Arduino Programming

What is Arduino?

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices.

Figure 1: Arduino Logo

IoT (Internet of Things)

The Internet of things describes physical objects with sensors, processing ability, software and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks.

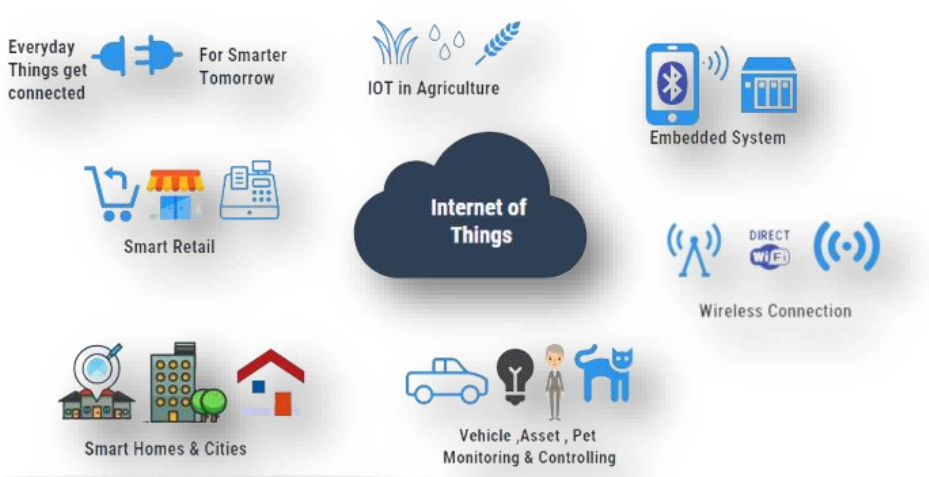

Figure 2: Internet of Things

Arduino Boards

There are a lot of different Arduino boards in the market and some of them are Arduino Mega, Arduino UNO, Arduino Micro, Arduino Pro Mini and Arduino Nano. *(Please refer to the figure 5 for more details.)* The UNO is arguably the most popular Arduino.

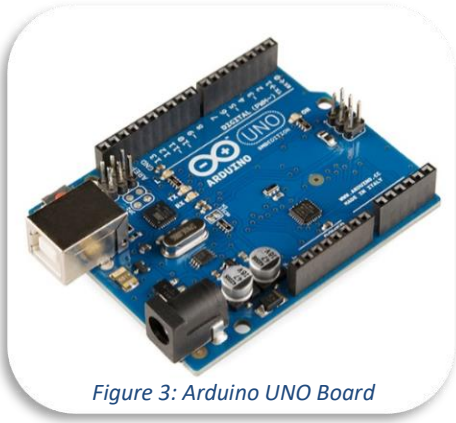

Figure 3: Arduino UNO Board

Arduino UNO

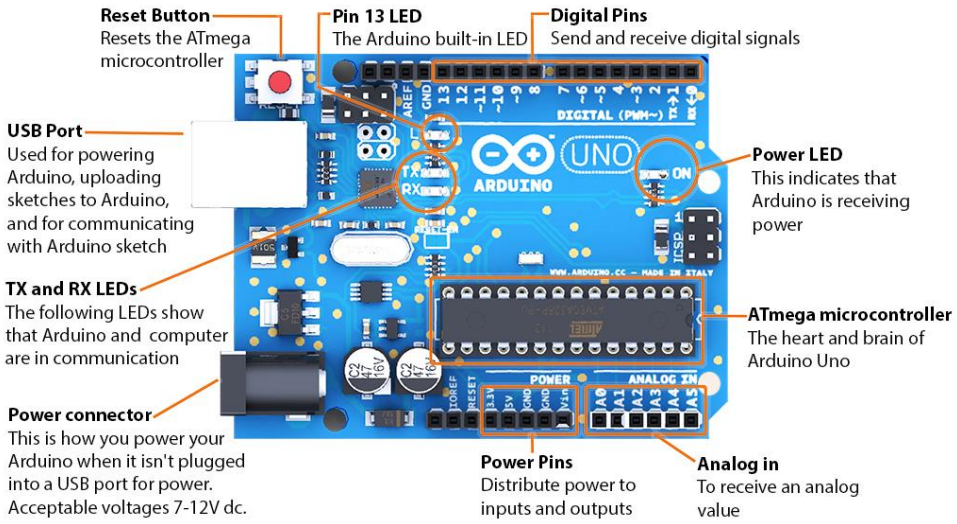

Figure 4: Different Parts of Arduino UNO Board

Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst-case scenario you can replace the chip for a few dollars and start over again.

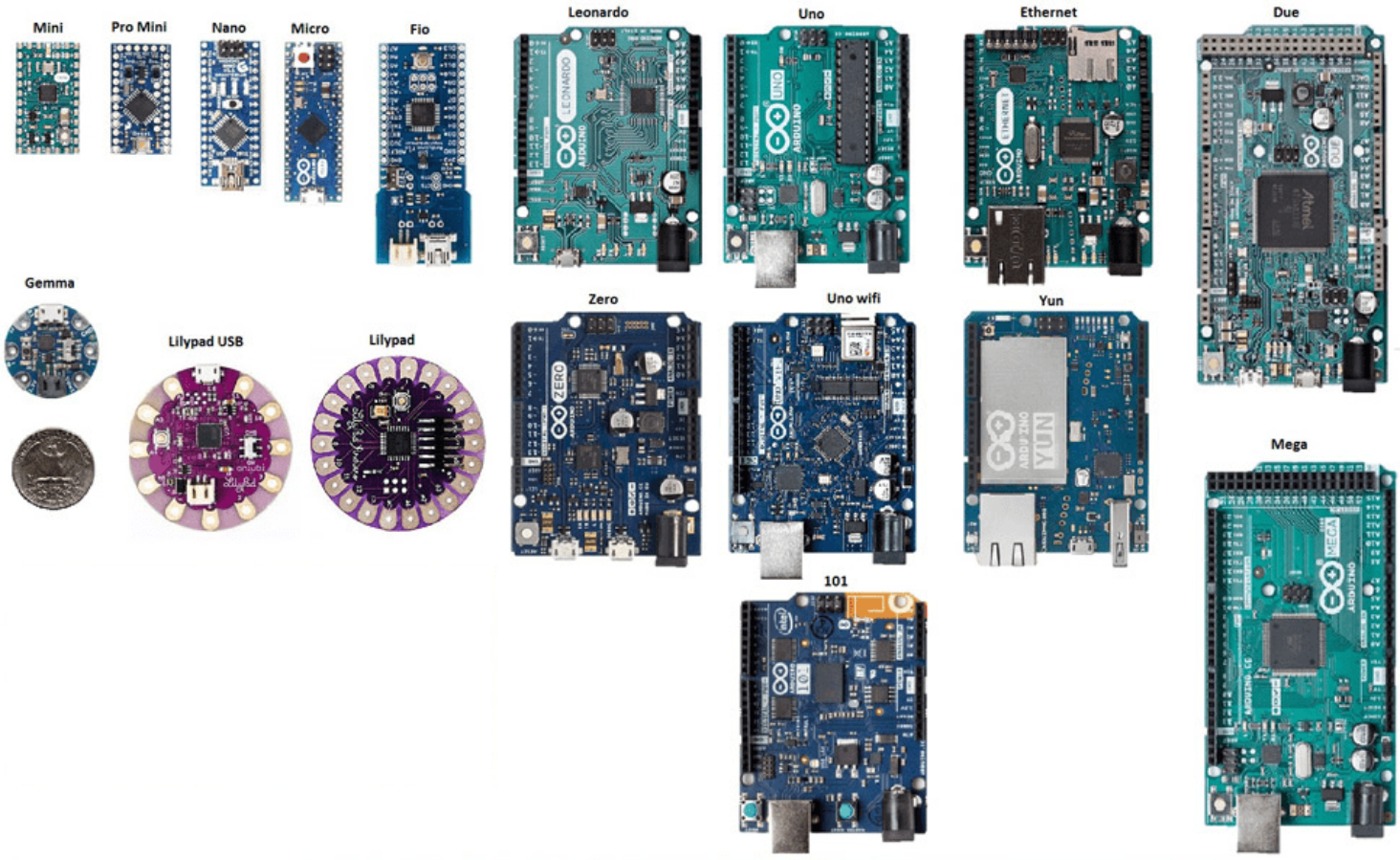

Figure 5: Different Types of Arduino Boards

Arduino IDE

Official Download Link: https://www.arduino.cc/en/software

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

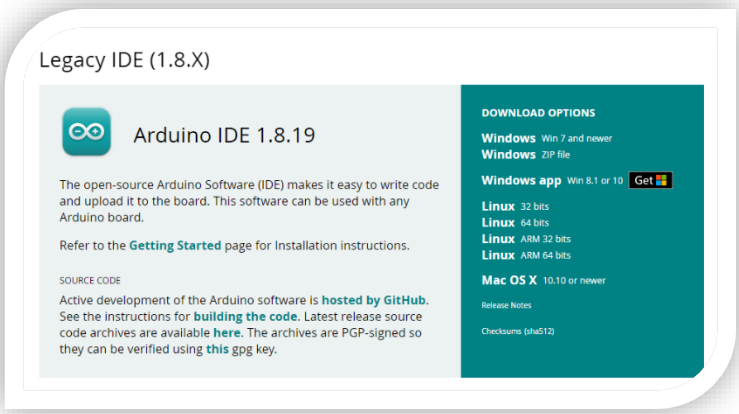

Figure 6: Arduino IDE Download Options

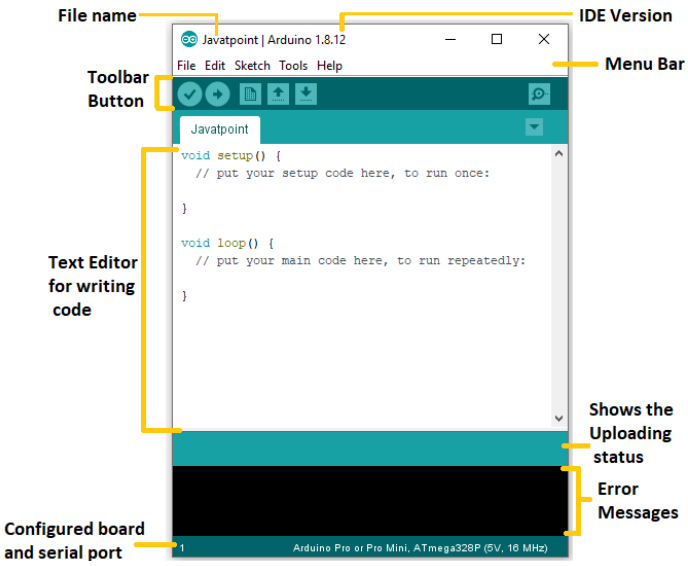

Figure 7: Arduino IDE Interface

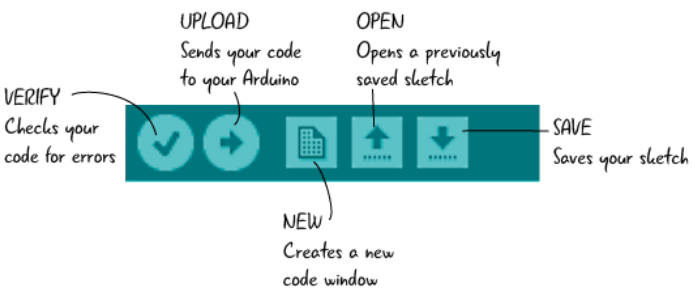

Figure 8: Arduino IDE Toolbar Buttons

void setup() & void loop() Functions

Void setup and void loop are in-built functions that do not return any value. The main difference between void setup and void loop is that void setup runs only once while void loop continuously repeated constantly.

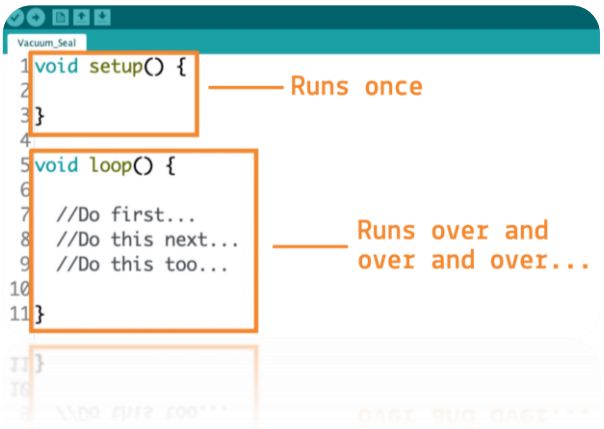

Figure 9: void setup() & void loop() Functions

void setup() Function

It starts with curly opening bracket & close with closing bracket. The void setup function executes only one time as soon as you upload, power up, or reset Arduino. In void setup, there are three things we should define and they are, Serial.begin() and pinMode(). *(You can find out more information about these functions in the next pages.)*

void loop() Function

The void loop function takes no argument inside it. Codes within void loop function repeat consecutively until Arduino is turned off. Whatever we write inside these curly brasses will run again and again. It starts with curly opening bracket & close with closing bracket. Anything inside this function will run from the top of the first line and whatever you write inside this function will be repeated over and over.

Arduino Programming Language Syntax and Program Flow

```
void setup() {
  // Curly braces are used to add scope

  // This is a comment
  // You can add a single line comment by using two slashes

  // int i = 5;
  // Indentation is given by two spaces

  pinMode(3, OUTPUT);
  // Semicolons are used to terminate statements
}

void loop() {
  /*digitalWrite(3, LOW);
  delay(1000);*/
  // slash star and star slash is used to add multiline comments
}
```

Figure 10: Arduino Syntax & Program Flow

Arduino uses its own programming language, which is similar to C++ and it uses a variant of the C++ programming language. The code is written in C++ with an addition of special methods and functions. Syntax in Arduino signifies the rules need to be followed for the successful uploading of the Arduino program to the board. The syntax of Arduino is like the grammar in English. It means that the rules must be followed to compile and run our code successfully.

Curly Braces

In Arduino programming curly braces or important constituents of Arduino sketch. They are the part of two main functions inside Arduino code which are setup() and loop() functions. Curly brackets are not limited to these functions only; they can also be used to define other blocks of code as well.

Semicolon

A semicolon is used at the end of every statement to separate one statement from the next. If a statement does not end with a semicolon it will result in a compile-time error. The error text for forgetting a semicolon is pretty obvious and will include the line number of the statement that is missing it.

Indentation

The reason you indent code is to make it easier for a reader to understand the code's flow. Simply use CTRL+T to automatically indent the selected code. Technically, it is fine to either indent using the tab key or with two spaces.

Comments

Comments are lines in Arduino sketch which help users to understand about how the program works. Comments can be either used for self-understanding or for the others to help them learn code instructions. Commenting is like disabling some lines of Arduino code. Double forward slash is the common way of commenting Arduino code. However, for multiline comments, we can use the second method which is forward slash and a star at the beginning of the comment and another star and a forward slash at the end of the comment. Also don't forget the shortcut key Ctrl+/ for commenting.

Configure Arduino Setup

Once you connect your Arduino to your pc / laptop via a USB connection, open the Arduino IDE and go to Tools menu, you will see an option called “Board”. Select your board name from there and for the “Port”, choose the correct port from the available list.

Figure 11: Configure Arduino Setup

pinMode() Function

The function to configure a pin as IN/OUT using Arduino IDE is pinMode(). This function is used to configure an Arduino pin as an input or as an output. On Industrial Shields equipment's is followed with the corresponding Pin-out. This function is normally used inside the void setup() function.

Table 1: pinMode()	
Syntax	pinMode(pin, mode)
Parameters	pin: the Arduino pin number to set the mode of. mode: INPUT, OUTPUT, or INPUT_PULLUP.
Example	pinMode(5, OUTPUT); pinMode(6, INPUT);

digitalWrite() Function

Write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode() , its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH , 0V (ground) for LOW

Table 2:* digitalWrite*()	
Syntax	digitalWrite(pin, value)
Parameters	pin: the Arduino pin number. value: HIGH or LOW.
Example	digitalWrite(13, HIGH); digitalWrite(12, LOW);

delay() Function

The delay() function allows you to pause the execution of your Arduino program for a specified period. For that purpose, the method requires you to supply it with a whole number that specifies how many milliseconds the program should wait. One second is equal to 1000 milliseconds.

Table 3:* delay*()	
Syntax	delay(ms)
Parameters	ms: the number of milliseconds to pause. Allowed data types: unsigned long.

Example	delay(1000); // waits for a second
	delay(1500); // waits for a second and a half

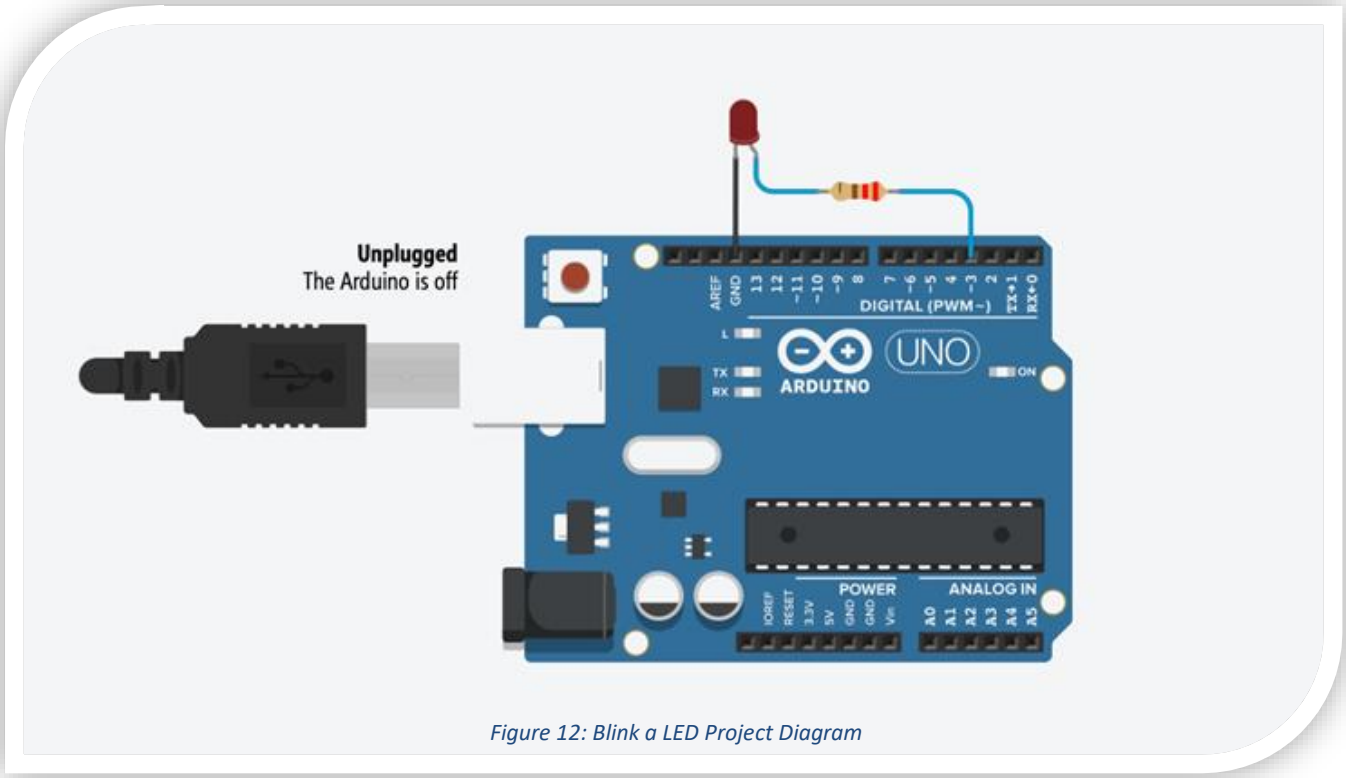

Figure 12: Blink a LED Project Diagram

Table 4: Blink a LED Project Pin Setup	
LED Anode (+)	Digital pin 3
LED Cathode (-)	GND (Ground)

First of all, connect a LED to the Arduino as shown in the Figure 11. The anode (+) of the LED should be in the digital pin 3 of the Arduino and the cathode (-) of the LED must be in one of the GND pins of the Arduino. Then type the codes which you can see in the figure 12 as it is in your Arduino IDE and save the file with a name you prefer. Once you done with the above steps, now it's time to upload your code to the Arduino. If you did all these steps successfully, you will be able to see your LED blinking continuously. After that, you can also try the figure 14 and 15 as well.

sketch_feb22a §

```
void setup() {
  pinMode(3, OUTPUT); // LED anode pin
}

void loop() {
  digitalWrite(3, HIGH); // Turn LED on
  delay(1000); // Wait for one second
  digitalWrite(3, LOW); // Turn LED off
  delay(1000); // Wait for one second
}
```

Figure 13: Blink a LED Project Code

```
void setup() {
  pinMode(3, OUTPUT);
}

void loop() {
  digitalWrite(3, HIGH);
  delay(500);
  digitalWrite(3, LOW);
  delay(100);
}
```

Figure 14: Blink a LED Project Another Code Example

```
void setup() {
  pinMode(3, OUTPUT);
}

void loop() {
  digitalWrite(3, HIGH);
  delay(500);
  digitalWrite(3, LOW);
  delay(100);
  digitalWrite(3, HIGH);
  delay(1000);
  digitalWrite(3, LOW);
  delay(100);
}
```

Figure 15: Blink a LED Project Another Code Example

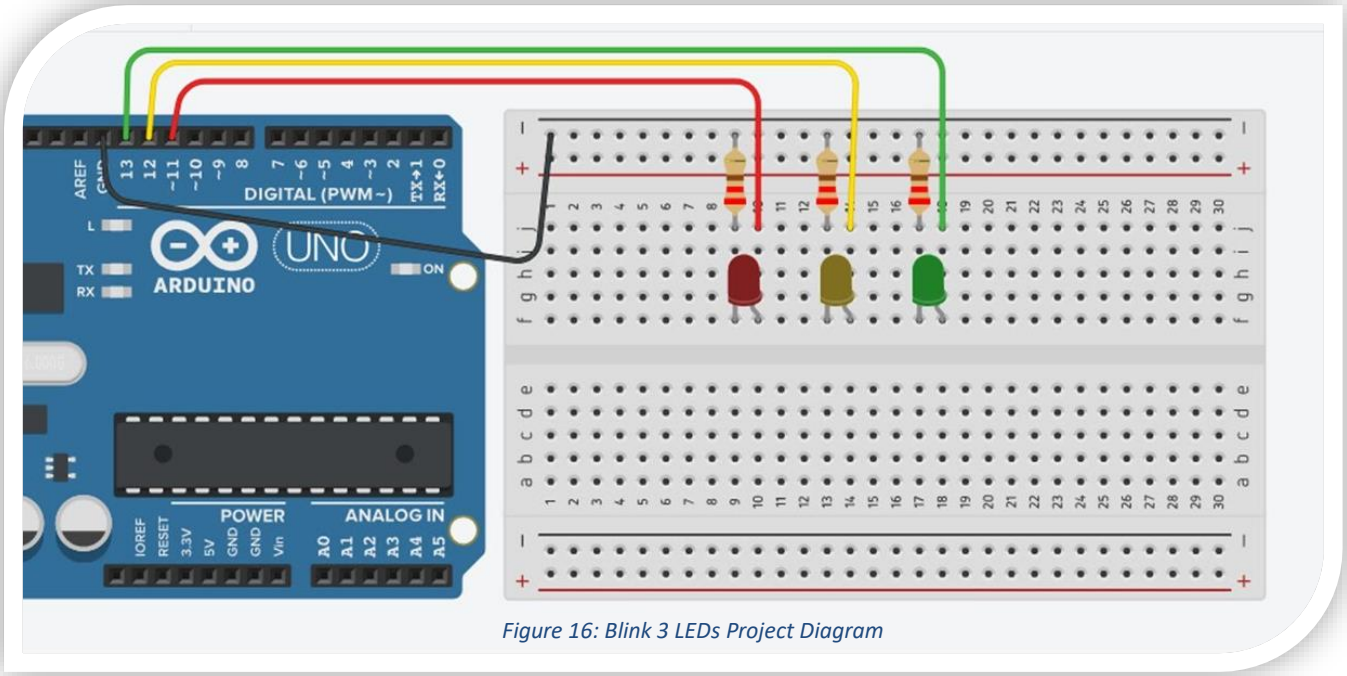

Figure 16: Blink 3 LEDs Project Diagram

Table 5: Blink 3 LEDs Project Pin Setup	
First LED Anode (+)	Digital pin 11
First LED Cathode (-)	GND (Ground)
Second LED Anode (+)	Digital pin 12
Second LED Cathode (-)	GND (Ground)
Third LED Anode (+)	Digital pin 13
Third LED Cathode (-)	GND (Ground)

Once you successfully connect 3 LEDs to the Arduino board as shown in the figure 16 and upload the figure 17 code to your Arduino, you will be able to see those LEDs blinking one by one. You can have some fun by changing the delays and removing some part of the code like “digitalWrite(11, LOW);” and also by adding more of them.

sketch_feb22a §

```
void setup() {
  pinMode(11, OUTPUT); // First LED anode pin
  pinMode(12, OUTPUT); // Second LED anode pin
  pinMode(13, OUTPUT); // Third LED anode pin
}

void loop() {
  digitalWrite(11, HIGH); // Turn first LED on
  delay(1000); // Wait for one second
  digitalWrite(11, LOW); // Turn first LED off
  delay(1000); // Wait for one second
  digitalWrite(12, HIGH); // Turn second LED on
  delay(1000); // Wait for one second
  digitalWrite(12, LOW); // Turn second LED off
  delay(1000); // Wait for one second
  digitalWrite(13, HIGH); // Turn third LED on
  delay(1000); // Wait for one second
  digitalWrite(13, LOW); // Turn third LED off
  delay(1000); // Wait for one second
}
```

Figure 17: Blink 3 LEDs Project Code

That’s all for now!

Thank you for reading so far and if you need any further clarification, please feel free to contact me.

Lecture conducted by,

Vishal Kalansooriya

CEO and the Chief Software Engineer of Webnifix

Undergraduate Software Engineer at SLIIT Academy

Phone: +94771130189

Email: vishalkalansooriya@gmail.com

WhatsApp: https://wa.me/+94771130189

Facebook: https://www.facebook.com/vishal.kalansooriya

RICHMOND ROBOTICS
THE EVOLUTION TO THE FUTURE

For an online version of this handout & the presentation, please visit:
https://github.com/vishal-kalansooriya/arduinoProgramming