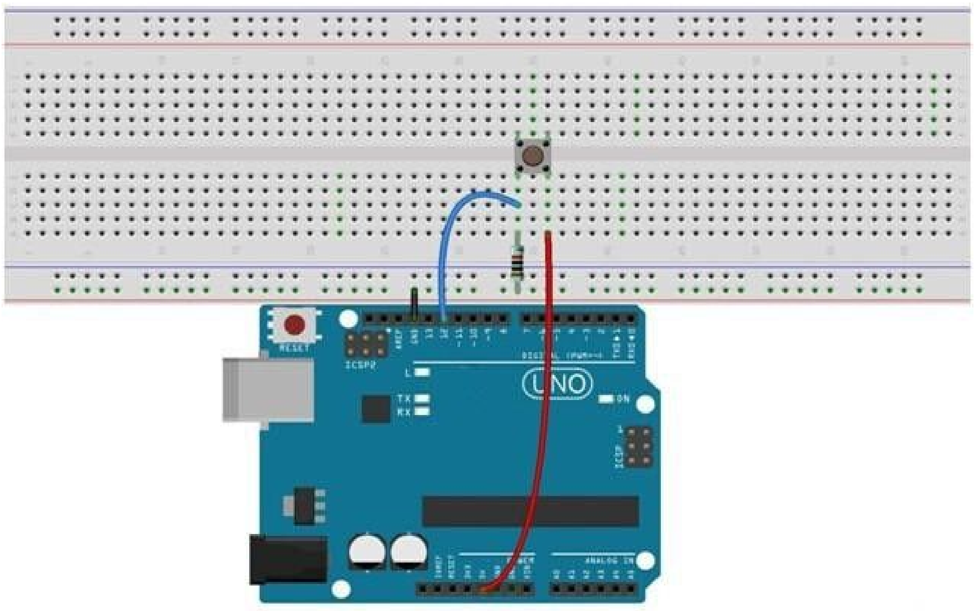
When the button is pressed, the pins pointed by the blue arrows will connect to the pins pointed by the red arrows.

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–10KΩ 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.

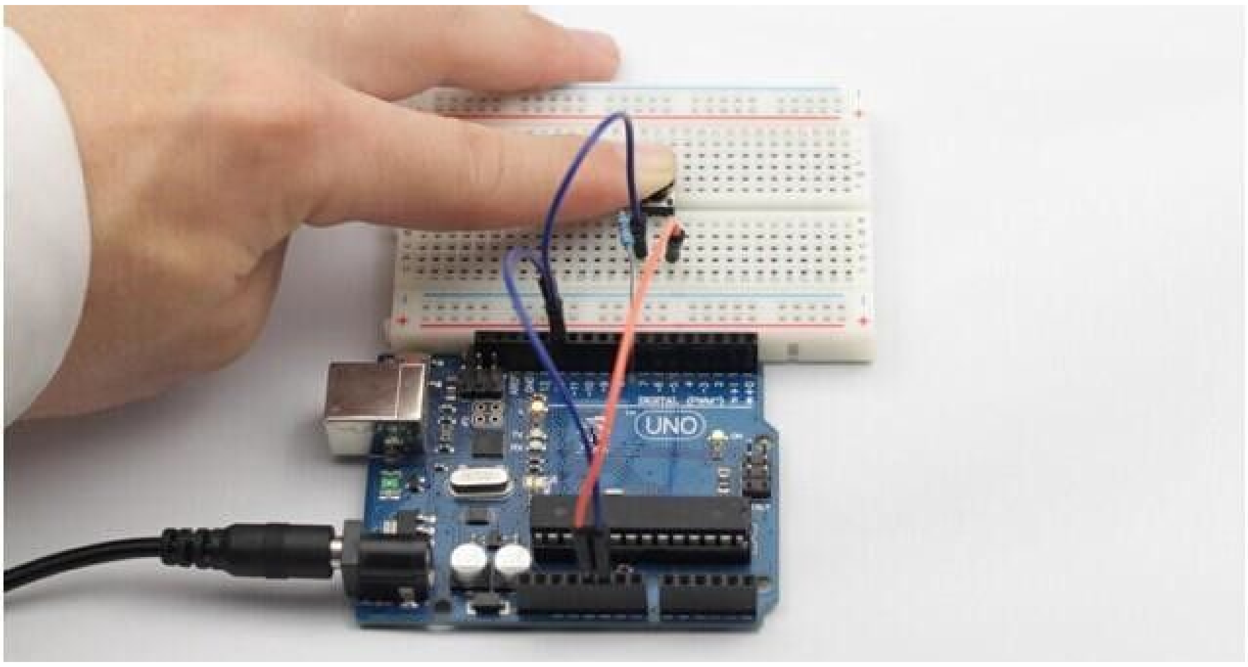
Experimental Procedures

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



Step 2: Program (please refer to the example code on the CD or http://bit.ly/tb1dayl2)
Step 3: Compile the program
Step 4: Burn the program into Arduino Uno board

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



Experimental Summary

Buttons are a very simple, very practical technology that is surprisingly easy to master. If you feel as though you're struggling, check out our video tutorials on www.thingerbits.com or ask us questions on our forum.

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

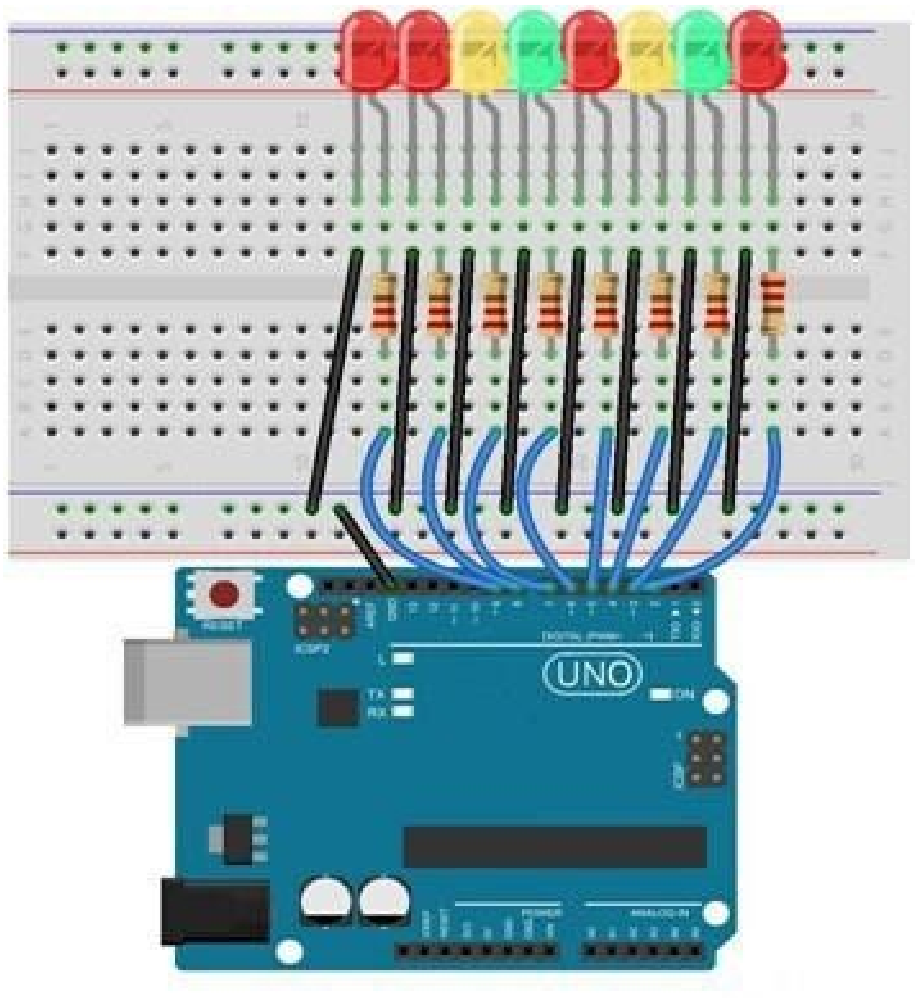
- 1 * Arduino Uno/Mega/Nano board
- 1 * Breadboard
- Jumper wires
- 8 * LED
- 8 * Resistor (220Ω)
- 1 * USB cable

Principle

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

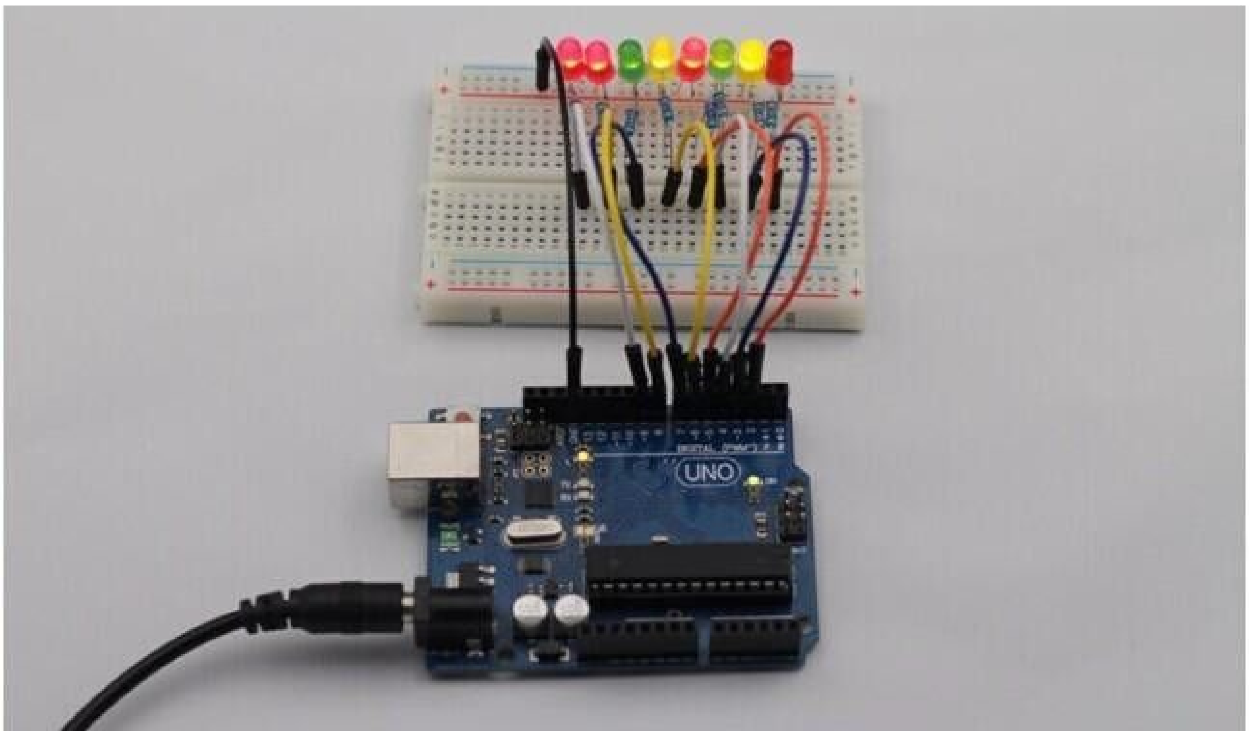
Experimental Procedures

Step 1: Connect circuit as shown in the following diagram



Step 2: Program (please refer to the example code on the CD or http://bit.ly/tb1dayl3)
Step 3: Compile the program
Step 4: Burn the program into Arduino 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

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

Experimental Conditions

- 1 * Arduino Uno/Mega/Nano 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.

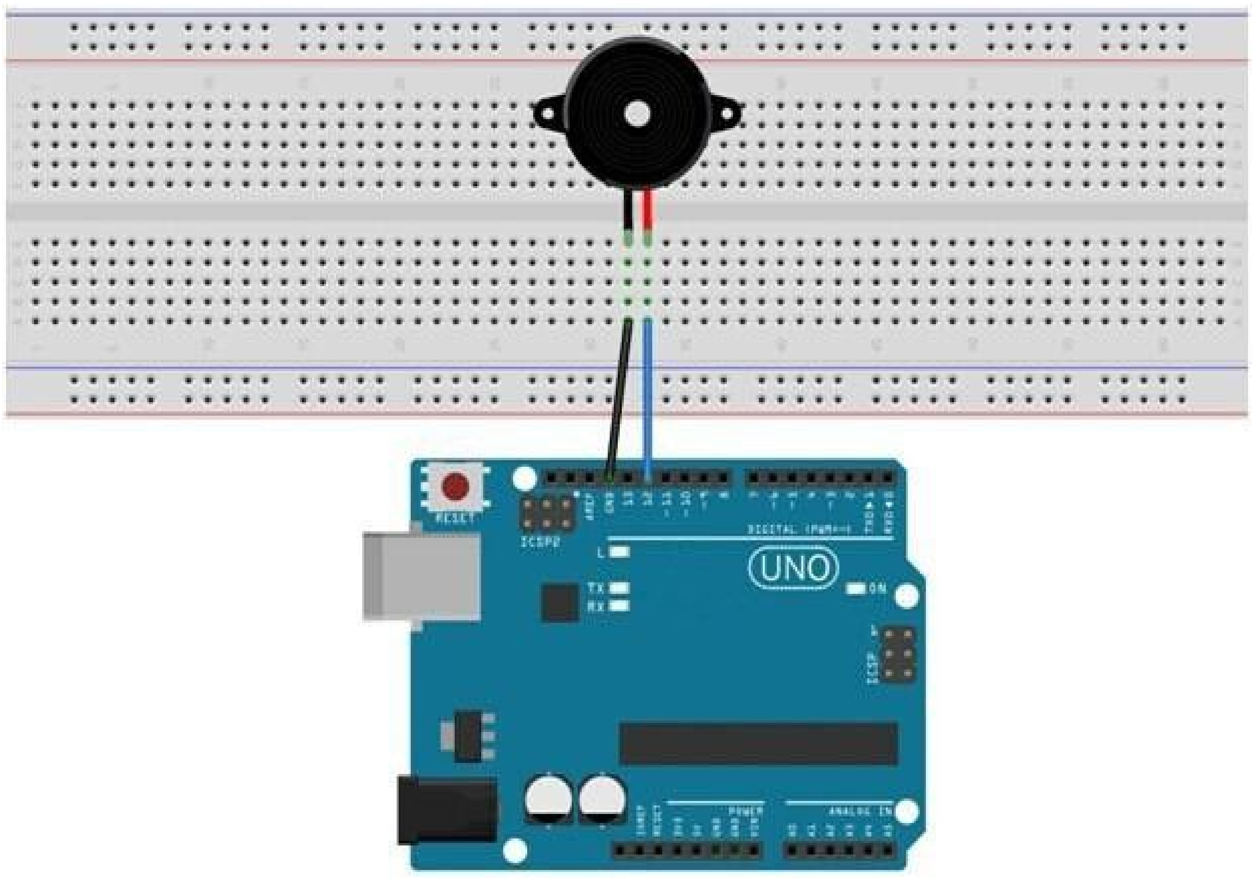
The difference between an **active buzzer** and a **passive buzzer** is:

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.

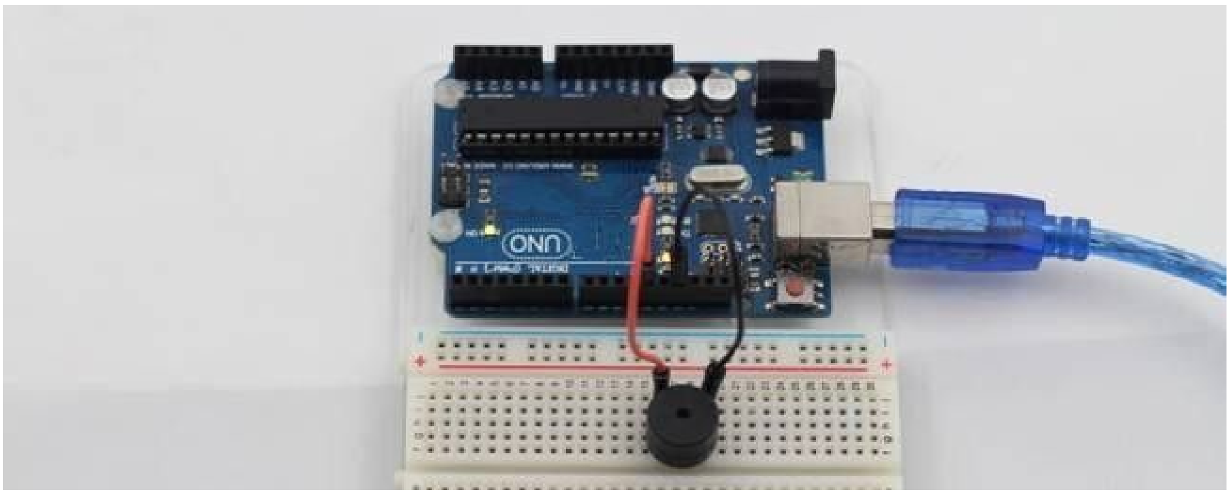
In this experiment, we use the active buzzer.

Experimental Procedures

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



Step 2: Program (please refer to the example code on the CD or http://bit.ly/tb1dayl4)
Step 3: Compile the program
Step 4: Burn the program into Arduino 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

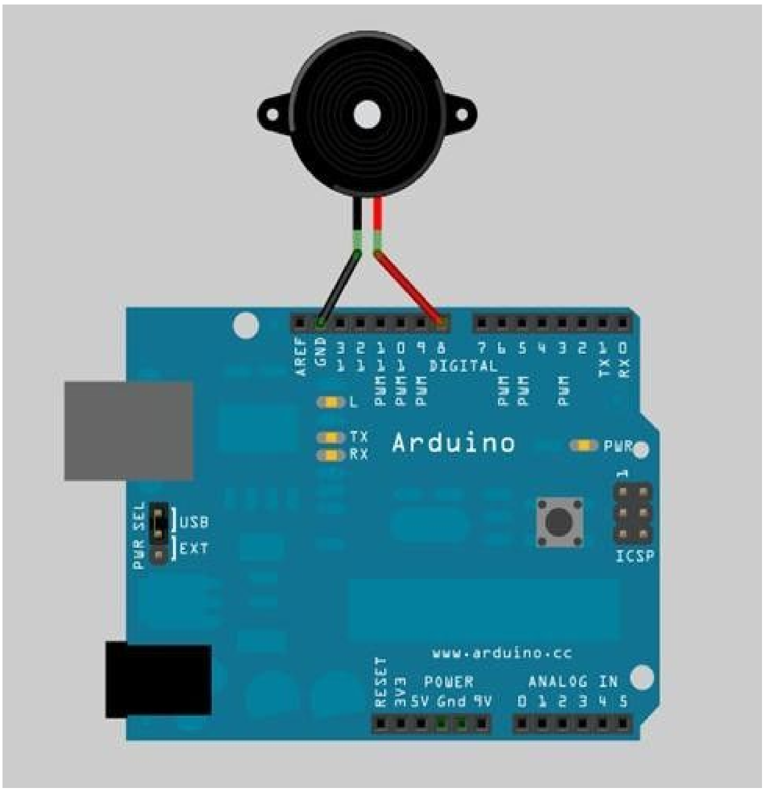
- 1 * Arduino Uno/Mega/Nano board
- 1 * USB data cable
- 1 * Passive Buzzer
- Several jumper wires
- 1 * Breadboard

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 on the CD or http://bit.ly/tb1dayl5)
Step 3: Compile the program
Step 4: Burn the program into Arduino 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 * Arduino Uno/Mega/Nano board
- 1 * USB data cable
- 1 * Photoresistor
- 1 * Resistor (10KΩ)
- 8 * LED
- 8 * Resistor (220Ω)
- Jumper wires
- 1 * Breadboard

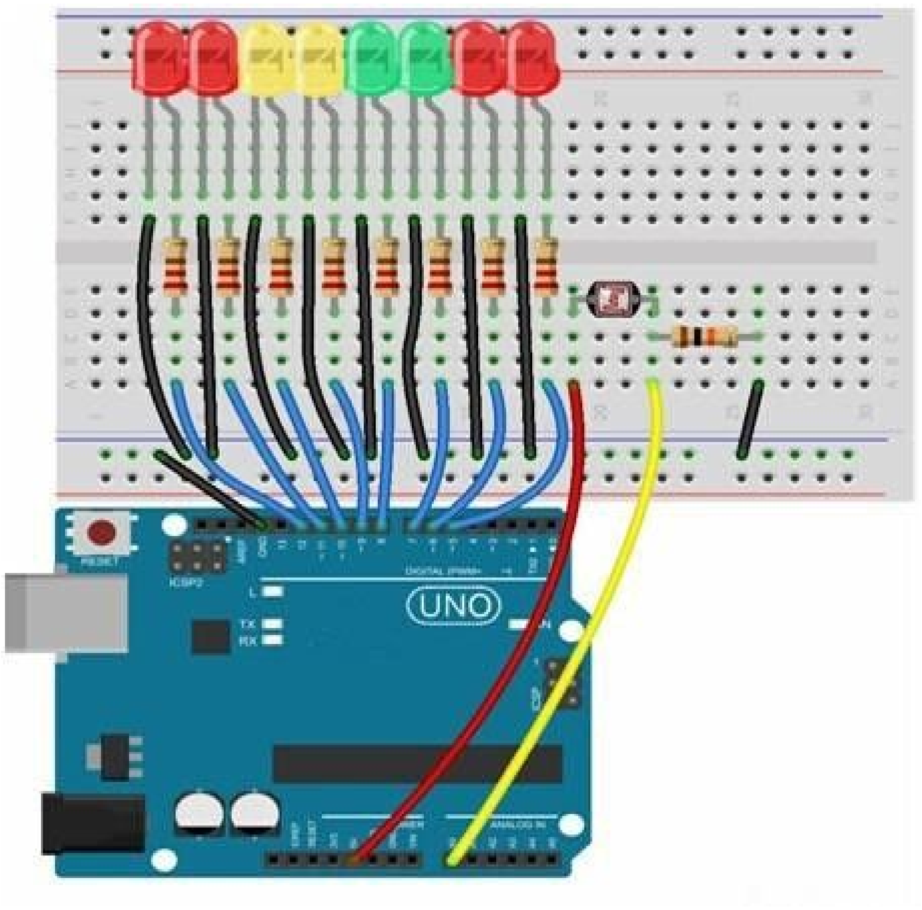
Experimental Principle

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:

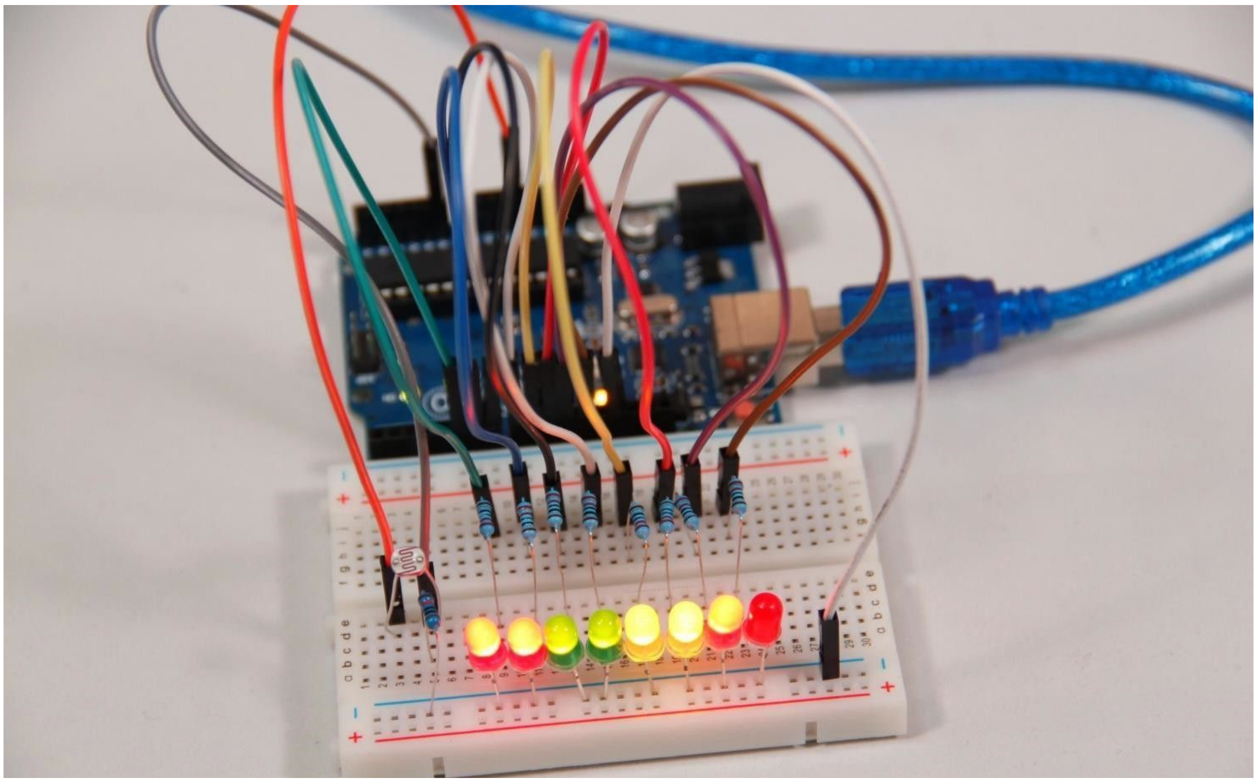


Step 2: Program (please refer to the example code on the CD or http://bit.ly/tb1dayl6)

Step 3: Compile the program

Step 4: Burn the program into Arduino Uno/Mega/Nano 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 * Arduino Uno/Mega/Nano board
- Jumper wires
- 1 * USB cable

Principle

Color Principle of RGB

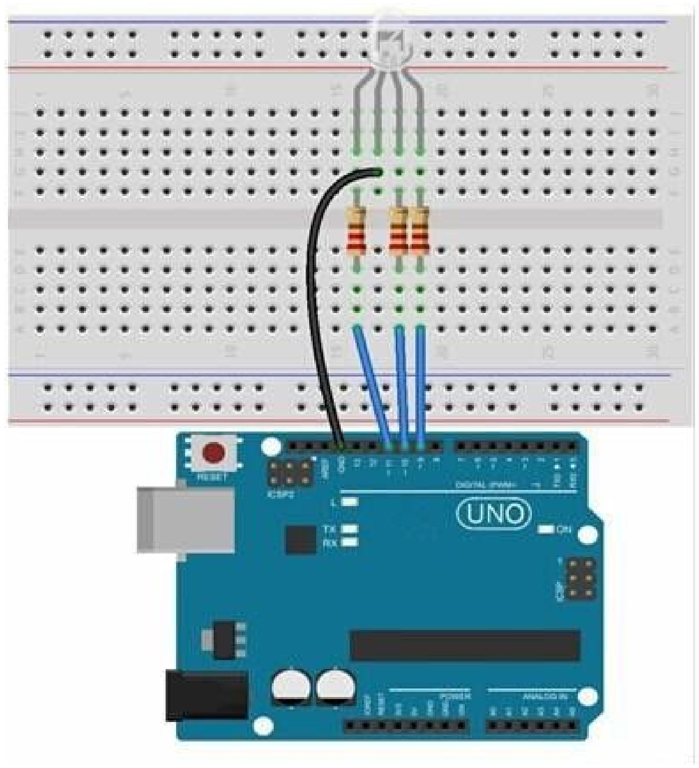
RGB stands for the red, green, and blue color channels and is an industry color standard. RGB displays various new colors by changing the three channels and superimposing them, which, according to statistics, can create 16,777,216 different colors. If you say the color displayed doesn't completely match 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.

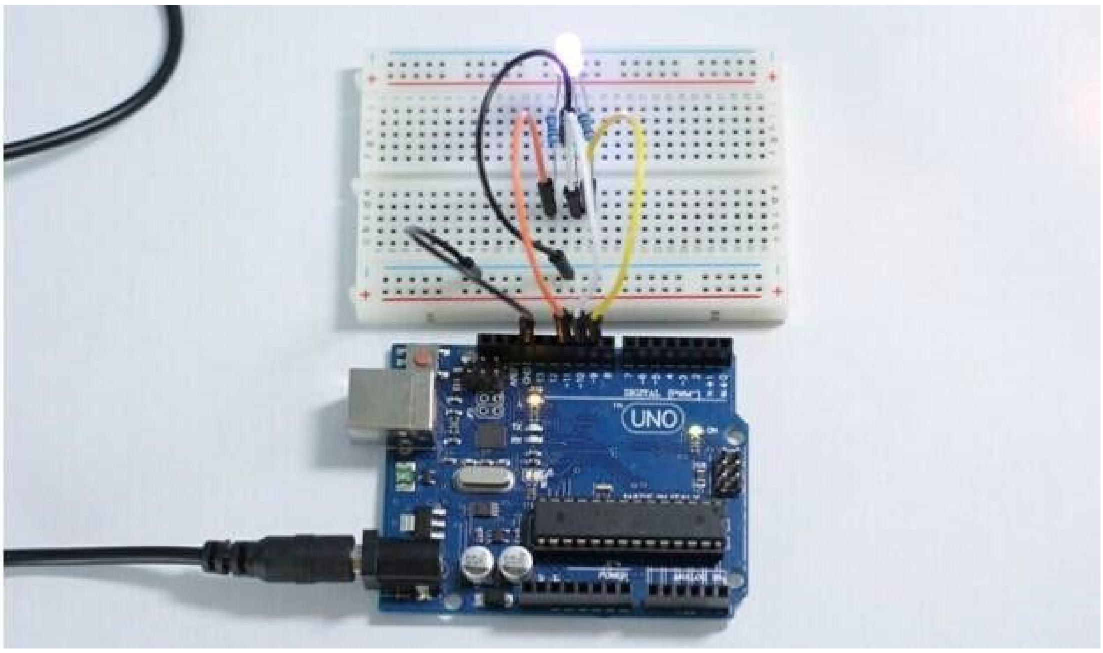
Experimental Procedures

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



Step 2: Program (please refer to the example code on the CD or http://bit.ly/tb1dayl7)
Step 3: Compile the program
Step 4: Burn the program into Arduino Uno/Mega/Nano board

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



Lesson 8 - LM35 Temperature Sensor

Introduction

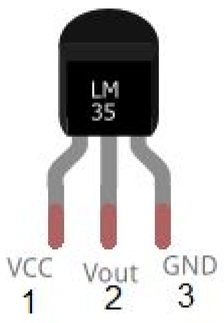
The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly proportional to the Centigrade temperature. LM35 is three terminal linear temperature sensor from National semiconductors. It can measure temperature from -55 degree Celsius to +150 degree Celsius. The voltage output of the LM35 increases 10mV per degree Celsius rise in temperature. LM35 can be operated from a 5V supply and the standby current is less than 60uA. The pin out of LM35 is shown in the figure below.

Components

- 1 * LM35
- 3 * Resistor (220Ω)
- 1 * Breadboard
- 1 * Arduino Uno/Mega/Nano board
- Breadboard jumper wires*several
- 1 * USB cable

Principle

LM35 Pinout

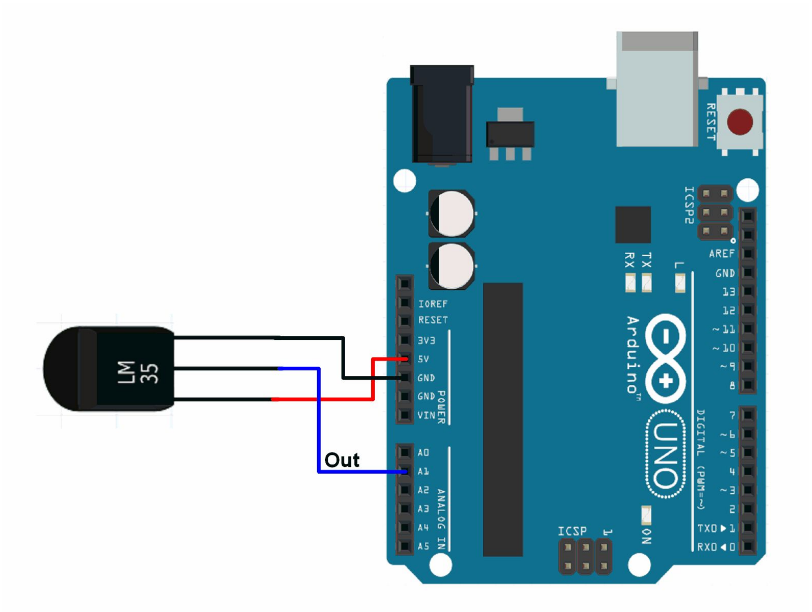


It has three pins:

- pin 1 is +Vcc
- pin 2 is Output
- pin 3 is Ground

Experimental Procedures

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



Step 2: Program (please refer to the example code on the CD or http://bit.ly/tb1dayl8)
Step 3: Compile the program
Step 4: Burn the program into Arduino Uno/Mega/Nano board

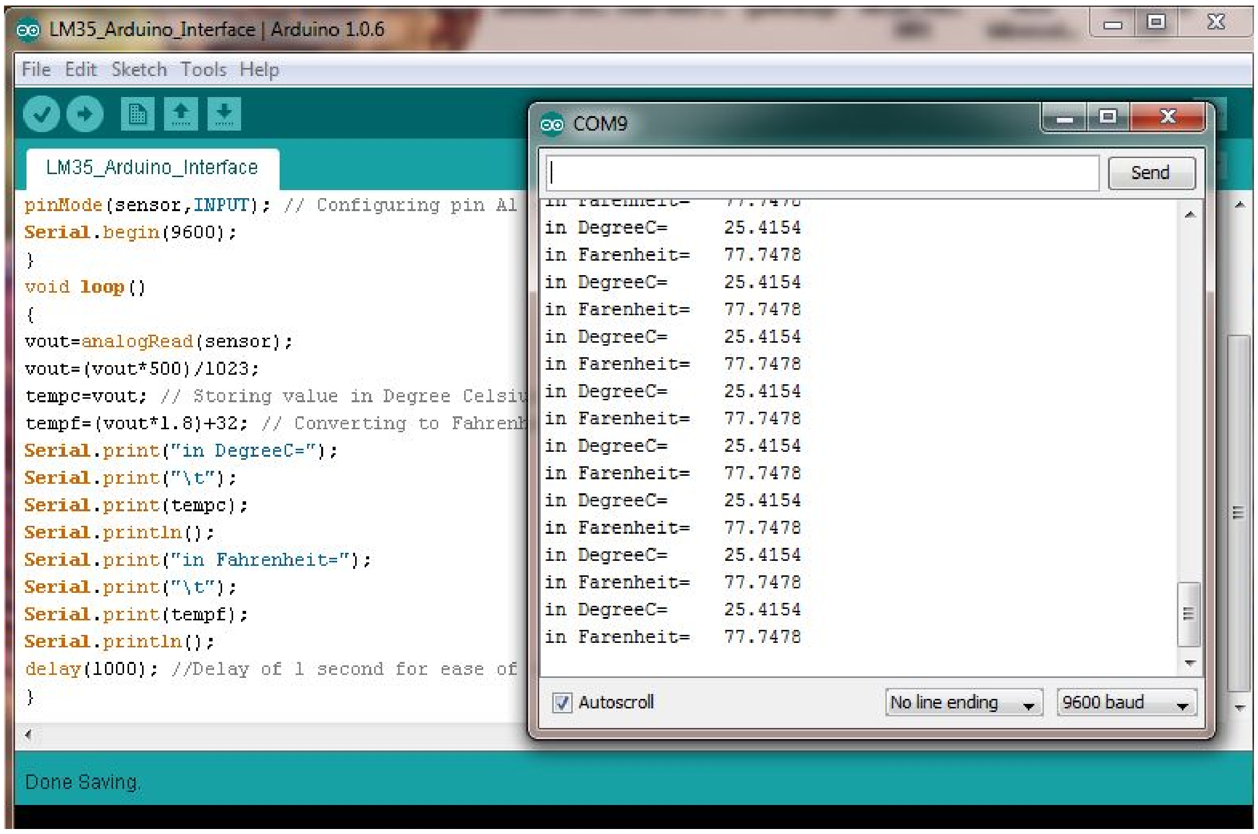
Measuring the temperature of surroundings using LM35 and displaying it on the serial monitor of Arduino.

Here, LM35 output is given to analog pin A1 of Arduino UNO. This analog voltage is converted to its digital form and processed to get the temperature reading.

Sketch for Temperature Measurement

```
const int sensor=A1; // Assigning analog pin A1 to variable 'sensor'
float tempc;  //variable to store temperature in degree Celsius
float tempf;  //variable to store temperature in Fahreinheit
float vout;  //temporary variable to hold sensor reading

void setup()
{
	pinMode(sensor,INPUT); // Configuring pin A1 as input
	Serial.begin(9600);
}

void loop()
{
	vout=analogRead(sensor);
	vout=(vout*500)/1023;
	tempc=vout; // Storing value in Degree Celsius
	tempf=(vout*1.8)+32; // Converting to Fahrenheit
	Serial.print("in DegreeC=");
	Serial.print("\t");
	Serial.print(tempc);
	Serial.println();
	Serial.print("in Fahrenheit=");
	Serial.print("\t");
	Serial.print(tempf);
	Serial.println();
	delay(1000); //Delay of 1 second for ease of viewing
}
```

So that’s the arduino lm35 code for reading temperature and displaying in degree Celsius and Fahrenheit. The program is self explanatory.



```
pinMode(sensor,INPUT); // Configuring pin A1
Serial.begin(9600);
}
void loop()
{
vout=analogRead(sensor);
vout=(vout*500)/1023;
tempc=vout; // Storing value in Degree Celsiu
tempf=(vout*1.8)+32; // Converting to Fahrenh
Serial.print("in DegreeC=");
Serial.print("\t");
Serial.print(tempc);
Serial.println();
Serial.print("in Fahrenheit=");
Serial.print("\t");
Serial.print(tempf);
Serial.println();
delay(1000); //Delay of 1 second for ease of
}
```

```
in DegreeC=     25.4154
in Farenheit=   77.7478
in DegreeC=     25.4154
in Farenheit=   77.7478
in DegreeC=     25.4154
in Farenheit=   77.7478
in DegreeC=     25.4154
in Farenheit=   77.7478
in DegreeC=     25.4154
in Farenheit=   77.7478
in DegreeC=     25.4154
in Farenheit=   77.7478
in DegreeC=     25.4154
in Farenheit=   77.7478
in DegreeC=     25.4154
in Farenheit=   77.7478
```

Lesson 9 - Magnetic Reed Switch Sensor

Introduction

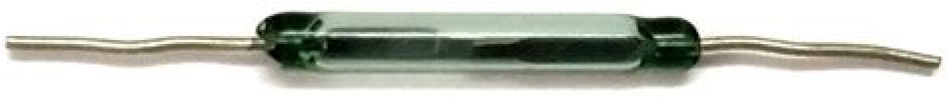
Reed switch is used in many of the real-life applications such as magnetic door switch, laptops, smartphones etc. In this Lesson, we learn about Reed Switch and guide you to Interface a Reed Switch with Arduino.

Components

- 1 * Arduino Uno board
- 1 * Breadboard
- 1 * Reed switch
- 1 * LED
- 1 * Magnet
- Jumper wires
- 1 * USB cable
- 220Ω Resistor and 10K Resistor

Principle

Reed switch is basically an electrical switch which is operated when a magnetic field is brought near to it. It was invented by W. B. Ellwood in 1936 at bell laboratories. It is made up of two small metal pieces kept inside a glass tube under vacuum. In a typical reed switch two metal pieces will be made of a ferromagnetic material and covered with rhodium or ruthenium to give them long life. The switch will be activated when there is a presence of magnetic field around the switch.

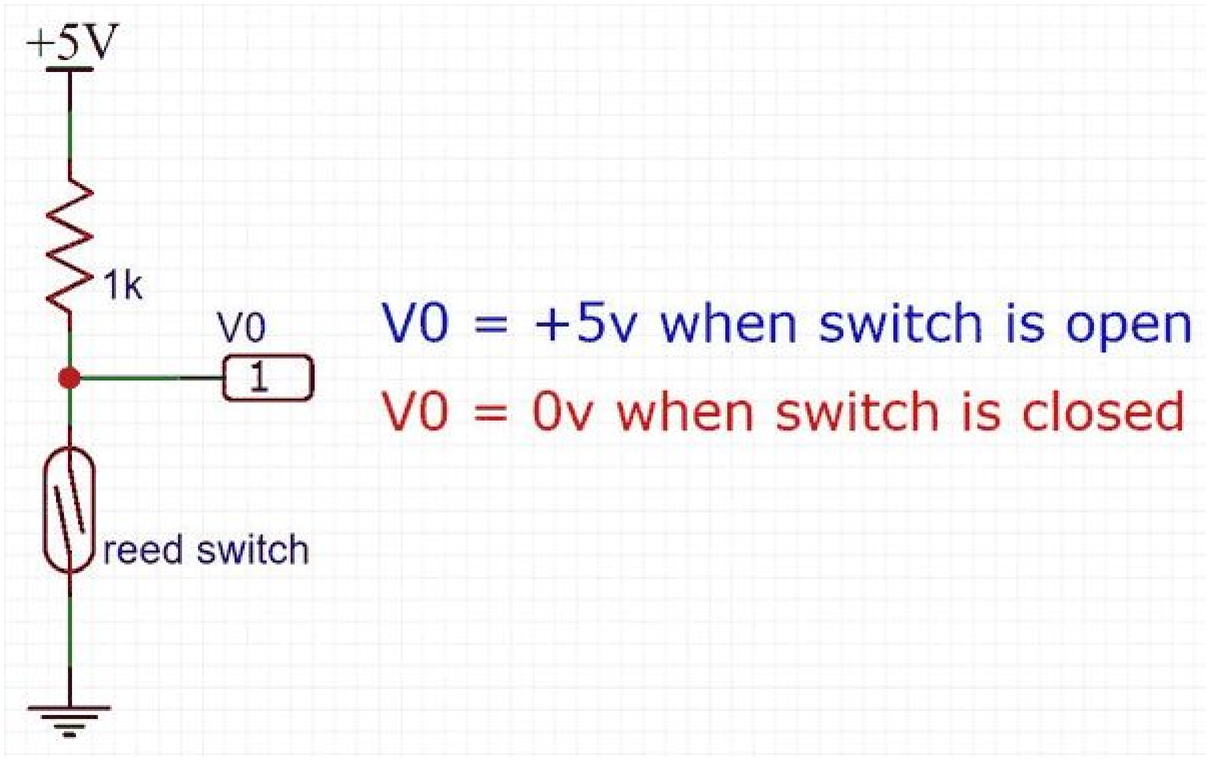


There are two types of reed switch.

a. **Normally open reed switch**
b. **Normally closed reed switch**

In normally open reed switch, switch is open in the absence of magnetic field and it is closed in the presence of magnetic field. Under the presence of magnetic field, two metal contacts inside the glass tube attract each other to make contact.

In normally closed reed switch, switch is closed in the absence of magnetic field and it is open in the presence of magnetic field.



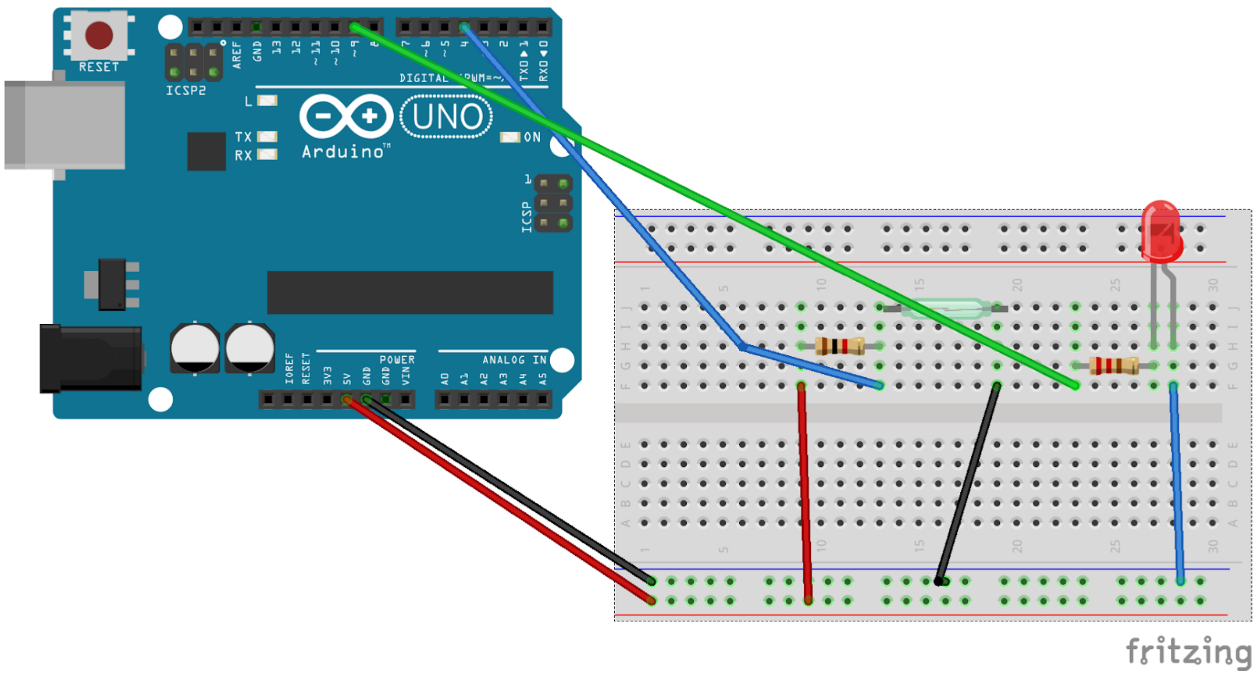
Applications of Reed switch

- Used in telephone exchange
- In laptops to put the screen on sleep if the lid is closed
- Used in window and door sensors in burglar alarm system

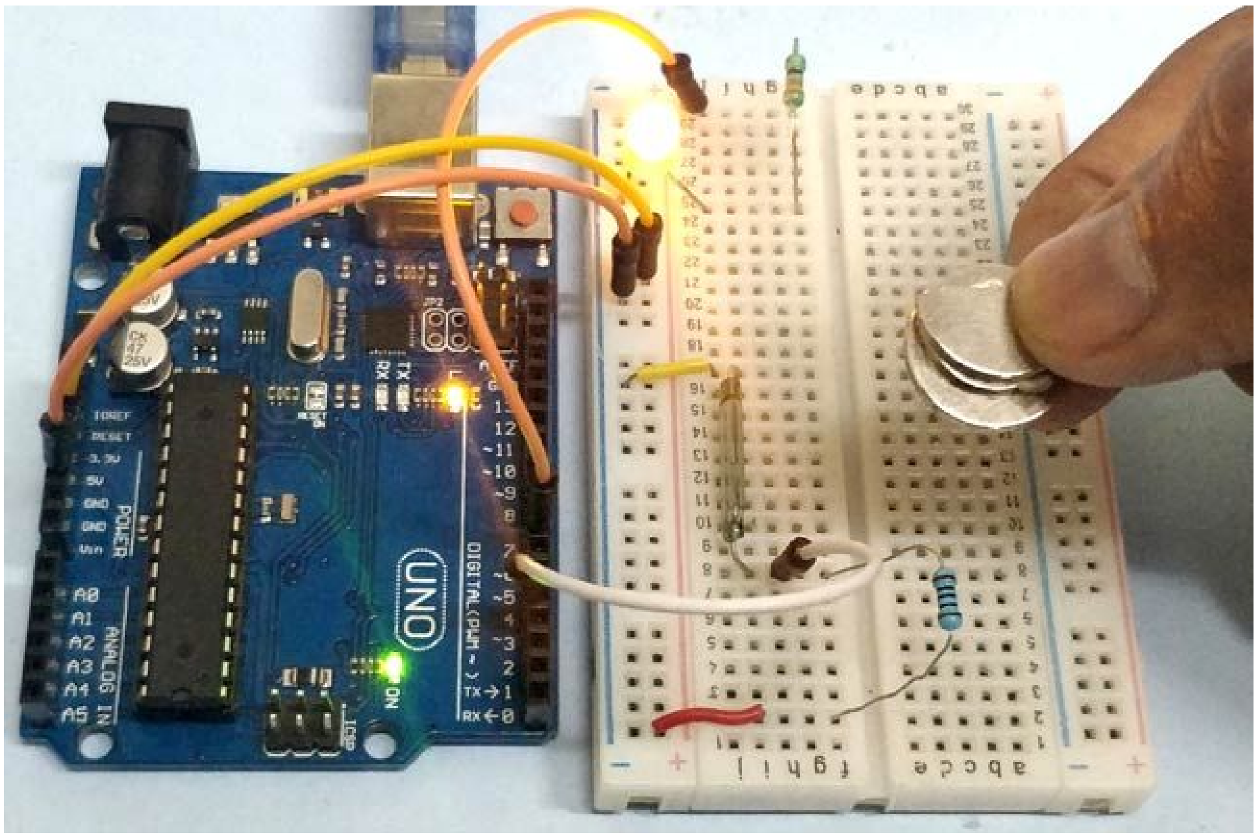
Experimental Procedures

To interface reed switch with Arduino we need to build a voltage divider circuit as shown in the figure below. Vo is +5V when the switch is open and 0V when the switch is closed. We are using a normally open reed switch in this project. Switch is closed in the presence of magnetic field and it is open in the absence of magnetic field.

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 on the CD or http://bit.ly/tb1dayl9)
Step 3: Compile the program
Step 4: Burn the program into Arduino Uno/Mega/Nano board



Code explanation

The complete code for this Arduino reed switch project is given at the end of this article. The code is split into small meaningful chunks and explained below.

In this part of the code we have to define pins on which Reed switch and LED which is connected to Arduino. Reed switch is connected to digital pin 4 of Arduino and LED is connected to digital pin 7 of Arduino through a current limiting resistor. The variable “reed_status” is used to hold the status of reed switch.

```
int LED = 7;
int reed_switch = 4;
int reed_status;
```

In this part of the code, we have to set status of pins on which LED and reed switch is connected. Pin number 4 is set as input and pin number 7 is set as output.

```
void setup()
{
  pinMode(LED, OUTPUT);
  pinMode(reed_switch, INPUT);
}
```

Next, we have to read the status of reed switch. If it is equal to 1, switch is open and LED is turned off. If it is equal to 0, switch is closed and we have to turn on LED. This process is repeated every second. This task is accomplished with this part of the code below.

```
void loop()
{
            reed_status = digitalRead(reed_switch);
            if (reed_status == 1)
              digitalWrite(LED, LOW);
            else
              digitalWrite(LED, HIGH);
            delay(1000);
}
```

So as you have seen its very easy to use Reed Switch with Arduino.

```
int LED = 7;
int reed_switch = 4;
int reed_status;
void setup()
{
            pinMode(LED, OUTPUT);
            pinMode(reed_switch, INPUT);
}
void loop()
{
            reed_status = digitalRead(reed_switch);
            if (reed_status == 1)
              digitalWrite(LED, LOW);
            else
              digitalWrite(LED, HIGH);
            delay(1000);
}
```

Lesson 10 - Controlling a DC Motor

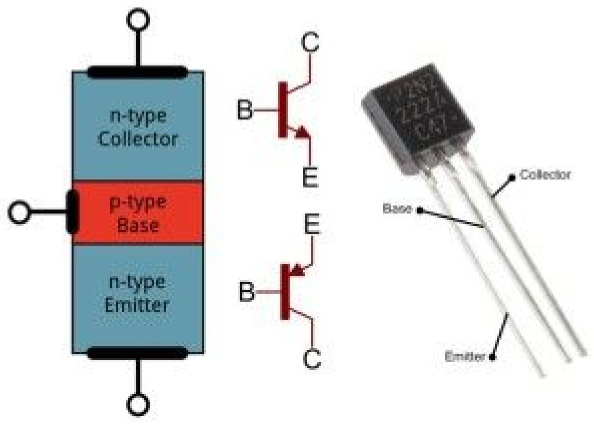
Introduction

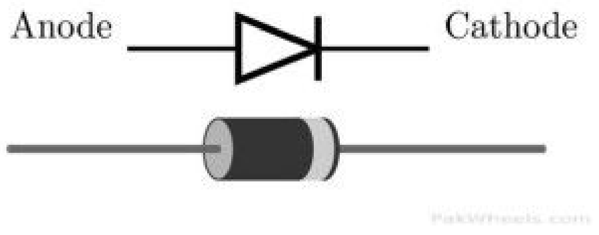
In this lesson you will going to learn how to control and drive the DC Motor on Arduino, you will also learn how to use the analog output PWM or Pulse width Modulation to control the speed of the motor by changing the value of 0 ~ 255 from the serial monitor.

Components

- 1 * Arduino Uno board
- 1 * Breadboard
- 1 * Small 6V DC Motor
- 1 * NPN Transistor
- 1 * USB cable
- 1 * 1N4001 Diode (Anode Negative– / Cathode Positive +)
- 1 * 220 Ω Resistor
- Jumper Wires

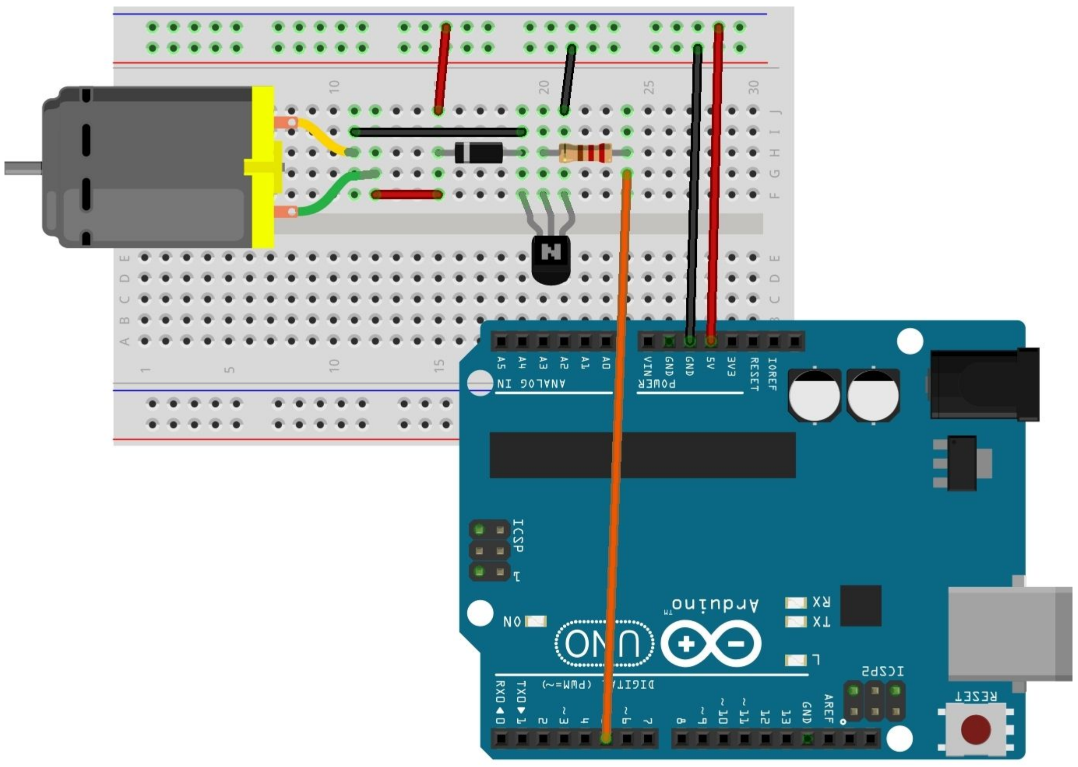
Principle





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):



Placing the transistor should be in the right way, please see the diagram above to know the transistor pin, the same with the diode. Diode only allows the flow of current in one direction. see also the diagram above for the diode pinout. the striped end of the diode should be towards into +5v at Arduino power line.

The PN222 Transistor has a 3 leads most of the electricity flows from the collector going to the emitter, this will only happen if a small amount of current flowing into the base connection. this small current will supplied by the Arduino digital output.

The digital 5 of the Arduino is connected to the resistor the same like using an LED this resistor will limits the current flowing into the transistor through the base, the diode connected across the connection of the motor, when you turn the power off to the motor, you will get a negative SPIKE of voltage that can damage your Arduino or your transistor, the diode protects against this SPIKE by shorting out any such reverse current from the motor.

```
int MOTOROLL =5;

void setup()
{
        pinMode(MOTOROLL, OUTPUT);
        Serial.begin(9600); // Opening the serial communication at baud rate 9600
        while(!Serial);
        Serial.println(“Turning Speed > 0 to 255″); // Serial Output Window
}
void loop()
{
        if(Serial.available())
        {
```

```
        int TurningSpeed = Serial.parseInt();
        if(TurningSpeed >= 0 && TurningSpeed <=255) // Turning Speed PWM at 0
        {
        analogWrite(MOTOROLL, TurningSpeed);
        }
        }
}

```

Lesson 11 - Simple Creation - Light Alarm

Introduction

This experiment is a very interesting one – a DIY photistor. DIY photistors use the principle of glow effect and the photoelectric effect of LEDs. That is, LEDs will generate weak currents when being shined on by light. We use a transistor to amplify the currents and trigger the thingerbits Uno board to detect them.

Components

- 1 * Arduino Uno/Mega/Nano board
- 1 * Breadboard
- 1 * USB cable
- Jumper wires
- 1 * Passive buzzer
- 1 * Resistor (10KΩ)
- 1 * LED
- 1 * NPN Transistor S8050

Principle

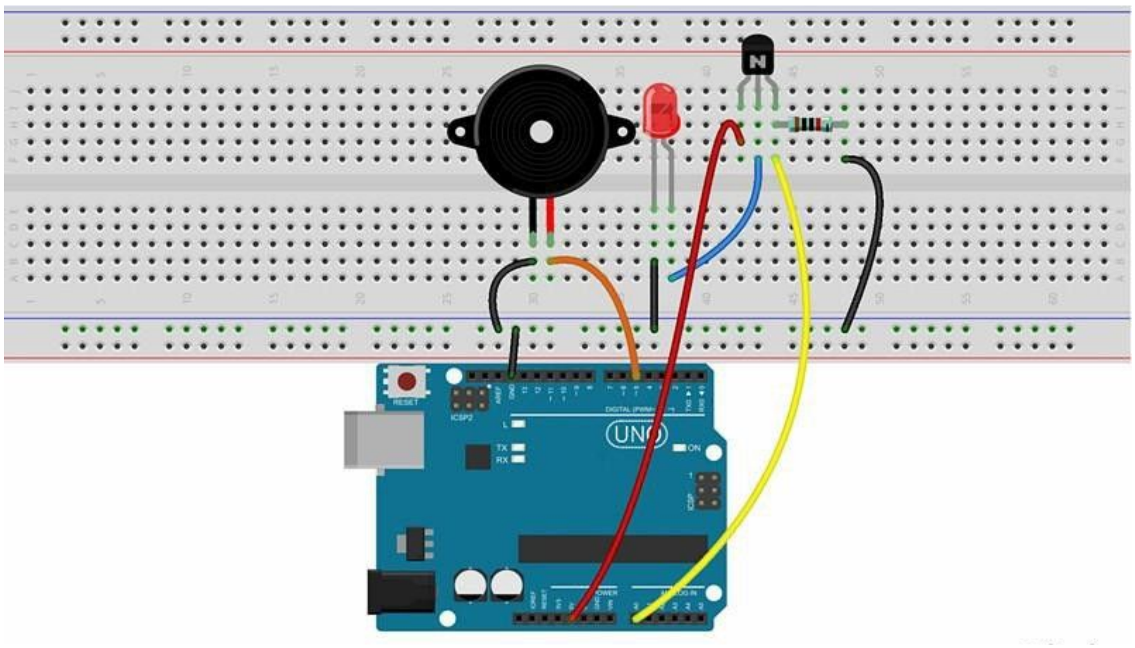
LEDs not only have a glow effect, but also a photoelectric effect. They will generate weak currents when exposed to light waves.

NPN Transistors are easy to use. The left pin is Emitter electrode, attached to the power source, the middle pin is Base, and the right pin is Collector. If only the middle pin has weak trigger currents, they will flow from left to right of the LED like a switch being opened.

A 10kΩ pull-down resistor is attached to the transistor output stage in order to avoid analog port suspending to interfere with signals and cause misjudgment.

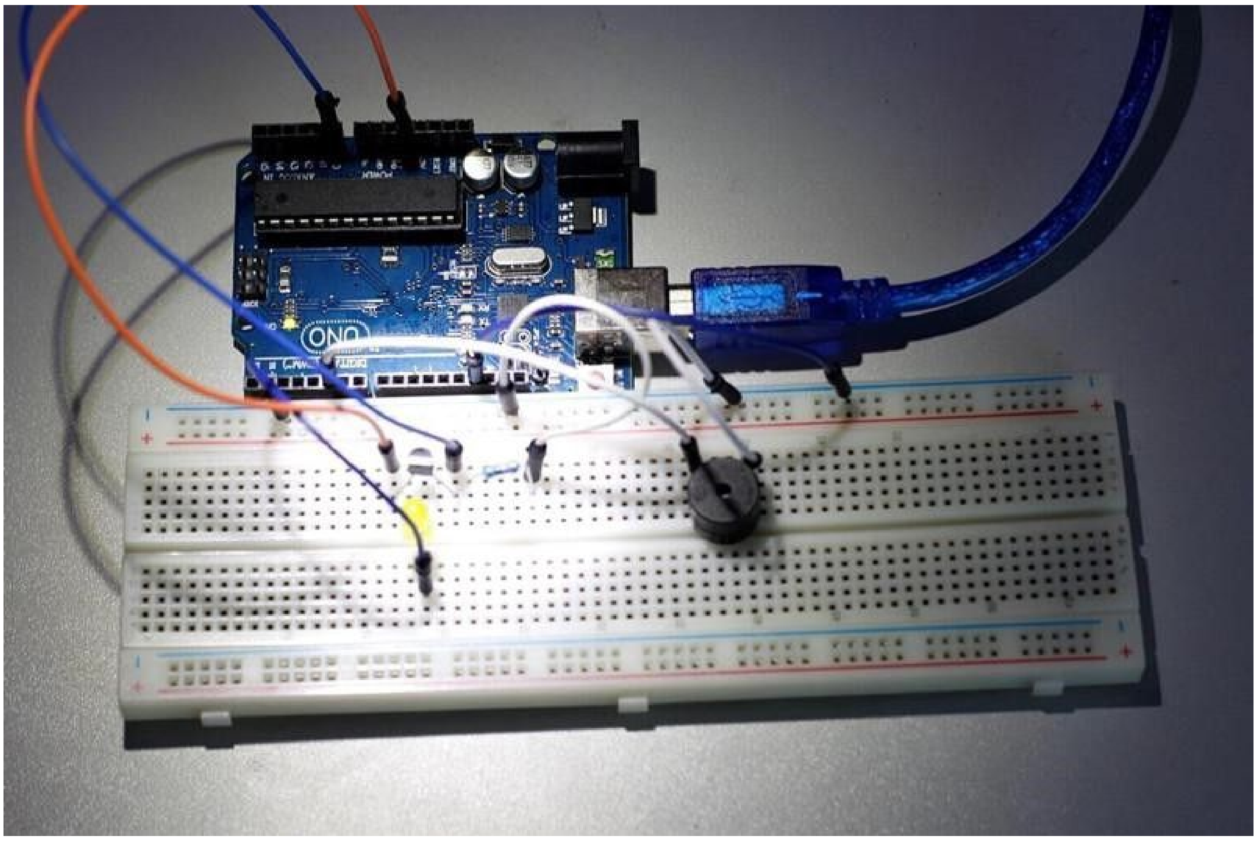
Experimental Procedures

Step 1: Connect circuit as shown in the following diagram



Step 2: Program (please go to our official website www.thingerbits.com to download related code or get the code from CD.)
Step 3: Compile the program
Step 4: Burn the program into Arduino Uno board

Now, you can hear that the buzzer make sounds when the LED is shined.



Experimental Summary

Through this experiment, you should have gained a basic understanding of analog circuits. You can also create many interesting interactive works by combining analog circuits and the Arduino Uno board. This is just the beginning!

What's next?

Thanks for your reading.

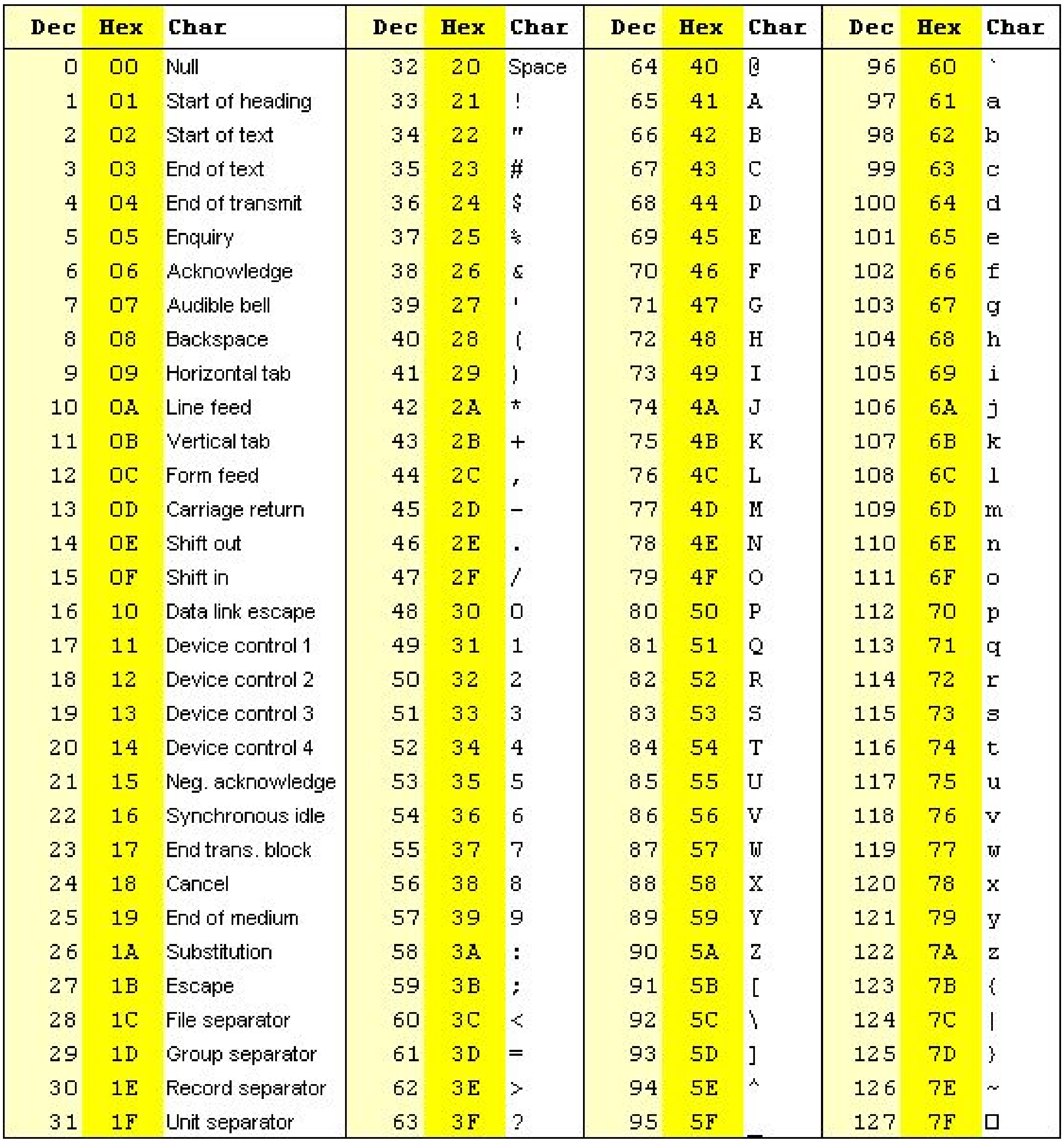
This book is all over here. If you find any mistakes, missions or you have other ideas and questions about contents of this book or the kit and ect, please feel free to contact us, and we will check and correct it as soon as possible.

If you want to learn more about Arduino, Raspberry Pi, smart cars, robots and other interesting products in science and technology, please continue to focus on our website. We will continue to launch cost-effective, innovative and exciting products.

Thank you again for choosing thingerbits products.

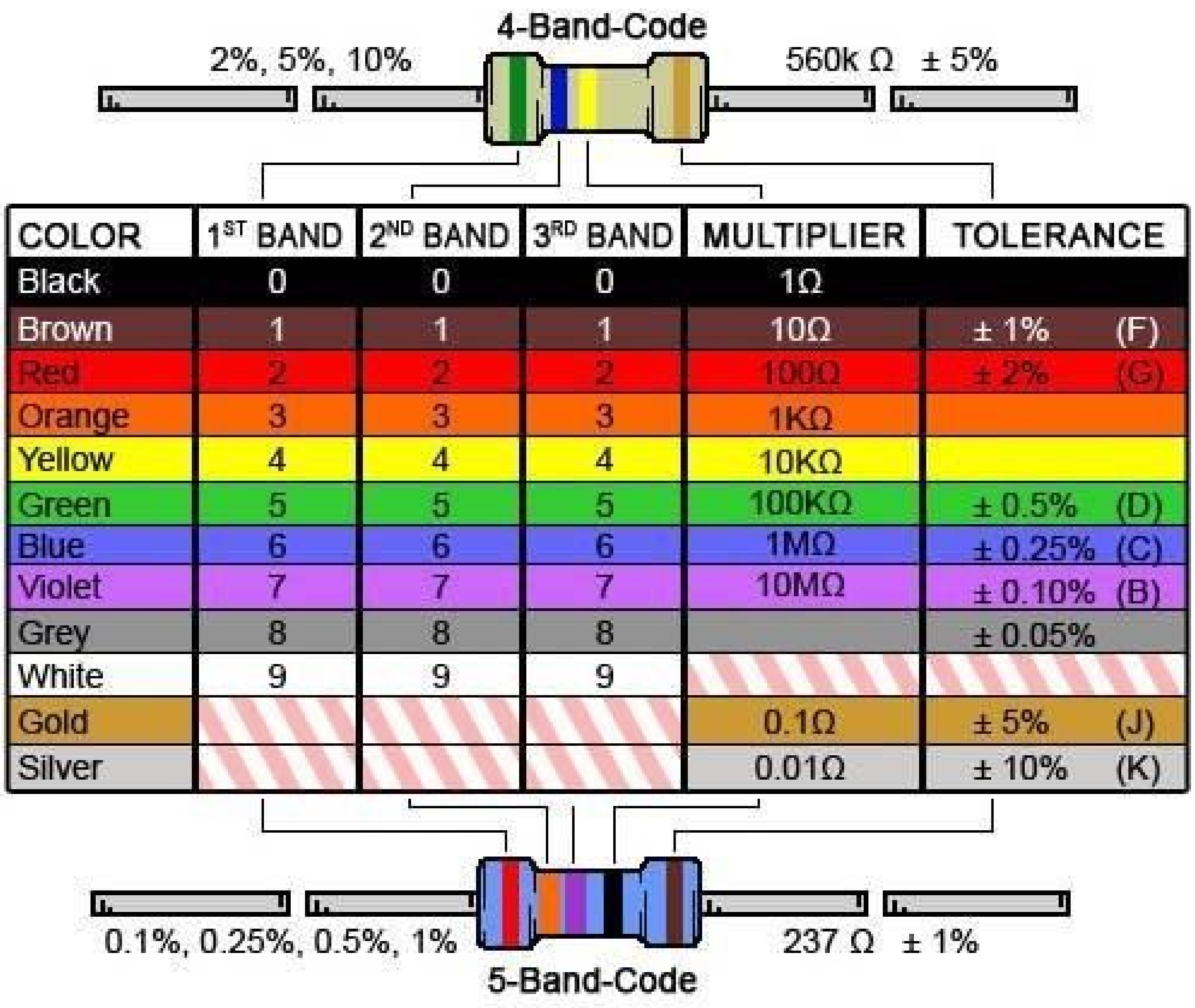
Appendix

Appendix



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Resistor color code


4-Band-Code
2%, 5%, 10%
560k Ω ± 5%
0.1%, 0.25%, 0.5%, 1%
237 Ω ± 1%
5-Band-Code

COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9	▨	▨
Gold	▨	▨	▨	0.1Ω	± 5% (J)
Silver	▨	▨	▨	0.01Ω	± 10% (K)

Language Reference

Arduino programming language can be divided in three main parts: structure, values (variables and constants), and functions.

FUNCTIONS

For controlling the Arduino board and performing computations.

Digital I/O

- digitalRead()
- digitalWrite()
- pinMode()

Analog I/O

- analogRead()
- analogReference()
- analogWrite()

Zero, Due & MKR Family

- analogReadResolution()
- analogWriteResolution()

Advanced I/O

- noTone()
- pulseIn()
- pulseInLong()
- shiftIn()
- shiftOut()
- tone()

Time

- delay()
- delayMicroseconds()
- micros()
- millis()

Math

- abs()
- constrain()
- map()
- max()
- min()
- pow()
- sq()
- sqrt()

Trigonometry

- cos()
- sin()
- tan()

Characters

- isAlpha()
- isAlphaNumeric()
- isAscii()
- isControl()
- isDigit()
- isGraph()
- isHexadecimalDigit()
- isLowerCase()
- isPrintable()
- isPunct()
- isSpace()
- isUpperCase()
- isWhitespace()

Random Numbers

- random()
- randomSeed()

Bits and Bytes

- bit()
- bitClear()
- bitRead()
- bitSet()
- bitWrite()
- highByte()
- lowByte()

External Interrupts

- attachInterrupt()
- detachInterrupt()

Interrupts

- interrupts()
- noInterrupts()

Communication

- Serial
- Stream

USB

- Keyboard
- Mouse

VARIABLES

Arduino data types and constants.

Constants

- Floating Point Constants
- Integer Constants
- HIGH | LOW
- INPUT | OUTPUT | INPUT_PULLUP
- LED_BUILTIN
- true | false

Conversion

- byte()
- char()
- float()
- int()
- long()
- word()

Data Types

- String()
- array
- bool
- boolean
- byte
- char
- double
- float
- int
- long
- short
- size_t
- string
- unsigned char
- unsigned int
- unsigned long
- void
- word

Variable Scope & Qualifiers

- const
- scope
- static
- volatile
- Utilities
- PROGMEM
- sizeof()

STRUCTURE

The elements of Arduino (C++) code.

Sketch

- loop()
- setup()

Control Structure

- break
- continue
- do...while
- else
- for
- goto
- if
- return
- switch...case
- while

Further Syntax

- #define (define)
- #include (include)
- /* */ (block comment)
- // (single line comment)
- ; (semicolon)
- {} (curly braces)

Arithmetic Operators

- % (remainder)
- * (multiplication)

- + (addition)
- - (subtraction)
- / (division)
- = (assignment operator)

Comparison Operators

- != (not equal to)
- < (less than)
- <= (less than or equal to)
- == (equal to)
- > (greater than)
- >= (greater than or equal to)

Boolean Operators

- ! (logical not)
- && (logical and)
- || (logical or)

Pointer Access Operators

- & (reference operator)
- * (dereference operator)

Bitwise Operators

- & (bitwise and)
- << (bitshift left)
- >> (bitshift right)
- ^ (bitwise xor)
- | (bitwise or)
- ~ (bitwise not)

Compound Operators

- %= (compound remainder)
- &= (compound bitwise and)
- *= (compound multiplication)
- ++ (increment)
- += (compound addition)
- -- (decrement)
- -= (compound subtraction)
- /= (compound division)
- ^= (compound bitwise xor)
- |= (compound bitwise or)