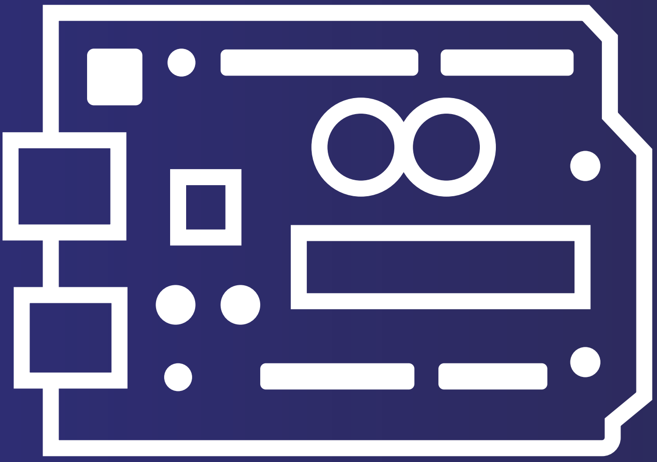
2021

Orange

37 in 1 Sensor Kit

For Beginners



ORANGE KIT

Less hassle , more Fun !

01. Index

02. Index

03. Index

04. Index

Orange 37 in 1 Sensor Kit

Hello Geek Thank you for purchasing our Orange Arduino 37 in 1 sensor Kit. We have specially designed this kit for those people who are interested in learning and Arduino.
After learning this kit, you will understand the following things:
- You will learn what is Arduino.
- You will learn about the active and passive components.
- You Will Understand the use of analog and digital read functions of the Arduino.

1. Arduino Introduction

In this section of this blog, we will talk about the Arduino Development Board.

Arduino is an open source project and was started with the intention of encouraging embedded systems students to learn about micro-controllers.

Before we begin, let me clear one very important doubt which every beginner faces in the initial stage of their learning phase and that is the difference between micro-controller and microprocessor.

Difference Between Micro-controller and Microprocessor

If you search this topic on the Internet you will find a lot of information on this topic but as a beginner to all these things, this information will obviously confuse you.

But don't worry, I have explained that thing in the below section in such a way that it will clear all your doubts and after reading this you will not need to look back on this topic.

So, both micro-controller and microprocessor are things that are designed to control applications, perform logical and mathematical operations.

If both are designed to perform the same operation, what is the difference between these two things? This question may be on your mind.

The difference is, micro-controllers are designed to perform only predefined tasks whereas microprocessors are designed to perform tasks by considering the conditions that occur at the runtime of the process.

The other difference is that if you want to use a microprocessor then you have to interface the memory unit, EEPROM and everything that is essential to perform the tasks. Micro-controllers, on the other hand, don't need all those things because those things are built-in with them.

So, this was about the difference between micro-controller and microprocess.

In the next section we will learn more about Arduino.

What is Arduino?

As discussed earlier, Arduino is a micro-controller that can be used to build small-scale applications. Nowadays, we can see the use of Arduino extensively in the automation industry, there are also a lot of jobs available in the Indian market for Arduino Masters.

The reason for the massive popularity of Arduino is that Arduino is an open-source project and is designed by the community that constantly works on it.

Since a large number of people contribute to this open-source project, Arduino users get quick solutions to their problems and their work never stops.

Varieties in board versions. This is another most important factor of Arduino boards.

The Arduino project was started in 2005. Aiming to provide a low-cost and easy way for novices and professionals.

Arduino software from there or you can download it from their official website.

2. After the download is complete, extract that zip file. You can use WinRAR software to extract the file.

3. After extracting that zip file, you will see a folder. In that folder you will find an .exe file.

4. Right click on that file and select 'Run As Administer' option.

5. When you will click on that option, the system will start installing Arduino software.

6. Follow the instructions asked by the installer and do not make any changes to the settings.

7. Those options are for the user who installed Arduino on their system and they are updating the software or reinstalling the software on their system.

8. In your case, you are installing the software afresh, so you don't need to make those changes in your Arduino installation process.

9. While installing the software, the software installer will ask you to install the driver on your system. You have to allow the installer to install that driver.

10. Those drivers are of Arduino board and if you do not install them on your system then your system will not recognize Arduino board.

11. After this process is over, you will see the Arduino IDE icon on your system's applications list.

Congratulation!! You have successfully installed Arduino IDE on your system.

There is one more driver that you need to install on your system. The name of that driver is CH340G driver and you will find the link to download that software on the product page

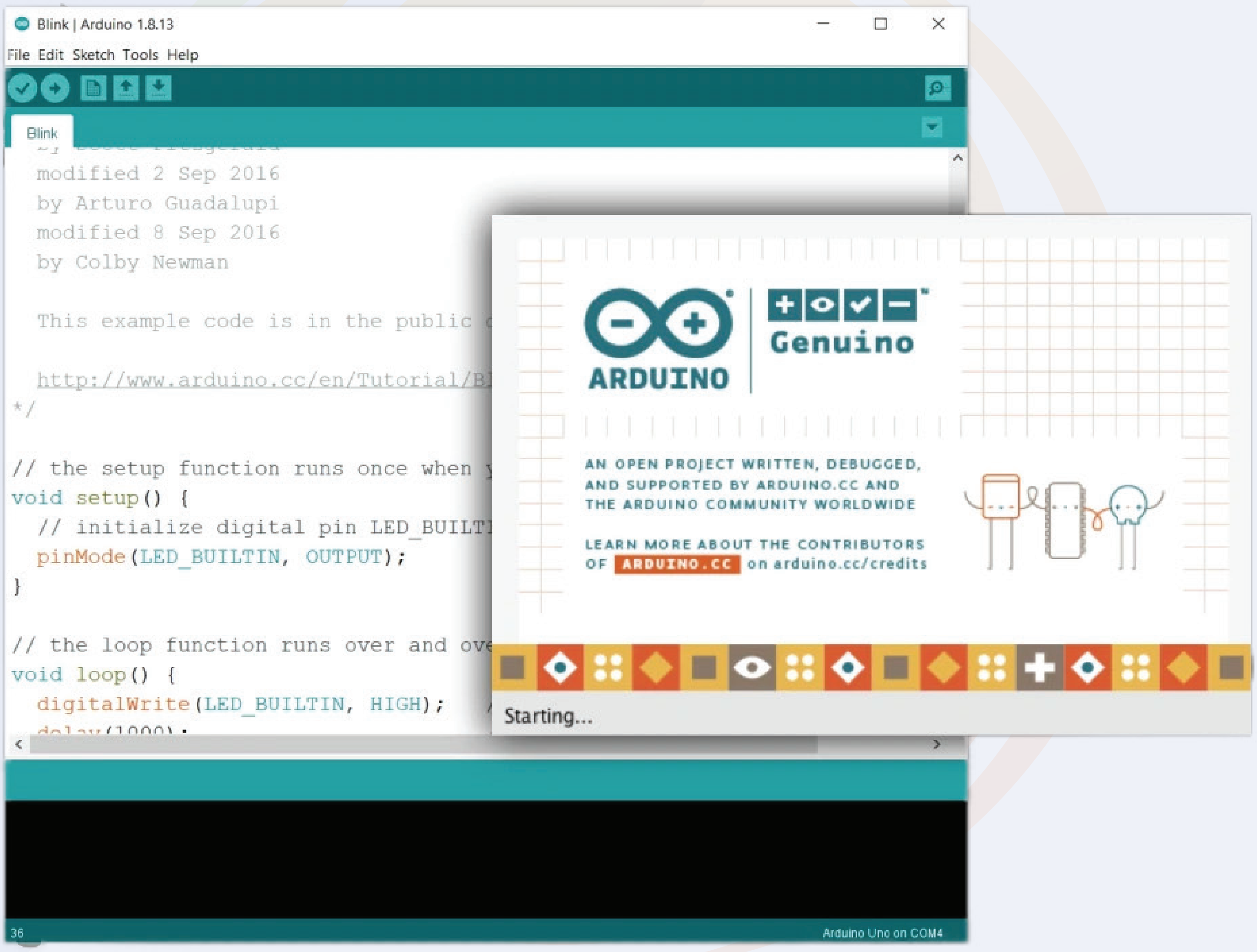
Please note - there is a misconception that boards that contain the ch340G driver are of poor quality but this is not true.

CH340G is just a driver used to transfer information from your computer to the controller board.

It is just a channel between the micro-controller and your computer. I have worked on those boards but never faced any problem.

So stop worrying about this.

In the next part of this blog, we will learn about the UI of Arduino IDE and understand the usage of buttons available in the application.

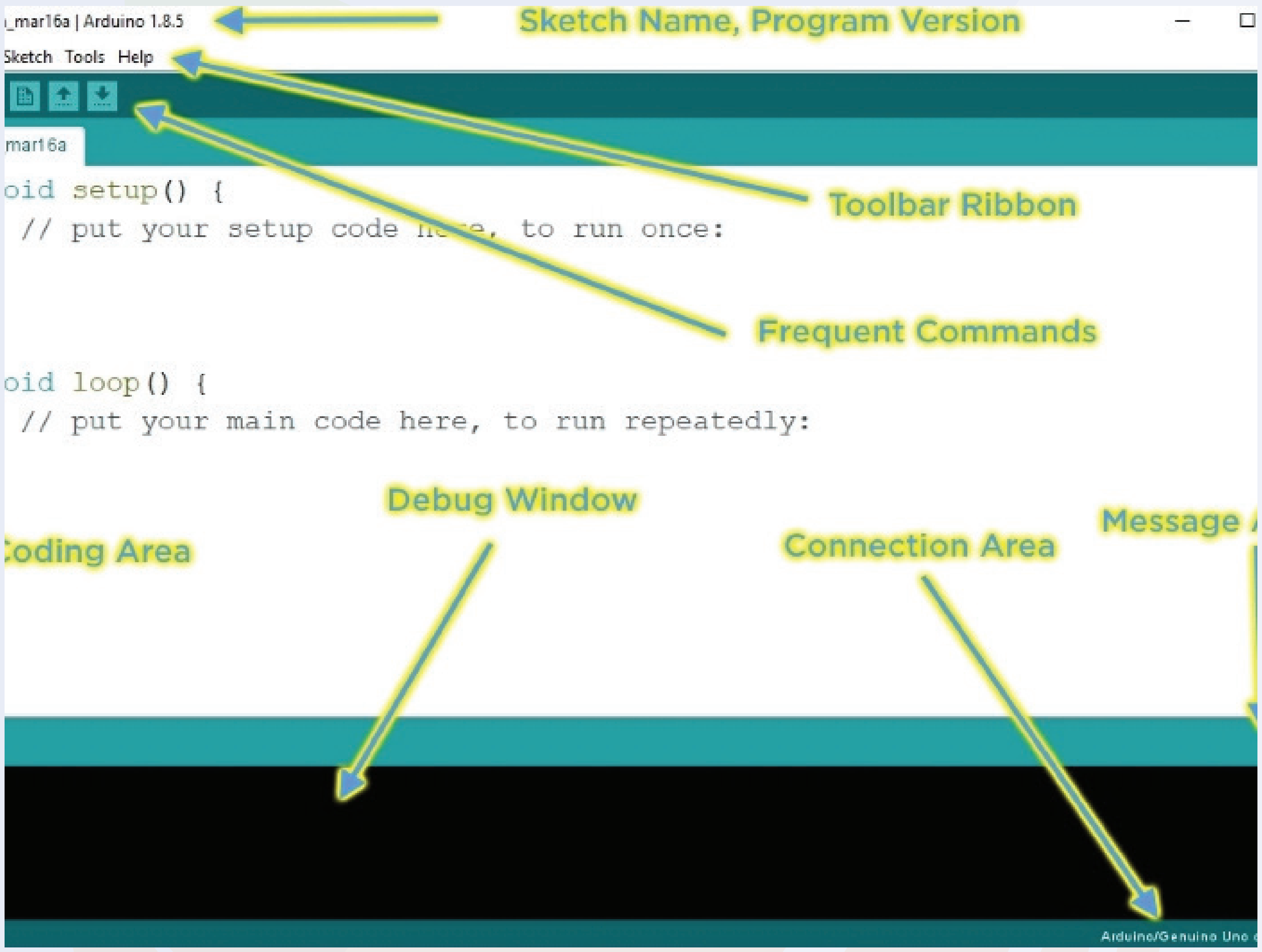

Blink | Arduino 1.8.13
File Edit Sketch Tools Help
Blink
ARDUINO
Genuino
AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE
LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits
Starting...
Arduino Uno on COM4

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

  This example code is in the public

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

// the setup function runs once when y
void setup() {
  // initialize digital pin LED_BUILTI
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and ove
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
```

2.0. Arduino IDE - User Interface

You have installed Arduino IDE on your system, now open it.

You will see a text editor as shown in the image below.



```
_mar16a | Arduino 1.8.5
Sketch  Tools  Help
mar16a
oid setup() {
  // put your setup code here, to run once:

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

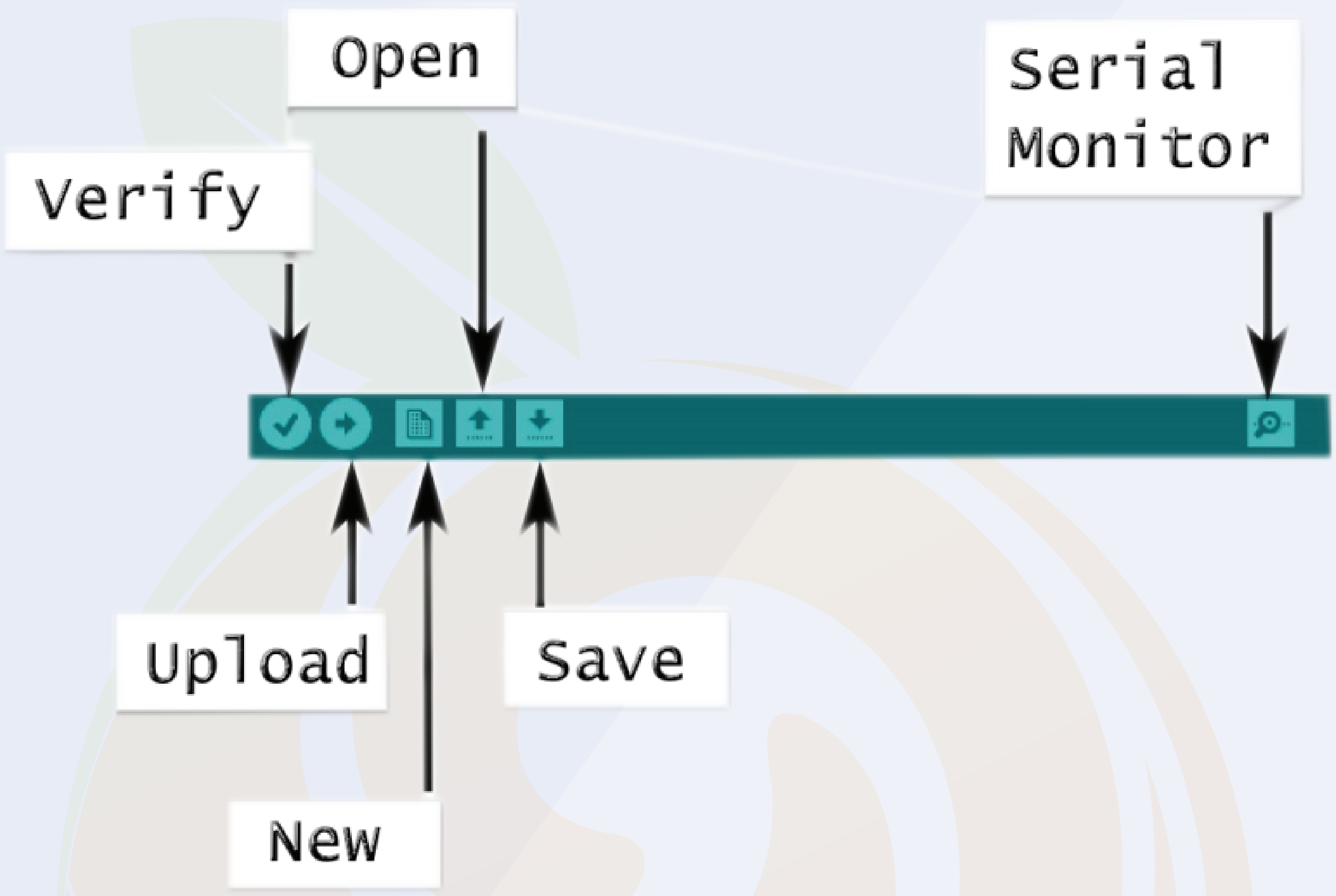
The Arduino IDE looks like this. In the above UI, you get all the functions that are useful for uploading code to your Arduino.

The IDE is divided into three sections. Text editor, output section and menu section.

You write your code in a text editor and you can upload your code to your Arduino with the help of the options available in the menu section.

2.1. Menu Option

Talking about the menu option, in that menu you get the following options.



The menu option looks like this. I have explained the working of those functions int the below sections please have a look.

2.1. Menu Option

Talking about the menu option, in that menu you get the following options.

1. File

- In this option you will get the option to save the file on the system and open the recent files if needed.

2. Edit

- Using this option, you can adjust the settings of your editor.

3. Sketch

- In this section you get the option to add libraries. To check what the library is all about, we request you to check the following section.

4. Tools

- This option is made for the selection of information related to the board. With the help of this option you can either add boards or install new boards in your IDE.

5.

- As the name suggests, you can use this section if you need any help regarding the software or program you have written.

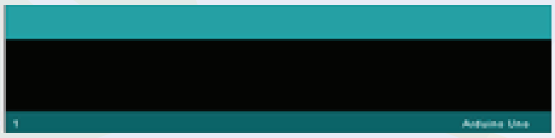
5. Search

- This option is of serial monitor and with the help of this option you can open serial monitor.

If your code is using the serial print function, you will see the code's output

2.1. Menu Option

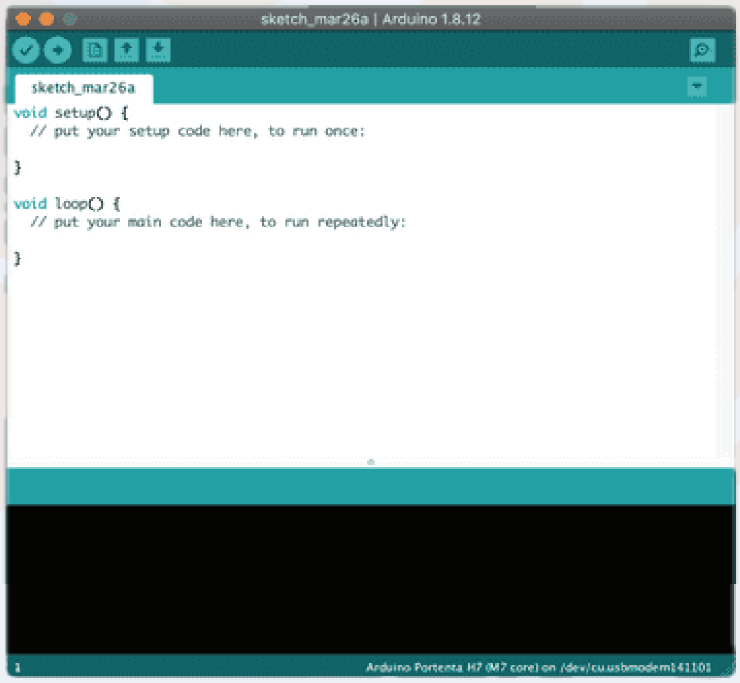
In this option you will see the update of the uploading process of the code. If there is any problem with your code then IDE will generate some error and those errors we can see in this section.



In this option you will see the update of the uploading process of the code. If there is any problem with your code then IDE will generate some error and those errors we can see in this section.

Arduino Text Editor

This section is designed for Arduino Code. In this section, we can write our Arduino.

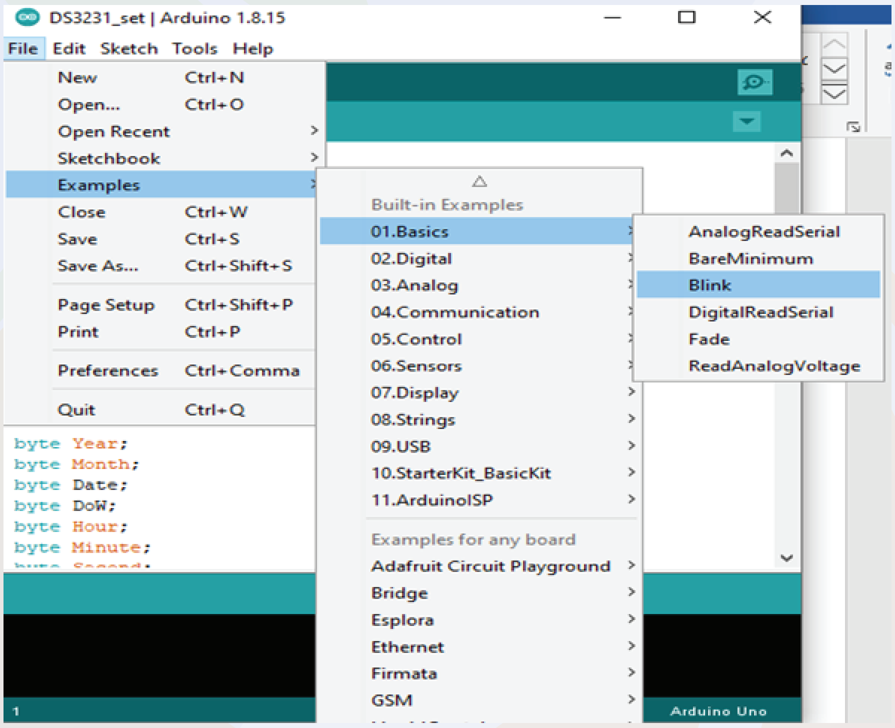

sketch_mar26a | Arduino 1.8.12
sketch_mar26a
1
Arduino Portenta H7 (M7 core) on /dev/cu.usbmodem141101

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

}

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

}
```

2.2. How To Upload The Arduino Code To The Arduino Board?

In the above section we have seen different sections of Arduino IDE. In this section we will use those sections of the Arduino IDE to upload our first Arduino code to the Arduino board.

We will try to upload the blink code to the board. We will get the code from file section.
Please see the following image to understand the process.



```
byte Year;
byte Month;
byte Date;
byte DoW;
byte Hour;
byte Minute;
```

When you will click on the blink option shown in the image, a new Arduino tab will open and in that tab, we will find the Arduino code, which we can use to toggle the 13-number pin.

So, now that we have the code, we are uploading this code to the Arduino and for uploading you can press the ctrl+U button on the keyboard to start the process of uploading the code.

So this is how we can upload the code to Arduino. In the next part of this blog, we will learn how to add libraries to that Arduino IDE?

3. What is an Arduino Library?

The Arduino library is a combination of code that has been written by an Arduino contributor.

The main purpose of designing libraries is to reduce code redundancy.

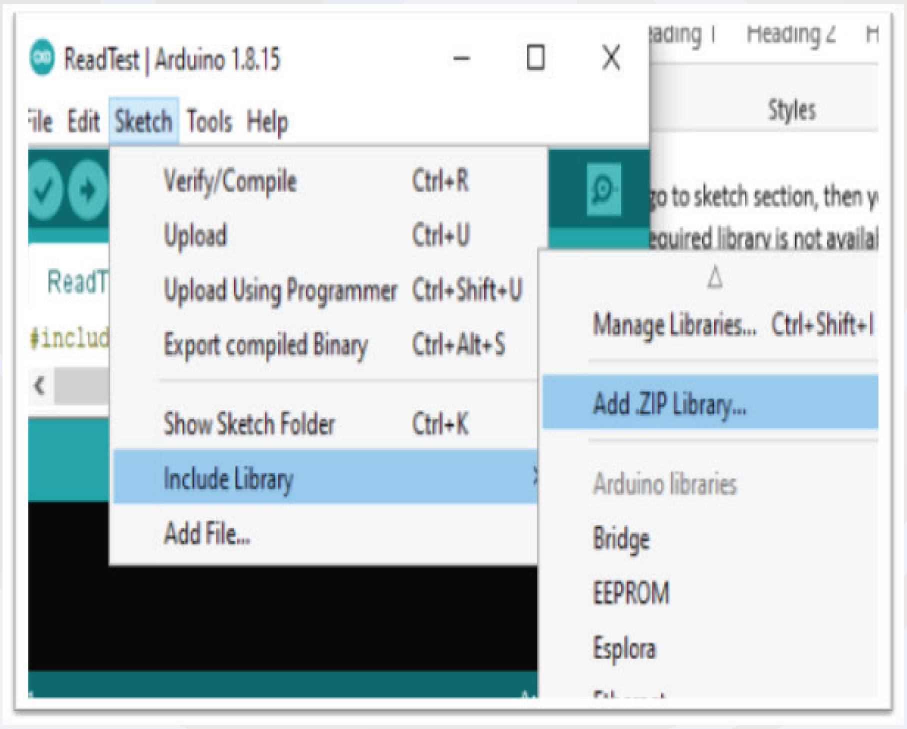
We use Arduino libraries, that means we use objects and methods of classes. So, even though we don't need to write much code, we can learn a lot by modifying existing code.

We can learn oops concepts, functional programming and much more.

So, that was about introduction to Arduino library, if you have any doubt you can contact me at info@robu.in.

How to Add an Arduino Library to the Arduino IDE?

There are two ways of adding library, either you can go to sketch section, then you can add the required library by clicking on the add library option or if the required library is not available on Arduino then we get ‘Add Zip Library’ option in that section.



ReadTest | Arduino 1.8.15
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
Bridge
EEPROM
Esplora

3.1. What is an Arduino Library?

I have faced this issue many times, I never got my required library in the Arduino IDE. I always used to download the zip files of the library from the Internet and then used to add that to the Arduino with the help of the 'Add zip Library' Option.

I hope the above section has cleared all your doubts about what is Arduino library and how to add a new Arduino library to the Arduino?

Arduino is an open source platform supported by huge community that continuously contributes to help others. So do not worry because the community will be always there to help

Void setup() is used for initializing the pins. This function executes once only.

If you are using a sensor that needs calibration before running, then you can do all those calibration settings in this section.

In Arduino, We use pinMode built in function to initialize the pins of the Arduino.

This function takes two parameters, pin number and the mode of operation. (Either output or input).

```
For Example: pinMode(12, INPUT);
             pinMode(12, OUTPUT);
```

Please check the above examples. In the above examples, the first line is defining 12 number pin as an input pin and the second line is defining the 12 number pin as an output pin.

Let me tell you a simple logic here, if you are defining any pin as an output pin that means you are putting some voltage on that GPIO pin or sending out some data.

Whereas if you are defining any pin as an input pin means, you are taking some

3.2. Void Loop And Void Setup

Voidloop()

This is one more inbuilt function that is used in the Arduino IDE. This loop keeps continuously running until someone don't turn Off the Arduino.

We can write our code-cases in this loop and those cases will keep on running.

Analog Write And Analog Read

Analog read and analog write these are the two functions that deals with the analog values.

AnalogWrite function is used to write the analog values whereas analogRead function is used to read the analog values that we pushing on the GPIO pin of the Arduino.

Ex, analogWrite(1, pwm)
    analogRead();

The analogwrite function take two parameters, pin number and pwm value.

The pwm value could be in range of 0 to 255.

Whereas the analogread function only take one parameter and that is pin number.

Store = analogRead(A1);

In the above example the analogRead function is reading the the data which is coming on the pin number A1 and then storing the information in the store variable.

So, this was about the analogRead function. This function is used when we are required to work with the variable voltages.

In next section of this blog we will learn about the digitalRead and digitalWrite function.

3.3. What is an Arduino Library?

I have faced this issue many times, I never got my required library in the Arduino IDE. I always used to download the zip files of the library from the internet and then used to add that to the Arduino with the help of the 'Add zip Library' Option.

I hope the above section has cleared all your doubts about what is Arduino library and how to add a new Arduino library to the Arduino?

Arduino is an open source platform supported by huge community that continuously contributes to help others. So do not worry because the community will be always there to help

Void setup() is used for initializing the pins. This function executes once only.

If you are using a sensor that needs calibration before running, then you can do all those calibration settings in this section.

In Arduino, We use pinMode built in function to initialize the pins of the Arduino.

This function takes two parameters, pin number and the mode of operation. (Either output or input).

```
For Example: pinMode(12, INPUT);
             pinMode(12, OUTPUT);
```

Please check the above examples. In the above examples, the first line is defining 12 number pin as an input pin and the second line is defining the 12 number pin as an output pin.

Let me tell you a simple logic here, if you are defining any pin as an output pin that means you are putting some voltage on that GPIO pin or sending out some data.

Whereas if you are defining any pin as an input pin means, you are taking some

3.4. Void Loop And Void Setup

Voidloop()

This is one more inbuilt function that is used in the Arduino IDE. This loop keeps continuously running until someone don't turn Off the Arduino.

We can write our code-cases in this loop and those cases will keep on running.

Analog Write And Analog Read

Analog read and analog write these are the two functions that deals with the analog values.

AnalogWrite function is used to write the analog values whereas analogRead function is used to read the analog values that we pushing on the GPIO pin of the Arduino.

```
Ex, analogWrite(1, pwm)
    analogRead();
```

The analogwrite function take two parameters, pin number and pwm value.

The pwm value could be in range of 0 to 255.

Whereas the analogread function only take one parameter and that is pin number.

```
Store = analogRead(A1);
```

In the above example the analogRead function is reading the the data which is coming on the pin number A1 and then storing the information in the store variable.

So, this was about the analogRead function. This function is used when we are required to work with the variable voltages.

In next section of this blog we will learn about the digitalRead and digitalWrite function.

3.4. Void Loop And Void Setup

Digital Read And Digital Write

DigitalRead and digitalWrite functions are used for reading and writing digital values.

Talking about the digitalWrite function, this function takes two parameters, pin number and HIGH/LOW String.

When we give HIGH parameter to the function, the function will put HIGH level signal on the GPIO pin of the Arduino and when we put LOW then the function will produce LOW level signal on the GPIO pin.

For Example,

```
digitalWrite (12, HIGH);
digitalWrite (12, LOW);
```

In the above line of code, the first line of code producing HIGH level signal on the 12 number pin and in the second line code the function is producing LOW level signal on the 12 number pin.

Talking about the digitalRead function, this function takes one parameter and that is pin number.

For example,

```
Store = digitalRead(12);
```

In the above function, the code is running the data of the pin number 12 and storing the received input in the ‘store’ variable.

4. I2C Communication

4.1 Introduction

So, in this section of this blog we will learn about the I2C communication Protocol.
We have already discussed the basic things of the I2C communication. In this section, we will learn explore this topic a little but more.

I2C communication was developed by Philips company. If we compare this communication protocol with the SPI and UART communication protocol then I2C communication is the fastest communication protocol compared to other communication protocols.

Till now we understood what is I2C communication protocol and the importance of it but do we know How it works?

You may have used the I2C communication before this, if not, don't worry, I am explaining everything from the scratch. And this section will cover all your doubts.

Before we talk about the technical things of the I2C communication, let me explain you this thing is laymen term first.

We understood the basics of the I2C communication, we know with the help of the I2C communication, we can connect multiple devices to the same line and can communicate with all those devices without any interference of the signals.

But do you know how this is possible? I2C communication uses address methodology. The slaves that are connected to the master has unique address.

The master uses that unique address to communicate with the slave devices that are connected to it.

Let's say we have connected 3 slave devices to the master and now master wants to transfer the data to the third slave device.

4.1 Introduction

In that case, the master will store the I2C address of the device and will connect to the device which I2C address matches with the I2C address the master has. After connecting with the I2C device the master and slave will start the transfer of the data.

So, this is how the I2C communication works.
In the next section, we will learn about the technical details of the I2C communication.

4.2 Technical Details Of the I2C communication

In the previous section, we learned about the I2C communication. In this section, we will learn about the technical aspect of the I2C communication.

So, we know that the master device is a main component of the I2C communication. In I2C communication protocol, the master is the first component which initiates the communication.

The slave devices that are connected to the master device, waits for the request from the master device and when they receive the request from the master, they either sends the data to the master or receives the data from the master.

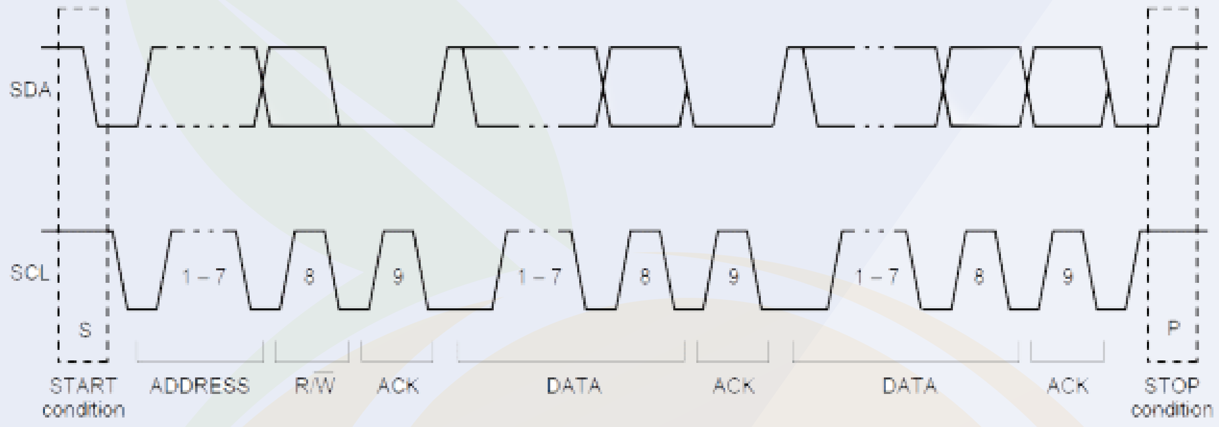
In I2C communication, the slave devices cannot start the communication at the first place and also cannot talk to the slave devices that are present in the network.

Starting the communication and collecting the data from the slave devices is the job of the I2C master.

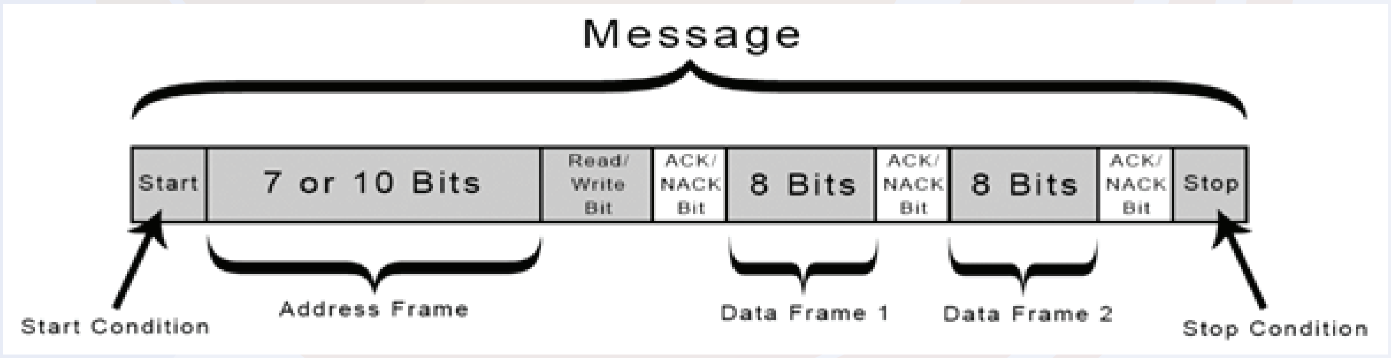
4.3. I2C Waveform – Explained

The Following image shows the waveform of the I2C Data communication. Before reading the following section, I request you check the following image.

4.3. I2C Waveform – Explained



Ok, I believe you have checked the above the image, now I want you to check the following image also.
Please do not skip taking look at these images, if skip these images then you will not understand whatever I will tell you in the coming sections of this blog.



4.3. I2C Waveform – Explained

The above image shows the message bit. The data that will be transferred from master to the slave devices will be broken up in to the messages.

Each of these messages contains the above shown information.

The information is shown in the above image is responsible for the successful transfer of the data from slaves to the masters or vice versa.

1) Start Condition –
2) Address Frame –
3) Read Write Bit –
4) ACK or NACK Bit –
5) Data Frame –
6) Stop Condition –

1) Start Condition –

This is first step of starting the communication between master and slave. At this stage, the master will make the SDA line low before the SCL line.
This signifies the start of the I2C communication.

2) Address Frame –

We know that every slave has its unique I2C address, at this this stage, the master will put the I2C address of the device in the message to which it wants to connect with.

3) Read/Write Bit –

This is a second important bit that will be sent by master. Based on this bit, the master reads or writes the data to the slave devices.

4) Data Frame –

After receiving the read / Write bit from the master the data transfer between the slave device and master starts.

6) End Condition –

This is final stage of the data transfer. In this stage, the master will make the SDA line low before the SCL line goes LOW.

Orange 37 in 1 Sensor Kit

Hello Geek, In this blog, we will talk about Orange 37 in 1 sensor kits. This kit is compatible with all micro-controllers and microprocessors.

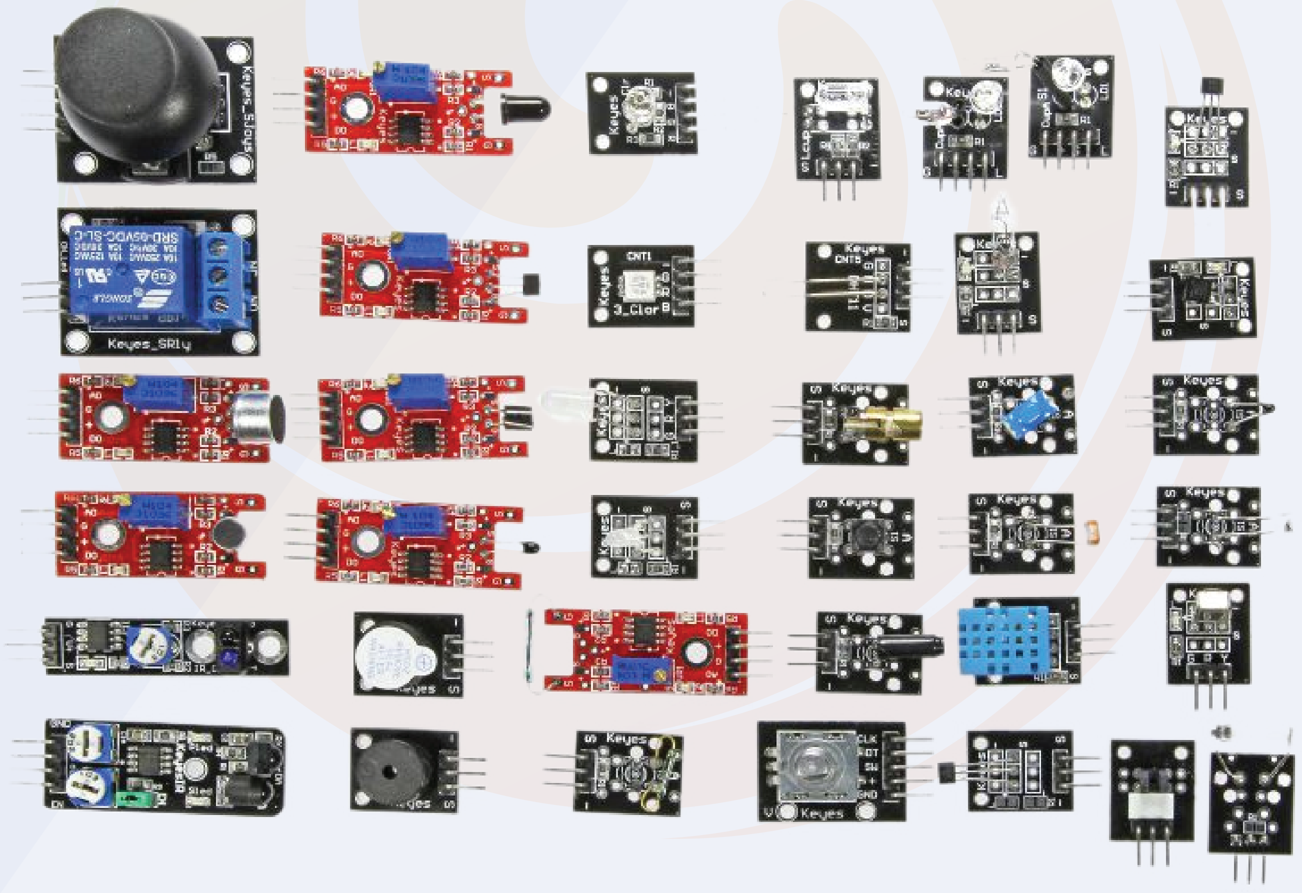
The only concern is the power supply. The sensor you get with this kit has an operating voltage of 5v.

So, if your controller is not capable of handling the 5V input voltage, I suggest you use either a voltage level converter or a resistor divider circuit.

The sensors you get with this kit are compatible with all types of Arduino boards. As I told you earlier you just need to take care of the operating voltage of the sensor.

If you have not bought any Arduino board yet then we request you to buy an Arduino board from here.

The following image shows the list of components we get with this kit. Please take a look



So, that was the introduction part of the orange sensor kit. In the next part of this blog, we will understand the working and interfacing of these sensors with the Arduino board.

Hall Effect Sensor – Orange 37 in 1 Sensor Kit

With this Orange 37 in 1 sensor kit, you are getting three types of hall effect sensors. The working principle of those three sensors is the same but the output of those sensors is slightly different.

The hall effect sensors we are getting with this kit are as follows:

1. Linear Hall Effect Sensor
2. Switch Hall Effect Sensor
3. Hall Effect Sensor with Differential Amplifier

These are the three types of sensors that you are getting with this orange 37 in 1 sensor kit.

If we talk about the output type of these sensors, then the output of linear hall effect sensor is analog and the output of switch hall effect sensor is digi-tal.

Talking about third hall effect sensor, we get two types of output analog output and digital output in that sensor.

This was about the hall effect sensor that we are getting with this kit.

In the next part of this blog, we will talk about interfacing Hall Effect sensor with Arduino.

Interfacing of the Hall Effect sensor with the Arduino:

You will need the following components to understand the working of the Hall Effect Sensor.

Components List

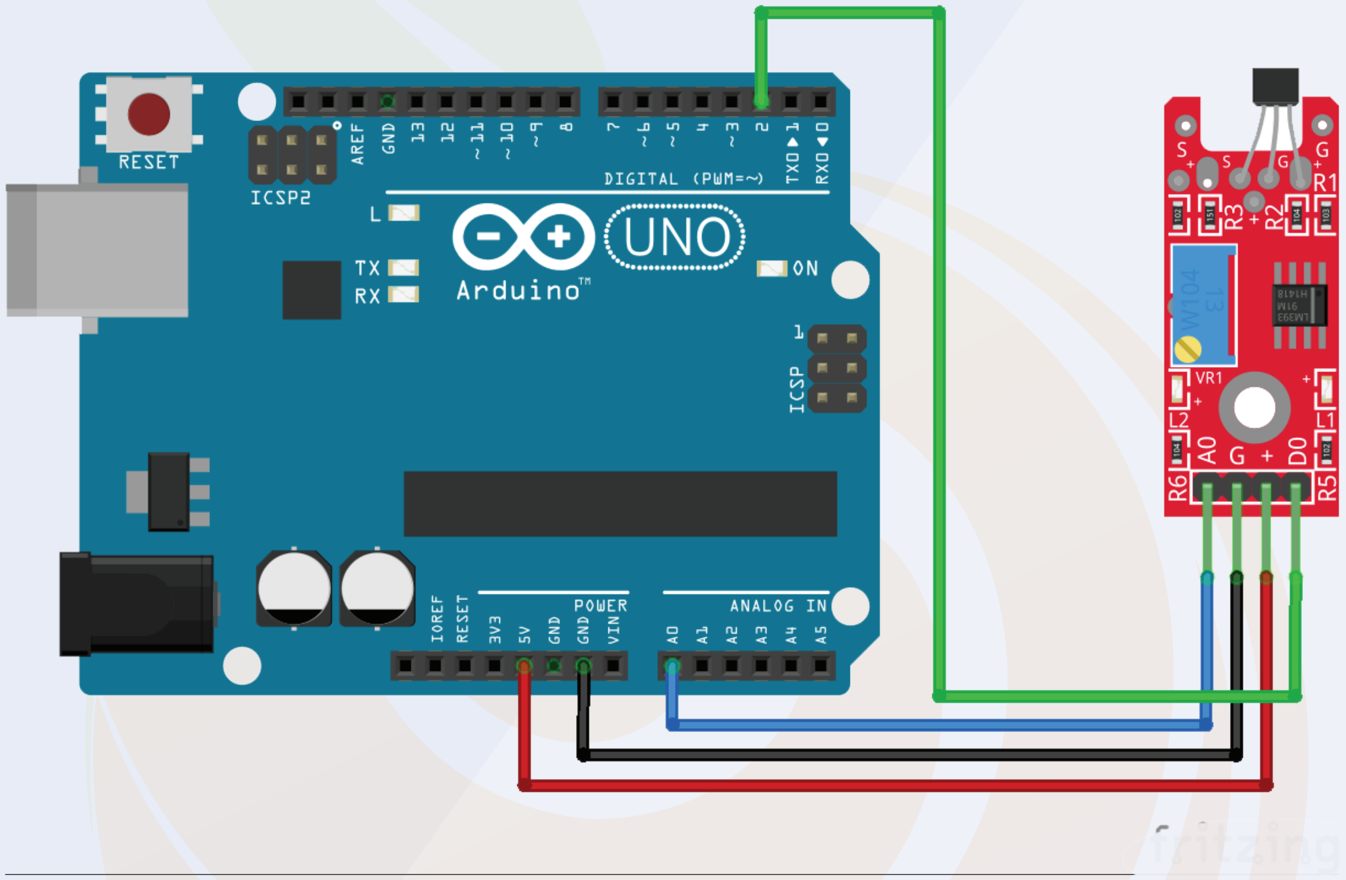
- Connecting Cable
- Arduino Board
- 5v Power Supply
- Breadboard

Interfacing The Hall Effect Sensor

As I told you earlier that the working of these three hall sensors is same, the only difference is that the output type of these sensors is different.

So, we have to make the connection according to the output type of the sensor.

I have shared the connection diagram below, please have a look and if you have any doubts please let us know.



4.1. Arduino Code For Orange Kit Hall Effect Sensor

Programming for the Orange Kit Hall Effect sensor is very easy. Since the output of these sensors can be either digital or analog, we just need to connect the signal pin of the sensor to the correct GPIO pin of the Arduino.

If you are using a linear Hall sensor or a number third sensor, you will need to connect the signal pin of that sensor to the analog pin of the Arduino.

4.1. Arduino Code For Orange Kit Hall Effect Sensor

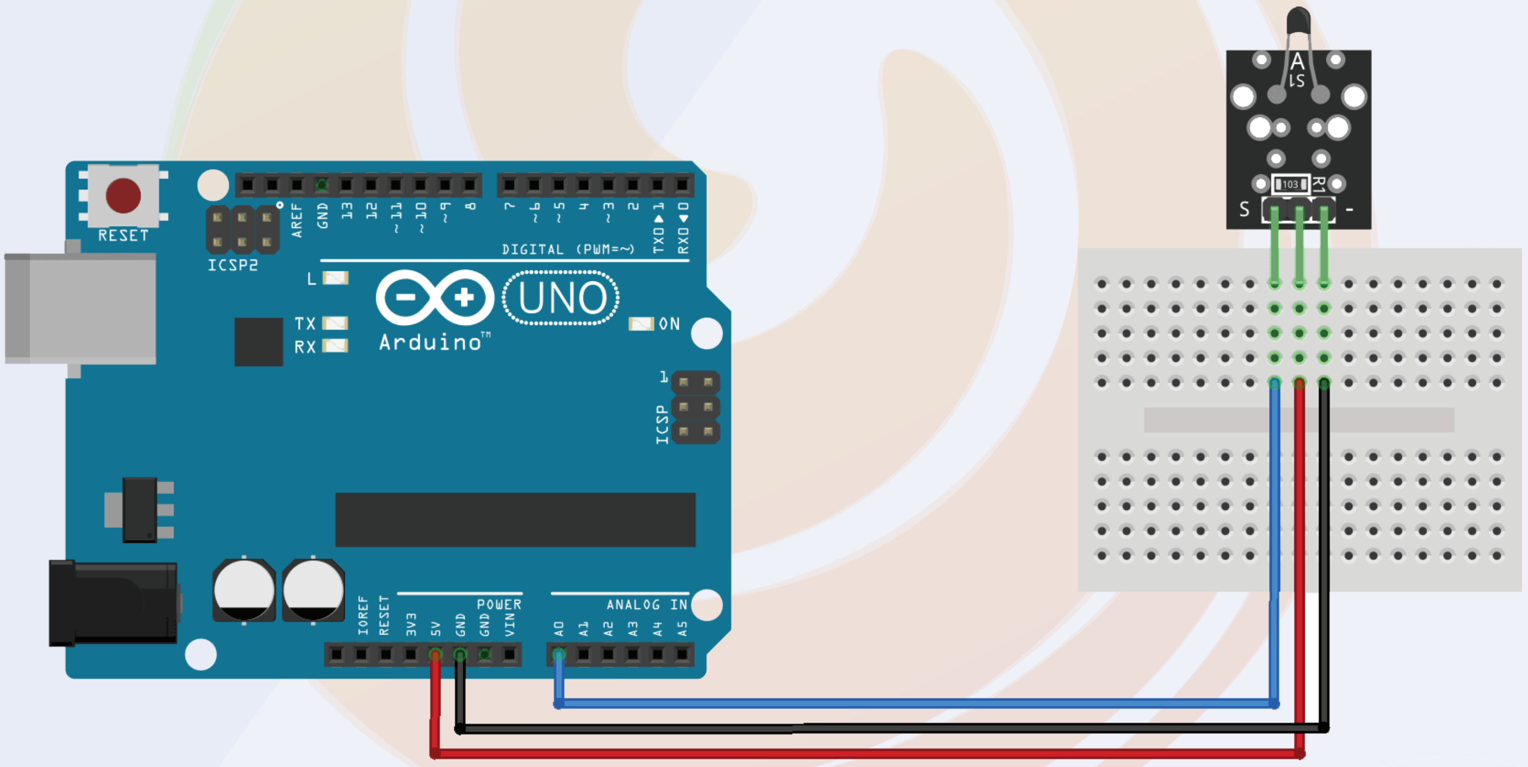
Sensor number three has four pins, two of those four pins are signal pins. You will find the sensor output on those pins.

You don't need to connect all those two pins to Arduino. You just need to connect the analog or digital pins to the Arduino. The selection of this PIN will depend on the requirement of your application.

If your application requires analog reading from the sensor then you can connect the hall effect sensor's analog output pin to Arduino and if your application requires the sensor's digital output as input then you can connect the sensor's digital pin Can connect to Arduino.

Talking about the switch type of Hall Effect sensor, the output type of this sensor is digital. So we have to connect this pin to any digital GPIO pin of Arduino.

4.2.1. Analog Hall- Effect Sensor.



Since the output type of the sensor will be either analog or digital, we can use either the analog read or digital read function to read the output of the sensor.

I have shared two code snippets below; One code snippet is for detecting analog output and the other is for digital output.

You can use either of these two snippets as per your application requirement.

4.2.1. Analog Hall- Effect Sensor.

```
const int hall_Sensor=2;
int inputVal = A1;

void setup()
{
  pinMode(13, OUTPUT);         // Pin 13 has an LED connected on most Arduino
boards:
  pinMode(hall_Sensor,INPUT);    //Pin 2 is connected to the output of proximi-
ty sensor
  Serial.begin(9600);
}

void loop()
{
  if(digitalRead(hall_Sensor)==HIGH)      //Check the sensor output
  {
    digitalWrite(13, HIGH);   // set the LED on
  }
  else
  {
    digitalWrite(13, LOW);    // set the LED off
  }
inputVal = digitalRead(hall_Sensor);
Serial.println(inputVal);
delay(1000);             // wait for a second
}
```

4.2 Digital Hall Effect Sensor

```
volatile byte half_revolutions;
 unsigned int rpm;
 unsigned long timeold;
 void setup()
 {
   Serial.begin(115200);
    attachInterrupt(0, magnet_detect, RISING);//Initialize the int-
terrupt pin (Arduino digital pin 2)
   half_revolutions = 0;
   rpm = 0;
   timeold = 0;
 }
 void loop()//Measure RPM
 {
   if (half_revolutions >= 20) {
     rpm = 30*1000/(millis() - timeold)*half_revolutions;
     timeold = millis();
     half_revolutions = 0;
     //Serial.println(rpm,DEC);
   }
 }
 void magnet_detect()//This function is called whenever a mag-
net/interrupt is detected by the arduino
 {
   half_revolutions++;
   Serial.println("detect");
 }
```

5. 2 Axis Joystick

We all used joystick in our childhood. The kind of Super Mario games we used to play when we were kids were controlled by a joystick controller. Plus, the electronic toy cars you used to play with as a child were controlled by joysticks.

So, as you have already used the joystick, it will be so much easier for you to understand the working of the joystick.

The joystick has two potentiometers. When we move the joystick, the position of the potentiometer changes.

Change in the position of the potentiometer means that the resistance of the potentiometer changes. Therefore, as the resistance is changing, the output voltage of the potentiometer also changes.

The output of those potentiometers is connected to the microcontroller.

When the microcontroller receives input from those potentiometers, the microcontroller takes further necessary action based on the logic that you have added to your code.

Coming back to our discussion, that joystick module has four output pins. We will discuss more about the interfacing diagram in the below section.

5.1. Interfacing the Joystick Module With the Arduino

The following components you may require to understand the working of the sensor.

5.1.1. Component List

- Arduino Board
- Connecting Cables
- Breadboard

5.1. Interfacing the Joystick Module With the Arduino

5.2. Arduino Code For Joystick Module

I have explained the function of those pins in the below section, please take a look.

1. X - This pin generates the analog output voltage. (0-255) and is used to monitor the movement of the joystick on the X-plane.

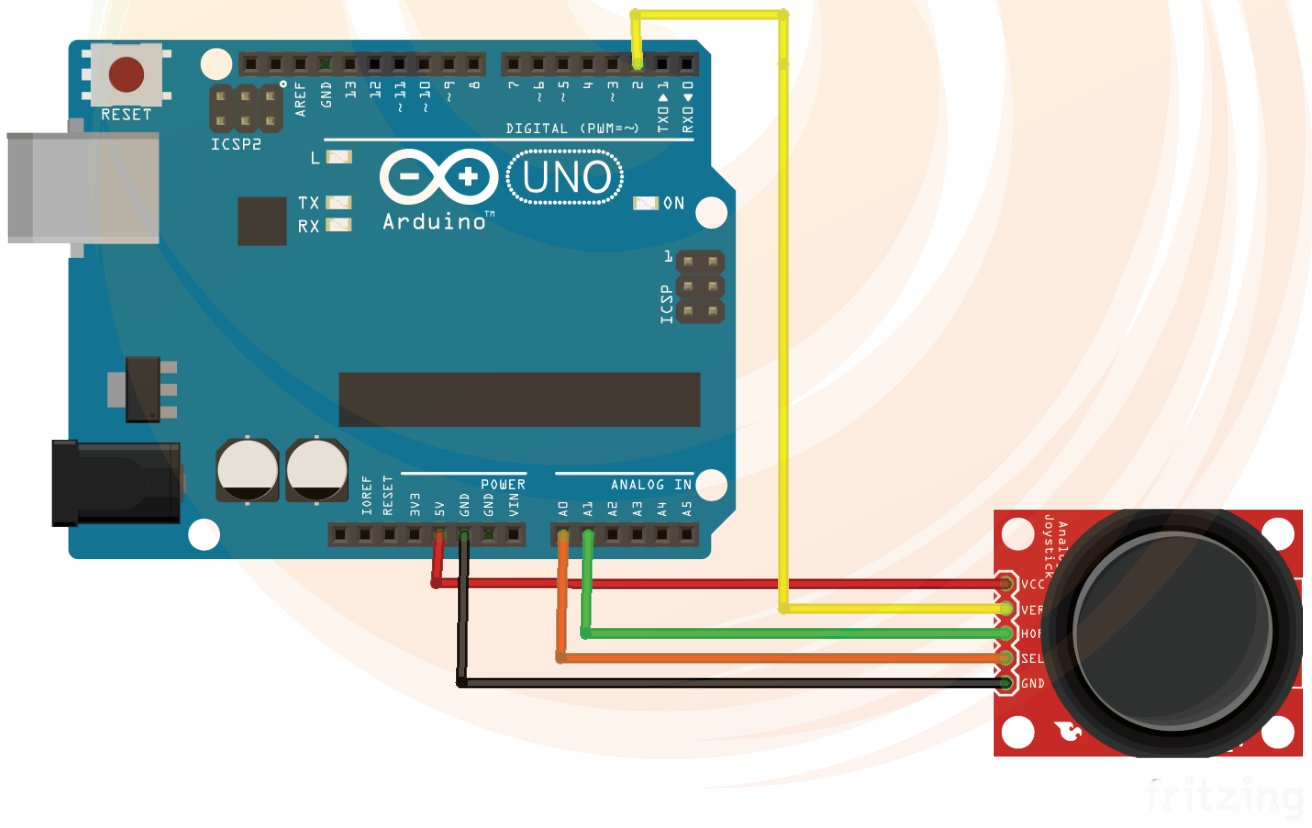
2. Y - This pin also generates the analog output voltage and is used to monitor the movement of the joystick on the Y-plane.

3. Switch - This pin is the output of the switch placed under the joystick. When we press the joystick, this pin generates high voltage. Sometimes this switch is also used to locate the Z-axis of the joystick.

Now that you know the function of each pin, we can now connect the pins of the joystick to the Arduino.

Analog pin of module pin 'X' and pin 'Y' You can connect analog pin of Arduino and digital pin of module to digital pin of sensor.

Please refer to the following image to understand the connection diagram.



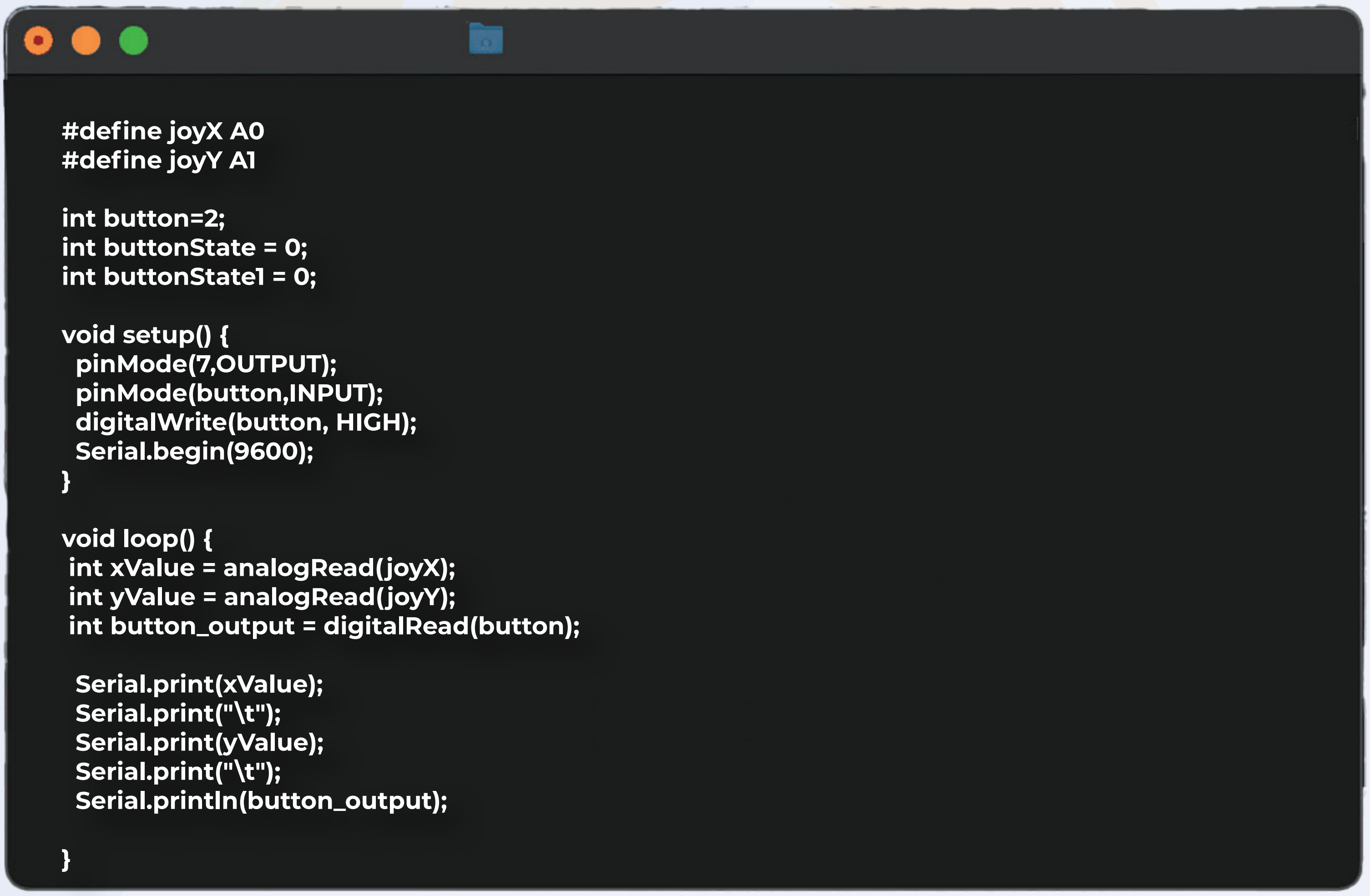
5.2. Arduino Code For Joystick Module

The joystick's output is either analog or digital. Therefore, we can use the analog read and digital read functions to read the output.

Since the joystick's output is in simple format, we don't need to install any library to work with this board, normal analog read function and digital read function will work for the application.

In the following code, we have not used any library. We have used a variable to store the value of the sensor and that value we are printing on the serial monitor using serial. print method.

It was about joystick code cases. If you have any doubts regarding the code then you can mention your doubts in the comment section.



```
#define joyX A0
#define joyY A1

int button=2;
int buttonState = 0;
int buttonState1 = 0;

void setup() {
  pinMode(7,OUTPUT);
  pinMode(button,INPUT);
  digitalWrite(button, HIGH);
  Serial.begin(9600);
}

void loop() {
 int xValue = analogRead(joyX);
 int yValue = analogRead(joyY);
 int button_output = digitalRead(button);

  Serial.print(xValue);
  Serial.print("\t");
  Serial.print(yValue);
  Serial.print("\t");
  Serial.println(button_output);

}
```

6. LDR (Light Dependent Switch)

Light Dependent Resistor are also known as photoresistor they are widely used in electronic sectors.

They are easily available is low prices so it becomes most preferable choice to the product selection engineer.

Although similar components such as photo-diode, phototransistor are also available in the market and used for the same applications but the low-cost feature of the LDR makes the deal to take away.

6.1. Applications of the LDR

LDR are used in many applications. The stud light that we see on streets that has LDR in it. The purpose of use an LDR in those stud lights is they are used to detect the day light.

Another application of the LDR is, they are used in the IR video recording cameras. The IR video recording cameras works in two mode.

- **Day light Mode**
- **Night Light Mode**

In first mode, they work similar to the normal camera but in second mode they turn on the IR lights to record the videos and to switch between these two modes the IR video recording cameras make use of LDR.

So, these are the applications of the LDR, in the next section of this blog we will learn about the interfacing of the LDR with the Arduino.
vv

6.2. Interfacing The LDR With The Arduino

Generally speaking, the LDR is nothing but a resistor. It has two pins and we can connect the LDR to the Arduino in a same fashion we connect the resistor to the Arduino.
In the following image you can see we have connected the one terminal of the LDR to the 5v pin of the Arduino and other end of the pin to the 7-number pin of the Arduino.
Please refer to the following image to understand the interfacing image of the LDR with Arduino.

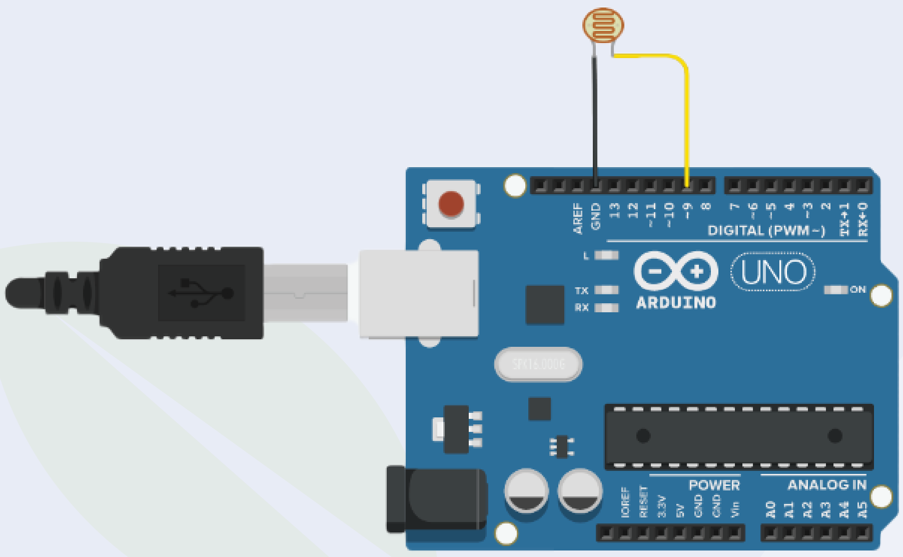
The LDR was designed in 1850 by the great electrical engineer Willoughby Smith.

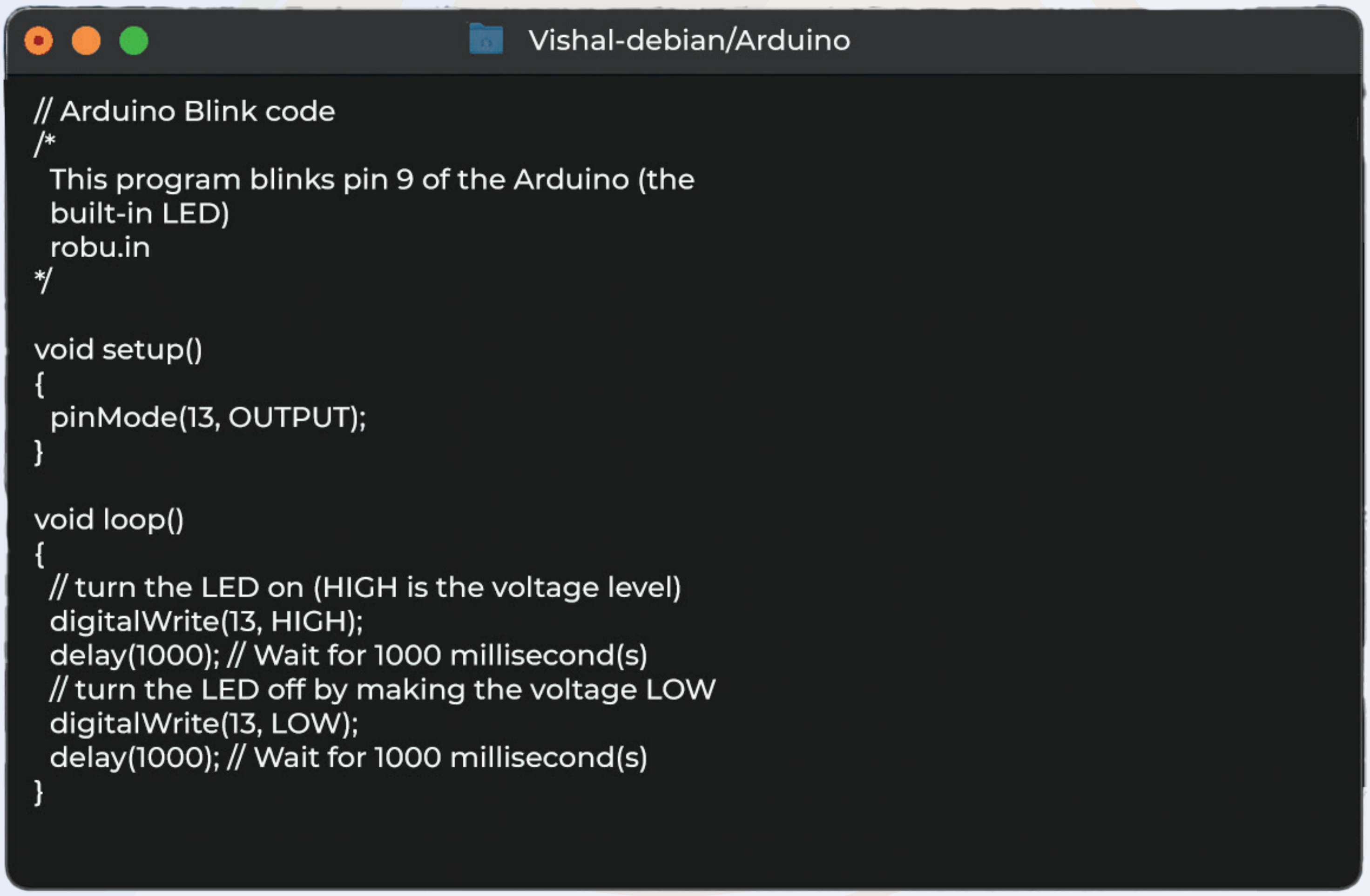
6.3. Arduino Code LDR

In this section, we will learn about the programming part of this section.
As I told you earlier, the interfacing the LDR is nothing but a switch which resistance changes according to change in the resistance of the light.
So, we have to monitor that change thorough the Arduino and to do that we will have to use the following methodology.

Here, when the intensity of light is increasing the resistance of the LDR is also changing. So to monitor that change, we will have read the output of the sensor and to read the output of the sensor we have used a variable.
That variable is storing the output of the sensor and then we are printing the value of that sensor on the serial monitor.
You can use the below code and can see the output of the sensor on the serial monitor.

6. LDR (Light Dependent Resistor)





```
// Arduino Blink code
/*
  This program blinks pin 9 of the Arduino (the
  built-in LED)
  robu.in
*/

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  // turn the LED on (HIGH is the voltage level)
  digitalWrite(13, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  // turn the LED off by making the voltage LOW
  digitalWrite(13, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

7. Common Cathode Two Color LED

We are getting command cathode two color LED module with the orange 37 in 1 sensor kit.
That module has two led on it and the anode termianal of those led is interconnected and the cathode terminal of those two led is left open.
The function of these two leds is same but the only difference is the packaging of the product.

You can see in the above image you are getting one SMD package LED and one DIP package LED.

Now you know about which LED you are getting with this kit but do you know the how LED will produce different color?
The common cathode two color LED has two LEDs in it. The cathode terminal of both LEDs is connected together and taken out of the led package whereas the anode terminals of those LEDs are not connected together they are separately taken out.

The LEDs that are inside of this LED can be turned on one after another or at the same.

We will have to connect the anode terminals of this LED to the positive voltage the particular LED will turn on.

And if we want to control the brightness of that particular LED then we will have to vary the input voltage to that LED.

And to vary the input voltage, we can use the PWM function.

PWM technique is a pulse width modulation technique. It is mainly used in audio ampliers, motor controllers.

The same technique we have used to control the brightness of the LED.

As I told earlier, we can turn on both LEDs at the same time or we can turn on each LED on after another.

You can try out turning on each LEDs on one after another and also you can try turning on them at the same using the below section discussed code.

But before that we will understand the interfacing of this LED with the Arduino.

7. Interfacing of Common Cathode LED with the Arduino

I have explained the function of those pins below please take a look.

Cathode Pin - The cathode pins of both LED are connected to this pin.

Anode Pin – This pin has three two anode terminals. One is of red LED and the other one is of green LED.

Note – Color of the LED may be different.

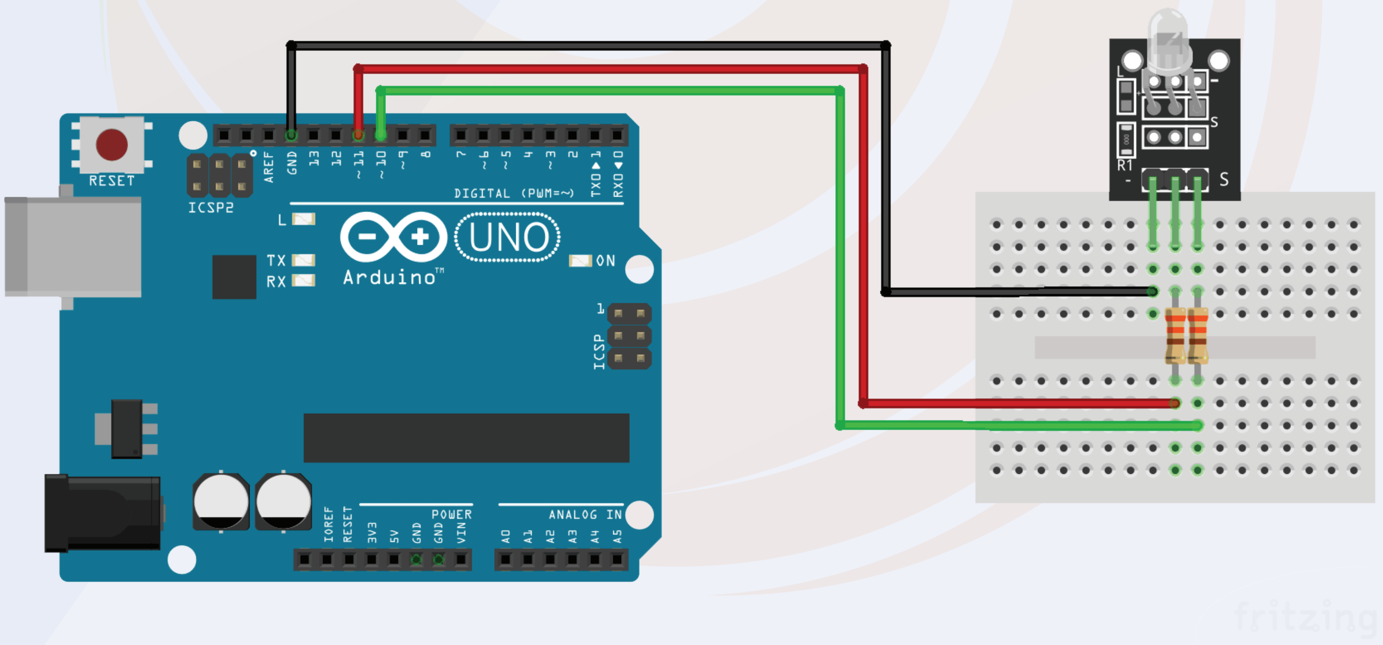
You can check the following image to understand the interfacing diagram properly.

In the following you can see we have connected the anode pin of the red LED to 6 pin number of the Arduino and the anode terminal of the green LED to the 7 number pin of the Arduino.

The reason behind connecting LEDs to those pins is those pins are PWM pins and as our application need PWM functionality we have used those.

You can use any GPIO pin of the Arduino to turn on the LEDs but if you want to control the brightness of those LEDs then you will have to use pwm pins only.

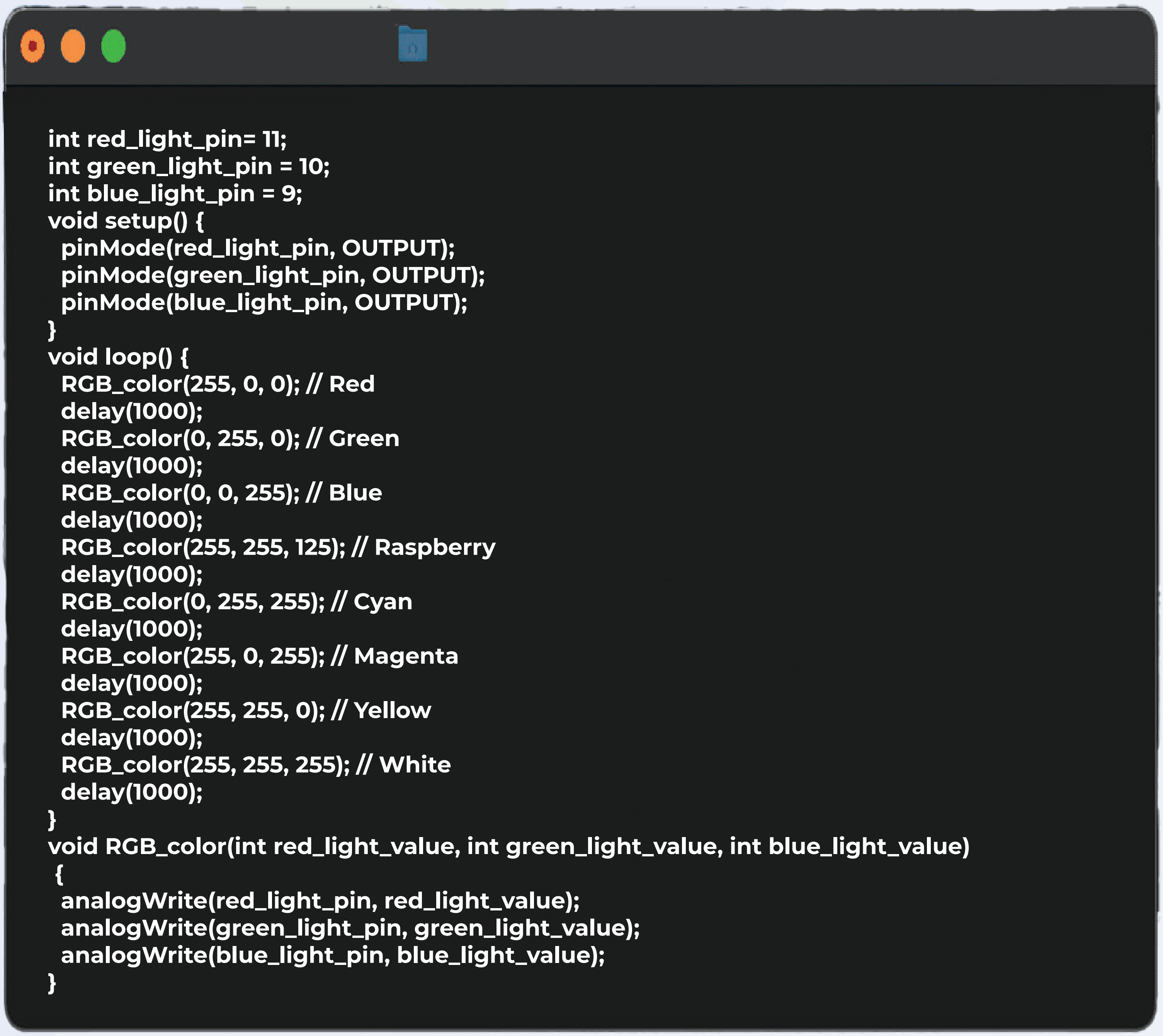
So, this was about the interfacing of the LEDs with the Arduino. In next section you will get the Arduino code for controlling the brightness of the LED.



7.2. Arduino Code For Common Cathode LED

We have already talked about the technique that we are going to use to control the Common cathode LED.

Now we will have to implement our logic. I have used my logic to implement the discussed things but if you better logic then you can use it.



```
int red_light_pin= 11;
int green_light_pin = 10;
int blue_light_pin = 9;
void setup() {
  pinMode(red_light_pin, OUTPUT);
  pinMode(green_light_pin, OUTPUT);
  pinMode(blue_light_pin, OUTPUT);
}
void loop() {
  RGB_color(255, 0, 0); // Red
  delay(1000);
  RGB_color(0, 255, 0); // Green
  delay(1000);
  RGB_color(0, 0, 255); // Blue
  delay(1000);
  RGB_color(255, 255, 125); // Raspberry
  delay(1000);
  RGB_color(0, 255, 255); // Cyan
  delay(1000);
  RGB_color(255, 0, 255); // Magenta
  delay(1000);
  RGB_color(255, 255, 0); // Yellow
  delay(1000);
  RGB_color(255, 255, 255); // White
  delay(1000);
}
void RGB_color(int red_light_value, int green_light_value, int blue_light_value)
 {
  analogWrite(red_light_pin, red_light_value);
  analogWrite(green_light_pin, green_light_value);
  analogWrite(blue_light_pin, blue_light_value);
}
```

8. RGB Colour LED

You may have used LEDs many times in your projects. In this kit, you are getting a LED but this LED is little bit different than normal LED. This LED produces three different types of color.

I know you may be interested in knowing about the working of the LED, in this section we will understand the working of the RGB LED.

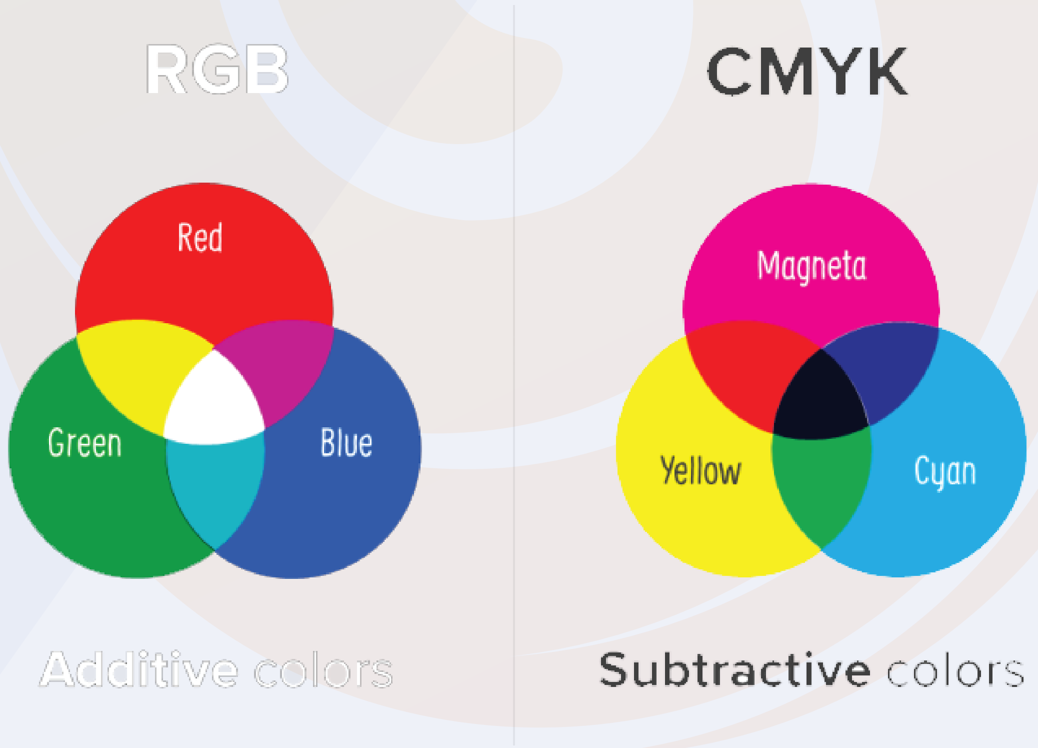
Before we talk about the working of the RGB led I would like to tell you the basic things of the RGB. So any color that you see in your surroundings is made up of three primary colors. Those Three primary colors are as follows.

1. Red
2. Green
3. Blue

And the following methods are used for creating different types of colors.

1. **Additive Mixing**
2. **Subtracting Mixing.**

The following images illustrate the color mixing methods.



8.1 RGB Colour LED

These are the two different LEDs we are getting with the kit. The function of these two leds is same but the only difference is the packaging of the product is different.

Till this point we were talking about the color mixing process of the RGB led but do you know how the RGB LED emit different colors?
The RGB led works on the PWM value, when the variable voltage is applied to the input terminal of the RGB LED, the LED start emitting different colors. So, we can say the color of the LED depends on the PWM value we are applying to the input pin of the LED.

We can generate PWM voltage by using the analogWrite function. There are other ways of generating PWM voltage also we have talked about those methods in the booklet of the kit.

You can check those methods there.

In the next section of this blog we will talk about the interfacing of the sensor with the Arduino.

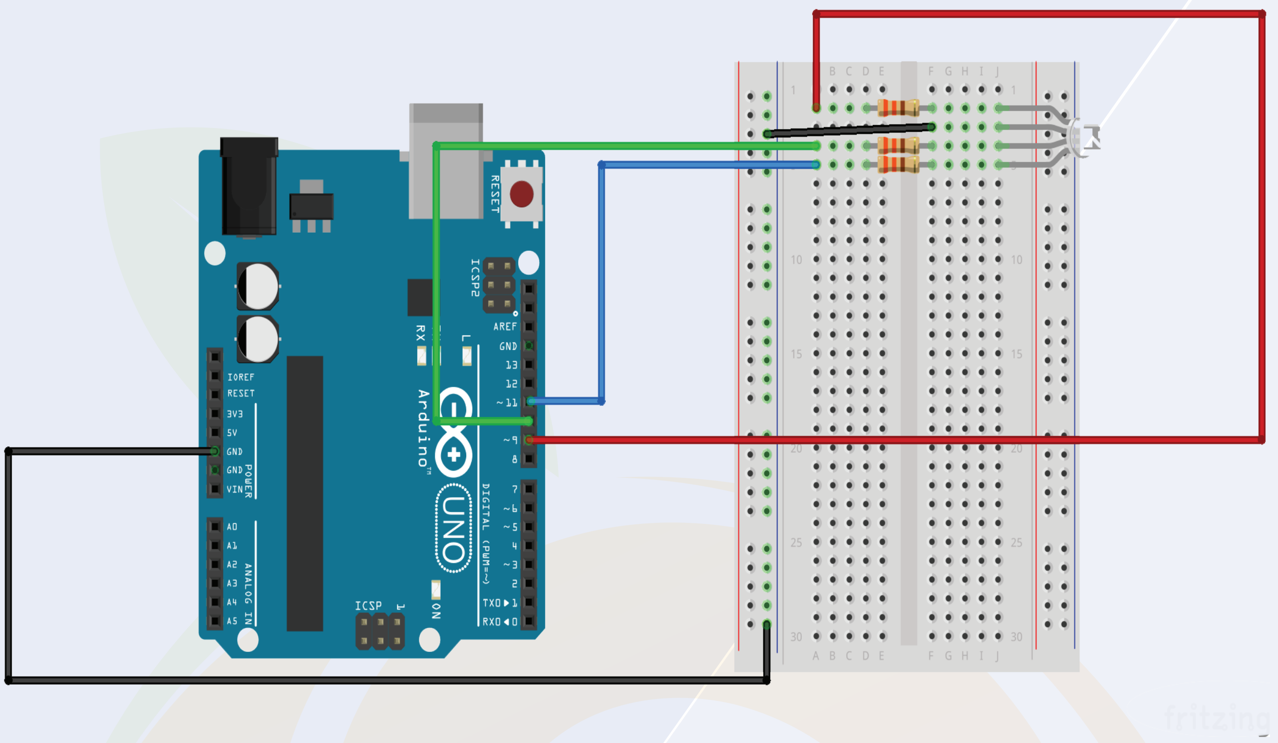
8.2. Interfacing of RGB LED with the Arduino

The RGB LED that you getting with this orange 37 in 1 sensor kit has only three pins. Vcc, gnd and signal. To turn on the LED, You will have to apply PWM signal to the signal pin of the RGB led.

In my case, I have connected the signal pin of my RGB to the 9-number pin of the Arduino. And the other power pins to the power pins of the Arduino.

You can check the following image to understand the interfacing diagram properly.

8.2. Interfacing of RGB LED with the Arduino



10.3 Arduino Code For RGB LED

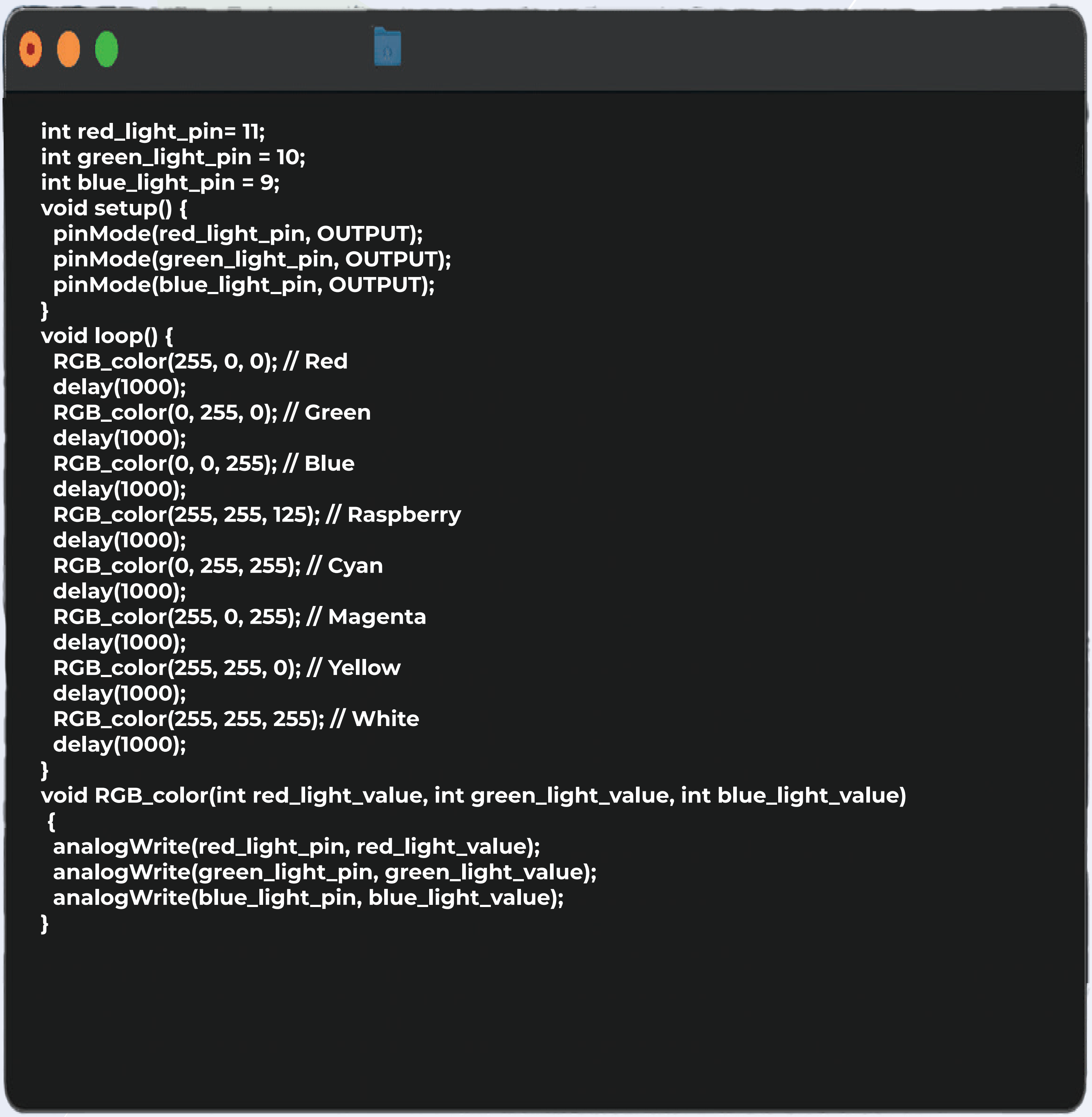
As discussed in the earlier sections, we have used analogRead function for writing the PWM values.

You can use the following code and if you have any doubt please let us know in the comment section.

8.3. Arduino Code For RGB LED

You may have used LEDs many times in your projects. In this kit, you are gettAs discussed in the earlier sections, we have used analogRead function for writing the PWM values.

You can use the following code and if you have any doubt please let us know in the comment section.



```
int red_light_pin= 11;
int green_light_pin = 10;
int blue_light_pin = 9;
void setup() {
  pinMode(red_light_pin, OUTPUT);
  pinMode(green_light_pin, OUTPUT);
  pinMode(blue_light_pin, OUTPUT);
}
void loop() {
  RGB_color(255, 0, 0); // Red
  delay(1000);
  RGB_color(0, 255, 0); // Green
  delay(1000);
  RGB_color(0, 0, 255); // Blue
  delay(1000);
  RGB_color(255, 255, 125); // Raspberry
  delay(1000);
  RGB_color(0, 255, 255); // Cyan
  delay(1000);
  RGB_color(255, 0, 255); // Magenta
  delay(1000);
  RGB_color(255, 255, 0); // Yellow
  delay(1000);
  RGB_color(255, 255, 255); // White
  delay(1000);
}
void RGB_color(int red_light_value, int green_light_value, int blue_light_value)
 {
  analogWrite(red_light_pin, red_light_value);
  analogWrite(green_light_pin, green_light_value);
  analogWrite(blue_light_pin, blue_light_value);
}
```

9. Shock Switch

This is shock sensor that we are getting with this kit. Now, where do we use this sensor?

This sensor can be used in many applications such shock sensor-based wake up system, toys and many more places.

But how does this sensor detect the shock? What part it has in that cylindrical part? We are going to discuss about all those things in the below section.

So, the shock switch has two main parts. One is thin metal spring and other one is conductive material.

The thin spring is wrapped around the conductive material when the sensors moves the spring also moves and while moving the spring touches to the conductive material and this is how the connection happens.

The output of this sensor will be digital so we can use digital read function to read the output of the sensor.

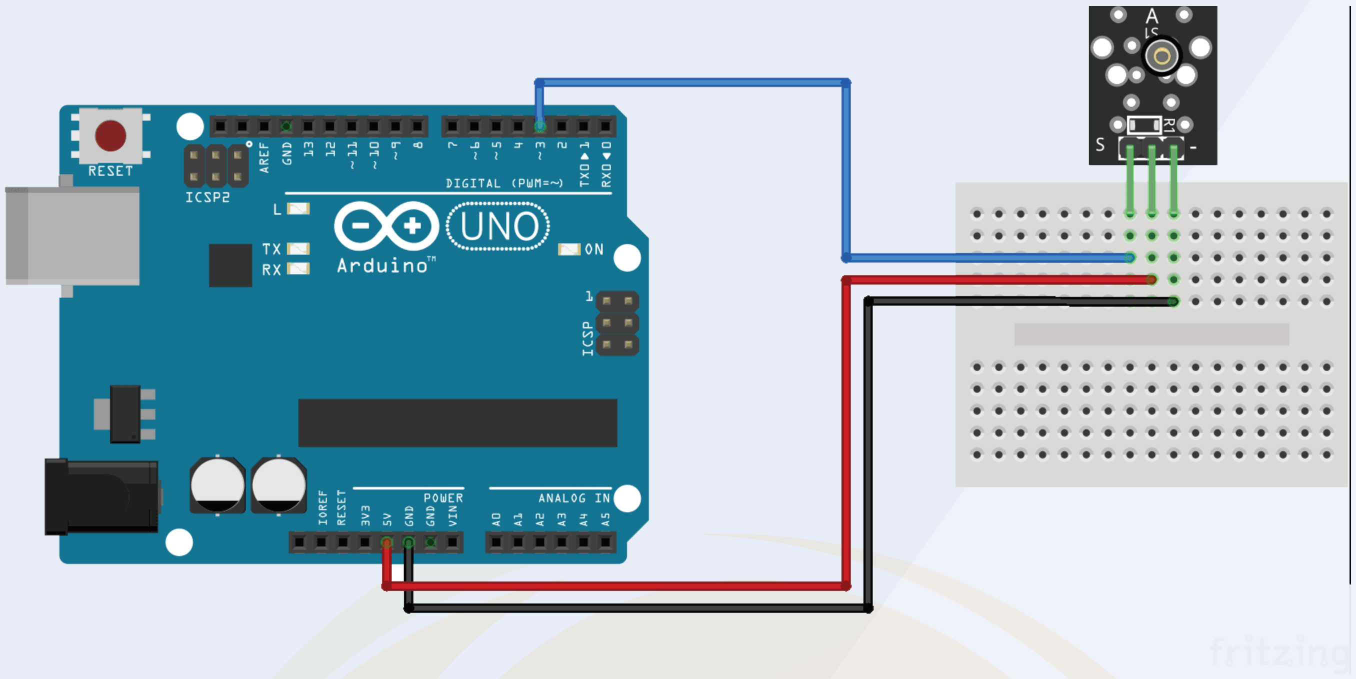
9.1 Interfacing The Shock Switch With The Arduino

The shock pin has three pins. Signal pin, Vcc and GND pin. The signal pin of the sensor we can connect to the any GPIO pin of the Arduino and the power pin of the sensor to the power pins of the Arduino.

Here while interfacing the shock sensor with the Arduino we will have to use either pull-up resister or pull-up resister.

I have shared the images of both methods. Please take a look and if you face any issues you can mention your issue in the comment section.

9.1 Interfacing The Shock Switch With The Arduino



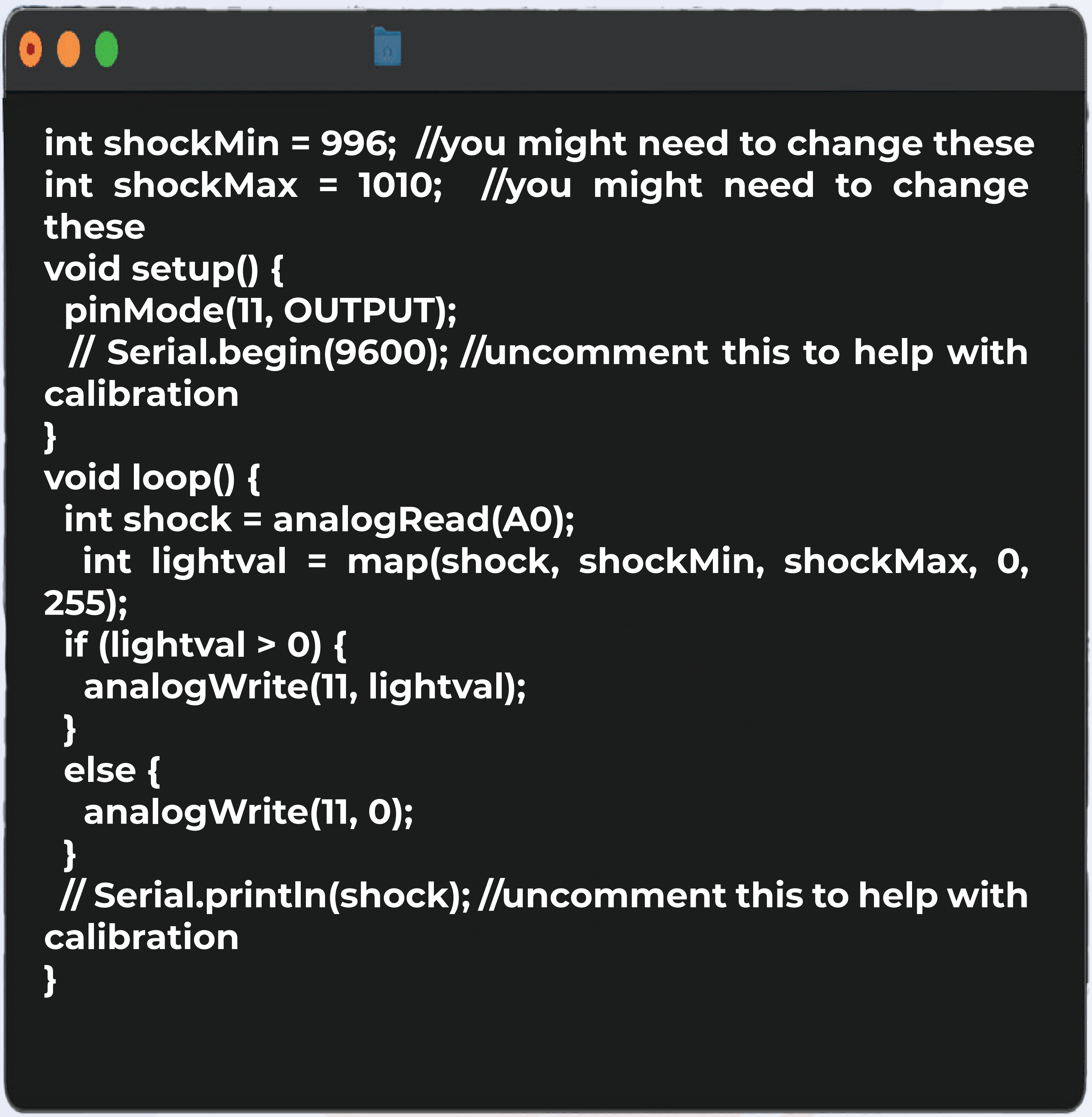
9.2 Arduino Code For shock Sensor

In the below section, I have shared the code for the Arduino. In this code I am using a digitalRead function to read the output of the sensor.

You can use the following code and can start working with the shock sensor. In the following code we are reading the output of the sensor and printing it on the serial monitor.

So, to check the output of the sensor you will have to open to the serial monitor.

9.2 Arduino Code For shock Sensor



```
int shockMin = 996;  //you might need to change these
int shockMax = 1010;  //you might need to change
these
void setup() {
  pinMode(11, OUTPUT);
  // Serial.begin(9600); //uncomment this to help with
calibration
}
void loop() {
  int shock = analogRead(A0);
    int lightval = map(shock, shockMin, shockMax, 0,
255);
  if (lightval > 0) {
    analogWrite(11, lightval);
  }
  else {
    analogWrite(11, 0);
  }
  // Serial.println(shock); //uncomment this to help with
calibration
}
```

10. Knock Sensor:

The working principle of the knock sensor is the same as that of the shock sensor.

When we rotate this sensor, this sensor generates a high-level voltage signal. It is used in doorbell applications and automatic switching boards.

The interfacing of these sensors is simple. In the next part of this blog, we will understand the interfacing of sensors with Arduino.

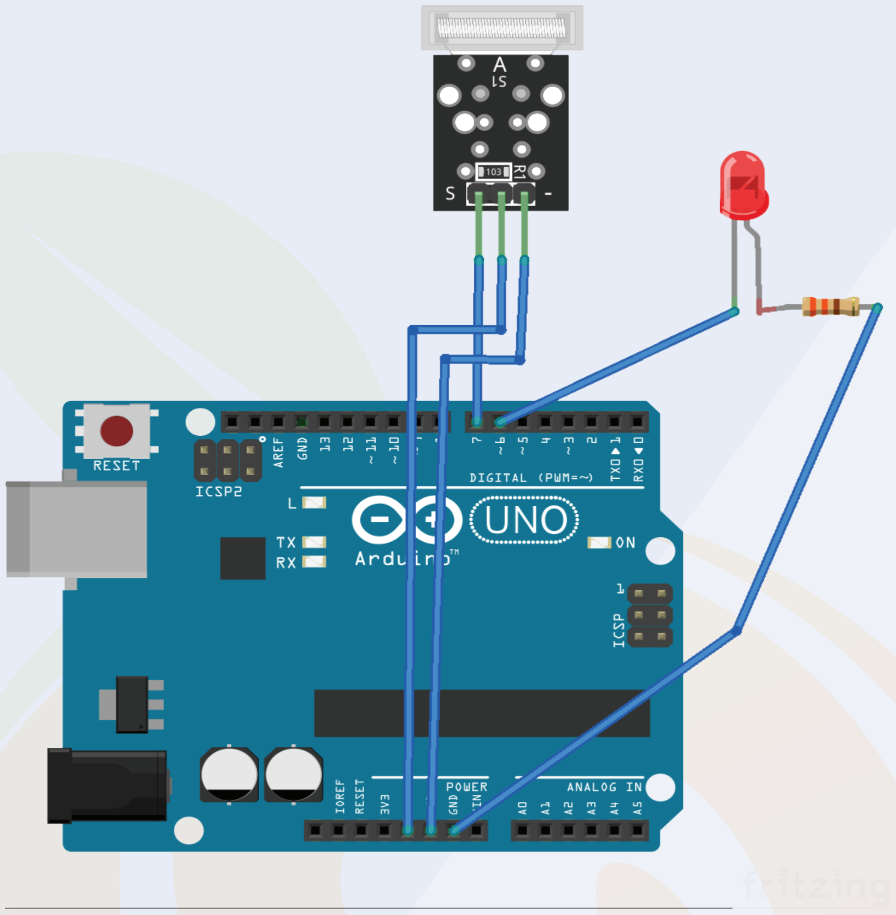
10.1 Interfacing the knock sensor with the Arduino

The interfacing of knock sensor is little bit different than the shock sensor. IN the interfacing of the shock sensor with the Arduino we were using the pull-up or pull-down resisters but in case of the knock sensor we don't need to add any external circuit.

In the following image you can see I have connected the output of the sensor with the Arduino and the power pins of the sensor with the power pins of the Arduino.

Please see the following image of the sensor to understand the interfacing of the sensor.

10.1 Interfacing The Shock Switch With The Arduino



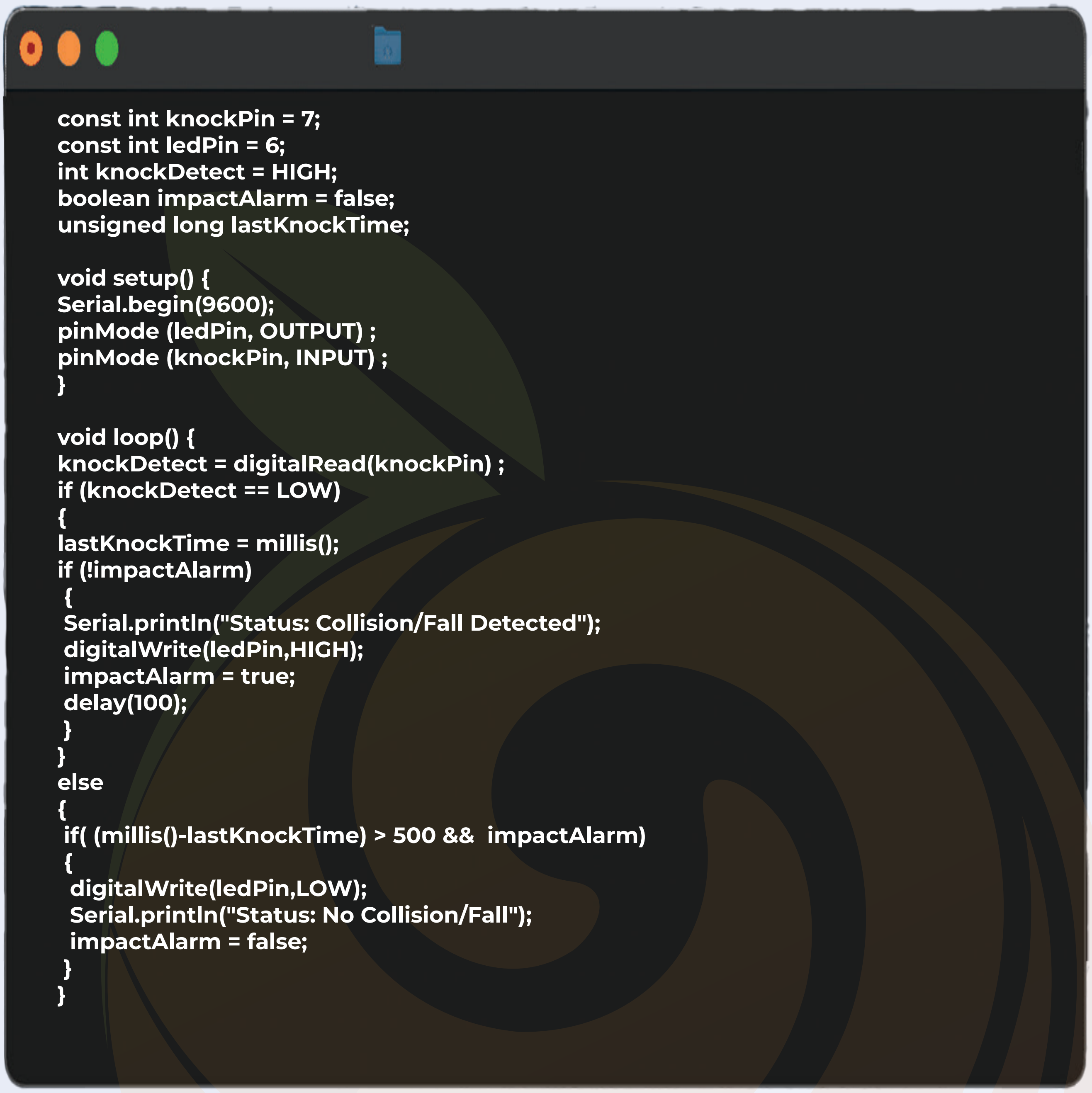
10.2 Arduino Code For shock Sensor

In the below section, I have shared the code for the Arduino. In this code I am using a digitalRead function to read the output of the sensor.

You can use the following code and can start working with the shock sensor. In the following code we are reading the output of the sensor and printing it on the serial monitor.

So, to check the output of the sensor you will have to open to the serial monitor.

10.2 Arduino Code For Shock Sensor



```
const int knockPin = 7;
const int ledPin = 6;
int knockDetect = HIGH;
boolean impactAlarm = false;
unsigned long lastKnockTime;

void setup() {
Serial.begin(9600);
pinMode (ledPin, OUTPUT) ;
pinMode (knockPin, INPUT) ;
}

void loop() {
knockDetect = digitalRead(knockPin) ;
if (knockDetect == LOW)
{
lastKnockTime = millis();
if (!impactAlarm)
 {
 Serial.println("Status: Collision/Fall Detected");
 digitalWrite(ledPin,HIGH);
 impactAlarm = true;
 delay(100);
 }
}
else
{
 if( (millis()-lastKnockTime) > 500 &&  impactAlarm)
 {
  digitalWrite(ledPin,LOW);
  Serial.println("Status: No Collision/Fall");
  impactAlarm = false;
 }
}
```

In the below section, I have shared the code for the Arduino. In this code I am using a digitalRead function to read the output of the sensor.

You can use the following code and can start working with the shock sensor. In the following code we are reading the output of the sensor and printing it on the serial monitor.

So, to check the output of the sensor you will have to open to the serial monitor.

11. Infrared Sensor Generator

The infrared sensor are used in many applications. You can see the infrared sensor in wireless remotes they are also can be used in data transmission systems.

In this section you will understand the working of the IR sensor and interfacing of the sensor with the Arduino.

Infrared sensor generates infrared rays. The switching speed of the infrared sensor is very fast that is why they are used in data transmission application also.

The wireless remote controller that we use in our daily life has an infrared sensor, control unit and power unit in it.

The power unit power the module and the control-unit is designed for generating the IR receiver acceptable code.

The code is nothing but a blinking pattern on the Infrared sensor. Our Arduino program will convert the control signals to encoded signals and this encoded signal will be then decoded at the IR receiver.

Talking about the IR receiver are the sensors that are used to receive the signals that are transmitted by the IR transmitter.

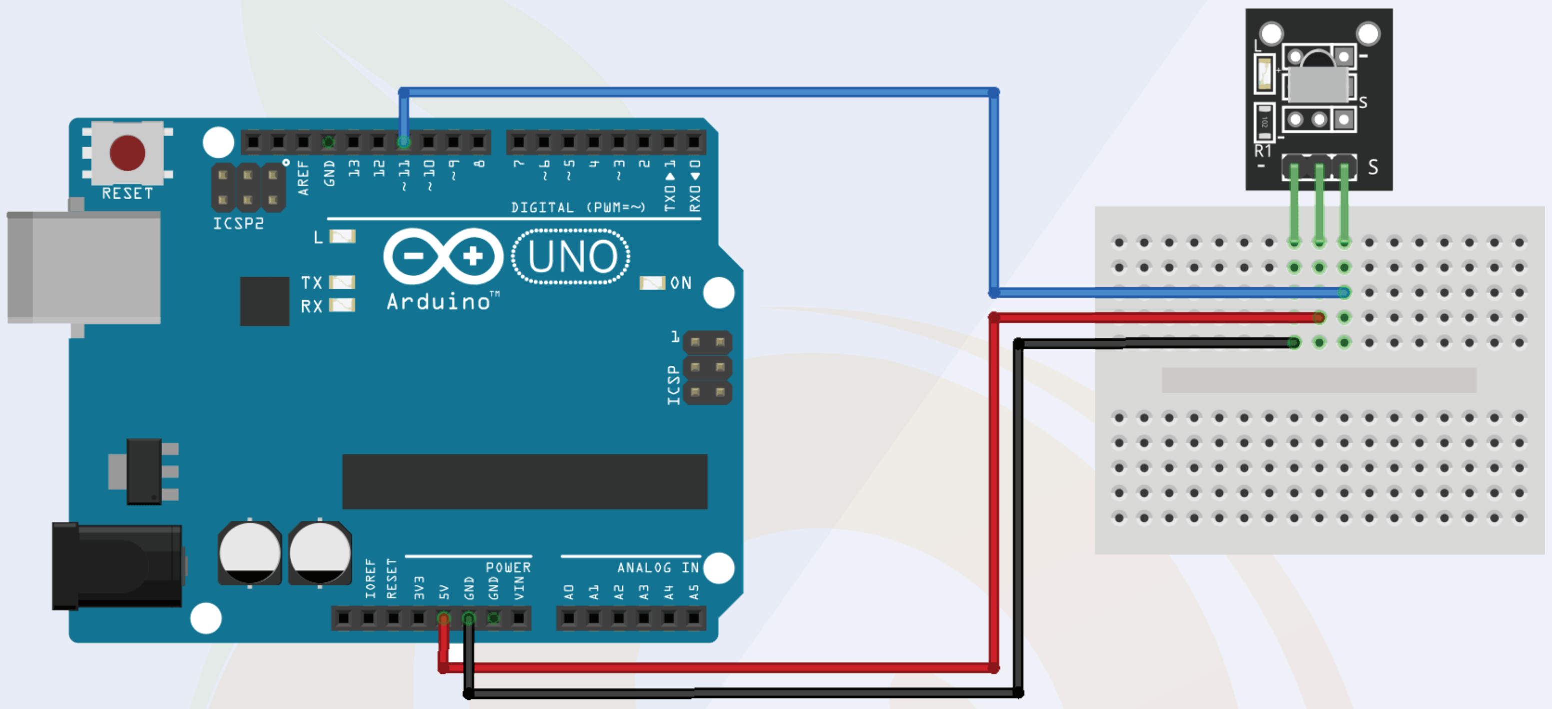
IR receiver receives the signal and pass those received signals to the control unit and then the control unit take the further required actions.

This Is how IR Receiver and IR Transmitter works. In the next section of this blog we will understand the interfacing of the IR transmitter.

11.1 Interfacing of the IR Sensor with the Arduino

11.1 Interfacing of the IR Sensor with the Arduino

Please check the following image to understand the interfacing of the IR sensor.



11.2 Arduino Code For Shock Sensor

We have connected the IR receiver to the Arduino. In this section, we will understand the interfacing of the Arduino with the IR receiver.
In the following code, we are using IR sensor library. You can download that library using the following link.

https://github.com/Arduino-IRremote/Arduino-IRremote

After installing the library, you can use the following code.

11.2 Arduino Code For Shock Sensor

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

int RECV_PIN = 13;          //  The digital pin that the signal pin of the
sensor is connected to
IRrecv receiver(RECV_PIN);  //  Create a new receiver object that
would decode signals to key codes
decode_results results;   //  A varuable that would be used by receiv-
er to put the key code into

void setup() {
 Serial.begin(9600);     // Setup serial port to send key codes to com-
puter
 receiver.enableIRIn();   //  Enable receiver so that it would start pro-
cessing infrared signals
}

void loop() {
 if(receiver.decode(&results)) {          //  Decode the button code and
put it in "results" variable
   Serial.println(results.value, HEX);     //  Print the code as a hexadec-
imal value
   receiver.resume();                   //  Continue listening for new signals
 }
}
```

12. IR Obstacle Sensor

IR sensor is an electronic device, that emits the light in order to sense some object of the surroundings.

An IR can measure the heat of an object as well as detects the motion. Usually, in infrared spectrum, all the objects radiate some form of thermal radiation.

These types of radiations are invisible to our eyes, but infrared sensor can detect these radiations.

This sensor has two modules. Those modules are as follows:

1. IR Receiver
2. IR Transmitter

12.1 IR Transmitter or IR LED

Infrared Transmitter is a light emitting diode (LED) which emits infrared radiations called as IR LED's.

Even though an IR LED looks like a normal LED, the radiation emitted by it is invisible to the human eye.

12.2 IR Receiver or Photodiode

Infrared receivers or infrared sensors detect the radiation from an IR transmitter. IR receivers come in the form of photodiodes and phototransistors. Infrared Photodiodes are different from normal photo diodes as they detect only infrared radiation. Below image shows the picture of an IR receiver or a photodiode.

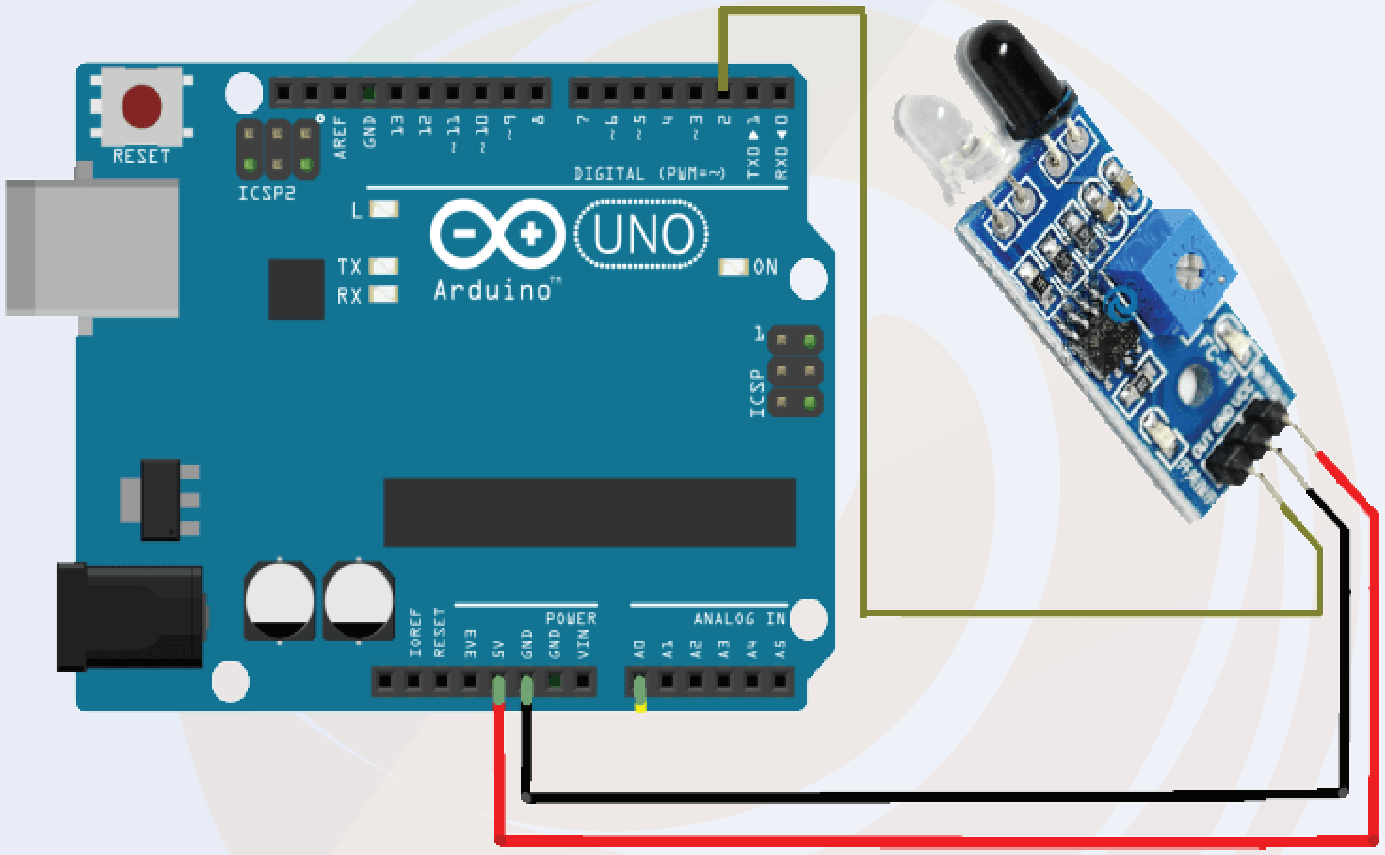
The third part of the IR obstacle sensor is Gain Amplifier.
The gain amplifier plays an important role when the quality of the signal is very low. It amplifies the week signal and generate the high-quality signals.
If the output of your module is not clear then you get potentiometer on the module. You can adjust that potentiometer to amplify the week signals.

12.3 Interfacing of the IR Sensor with the Arduino

The interfacing of the IR obstacle sensor with the Arduino is very. This module has only three pins out of those three pin two are power pins and one is signal pin.

Those power pins you can connect to the power pins of the Arduino and you can connect the signal pin to the analog pin of the Arduino.

The following image will clear your doubts.

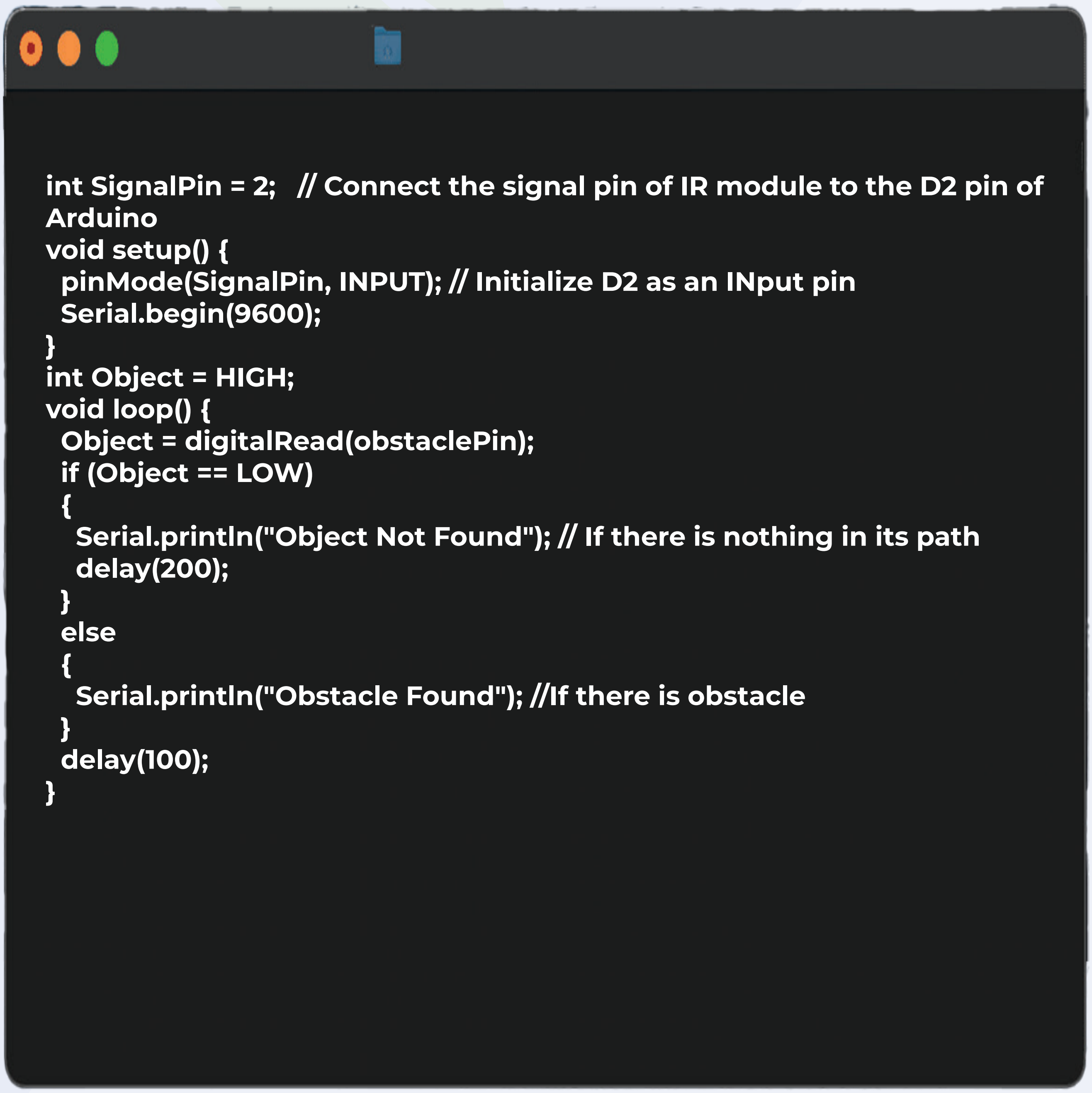


12.4. Arduino Code IR Obstacle Sensor

In this section we will understand about the Arduino code for the IR Sensor. In the code below you can see we have used analogRead function to read the output of the IR sensor.

After that you can see there we are storing the output of the sensor in a variable. And then we have used serial.print function to print the output of the sensor.

Please check the code below and if you have any doubt please mention in the comment section.



```
int SignalPin = 2;   // Connect the signal pin of IR module to the D2 pin of
Arduino
void setup() {
  pinMode(SignalPin, INPUT); // Initialize D2 as an INput pin
  Serial.begin(9600);
}
int Object = HIGH;
void loop() {
  Object = digitalRead(obstaclePin);
  if (Object == LOW)
  {
    Serial.println("Object Not Found"); // If there is nothing in its path
    delay(200);
  }
  else
  {
    Serial.println("Obstacle Found"); //If there is obstacle
  }
  delay(100);
}
```

13. IR Flame Sensor

Different flame sensors are available in the market. What you are getting with this kit is the IR flame sensor.

IR flame sensors are used in fire systems. They detect the fire and send a signal to the controller unit, then the controller unit takes the necessary actions to avoid damage from the fire. But do you know how the IR flame sensor detects fire.

The answer to the question is, IR sensors are designed to capture gases that appear in the infrared spectral band.

When an explosion occurs, the light produced by the flames appears in the infrared spectral band and produces a flame sensor signal based on the received frequency of light.

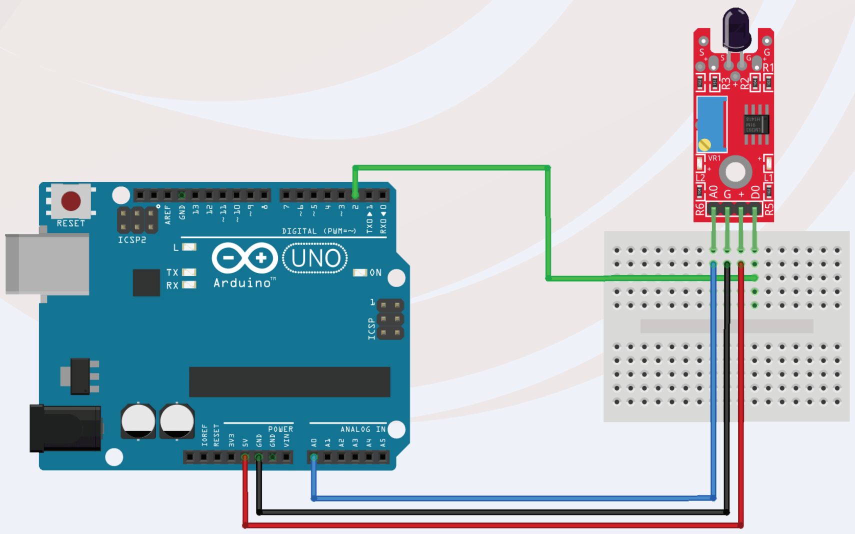
It was about the introduction of flame sensors; In the next part of the blog we will learn about interfacing of flame sensors with Arduino.

13.1 Interfacing Flame Sensor With Arduino

As we know that IR Flame Sensor has three pins, interfacing of IR Flame Sensor with Arduino is so easy.

You just need to connect the signal pin of the flame sensor to the analog pin of the Arduino and the other power pin of the sensor to the power pin of the Arduino.

Please refer to the image below to understand the wiring diagram.



13.2 Arduino Code For Flame Sensor

So, as you connected the IR Flame Sensor to the Arduino, we will now move to the Arduino Code section of the Flame Sensor.

The code is very simple and requires no explanation. In this code, we have used a variable and the function of that variable is to collect the output of the IR flame sensor.

Then we print that output on the serial monitor.

Please use the below code.

```
// lowest and highest sensor readings:
const int sensorMin = 0;     // sensor minimum
const int sensorMax = 1024;  // sensor maximum

void setup() {
  // initialize serial communication @ 9600 baud:
  Serial.begin(9600);
}
void loop() {
  // read the sensor on analog 2:
        int sensorReading = analogRead(2);
  // map the sensor range (four options):
  // ex: 'long int map(long int, long int, long int, long int, long int)'
        int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

  // range value:
  switch (range) {
  case 0:    // A fire closer than 1.5 feet away.
    Serial.println("** Close Fire **");
    break;
  case 1:    // A fire between 1-3 feet away.
    Serial.println("** Distant Fire **");
    break;
  case 2:    // No fire detected.
    Serial.println("No Fire");
    break;
  }
```

14. Buzzer

Buzzers are used in many devices. It is used as an indicator in many systems such as home appliances, alarm systems, electronic bells.
Talking about the working principle of the buzzer, the buzzer converts electrical energy into sound energy.

When the voltage is applied to the buzzer, the piezo crystal expands and shrinks inside the plastic casing. This causes the plate vibrates near the crystal and the sound you hear is of the same vibration.

Changing the frequency to buzzer changes the speed of vibration and as a result you hear different types of sound.

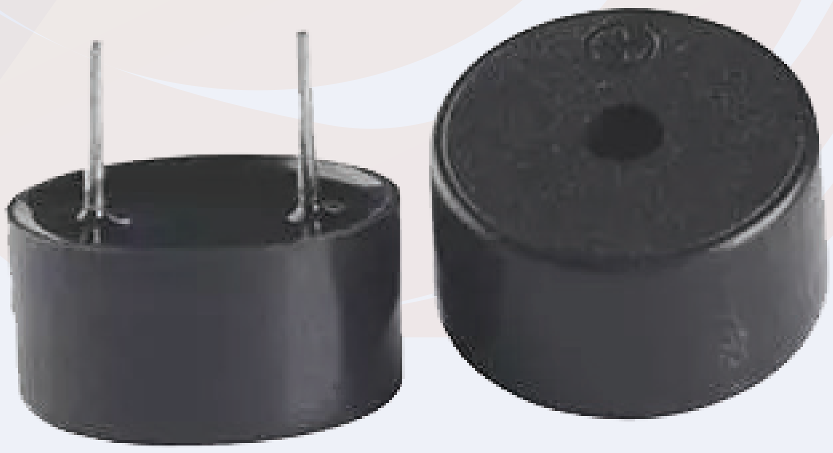
So, it was about the buzzer. There are two main types of buzzer. Active buzzer and passive buzzer. In the next part of this blog, we will understand the difference between these two types.

14.1 Difference Between Active and Passive Buzzer

As I mentioned earlier, there are many types of buzzers. Active buzzer and passive buzzer.

Talking about active buzzer, it has inbuilt oscillating source. As soon as you power it up, the active buzzer starts to sound but in the case of a passive buzzer they do not have an inbuilt oscillating source.

If you want a passive buzzer to generate a sound signal, then you have to give the buzzer a different frequency.



B

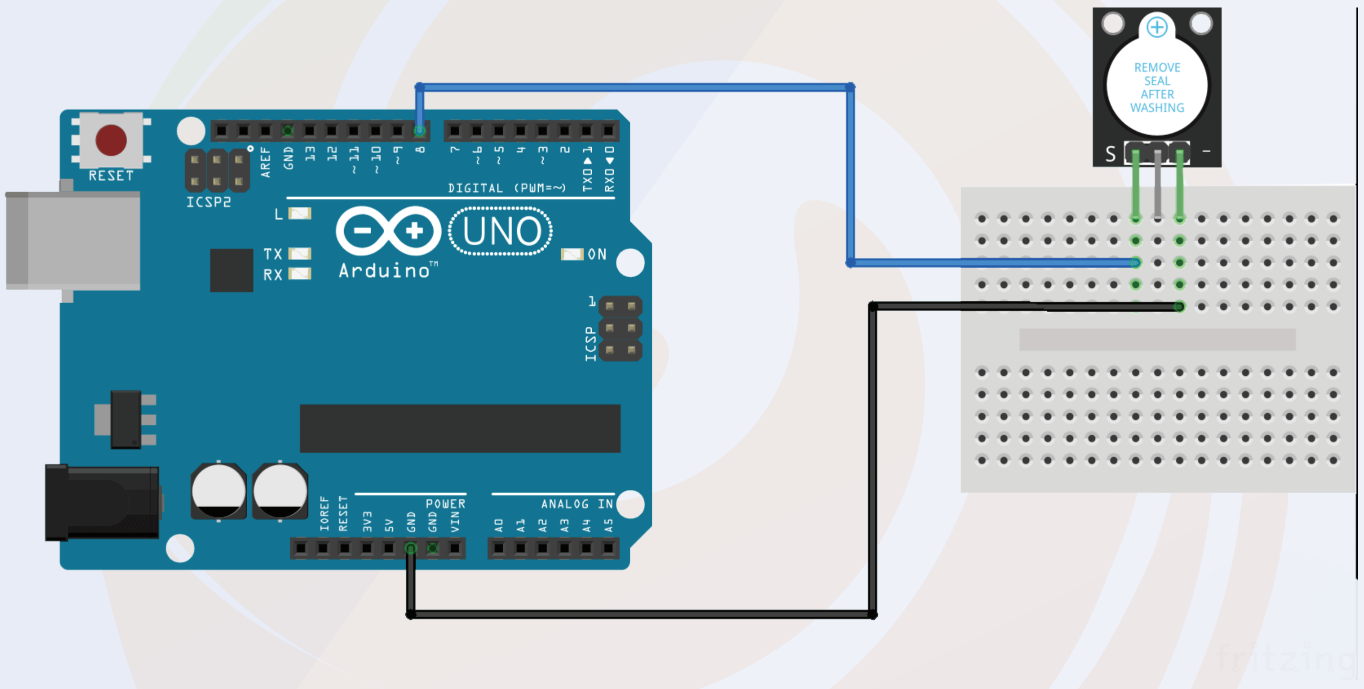
14.2. Buzzer Interfacing with Arduino

pin directly to the GPIO pin of the Arduino and the GND pin of the buzzer to the GND pin of the Arduino.

In my case, I have used an active buzzer. But you can make use of passive buzzer, but remember, if you want to make a different sound from the passive buzzer here, then you have to give different frequency to the VCC pin of the buzzer.

Please check the interfacing diagram below to understand the interfacing diagram.

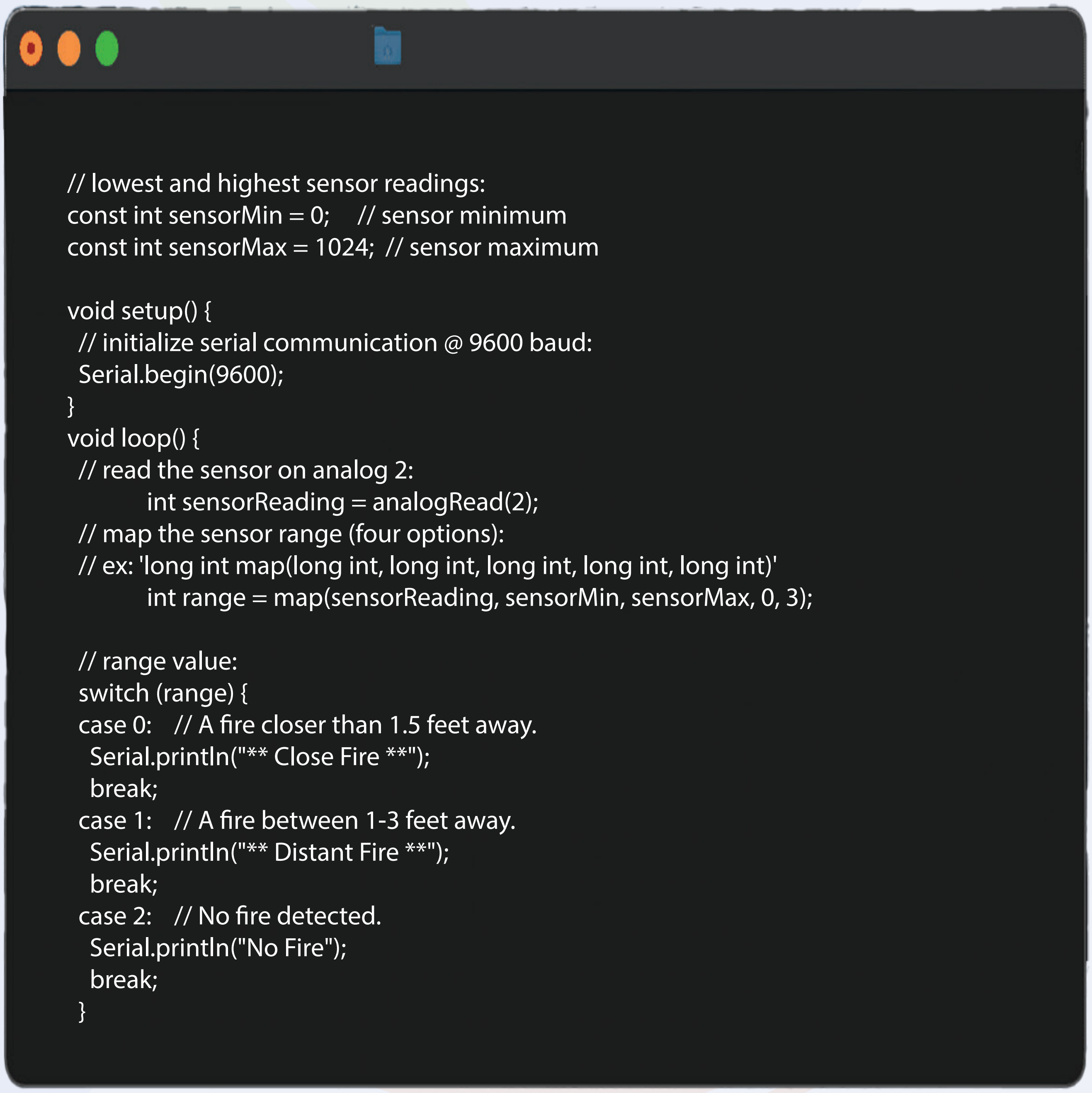
As you have wired the buzzer. Now, we will move to the Arduino code for buzzer.



14.2 Arduino Code For Buzzer

The code for the buzzer is very simple. You just need to use the digitalWrite function and apply high or low voltage to turn on or off the buzzer.

You can use the following code to turn on the buzzer.



```
// lowest and highest sensor readings:
const int sensorMin = 0;     // sensor minimum
const int sensorMax = 1024;  // sensor maximum

void setup() {
  // initialize serial communication @ 9600 baud:
  Serial.begin(9600);
}
void loop() {
  // read the sensor on analog 2:
            int sensorReading = analogRead(2);
  // map the sensor range (four options):
  // ex: 'long int map(long int, long int, long int, long int, long int)'
            int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

  // range value:
  switch (range) {
  case 0:    // A fire closer than 1.5 feet away.
    Serial.println("** Close Fire **");
    break;
  case 1:    // A fire between 1-3 feet away.
    Serial.println("** Distant Fire **");
    break;
  case 2:    // No fire detected.
    Serial.println("No Fire");
    break;
  }
```

15. Vibration Sensor

Vibration sensors are used in many devices such as mechanical machines. It is used to measure the frequency of vibrations generated by machines and those measured frequencies are then converted into voltage signals.

The Vibration sensor we are getting with this kit has an onboard gain amplifier. That amplifier is used to adjust the gain of the module. If the signals generated by the vibration sensor are not accurate then you can adjust the trimmer next to the gain amplifier.

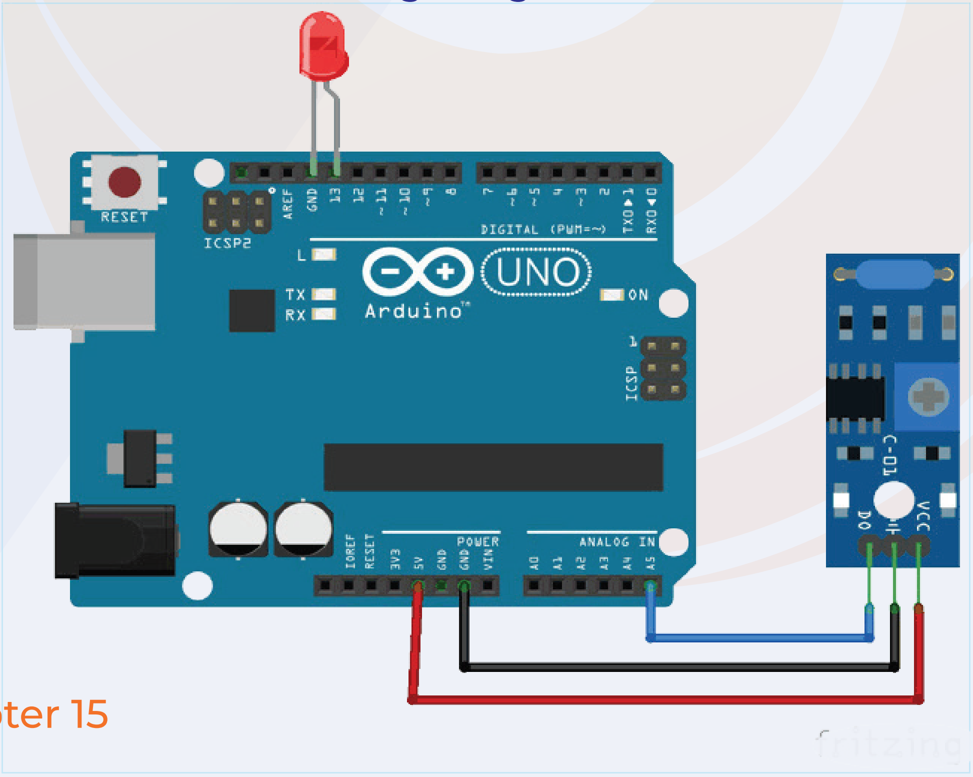
There is a lot of information about vibration sensors, now we will talk about the interfacing of vibration sensors with Arduino.

15.1 Interfacing Vibration Sensor With Arduino

Vibration sensor interfacing with the Arduino is super easy. It has three pins, two for powering the module and one for output.

We have to connect the power pin of the vibration sensor to the Arduino and the signal pin to any GPIO pin of the Arduino.

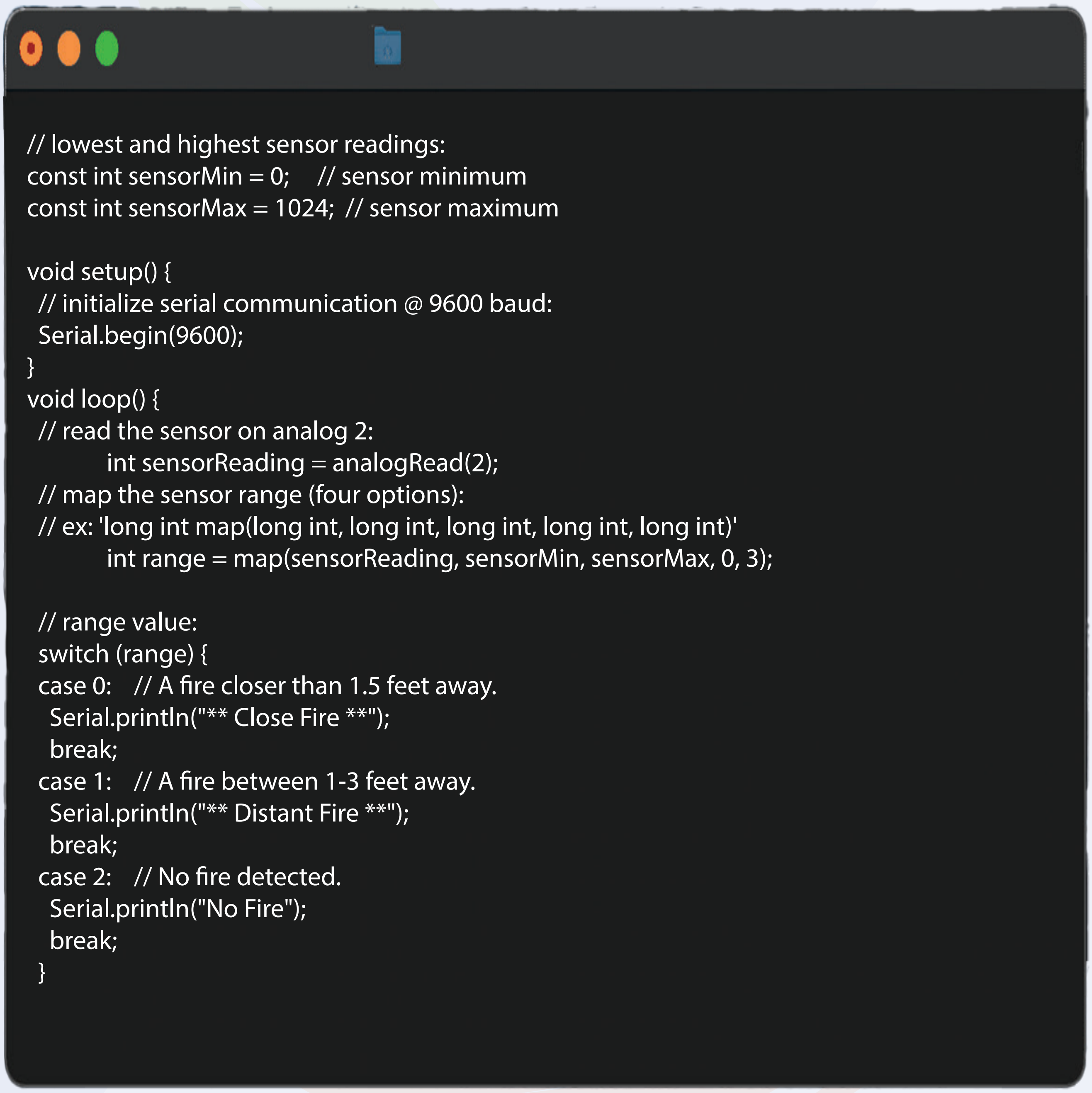
You can refer to the following image to understand the wiring diagram.



15.2 Arduino Code For Vibration Sensor

The code for the buzzer is very simple. You just need to use the digitalWrite function and apply high or low voltage to turn on or off the buzzer.

You can use the following code to turn on the buzzer.



```
// lowest and highest sensor readings:
const int sensorMin = 0;     // sensor minimum
const int sensorMax = 1024;  // sensor maximum

void setup() {
 // initialize serial communication @ 9600 baud:
 Serial.begin(9600);
}
void loop() {
 // read the sensor on analog 2:
        int sensorReading = analogRead(2);
 // map the sensor range (four options):
 // ex: 'long int map(long int, long int, long int, long int, long int)'
        int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

 // range value:
 switch (range) {
 case 0:    // A fire closer than 1.5 feet away.
  Serial.println("** Close Fire **");
  break;
 case 1:    // A fire between 1-3 feet away.
  Serial.println("** Distant Fire **");
  break;
 case 2:    // No fire detected.
  Serial.println("No Fire");
  break;
 }
```

16. 5V Single Channel

A relay is an electrically operated switch that can be turned on or off, letting the current go Relay Module can be found in many electronic devices. It is an electromagnetic switch that is used for switching high voltage requirement appliances.

In this section we are going to understand the interfacing of the relay with the Arduino and will understand a few basic things of the relay module.

Relay module has five pins, out of those five pins three are of input and two are of output. I have explained the function of those pins below please have a look.

No Normally Open –

This is output pin of the relay module. This pin remain open until the coil of the relay module is not energized.

NC Normally Close –

This is also an Output pin. This pin stays connected to the common terminal until the relay module is not energized.

Com Common Pin –

This is an input pin. When the relay module is powered on the Normally open pin will be connected to the common terminal pin and when the power to the relay module is turned off then the common terminal will be connected to the normally close terminal.

VCC Pin and GND Pin –

These two pins are the power pins and are connected to the internal coil of the relay.

The operating voltage of the coil of the relay module that we are getting with this kit, is 5V.
So, we can directly connect the 5V supply to this pin.

16. 5V Single Channel Relay

Signal Pin –

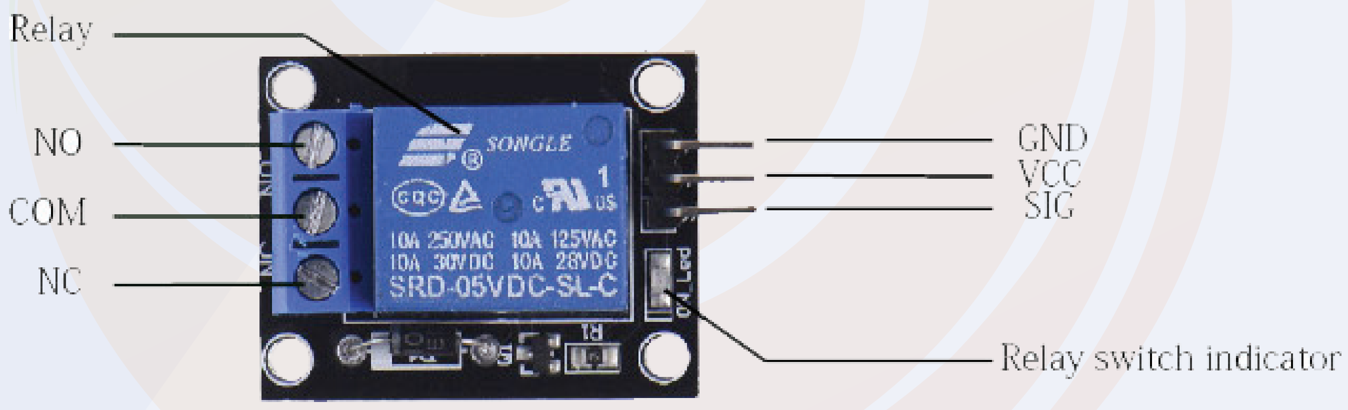
This pin is connected to the base terminal of the SMD transistor which is present on the relay module. When we give 5V to this pin, this pin will turn on the power to the relay coil.

So, this was about the introduction of the relay module in the next section of this blog we will understand the interfacing of the relay module.

16.1. Interfacing The Relay Module With The Arduino –

In this section, we will be learning about the interfacing of the relay module with the Arduino.

As discussed earlier, the relay module has six pins and out of those six pins, five pins three are input pin.



16.1. Interfacing The Relay Module With The Arduino –

You can see those three pins in the above image.

You can see GND, VCC and signal pin over there. The GND and VCC pin of the relay module you can connect to the VCC and GND pin of the Arduino and the Signal pin of the relay module you can connect to any GPIO pin of the Arduino.

This was about the one end of the relay module now we will understand the connections of the other side.

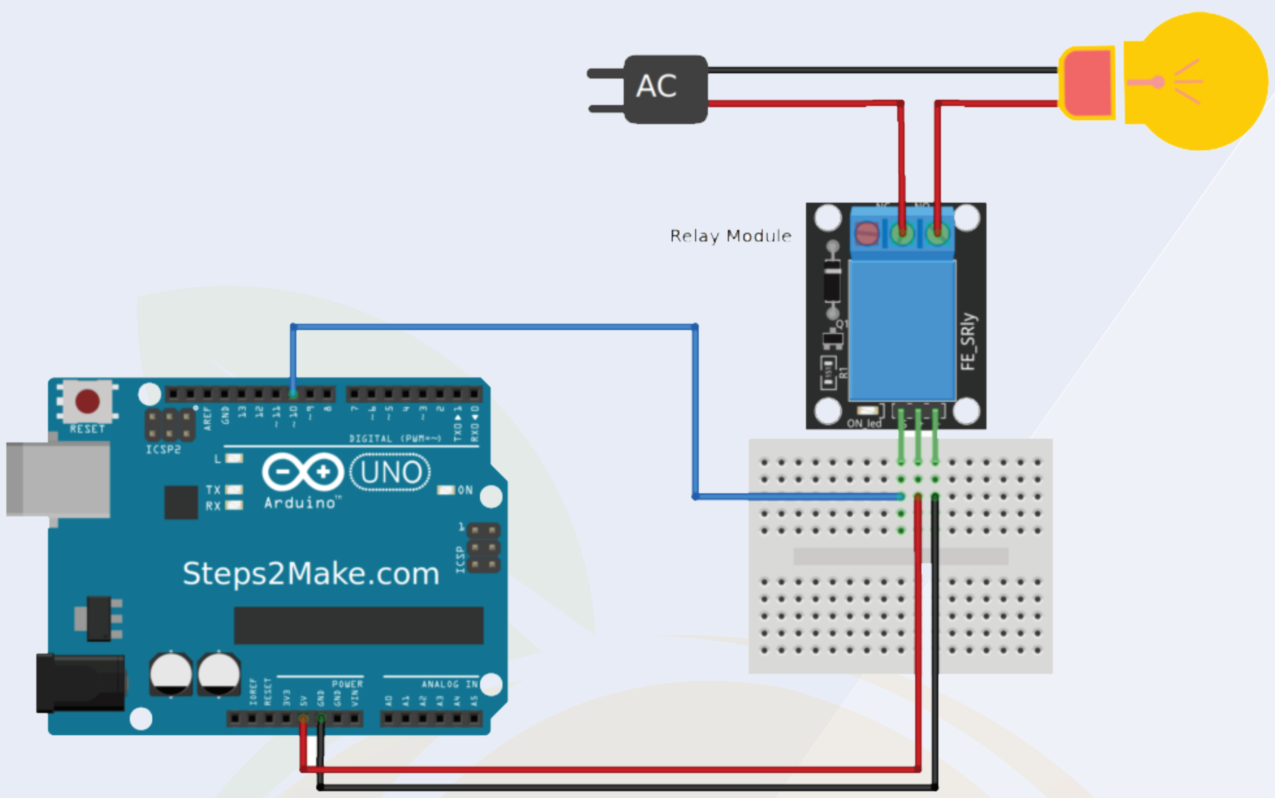
In the above image you can see COM, NC and NO pins. These are connected to the load.

Let's take an example, let's say you want to turn on an LED lamp with the help of the relay module.

In order to turn on the led we will have to connect the positive pin of the Lamp to the mains supply and the negative pin of supply to the common pin of the relay module and to the NO (Normally Open) pin we can connect the negative pin of the LED Lamp.

16.1. Interfacing The Relay Module With The Arduino –

16.2. Interfacing The Relay Module With The Arduino –



16.3. Arduino Code For Controlling the Relay Module –

In this section we will understand the Arduino code that we will use to the relay module.

So, the relay is connected to the Arduino and Arduino is powering the relay module so we will have to use digitalWrite function.

I have added 200 sec below between the execution of two functions. The reason behind adding delay between two functions is, if there is no delay between the relay turning on event and turning off event then we will not understand the difference between these two events.

So, to understand the difference between these two functions, I have used the delay function.

If you don't have any doubts about the code then you can use the following Arduino. If you have any doubts then mention your issues in the comment section below.

16.1. Arduino Code For Controlling the Relay Module –

```
int relay = 8;
volatile byte relayState = LOW;

// PIR Motion Sensor is connected to D2.
int PIRInterrupt = 2;

// Timer Variables
long lastDebounceTime = 0;
long debounceDelay = 10000;

void setup() {
  // Pin for relay module set as output
  pinMode(relay, OUTPUT);
  digitalWrite(relay, HIGH);
  // PIR motion sensor set as an input
  pinMode(PIRInterrupt, INPUT);
  // Triggers detectMotion function on rising mode to turn the relay on, if the condition is met
  attachInterrupt(digitalPinToInterrupt(PIRInterrupt), detectMotion, RISING);
  // Serial communication for debugging purposes
  Serial.begin(9600);
}

void loop() {
  // If 10 seconds have passed, the relay is turned off
  if((millis() - lastDebounceTime) > debounceDelay && relayState == HIGH){
    digitalWrite(relay, HIGH);
    relayState = LOW;
    Serial.println("OFF");
  }
  delay(50);
}

void detectMotion() {
  Serial.println("Motion");
  if(relayState == LOW){
    digitalWrite(relay, LOW);
  }
  relayState = HIGH;
  Serial.println("ON");
  lastDebounceTime = millis();
}
```

17. Reed Switch Module

Reed Switch are one of the contactless switches. These switches are used in many applications such as cars, washing machines but the burglar system is one of the most popular application.

In this section of this blog we will understand working of the reed switch.

We are getting two types of the Reed Switches with the kits I have shown the images of those two modules below please take a look.

So, this two Reed switches we are getting with this kit. In the next section of this blog we will understand the basics of the functions.

The reed switch has two ferromagnetic contacts inside. These contacts are placed two microns apart from each other.

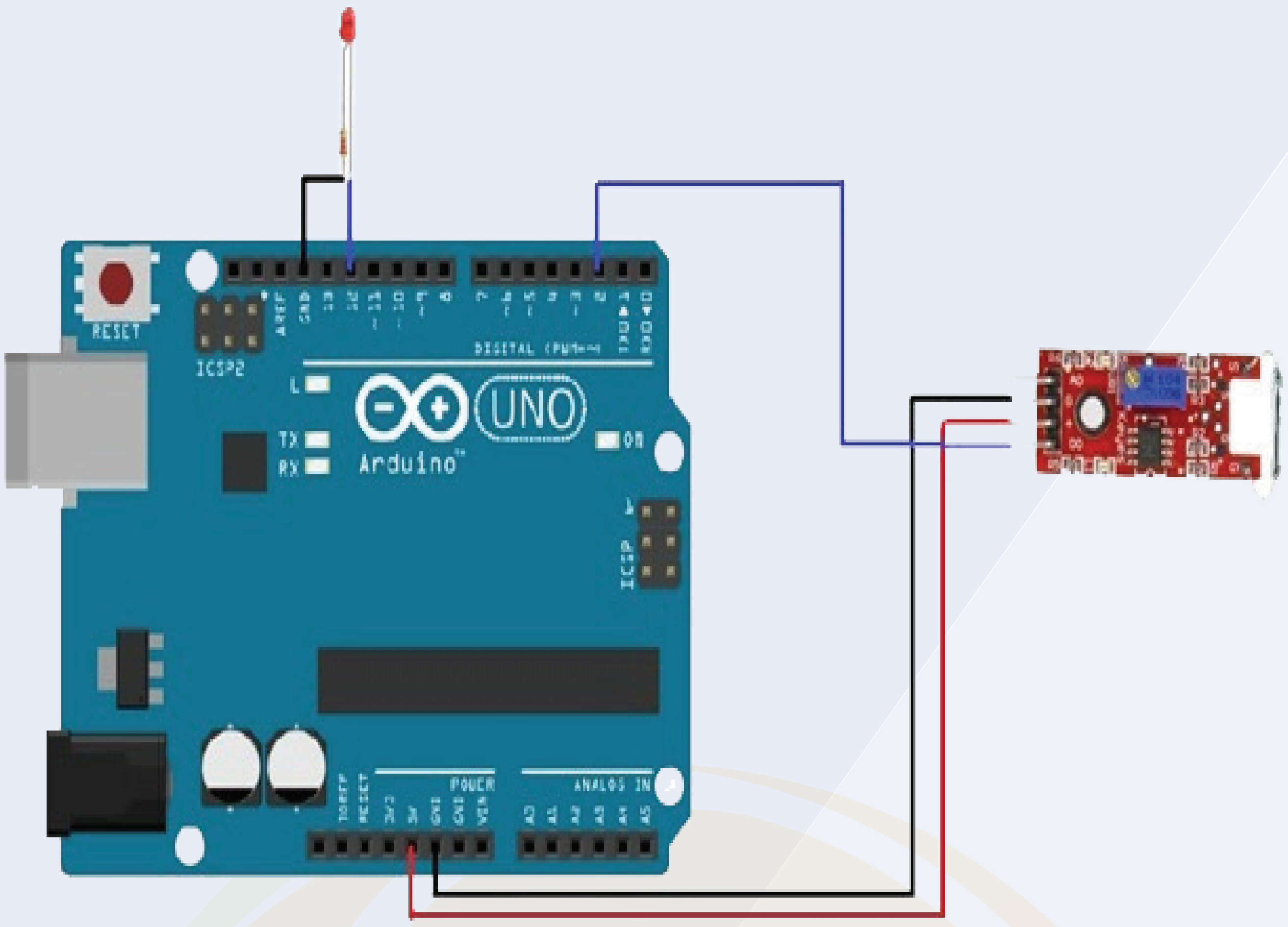
When the magnet is placed near this switch the ferromagnet blades get attracted towards one another.

In this way the connection happens and the switch acts as a closed switch.

16.2. Interfacing The Relay Module With The Arduino –

The reed switch has two pins. You can connect one end of the reed switch to the 5v pin of the Arduino and the other end of the reed switch to any GPIO pin of the Arduino.

The following image shows the interfacing diagram of the reed switch with the Arduino.



16.3. Arduino Code For Reed Switch

In the following Arduino code, we have used digitalRead function to read the output of the sensor.

You can upload the code to your Arduino and can see the output of the code on the serial monitor.

Orange 37 in 1 Sensor Kit

Hello Geek, In this blog, we will talk about Orange 37 in 1 sensor kits. This kit is compatible with all micro-controllers and microprocessors.

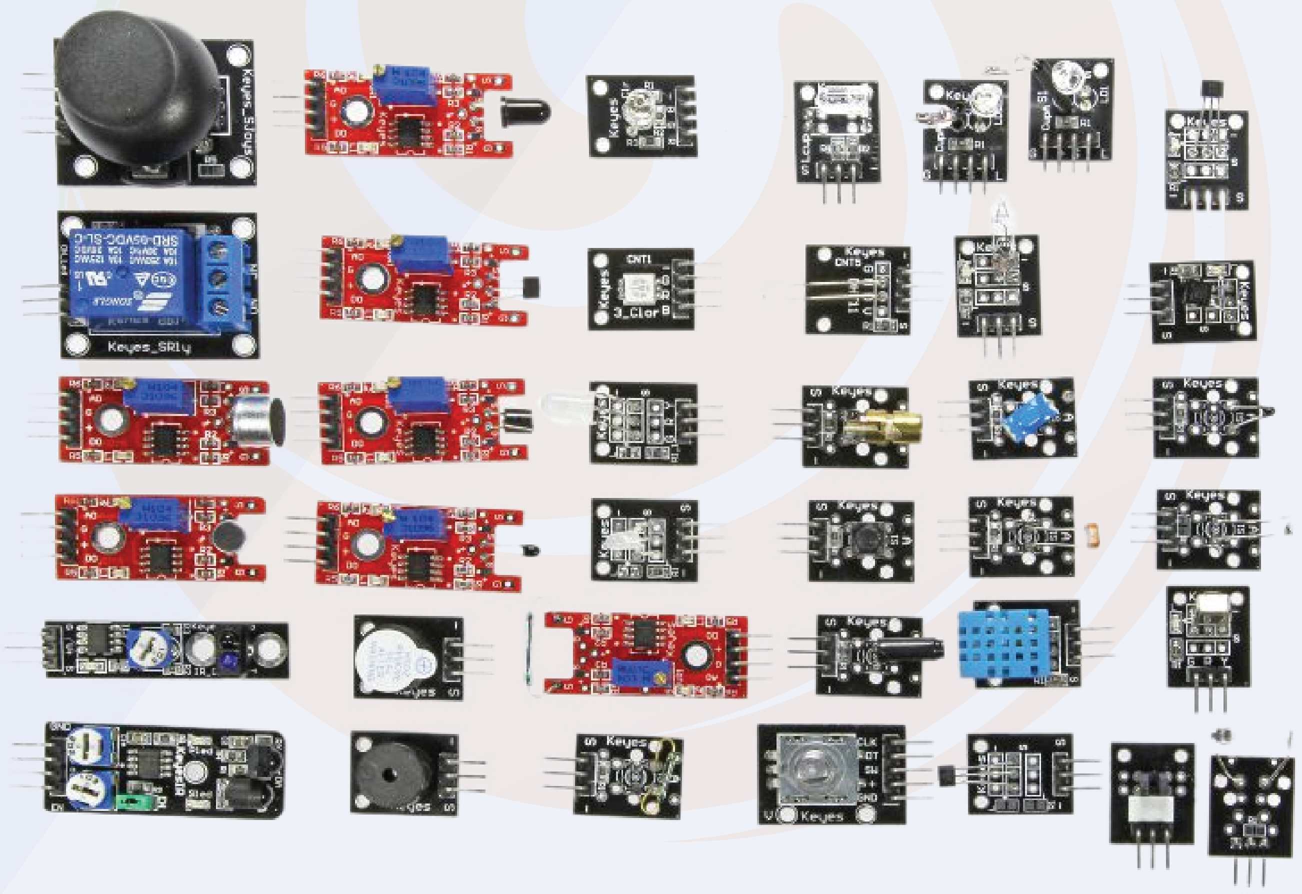
The only concern is the power supply. The sensor you get with this kit has an operating voltage of 5v.

So, if your controller is not capable of handling the 5V input voltage, I suggest you use either a voltage level converter or a resistor divider circuit.

The sensors you get with this kit are compatible with all types of Arduino boards. As I told you earlier you just need to take care of the operating voltage of the sensor.

If you have not bought any Arduino board yet then we request you to buy an Arduino board from here.

The following image shows the list of components we get with this kit. Please take a look



17. Analog and Digital Temperature Sensor

We are getting two temperature sensors with this kit. We can use these two sensors in our IOT projects.

- Analog Temperature Sensor
- Digital Temperature Sensor

The analog temperature sensor has a thermocouple sensor in it, the output of the thermocouple sensor changes as per the change in the temperature.

Talking about the digital temperature sensor, this sensor generates either high voltage or Low Voltage depending on the threshold temperature we set.

This was the basic introduction of the temperature sensor; in the next section of this blog we will discuss about the interfacing of the analog and digital sensor with the Arduino.

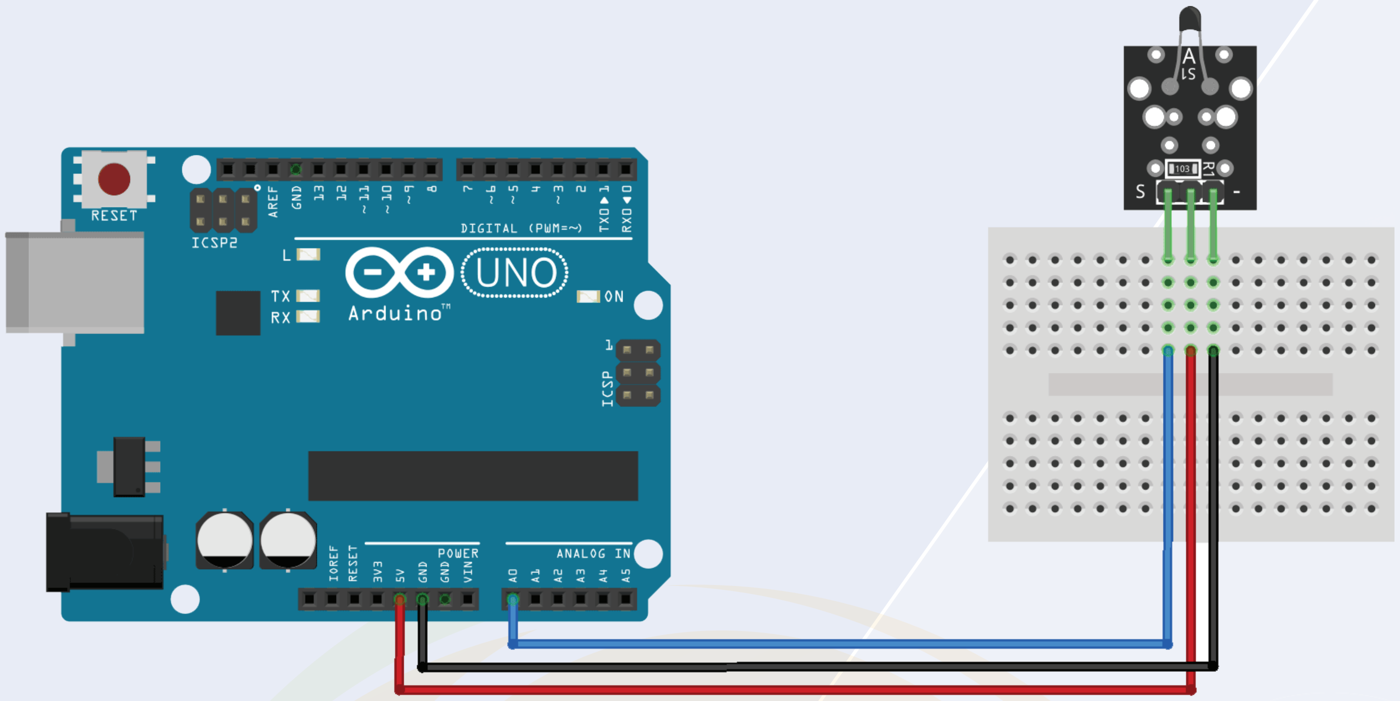
17.2. Interfacing the Temperature sensor with the Arduino

The analog and digital temperature sensor both has three pins. Out of those three pins two pins are power pins and on is signal pin.

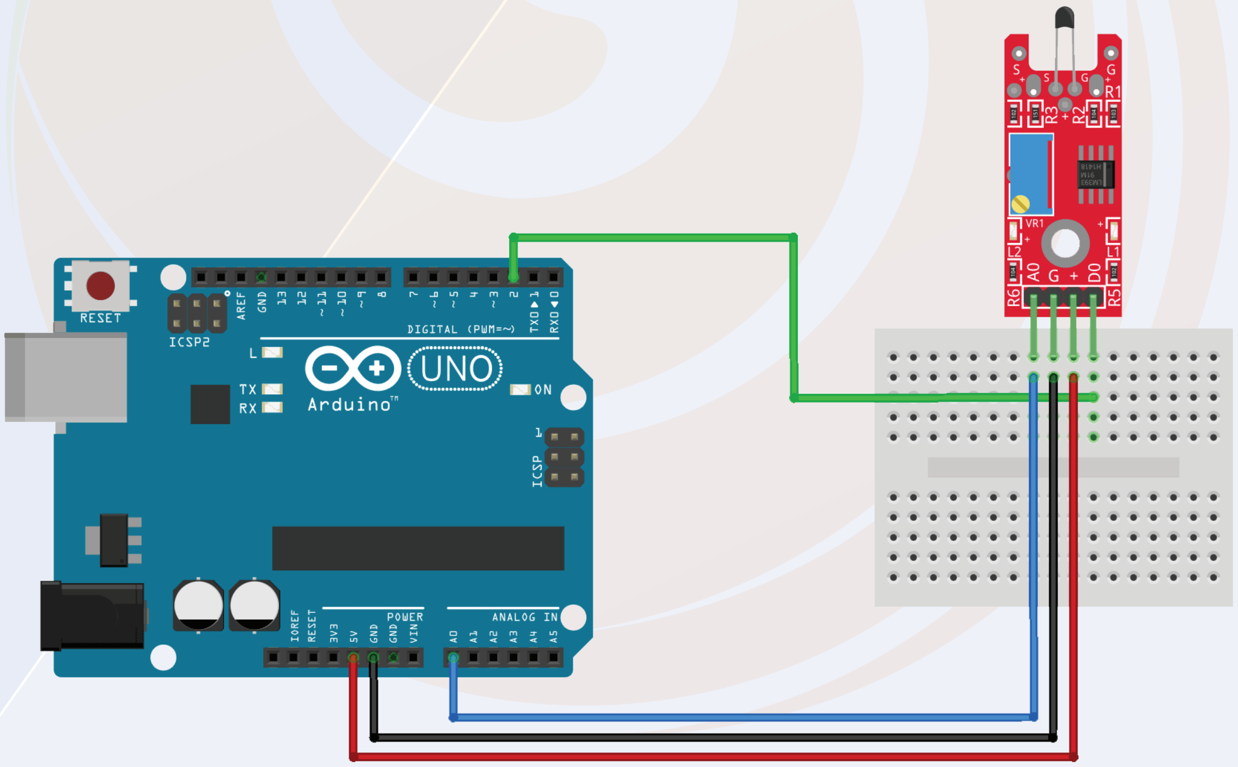
The signal pin of those sensor we can connect to any GPIO pin of the arduiuno.

Please check the following images to understanding the connection diagram.

17.2. Interfacing the analog Temperature sensor with the Arduino



17.2. Interfacing the digital Temperature sensor with the Arduino



17.2. Interfacing the Temperature sensor with the Arduino

We are getting two temperature sensors with this kit. We can use these two sensors in our IOT projects.

- Analog Temperature Sensor
- Digital Temperature Sensor

The analog temperature sensor has a thermocouple sensor in it, the output of the thermocouple sensor changes as per the change in the temperature.

Talking about the digital temperature sensor, this sensor generates either high voltage or Low Voltage depending on the threshold temperature we set.

This was the basic introduction of the temperature sensor; in the next section of this blog we will discuss about the interfacing of the analog and digital sensor with the Arduino.

17.2. Interfacing the Temperature sensor with the Arduino

The analog and digital temperature sensor both has three pins. Out of those three pins two pins are power pins and on is signal pin.

The signal pin of those sensor we can connect to any GPIO pin of the arduiuno.

Please check the following images to understanding the connection diagram.

17.3. Arduino Code For Analog temperature sensor –

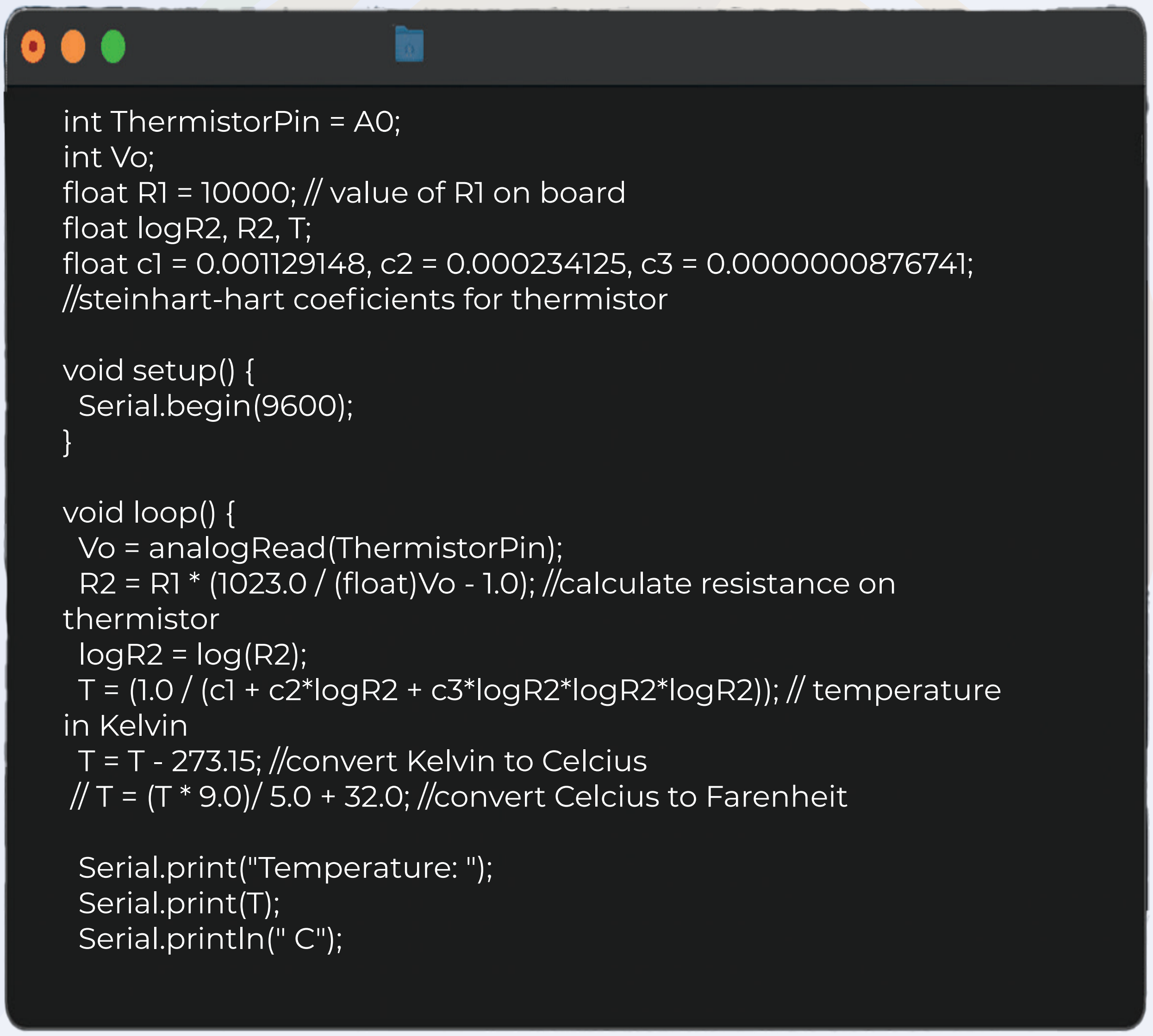
We have learned about the interfacing of the sensor with the Arduino. Now, as we know how the analog and digital sensor work, we can design our logic to read the output of the sensor.

So far, we understood that the analogRead function is used to read the analog values and digitalRead function is used to read the digital output of the sensor.

In the temperature sensor code, we are using both of these functions to get the output of the sensor.

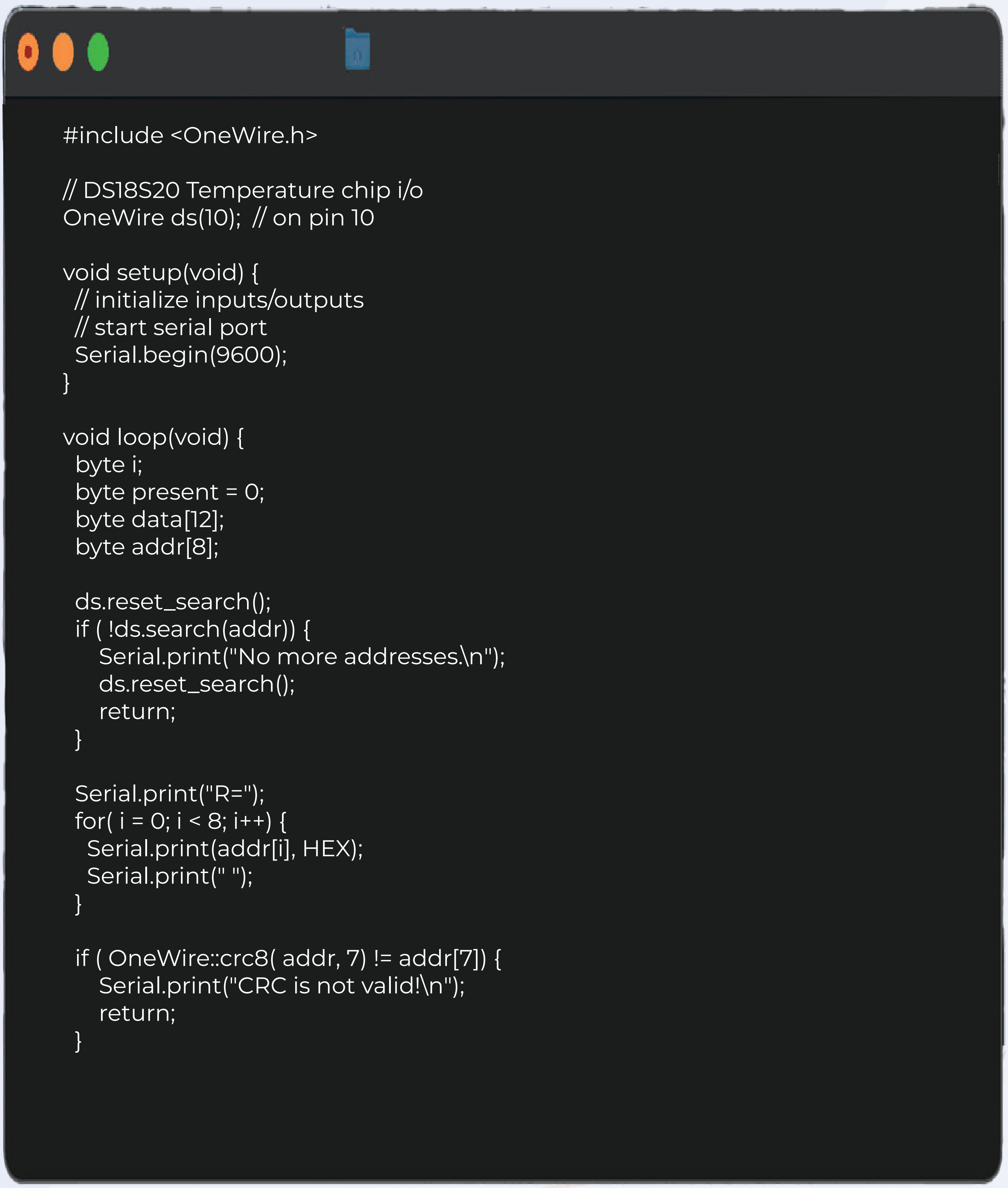
I have shared two code snippets below. One code snippet is for analog temperature sensor and other one is of digital temperature sensor.

Please upload the code to your Arduino and see the output of the sensor on the serial monitor.



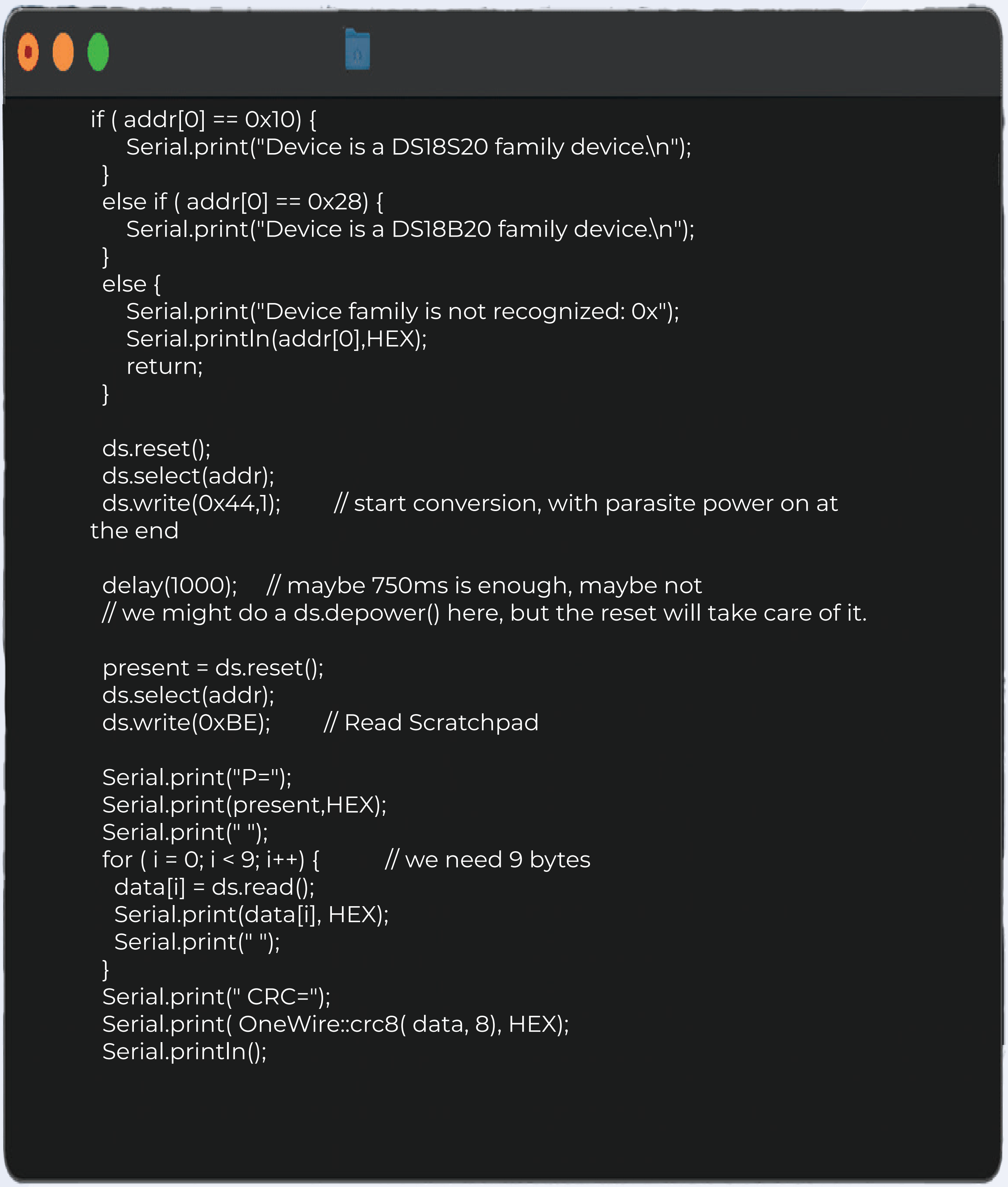
```
int ThermistorPin = A0;
int Vo;
float R1 = 10000; // value of R1 on board
float logR2, R2, T;
float c1 = 0.001129148, c2 = 0.000234125, c3 = 0.0000000876741;
//steinhart-hart coeficients for thermistor

void setup() {
  Serial.begin(9600);
}

void loop() {
  Vo = analogRead(ThermistorPin);
  R2 = R1 * (1023.0 / (float)Vo - 1.0); //calculate resistance on
thermistor
  logR2 = log(R2);
  T = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2)); // temperature
in Kelvin
  T = T - 273.15; //convert Kelvin to Celcius
 // T = (T * 9.0)/ 5.0 + 32.0; //convert Celcius to Farenheit

  Serial.print("Temperature: ");
  Serial.print(T);
  Serial.println(" C");
```

17.3. Arduino Code For Digital temperature sensor –



```
#include <OneWire.h>

// DS18S20 Temperature chip i/o
OneWire ds(10);  // on pin 10

void setup(void) {
  // initialize inputs/outputs
  // start serial port
  Serial.begin(9600);
}

void loop(void) {
  byte i;
  byte present = 0;
  byte data[12];
  byte addr[8];

  ds.reset_search();
  if ( !ds.search(addr)) {
      Serial.print("No more addresses.\n");
      ds.reset_search();
      return;
  }

  Serial.print("R=");
  for( i = 0; i < 8; i++) {
    Serial.print(addr[i], HEX);
    Serial.print(" ");
  }

  if ( OneWire::crc8( addr, 7) != addr[7]) {
      Serial.print("CRC is not valid!\n");
      return;
  }
```

17.3. Arduino Code For digitaltemperature sensor –



```
if ( addr[0] == 0x10) {
      Serial.print("Device is a DS18S20 family device.\n");
  }
  else if ( addr[0] == 0x28) {
      Serial.print("Device is a DS18B20 family device.\n");
  }
  else {
      Serial.print("Device family is not recognized: 0x");
      Serial.println(addr[0],HEX);
      return;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44,1);         // start conversion, with parasite power on at
the end

  delay(1000);     // maybe 750ms is enough, maybe not
  // we might do a ds.depower() here, but the reset will take care of it.

  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);         // Read Scratchpad

  Serial.print("P=");
  Serial.print(present,HEX);
  Serial.print(" ");
  for ( i = 0; i < 9; i++) {           // we need 9 bytes
    data[i] = ds.read();
    Serial.print(data[i], HEX);
    Serial.print(" ");
  }
  Serial.print(" CRC=");
  Serial.print( OneWire::crc8( data, 8), HEX);
  Serial.println();
```

18. DHT11 Temperature and Humidity Sensor Module

If you are working on the IOT project then this sensor will help you in gathering the surrounding atmosphere. This sensor has inbuilt capacitive temperature sensor and thermometer in it.

The DHT11 sensor is designed in such way that it will hold the moisture. When the water vapor gets collected on the subtract of the sensor. The ions will be released by the subtract. Increase in the ions will reduce the resistance between two electrodes.

Now as the resistance is decreasing between two electrodes, we will get the variations in the voltage readings. Based on the variations we received from the sensor we can calculate the humidity.

Talking about the temperature sensor, DHT11 Sensor has NTC type thermistor in it. This thermistor is nothing but a temperature sensor.

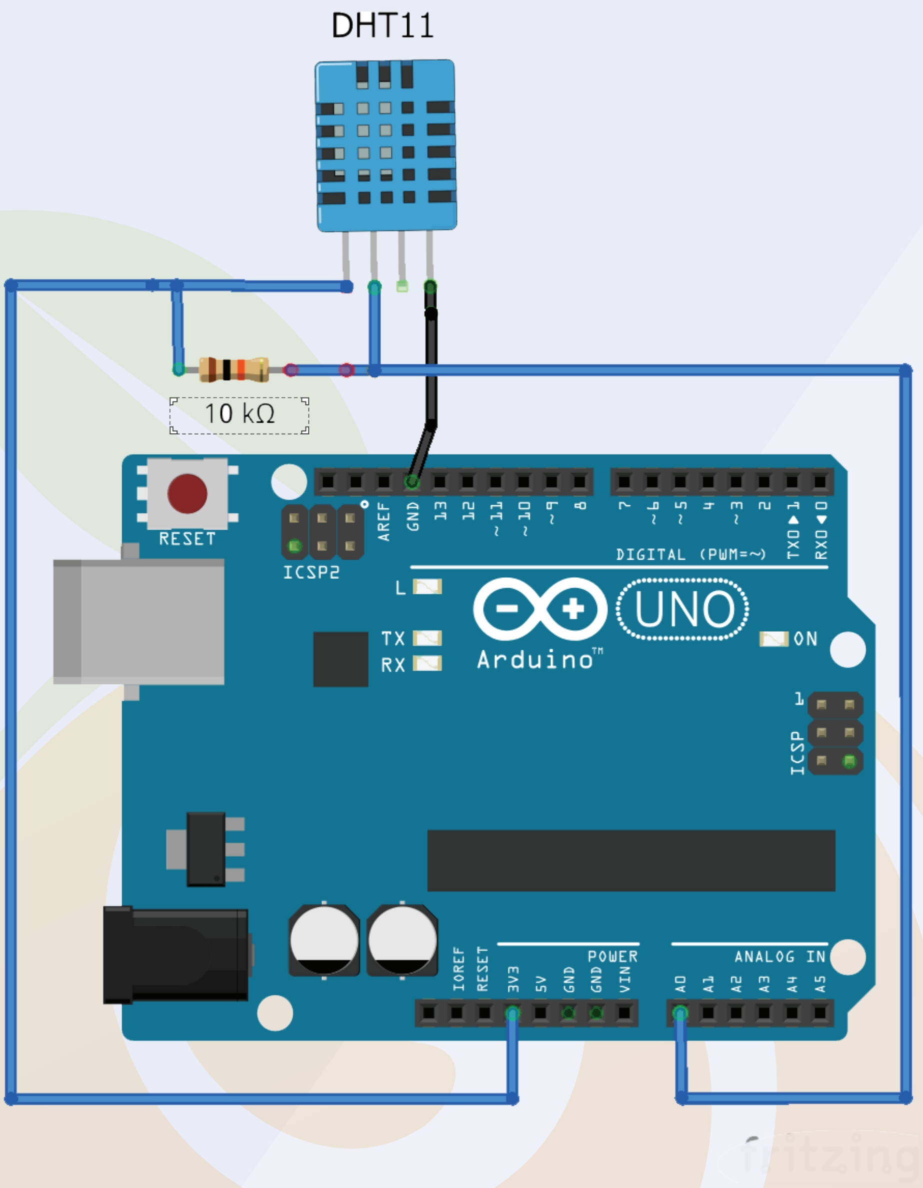
The output value of the thermistor changes as per the change in the surrounding temperature. So by calculating the change in the output voltage of the sensor we can find out the reading of the temperature.

18.3. Interfacing DHT11 Sensor With Arduino

In the following section I have shown the interfacing diagram of the DHT11 sensor with the Arduino.

The DHT11 Sensor has three pins. You can connect those three pins to the Arduino as shown in the below figure.

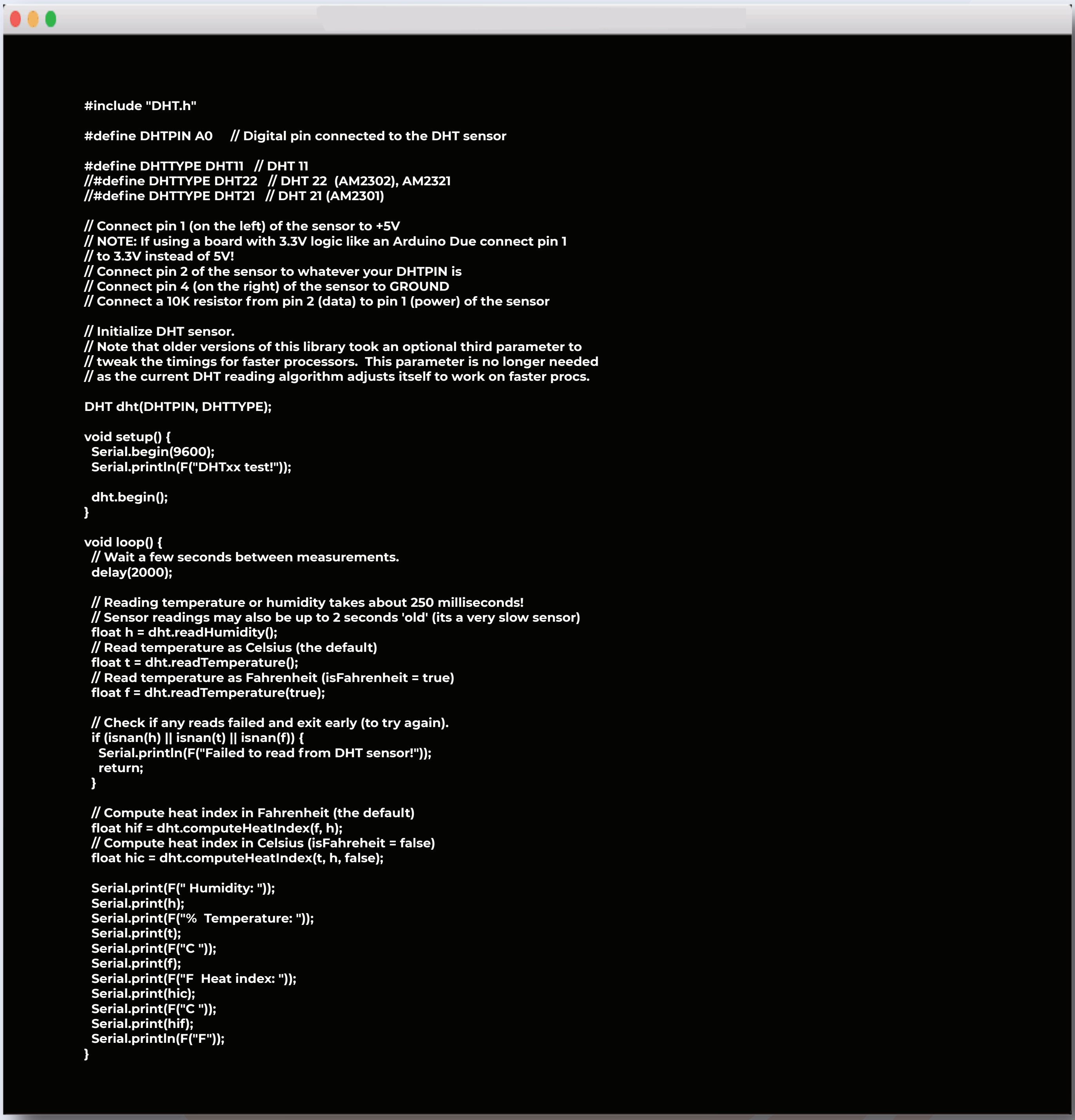
18.3. Interfacing DHT11 Sensor With Arduino



In the following section I have shown the interfacing diagram of the DHT11 sensor with the Arduino.

The DHT11 Sensor has three pins. You can connect those three pins to the Arduino as shown in the below figure.

18.4. Arduino Code DHT11 Interfacing With Arduino



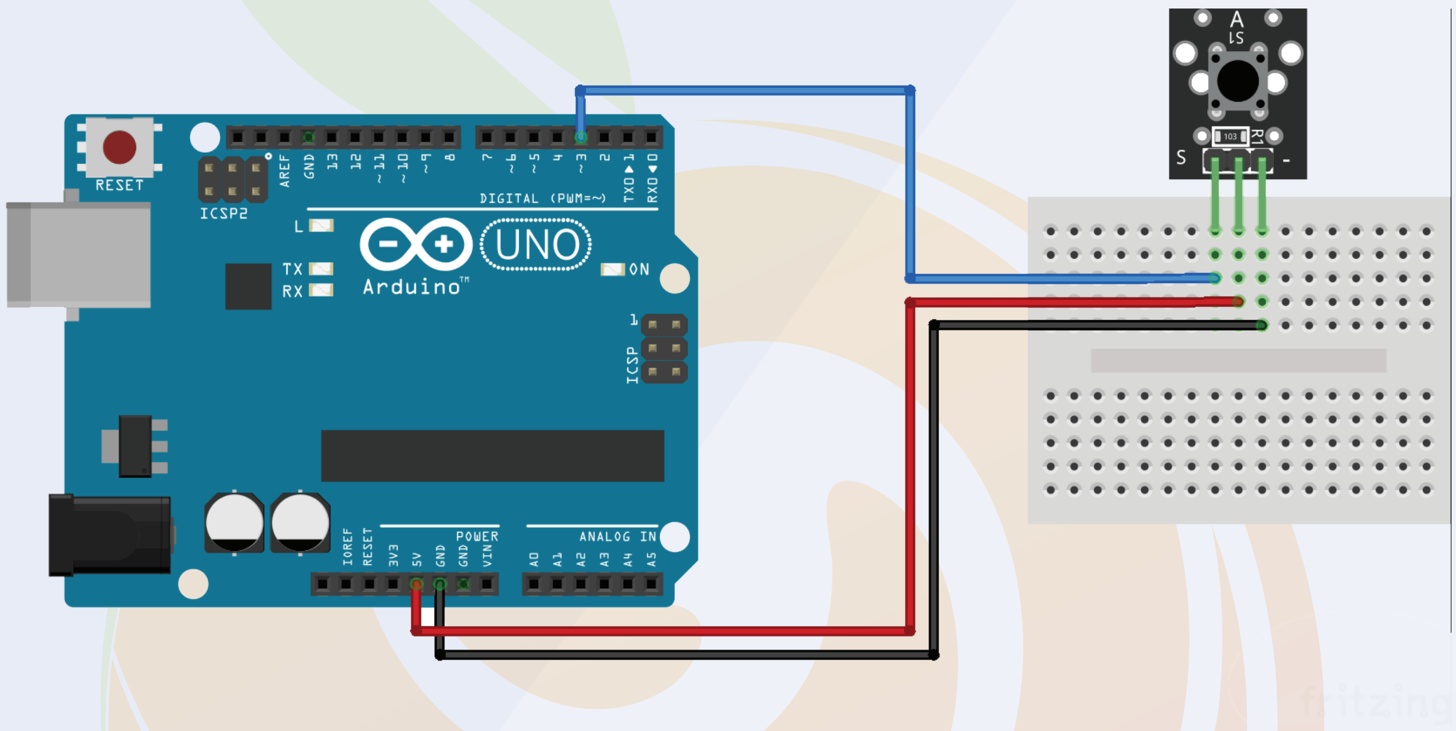
```
#include "DHT.h"

#define DHTPIN A0     // Digital pin connected to the DHT sensor

#define DHTTYPE DHT11   // DHT 11
//#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
//#define DHTTYPE DHT21   // DHT 21 (AM2301)

// Connect pin 1 (on the left) of the sensor to +5V
// NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
// to 3.3V instead of 5V!
// Connect pin 2 of the sensor to whatever your DHTPIN is
// Connect pin 4 (on the right) of the sensor to GROUND
// Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor

// Initialize DHT sensor.
// Note that older versions of this library took an optional third parameter to
// tweak the timings for faster processors.  This parameter is no longer needed
// as the current DHT reading algorithm adjusts itself to work on faster procs.

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println(F("DHTxx test!"));

  dht.begin();
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  // Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print(F(" Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("C "));
  Serial.print(f);
  Serial.print(F("F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("C "));
  Serial.print(hif);
  Serial.println(F("F"));
}
```

19. Switch Module

We are getting the switch module with this kit. If your working on a project that needs switch then you can use this module.

This switch module has three pins. Out of those three pins two pins are power pins and one pin is a signal pin.

In the following image I have shown the interfacing diagram of the switch please take a look and if there is any doubt please let us know in the comment section.



19.1. Arduino Code for Switch Module

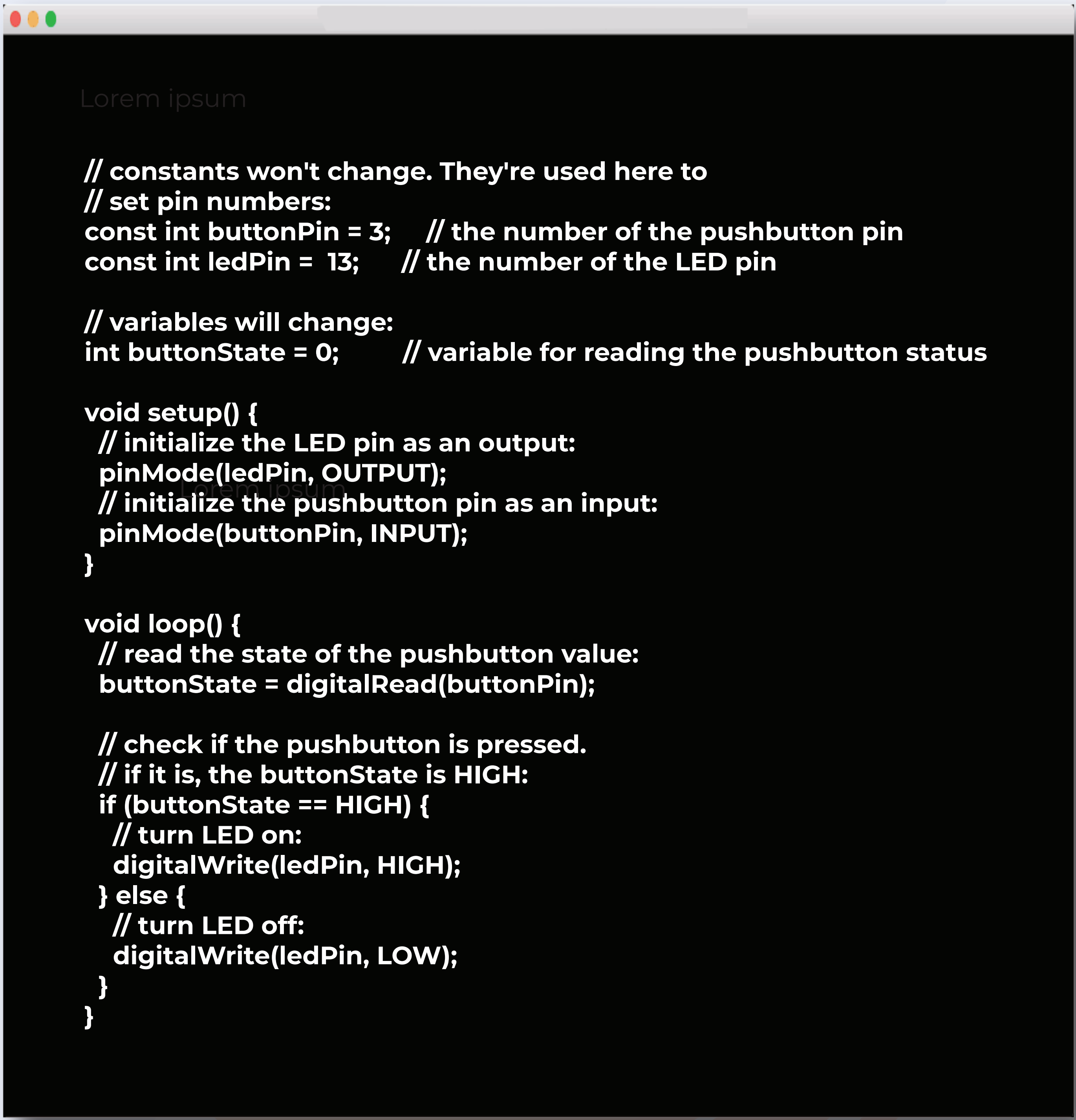
When we press the switch the switch module will generate HIGH level signal.

So, if we want to read the output of the sensor then we will have to use the digitalRead function.

In the following code you can see we have used the digitalRead function and then printing it on the serial monitor.

Please use the code below to test the Switch module

19.1. Arduino Code For Switch Module



```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 3;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

We are getting a Mercury Switch Module with this product. The switch can be used for measuring the tilt.

Talking about the construction of this switch, it has two electric contacts enclosed in a cylindrical shape made up of glass material. When the switch is tilted the mercury that is inside the glass envelop material gets connected to the terminals of the switch and the switch act as a closed switch.

Usually in normal switches, when we press them or turn them on a little arcing can be seen that arcing produces EMI (Electromagnetic interference) in the electronic circuits.

But in the case of mercury switch, the arcing is absorbed by the mercury. So, the use of mercury switch modules reduces the effect of the (Electromagnetic interference) EMI and that is why it is most favorable switch module in the applications where EMI is the most concern.

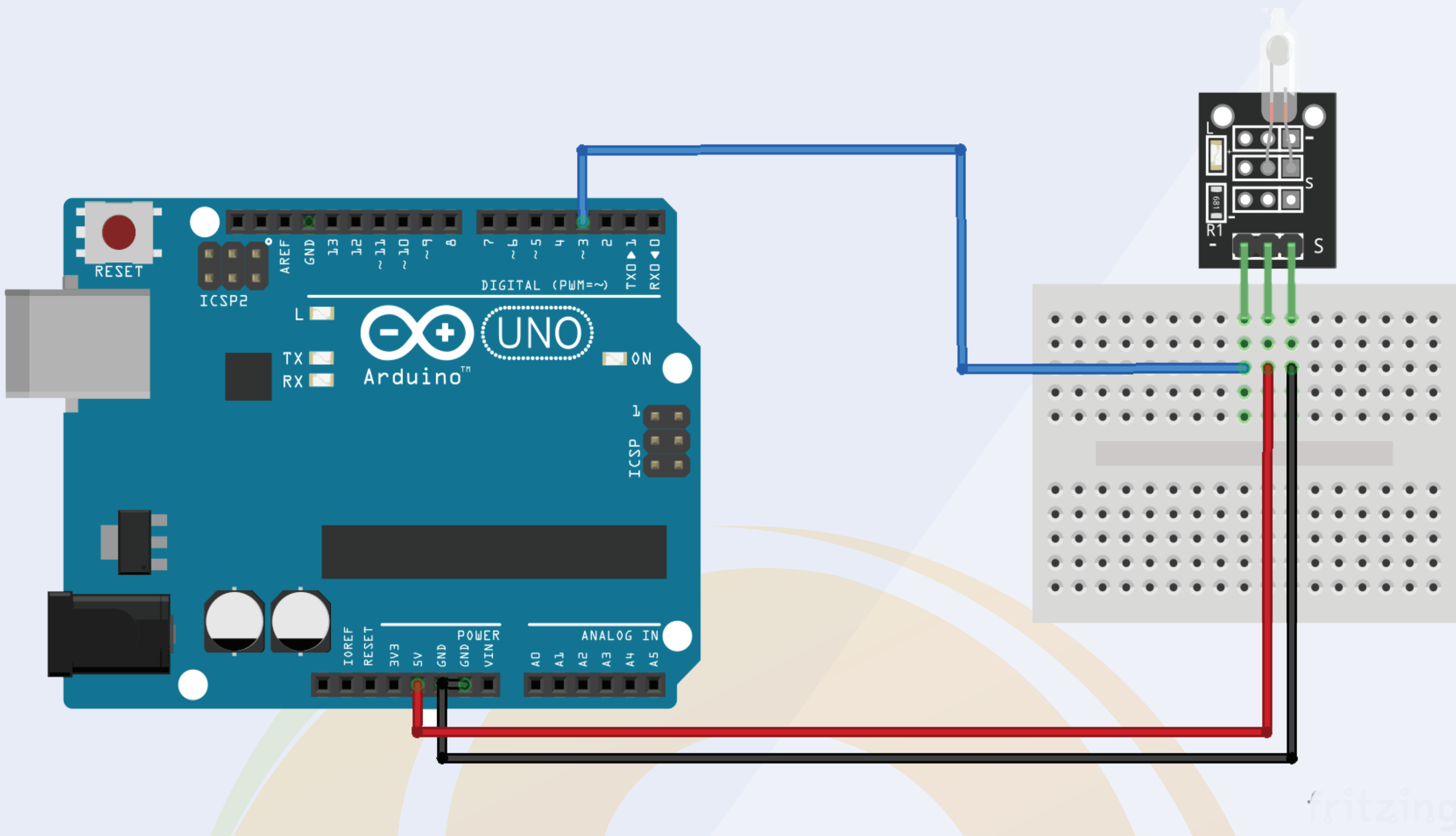
20.1. Interfacing Mercury-switch Module With The Arduino

Talking about the connection diagram of the mercury-switch module, the mercury-switch module has three pins that we are getting with the kit only has three pins.

- VCC
- GND
- Signal

VCC and the GND pins are the power pins of the sensor and the signal pin we can connect to any GPIO of the Arduino.
Please see the following Image to understand the working of the Mercury-Switch Module.

20.1. Interfacing Mercury-switch Module With The Arduino



20.2. Arduino Code for Mercury Switch Module

As discussed earlier, when we will tilt the sensor, the sensor will produce high level signal and if we are using an Arduino then to read the output of the sensor, we will have to use digitalRead function.

In the following code you can see we are using digitalRead function and storing the output of the sensor to the variable.

As we are printing the output of the sensor on the serial monitor, we will have to open the serial monitor to see the output.

20.3 Arduino Code for Mercury Switch Module

```
int led_pin = 13; // Define the LED interface
int switch_pin = 3; // Definition of mercury tilt switch sensor inter-
face
int val; // Defines a numeric variable

void setup()
{
        pinMode(led_pin, OUTPUT);
        pinMode(switch_pin, INPUT);
}

void loop()
{
        val = digitalRead(switch_pin); // check mercury switch state
        if(val == HIGH)
        {
                digitalWrite(led_pin, HIGH);
        }
        else
        {
                digitalWrite(led_pin, LOW);
        }
}
```

21. Magic Cup Module

The Working principle of the magic cup module is similar to the Mercury switch module. The only difference is, you get Two LEDs on this module.

Those LEDs are used for indication purpose. When you will tilt the sensor, the brightness of one LED will start increasing and the brightness of other led start decreasing.

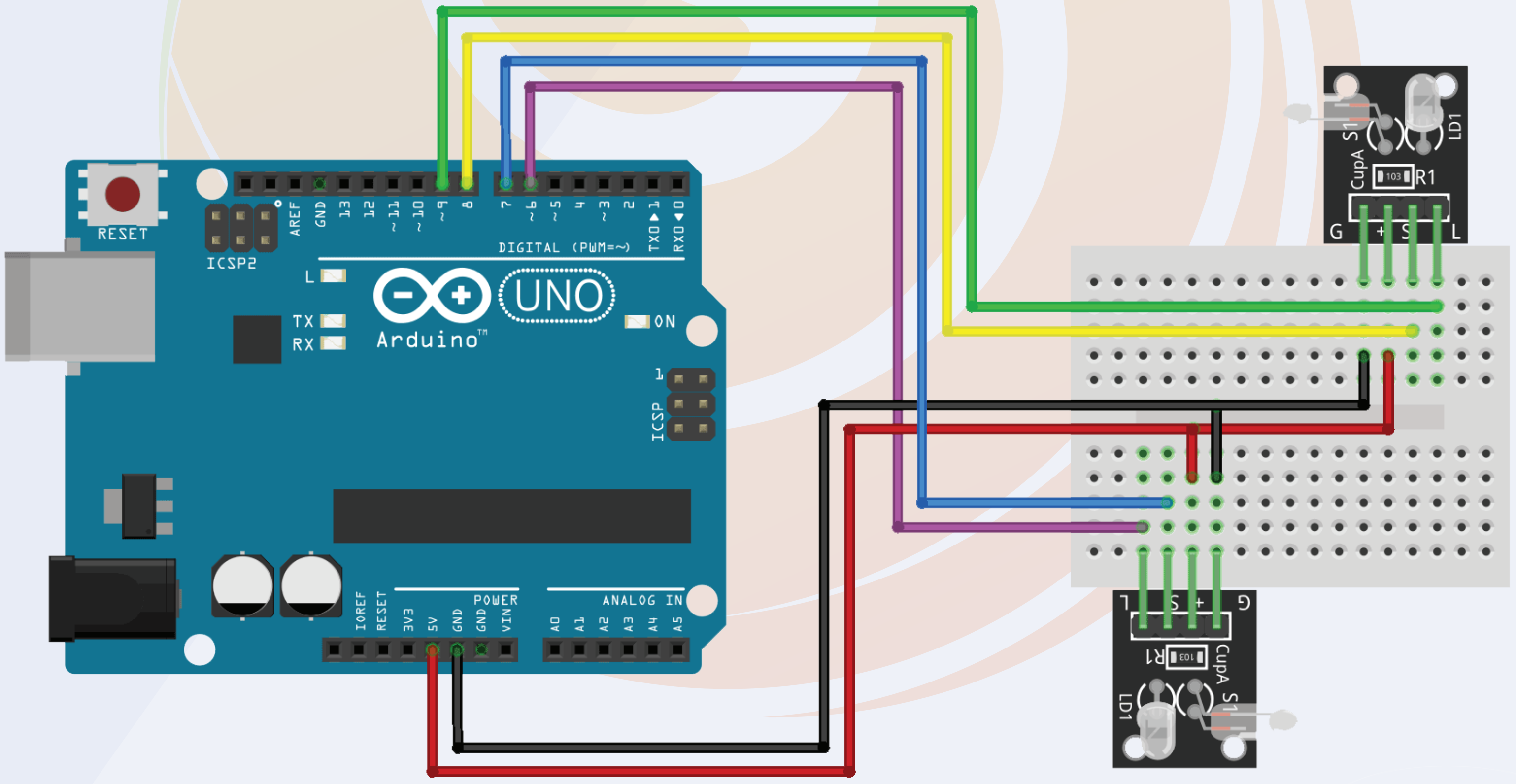
So, this was the basic introduction of the Magic cup module with the Arduino. In the next section of this blog we will understand the interfacing of the magic cup module.

21.1. Interfacing of the Magic Cup Module

As we have connected the magic cup module to the Arduino, now we can write our code for getting the output from the module.

As discussed earlier, we will get the sensor output on the signal pin. In the following code, we have used a variable to store the output of the sensor.

In the code below, we are continuously checking the output of the sensor and once the change in the state of the variable is occurring, we are tuning on the LED.



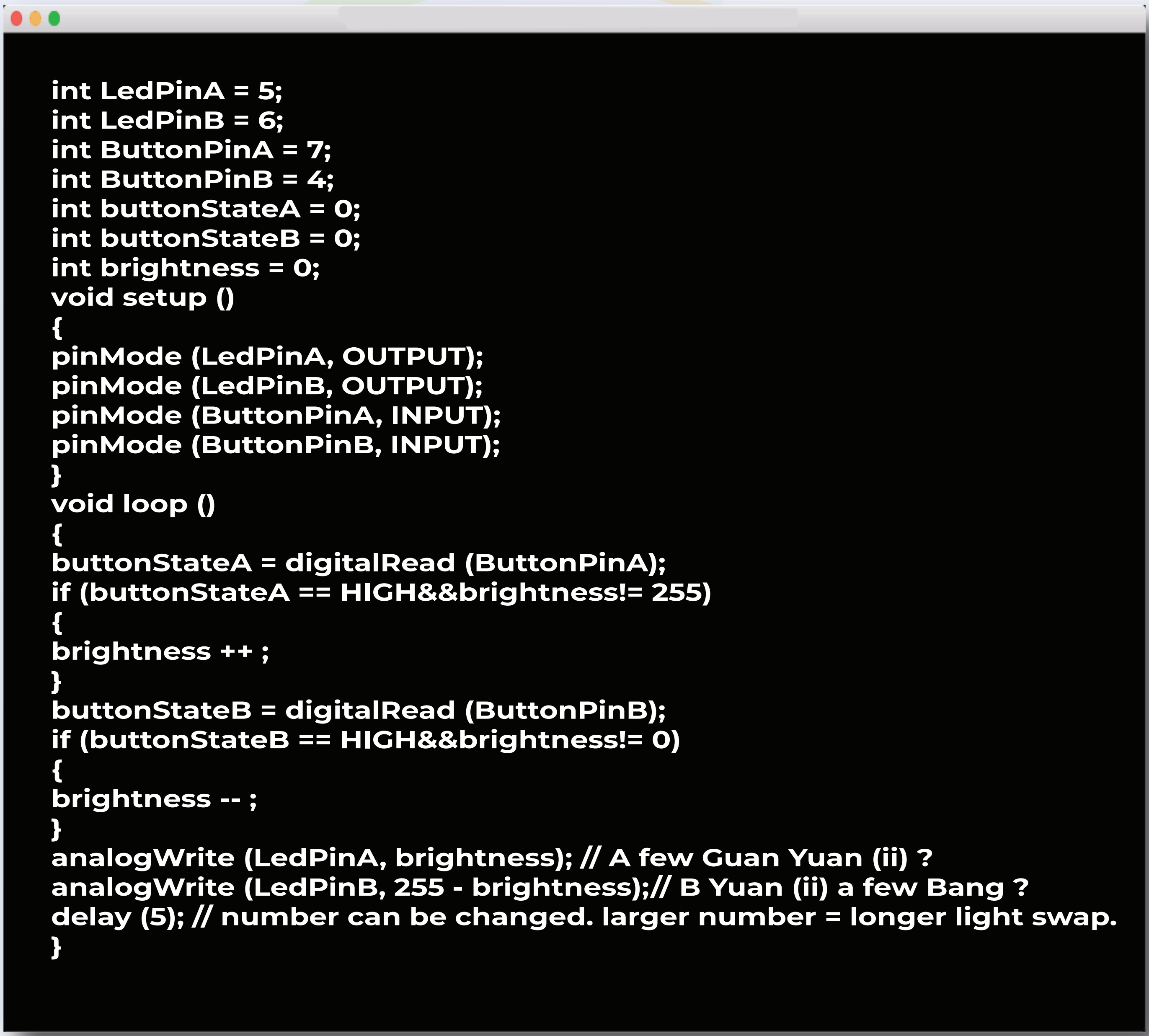
21.2. Arduino Code For Magic Cup Module

As we have connected the magic cup module to the Arduino, now we can write our code for getting the output from the module.

As discussed earlier, we will get the sensor output on the signal pin. In the following code, we have used a variable to store the output of the sensor.

In the code below, we are continuously checking the output of the sensor and once the change in the state of the variable is occurring, we are tuning on the LED.

Please take a look at the following code and if you have any doubts please let us know.



```
int LedPinA = 5;
int LedPinB = 6;
int ButtonPinA = 7;
int ButtonPinB = 4;
int buttonStateA = 0;
int buttonStateB = 0;
int brightness = 0;
void setup ()
{
pinMode (LedPinA, OUTPUT);
pinMode (LedPinB, OUTPUT);
pinMode (ButtonPinA, INPUT);
pinMode (ButtonPinB, INPUT);
}
void loop ()
{
buttonStateA = digitalRead (ButtonPinA);
if (buttonStateA == HIGH&&brightness!= 255)
{
brightness ++ ;
}
buttonStateB = digitalRead (ButtonPinB);
if (buttonStateB == HIGH&&brightness!= 0)
{
brightness -- ;
}
analogWrite (LedPinA, brightness); // A few Guan Yuan (ii) ?
analogWrite (LedPinB, 255 - brightness);// B Yuan (ii) a few Bang ?
delay (5); // number can be changed. larger number = longer light swap.
}
```

22. Microphone Sound Sensor

We are getting two types of sound sensors with this kit.

The working of these sound sensors is same. In the following section we will understand the working of the sound sensor.

Sound sensor are designed for converting analog signals to digital signals. They convert any sound signals into digital signals.

We can see these sensors in everywhere. These sensors act as an ear of the electronic circuits. But do you know how they work?

These sensors consist diaphragm, a sort of piezoelectric transducer that converts sound frequency into electrical signals.

When the sound frequency hit on this sensor, the vibrations will be produced on the surface of the diaphragm and then based on the intensity of the sound signals that diaphragm produces the voltage.

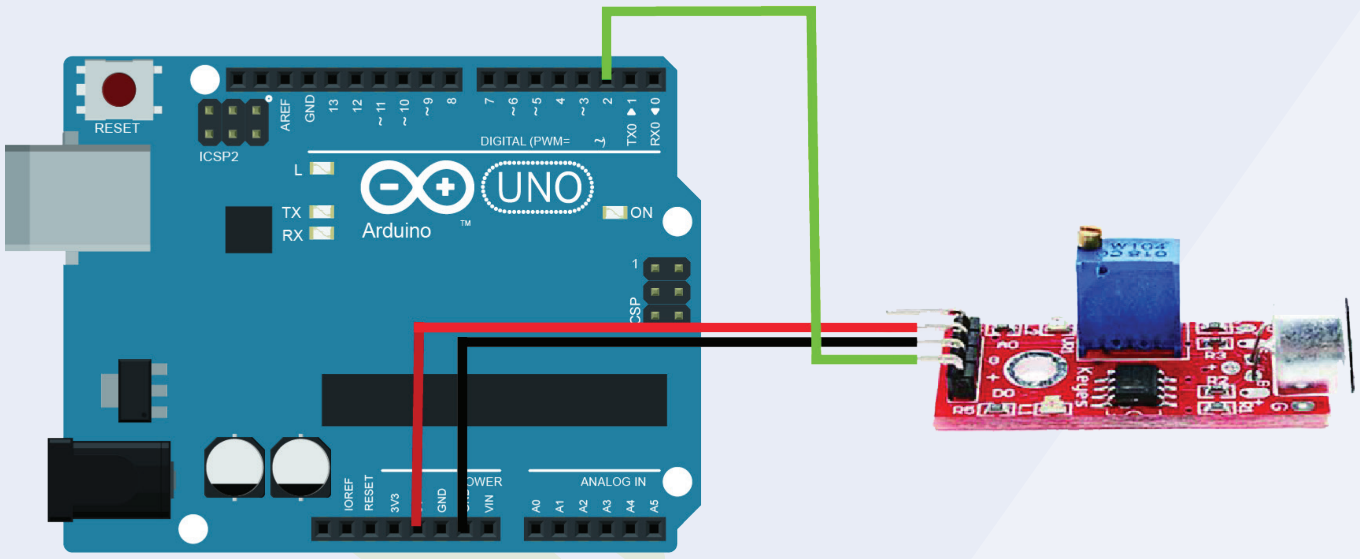
This how the sound sensor converts the sound frequency into electrical signals.

22.1. Interfacing of the sound sensor with the Arduino

The sound sensor has three pins and we get the output of the sensor on the signal pin of the module.

We have connected the signal pin of the module to the seven number GPIO pin of the Arduino.

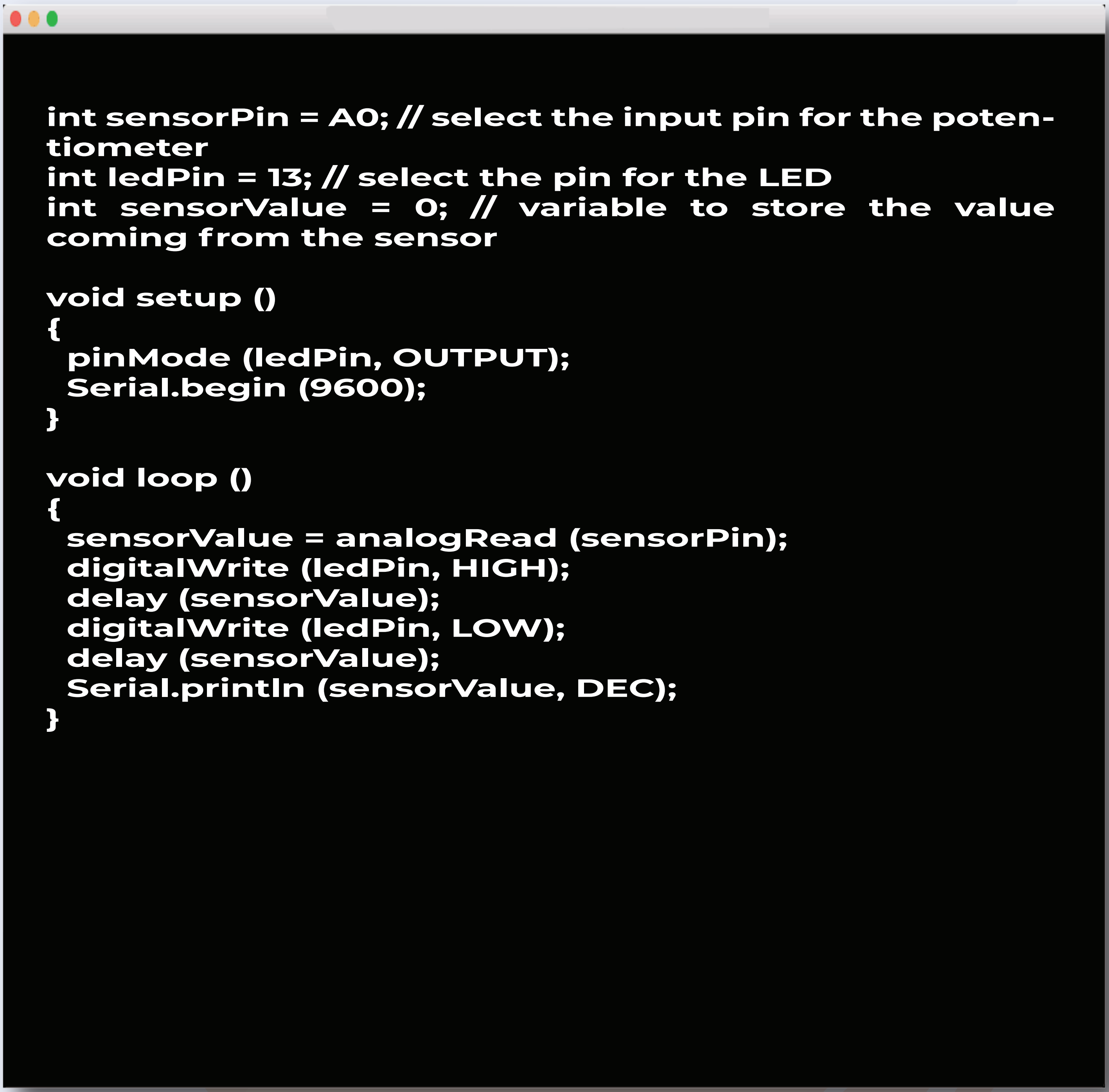
22.1. Interfacing of the sound sensor with the Arduino



22.2. Arduino Code For Digital Microphone Sensor

```
int Led = 13 ;// define LED Interface
int buttonpin = 2; // define D0 Sensor Interface
int val = 0;// define numeric variables val

void setup ()
{
  pinMode (Led, OUTPUT) ;// define LED as output interface
  pinMode (buttonpin, INPUT) ;// output interface D0 is de-
fined sensor
}

void loop ()
{
  val = digitalRead(buttonpin);// digital interface will be as-
signed a value of pin 3 to read val
  if (val == HIGH) // When the sound detection module de-
tects a signal, LED flashes
  {
    digitalWrite (Led, HIGH);
  }
  else
  {
    digitalWrite (Led, LOW);
  }
}
```

22.2. Arduino Code For Analog Microphone Sensor



```
int sensorPin = A0; // select the input pin for the poten-
tiometer
int ledPin = 13; // select the pin for the LED
int sensorValue = 0; // variable to store the value
coming from the sensor

void setup ()
{
  pinMode (ledPin, OUTPUT);
  Serial.begin (9600);
}

void loop ()
{
  sensorValue = analogRead (sensorPin);
  digitalWrite (ledPin, HIGH);
  delay (sensorValue);
  digitalWrite (ledPin, LOW);
  delay (sensorValue);
  Serial.println (sensorValue, DEC);
}
```

23. Broken LED Module/ Photocoupler

We are getting this very useful module with this kit. This type of switches is mostly used in 3D printing machines and the applications where electronic interference issues are biggest concern.

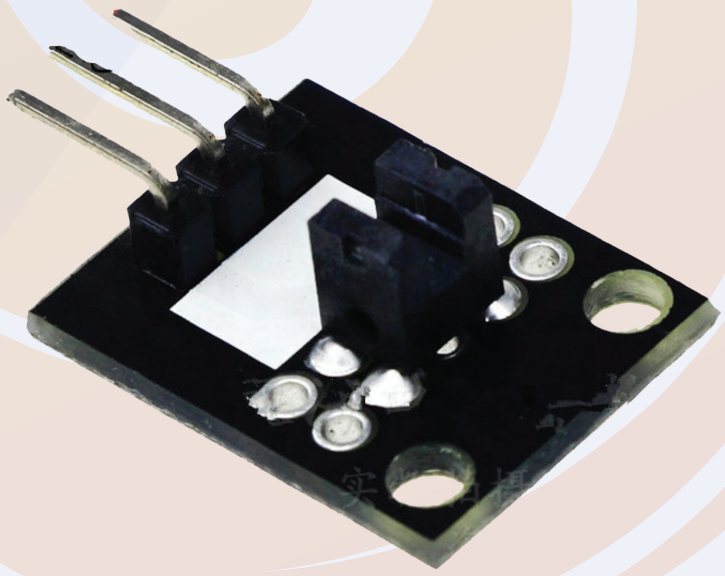
These switches consist an optical emitter and an optical receiver.

Talking about the optical emitter, it is nothing but LED which continuously emits the light and the optical receiver which is placed just in front of the optical emitter, the keep receiving the light that is being transmitted by the optical emitter.

The optical receiver is nothing but a photodiode. When the light rays fall on the photodiode the barrier gap between PN junction of the photodiode starts decreasing and at one point that barrier gap becomes zero and the photodiode acts as a switch.

In this way, until the light is falling on the photodiode the photodiode will be operating as a closed switch and when the light source is gone then it acts as an open switch.

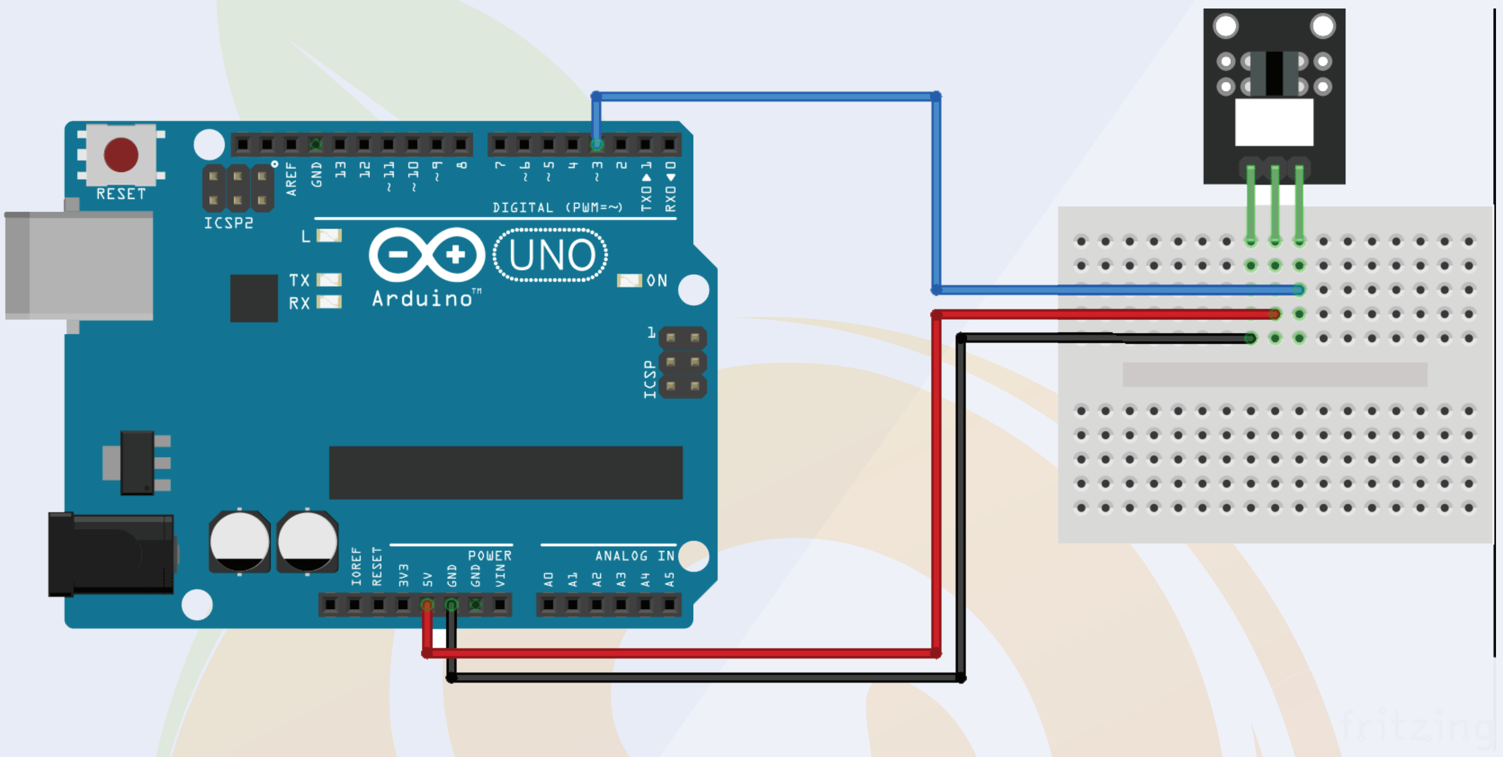
Please check the following image for better understanding.



23.1. Interfacing of the Broken LED Module With the LED

In the following image I have shown the interfacing of the LED with the Arduino. The Broken LED module has three pins. Out of those three pins S is the signal pin. I have connected that pin to the seven number pin of the Arduino.

Please see the following image to understand the connection diagram.



23.2. Arduino Code For Broken LED Module

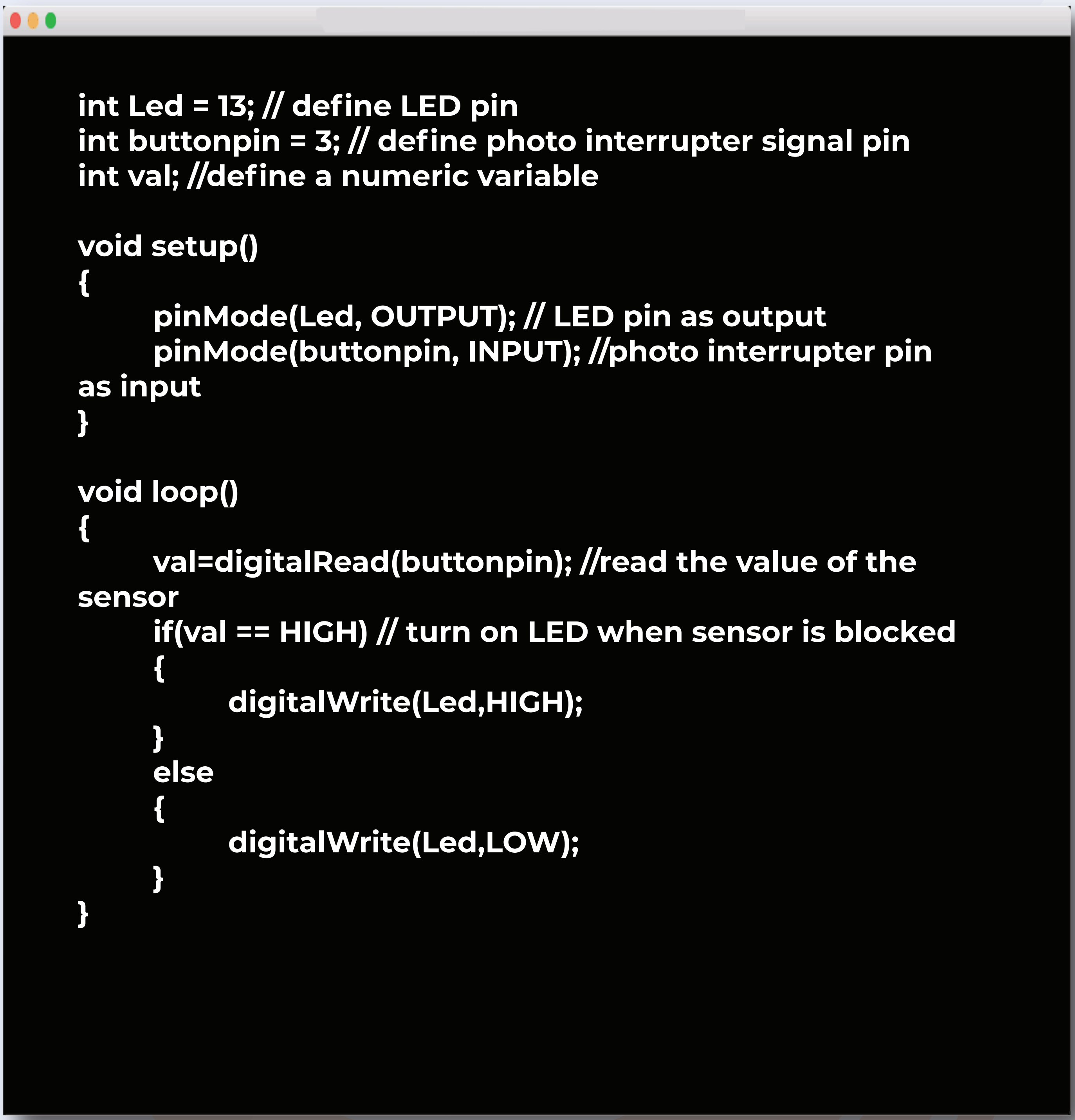
In this section we are discussing about the code for the broken light LED. As we have discussed earlier the broken light LED Module either produces HIGH voltage or LOW voltage.

So, to read the output of the sensor we will have to use the digitalRead function. In the code below we have used digitalRead function and we are printing the output of the sensor on the serial monitor.

You can see the output of the sensor by opening serial monitor.

Initially when there will be no obstacle between the optical emitter and optical receiver, we will see 'Switch is on signal' on the screen but when we put obstacle between the optical emitter and optical receiver then we will see 'switch is off' on the screen.

23.2. Arduino Code For Broken LED Module



```
int Led = 13; // define LED pin
int buttonpin = 3; // define photo interrupter signal pin
int val; //define a numeric variable

void setup()
{
        pinMode(Led, OUTPUT); // LED pin as output
        pinMode(buttonpin, INPUT); //photo interrupter pin
as input
}

void loop()
{
        val=digitalRead(buttonpin); //read the value of the
sensor
        if(val == HIGH) // turn on LED when sensor is blocked
        {
                digitalWrite(Led,HIGH);
        }
        else
        {
                digitalWrite(Led,LOW);
        }
}
```

24. Rotary Encode/ Potentiometer

We are getting this very useful module with this kit. This type of switches is mostly used in 3D printing machines and the applications where electronic interference issues are biggest concern.

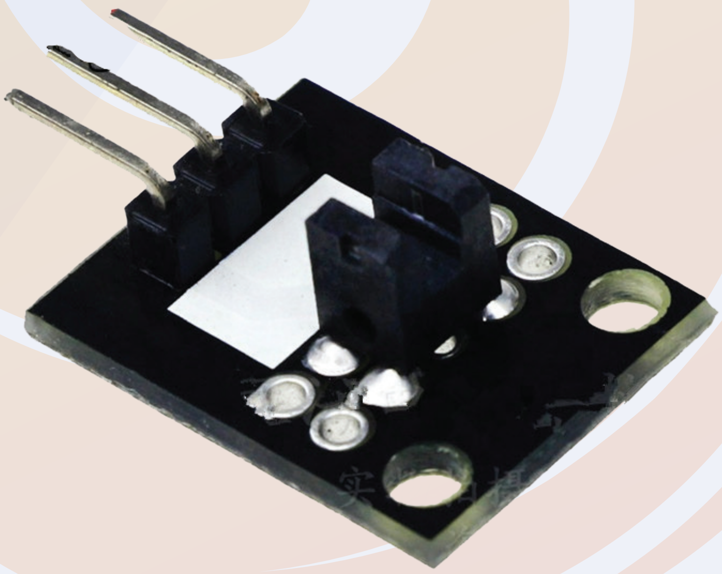
These switches consist an optical emitter and an optical receiver.

Talking about the optical emitter, it is nothing but LED which continuously emits the light and the optical receiver which is placed just in front of the optical emitter, the keep receiving the light that is being transmitted by the optical emitter.

The optical receiver is nothing but a photodiode. When the light rays fall on the photodiode the barrier gap between PN junction of the photodiode starts decreasing and at one point that barrier gap becomes zero and the photodiode acts as a switch.

In this way, until the light is falling on the photodiode the photodiode will be operating as a closed switch and when the light source is gone then it acts as an open switch.

Please check the following image for better understanding.



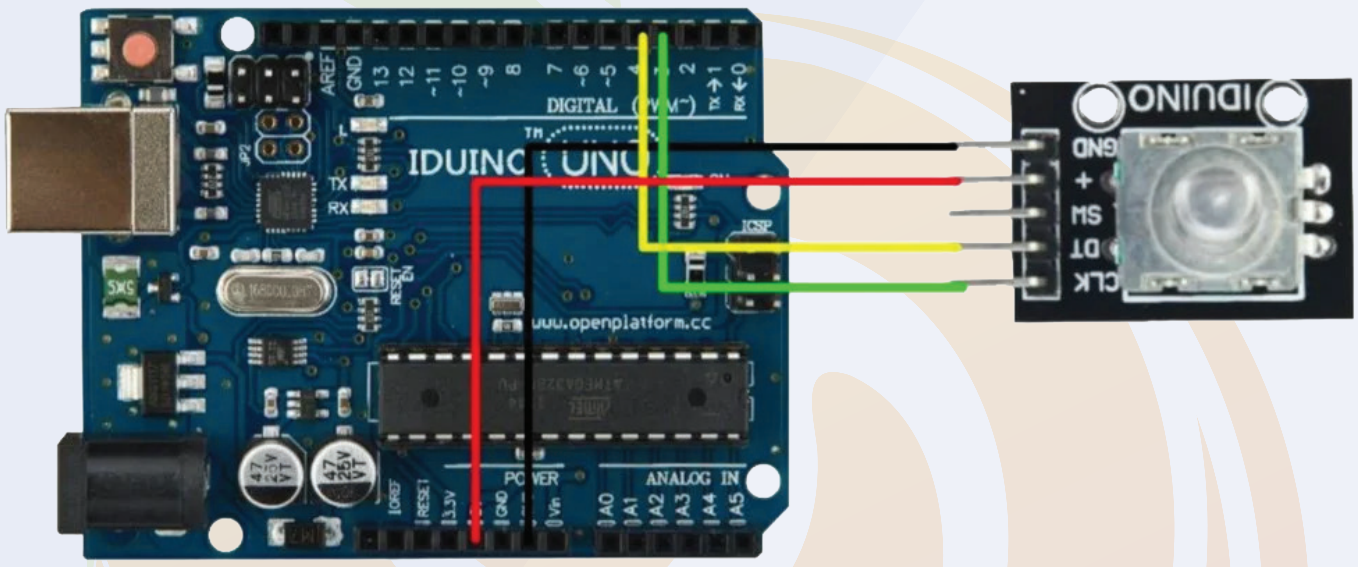
24.1. Interfacing The rotary Encoder with the Arduino

In this section, we are understanding the interfacing of the rotary encoder with the Arduino.

The rotary encoder has three pins. Output, vcc and GND.

The output pin of the rotary encoder we can a to the any analog pin of the Arduino.

Please refer to the following Image to understand the interfacing diagram of the sensor with the Arduino.



24.2. Arduino Code For Rotary Encoder

The output of the sensor is connected to the analog pin and output type of the sensor is analog so we will have to use analogRead function to read the output of the sensor.

Please refer to the following code to work with the rotary encoder.

Thank You...!

A beginner at electronics systems comes across this common problem “How should I step into learning embedded systems? Which components should I buy? This is why Robu has designed the kits which will assist you to understand the embedded system from zero to hero level. This will solve the dilemma while selecting the proper products for your need.

Moreover, free e booklets and Video tutorials that are being shared with these kits have everything in it that's being taught within the paid lectures. That means you'll master those technical skills without paying an enormous amount to training institutes. So prepare to be the master of your dreams and start gaining the knowledge which is effective to you!

Happy Learning !!!