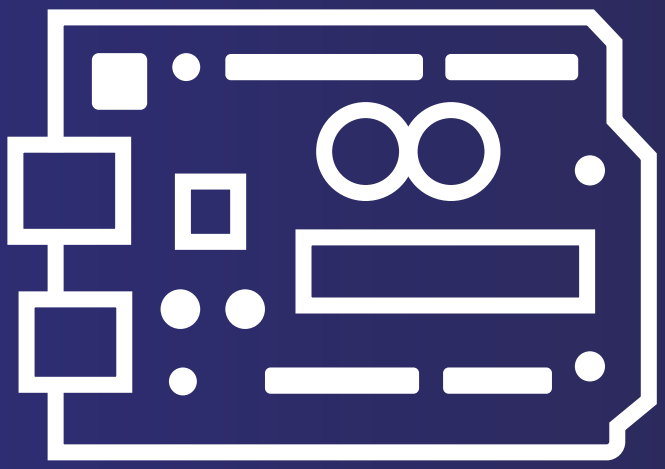
2021

Orange

Advance Kit

For Beginners



ORANGE KIT

Less hassle , more Fun !

01. Index

02. Index

03. Index

Orange Advance Kit

Hello Geek Thank you for purchasing our Orange Arduino Intermediate 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 microcontroller and microprocessor.

Difference Between Microcontroller 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 microcontroller 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, microcontrollers 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. Microcontrollers, 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 microcontroller and microprocess.

In the next section we will learn more about Arduino.

What is Arduino?

As discussed earlier, Arduino is a microcontroller 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.

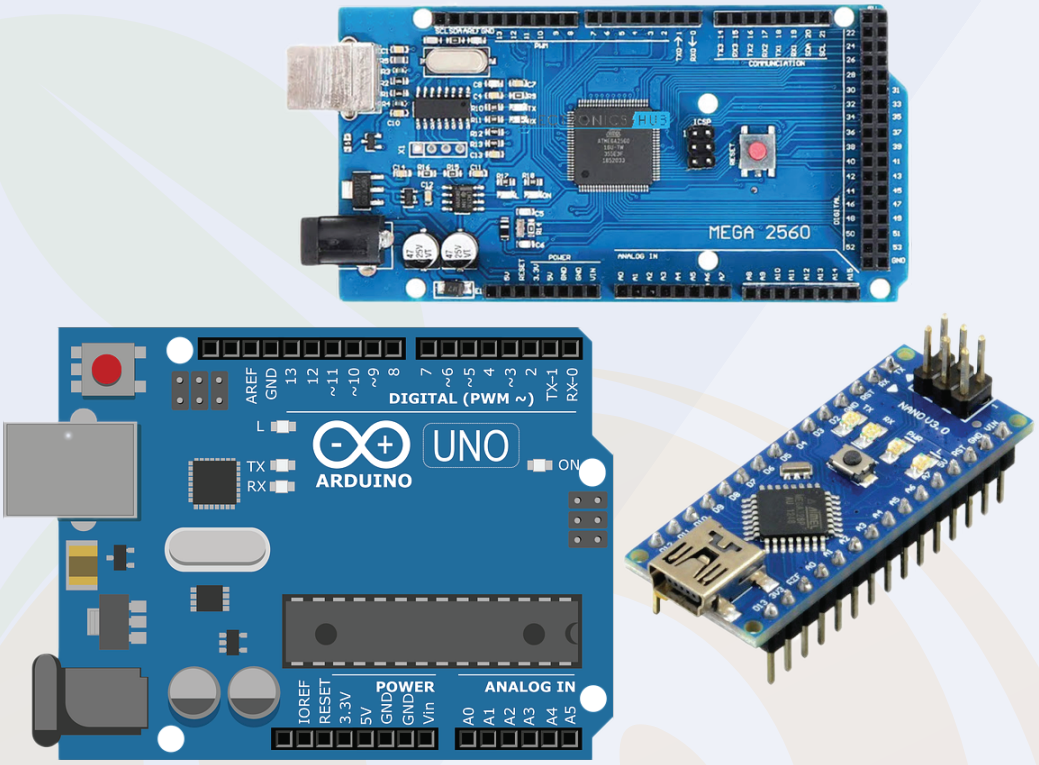
Since a large number of people contribute to this open-source project, Arduino users get quick solutions to their problems and their work never stops.

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.

1. Orange Arduino Uno Starter Kit
2. Orange Arduino Nano Starter Kit
3. Orange Arduino Mega Starter Kit

This was about the Arduino introduction in the next section of this blog we will learn to install the Arduino IDE on your system.



How to Install Arduino IDE On Windows?

Arduino IDE is a lightweight version software used to upload your written programs to Arduino boards.

This IDE gives us an easy-to-use UI that makes complex stuff so easy to understand.

In the following part, I have explained all these things step by step. Please take a look

To install Arduino IDE on your system, you need to download software package from internet.

How to Install Arduino IDE On Windows?

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 microcontroller and your computer. I have worked on those bords but never faced any problem.

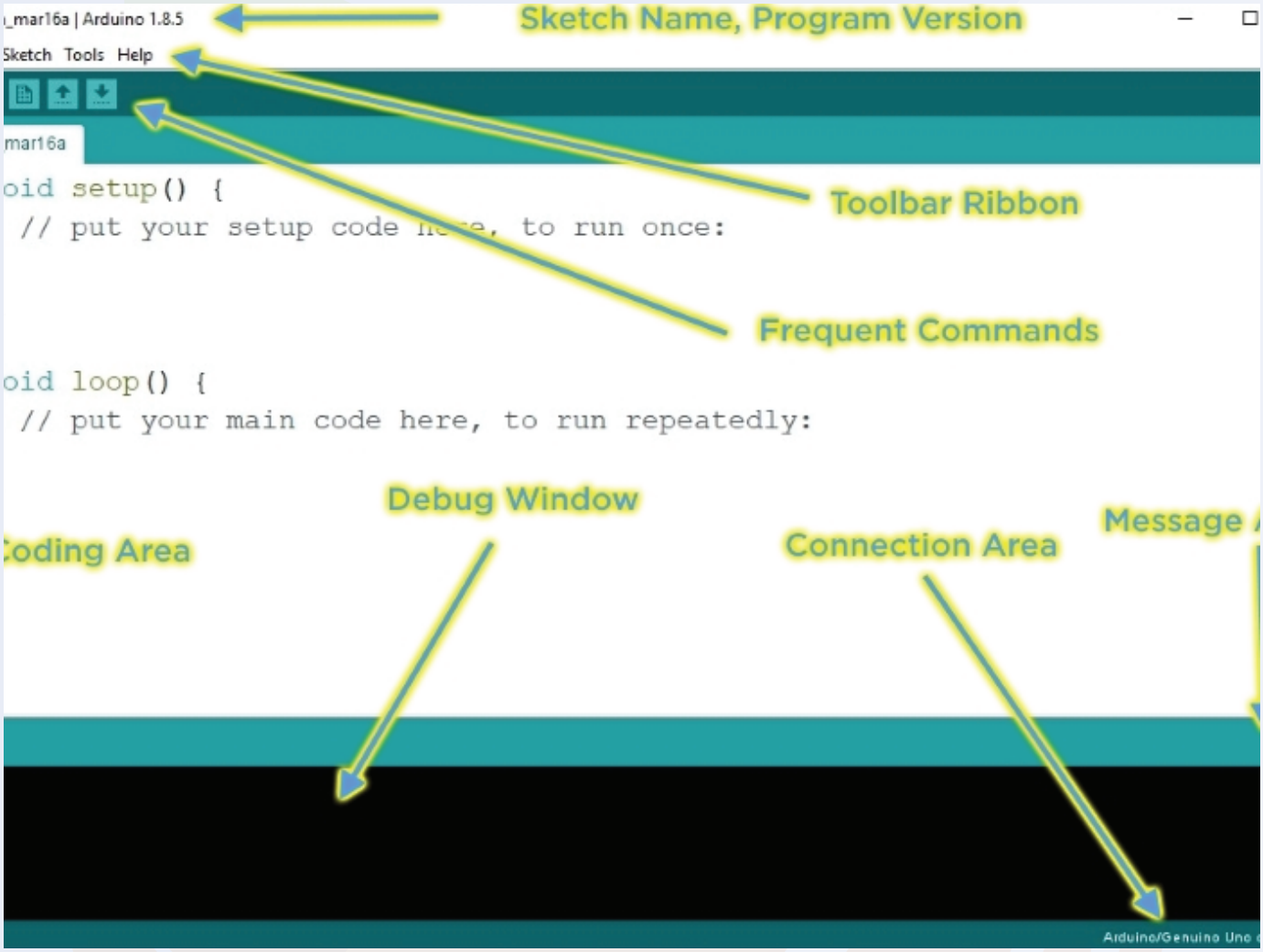
So stop worrying about this.

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.

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.



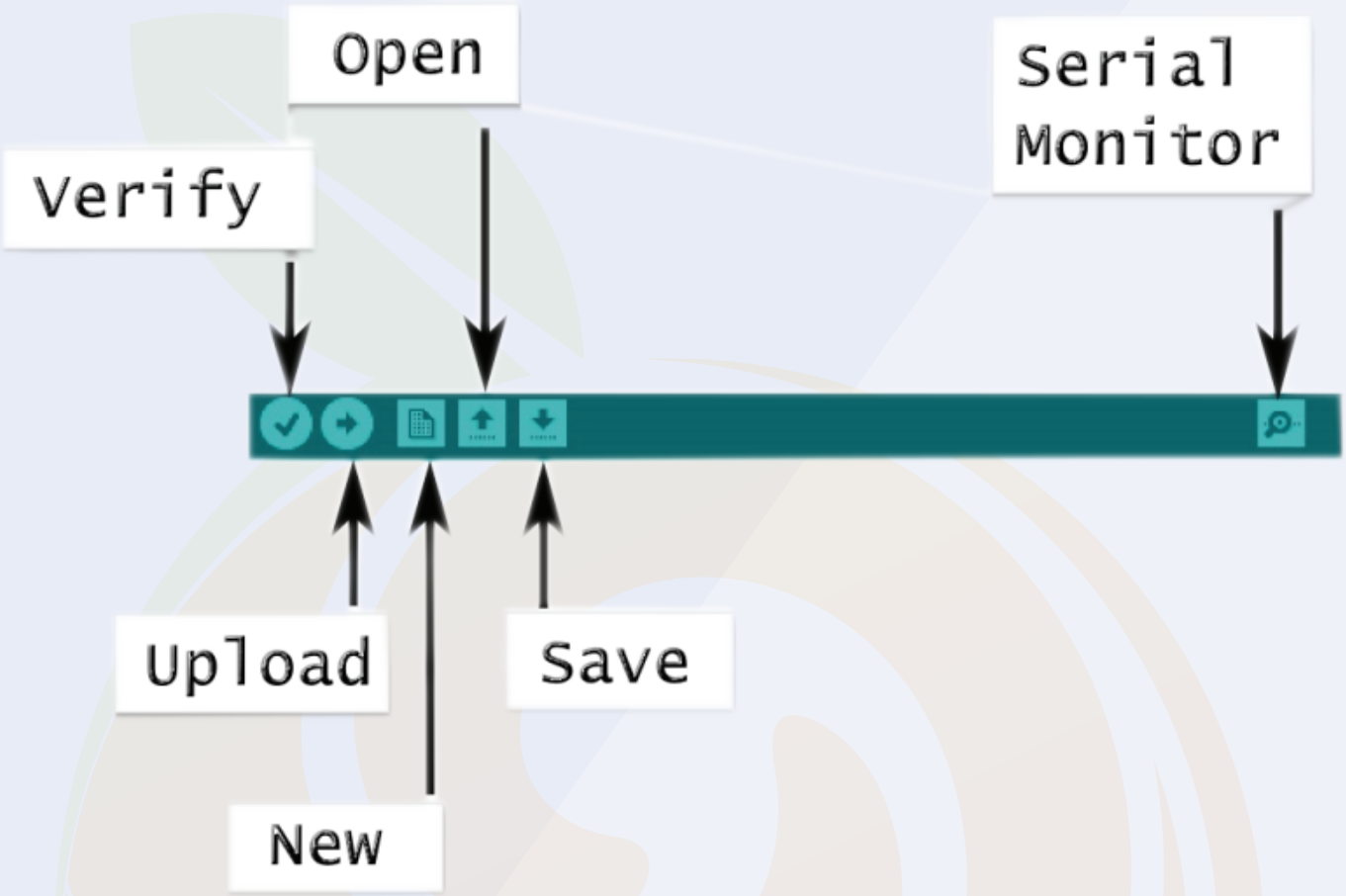
The Arduino IDE looks like this. In the above UI, you get all the functions that are useful for uploading code to your Arduino.

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 below 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.

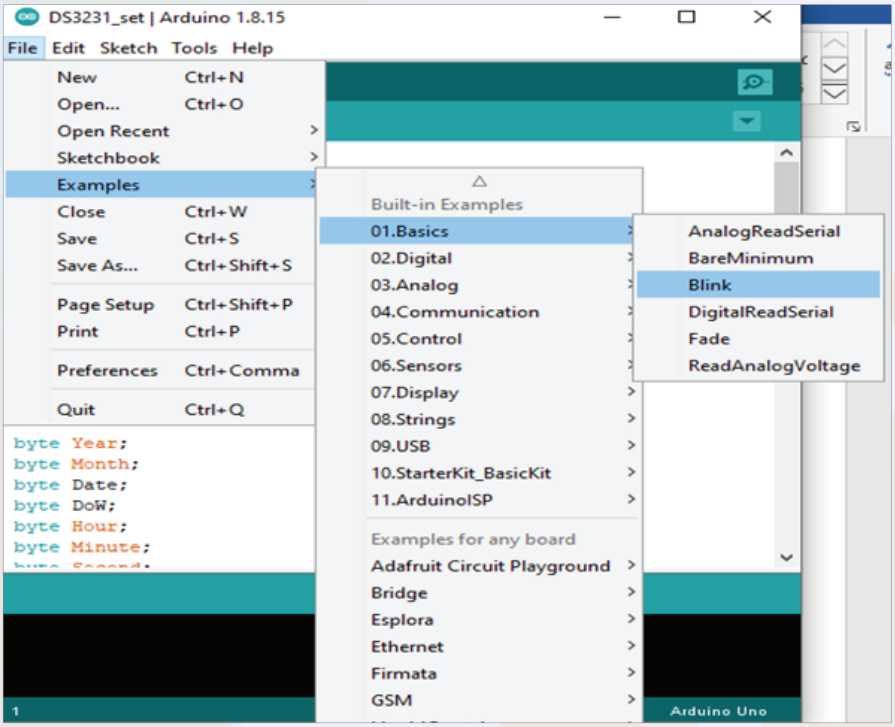
Arduino Text Editor

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

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.



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.

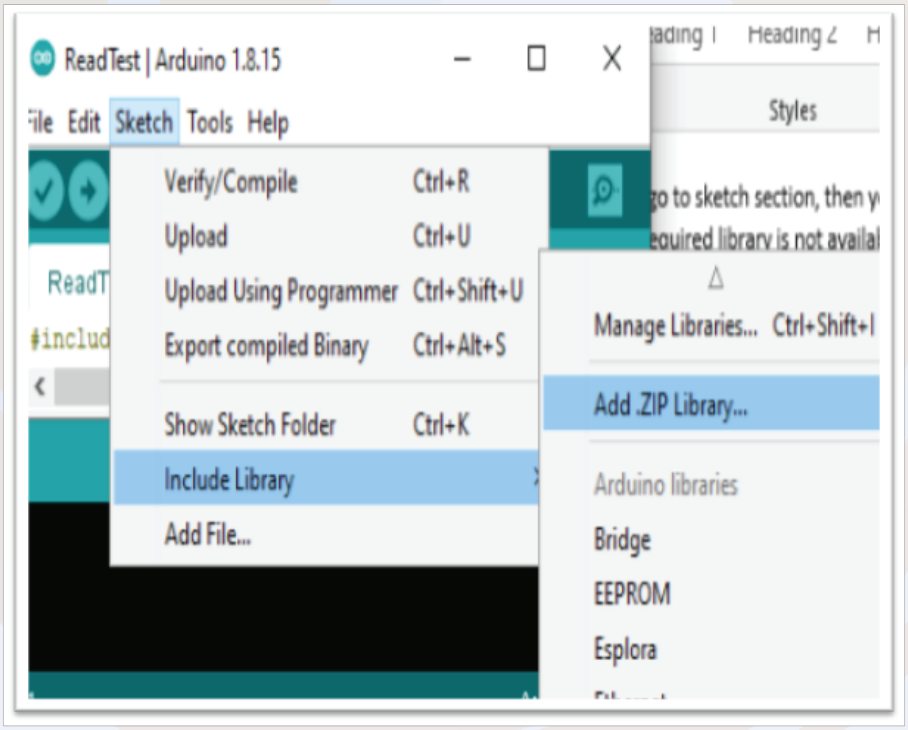
We use Arduino libraries, that means we use objects and methods of classes. So, even though we don't need to write much code, we can learn a lot by modifying existing code.

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.



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.

4.1 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.

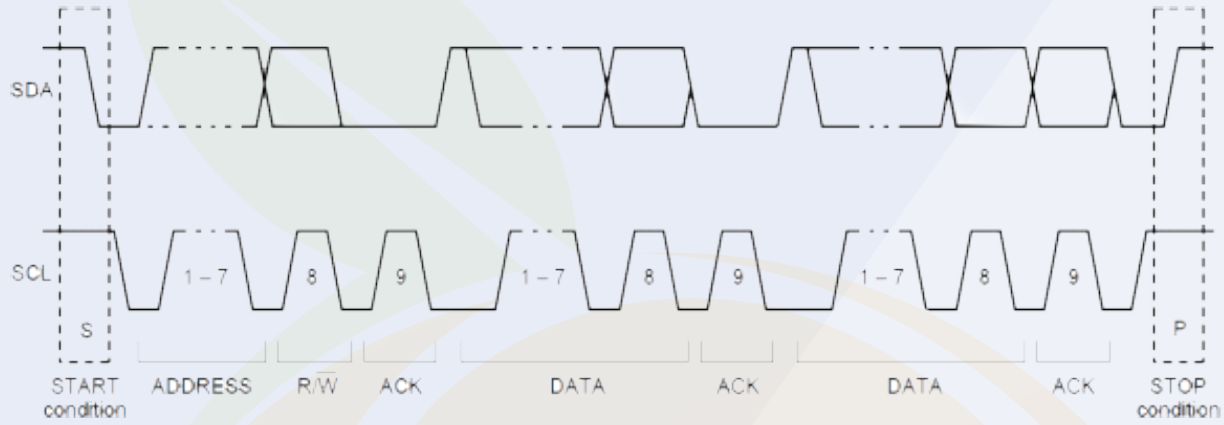
In I2C communication, the slave devices cannot start the communication at the first place and also cannot talk to the slave devices that are present in the network.

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

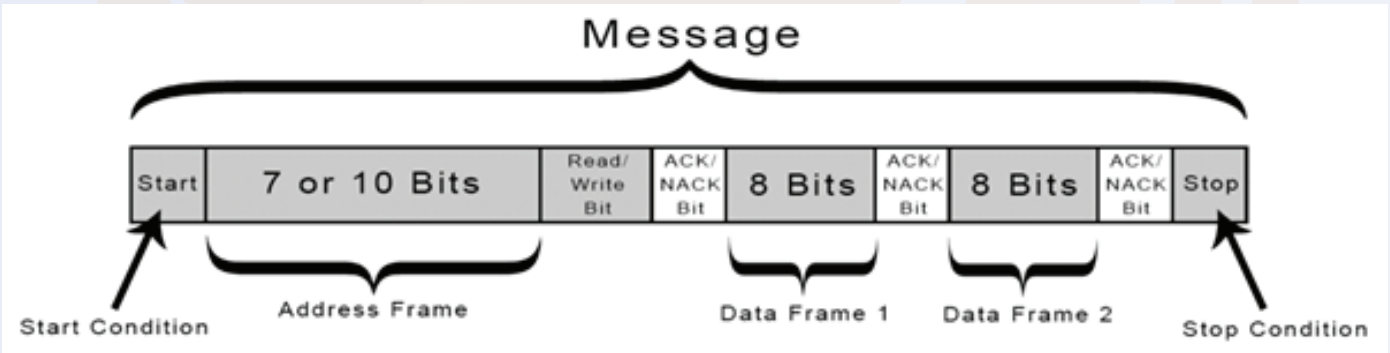
4.3. I2C Waveform – Explained

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.

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

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.

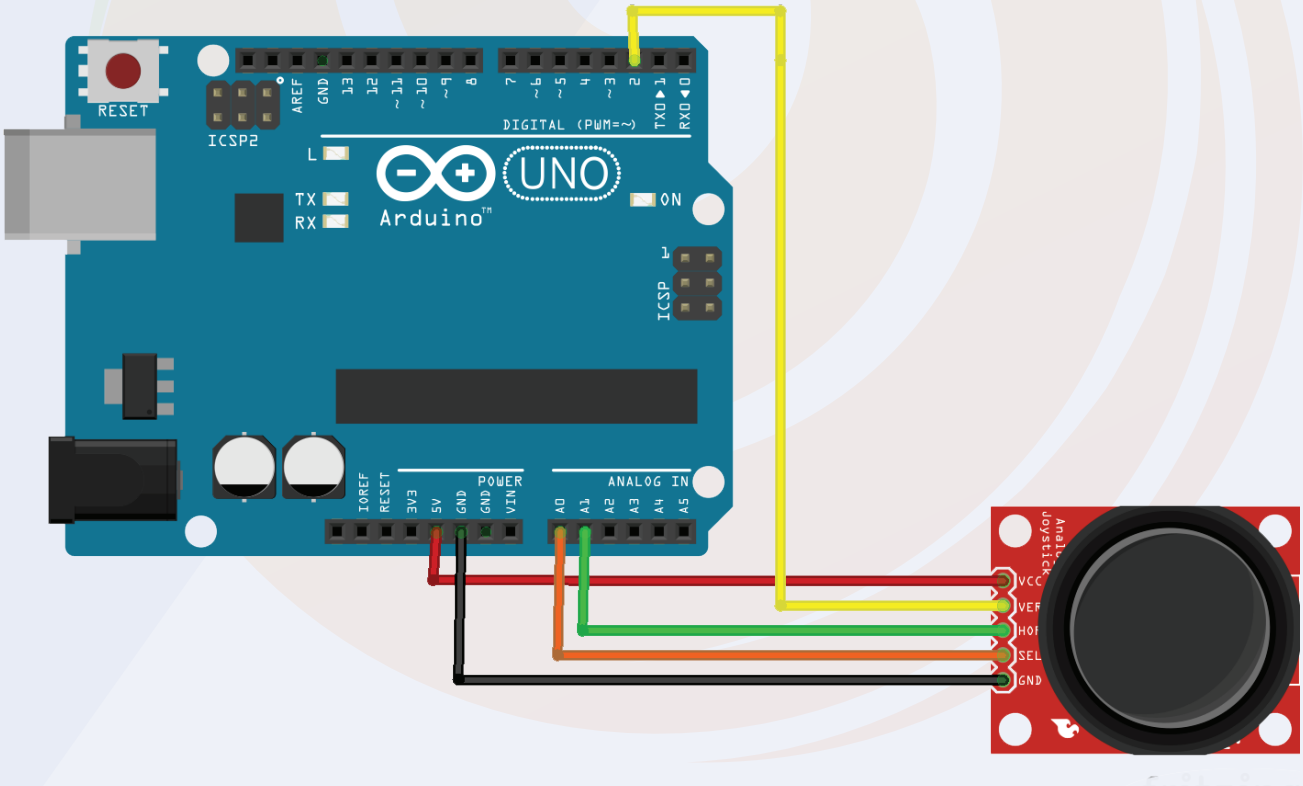
2. Y - This pin also generates the analog output voltage and is used to monitor the movement of the joystick on the Y-plane.

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. 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.

6. 5V Single Channel Relay

Signal Pin –

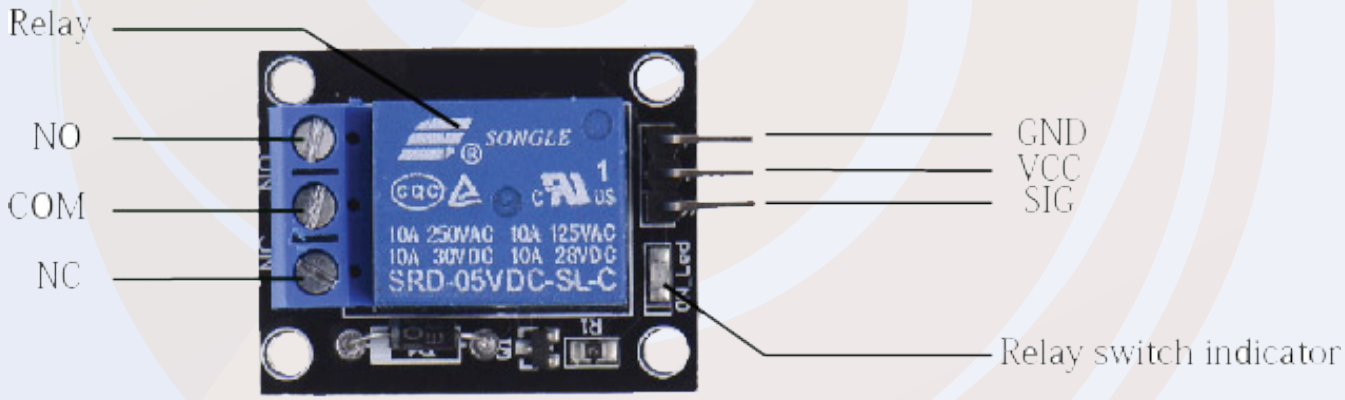
This pin is connected to the base terminal of the SMD transistor which is present on the relay module. When we give 5V to this pin, this pin will turn on the power to the relay coil.

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.

6.1. 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.

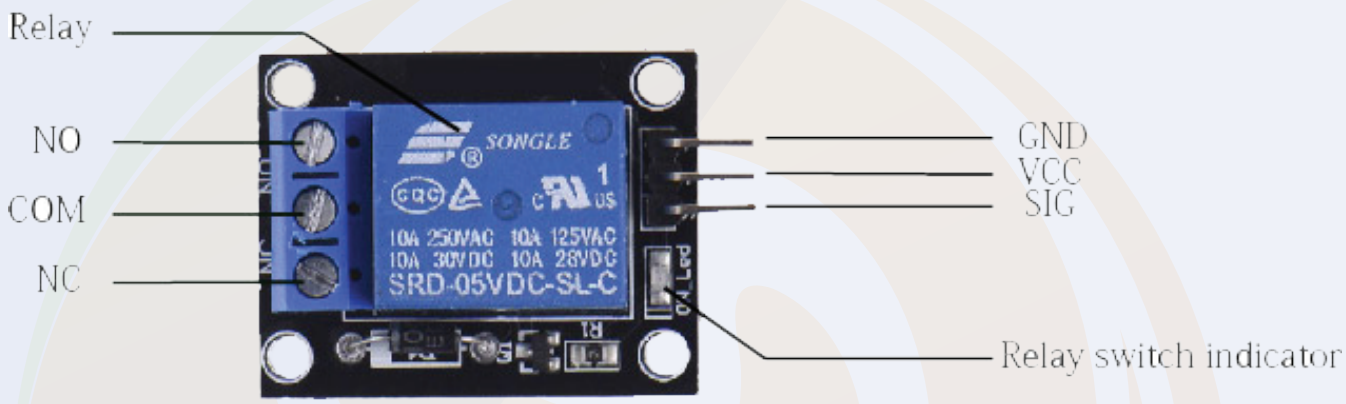


6.1. Interfacing The Relay Module With The Arduino –

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.

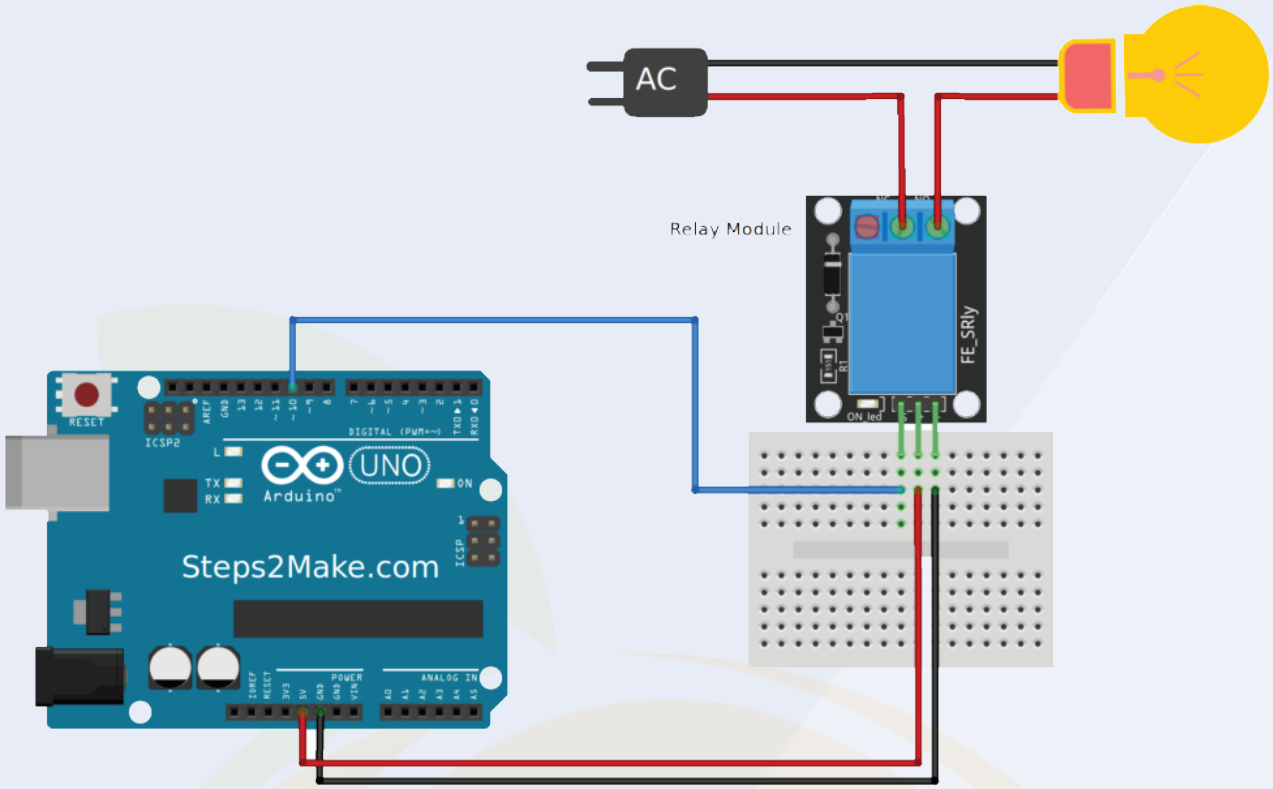


In the above image you can see COM, NC and NO pins. These are connected to the load.

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.

6.1. Interfacing The Relay Module With The Arduino –



6.2. 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.

6.1. 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();
}
```

7. Ultrasonic Sensor Module

Ultrasonic sensors are mainly used in distance detection applications. You may have heard or used this sensor before this.

If not, don't worry, in this section of the blog, you will understand everything that is important about the ultrasonic sensor.

We know every sound has some frequency. Some of the frequency is audible to us and some of it, is not.

The sound frequency that is generated by the ultrasonic sensor is not audible to human ears. Ultrasonic sensor generates ultrasonic frequency signals to detect the object.

Now you may be thinking, How an object can tell the difference by generating some ultrasonic frequency?

The answer is so simple, the control unit to which the ultrasonic sensor is connected that uses some time dependent logic.

When the ultrasonic sensor is powered, it generates some ultrasonic frequency, that frequency propagates through air medium with known speed and velocity.

These waves create a path while propagating through the air. When that path is interrupted by any object, the ultrasonic signals travel back to receiver from there. They do not penetrate through the object like electromagnetic waves does.

Those returning signals are then captured by the receiver part of the ultrasonic sensor.

Now, the control unit, do the work of time calculations. It calculates the time travelled by the sensor and the time taken by the signal to reach back.

This is how we get the output of the ultrasonic sensor.

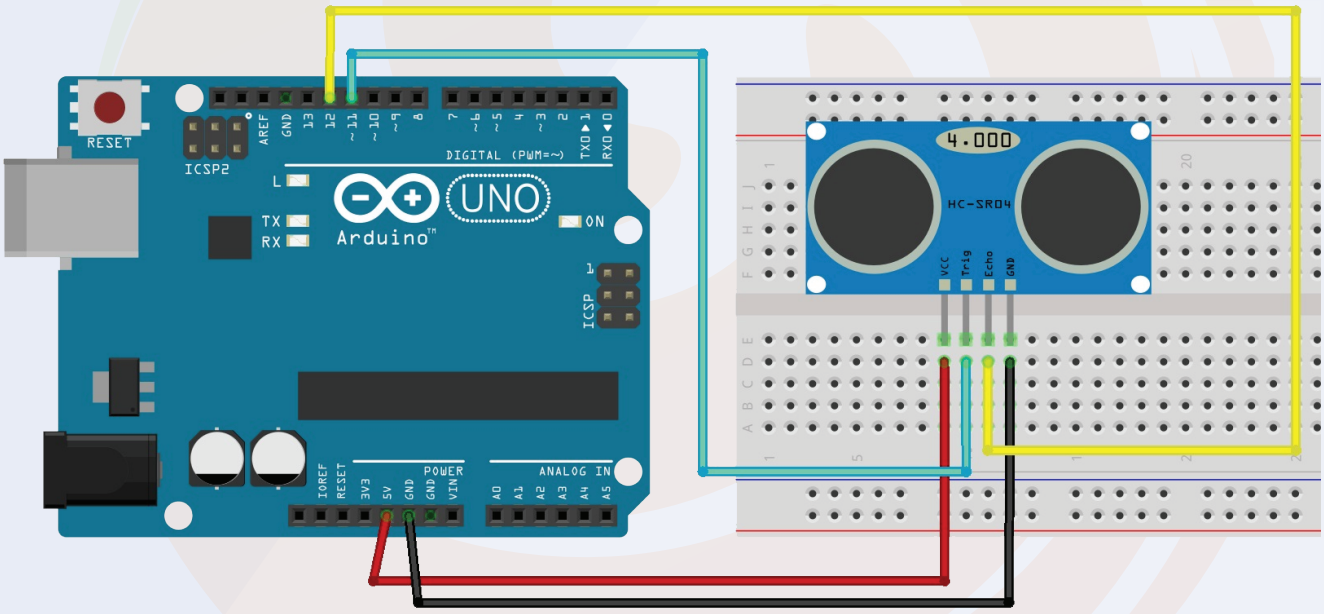
7. Ultrasonic Sensor Module

7.1. Interfacing the Ultrasonic Sensor With The Arduino

The HC-SR04 Ultrasonic Module has 4 pins, Ground, VCC, Trig and Echo.

The Ground and the VCC pins of the module needs to be connected to the Ground and the 5 volts pins on the Arduino Board respectively and the trig and echo pins to any Digital I/O pin on the Arduino Board.

Please see the following image to understand the interfacing of the Ultrasonic sensor with the Arduino.



7.2. Arduino Code Ultrasonic Sensor

```
const int trigPin = 9;
const int echoPin = 10;
// defines variables
long duration;
int distance;
void setup() {
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
Serial.begin(9600); // Starts the serial communication
}
void loop() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance= duration*0.034/2;
// Prints the distance on the Serial Monitor
Serial.print("Distance: ");
Serial.println(distance);
}
```

8. PIR Sensor Module

PIR sensor are used for motion detection.

These sensors measure the heat radiated by the objects and detects the motion of the of objects.

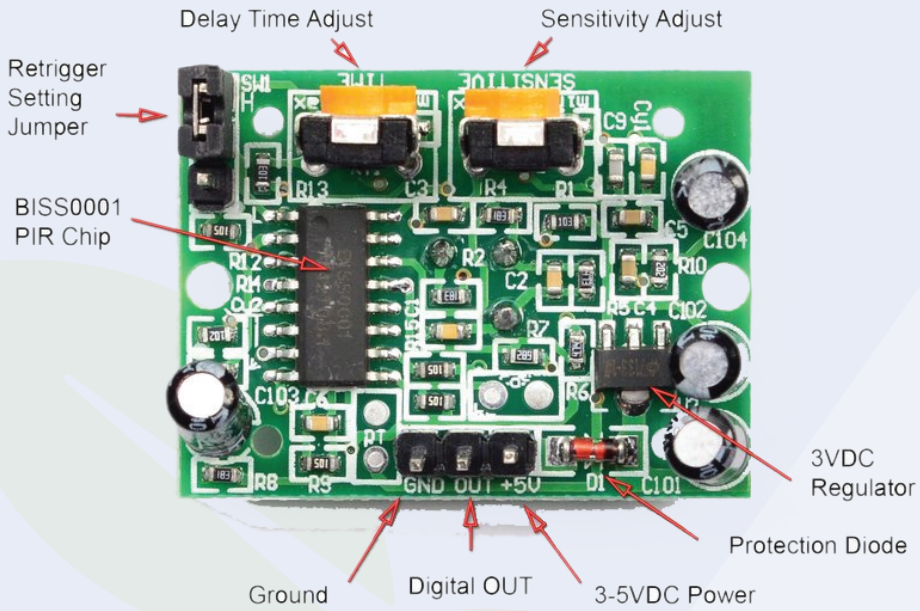
Unlike an active infrared sensor, passive IR sensors do not require a separate transmitter and a receiver.

These sensors mainly used in home surveillance applications.

Apart from all these basics, if you check the module, you will see two potentiometers on the PIR sensor.

Please check the following image.

8.1. PIR Sensor Module Introduction.



8.2. Troubleshoot if the PIR module doesn't work

A potentiometer designated as sensitivity used to adjust the sensing the range. And a potentiometer designated as Output-timing used to adjust the delay.

8.3. Troubleshooting - The Module is Not Giving Any Output in The first Minute After Turning it on

If you are having this problem, don't worry, it is not a problem, in the first minute after the power is applied, the PIR sensor starts reading the IR rays around it. Therefore, let it be stable in the surrounding environment.

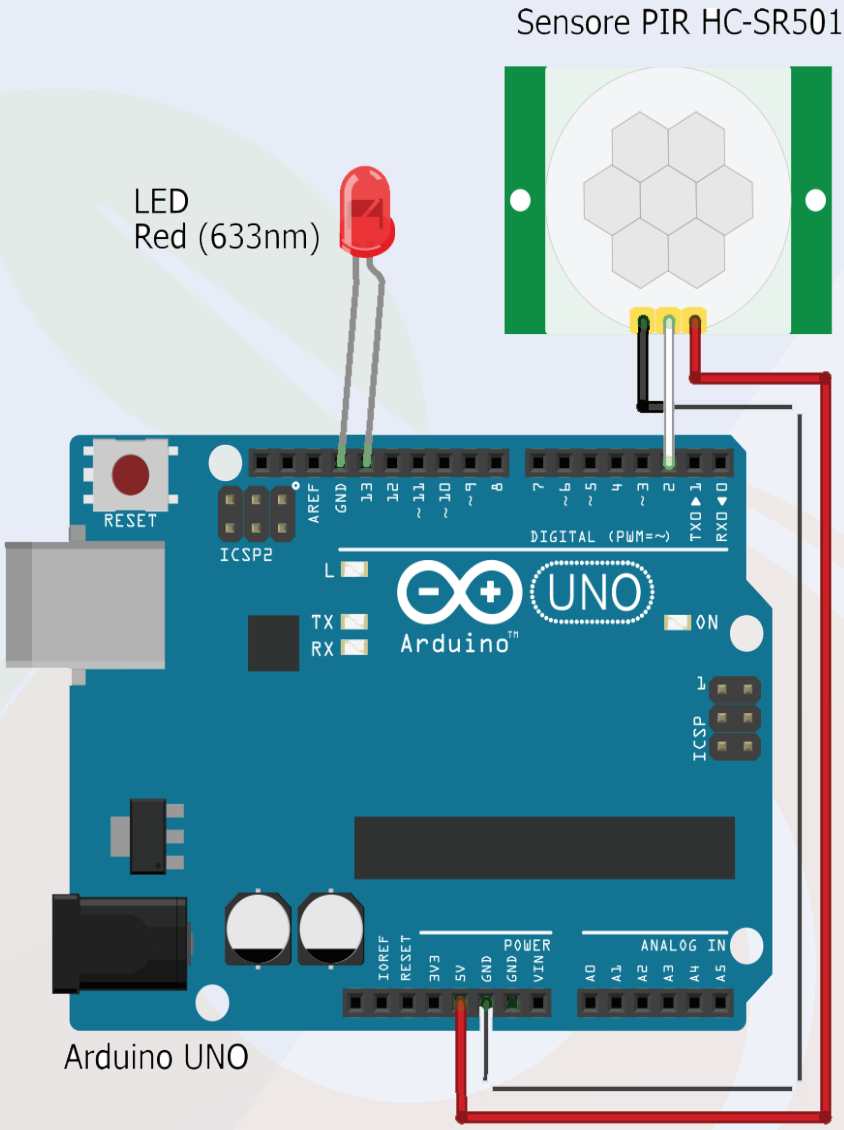
8.3. The output is not stable

If you are experiencing this problem, there are two reasons, first, that you have mistakenly changed the first potentiometer (sensitivity) of the module to its maximum position and the second cause of the problem may be the incorrect positioning of the PIR sensor.

If you have installed the PIR sensor at a point where the PIR sensor comes in direct contact with air and light. This could then be the reason that the sensor is producing the wrong output.

8.4. Interfacing the PIR Sensor With the Arduino

The following fig. Shows the PIR sensor's interface with the Arduino board. In this, we connected the signal pin of the PIR sensor to the D9 pin of the Arduino board and powering the sensor through the Arduino itself.



8.5. Arduino Code For PIR Sensor

The following code you can use for PIR Sensor. If you face any issues you can contact me and if there is any doubt please let us know we will be happy to assist you.

8.5. Arduino Code For PIR Sensor

```
#define pirPin 2
int calibrationTime = 30;
long unsigned int lowIn;
long unsigned int pause = 5000;
boolean lockLow = true;
boolean takeLowTime;
int PIRValue = 0;

void setup() {
  Serial.begin(9600);
  pinMode(pirPin, INPUT);
}

void loop() {
  PIRSensor();
}

void PIRSensor() {
  if(digitalRead(pirPin) == HIGH) {
    if(lockLow) {
      PIRValue = 1;
      lockLow = false;
      Serial.println("Motion detected.");
      delay(50);
    }
    takeLowTime = true;
  }
  if(digitalRead(pirPin) == LOW) {
    if(takeLowTime){
      lowIn = millis();takeLowTime = false;
    }
    if(!lockLow && millis() - lowIn > pause) {
      PIRValue = 0;
      lockLow = true;
      Serial.println("Motion ended.");
      delay(50);
    }
  }
}
```

The following code you can use for PIR Sensor. If you face any issues you can contact me and if there is any doubt please let us know we will be happy to assist you.

9. DC Motor

DC motor converts electrical energy in the form of Direct Current into mechanical energy in the form of rotational motion of the motor shaft.

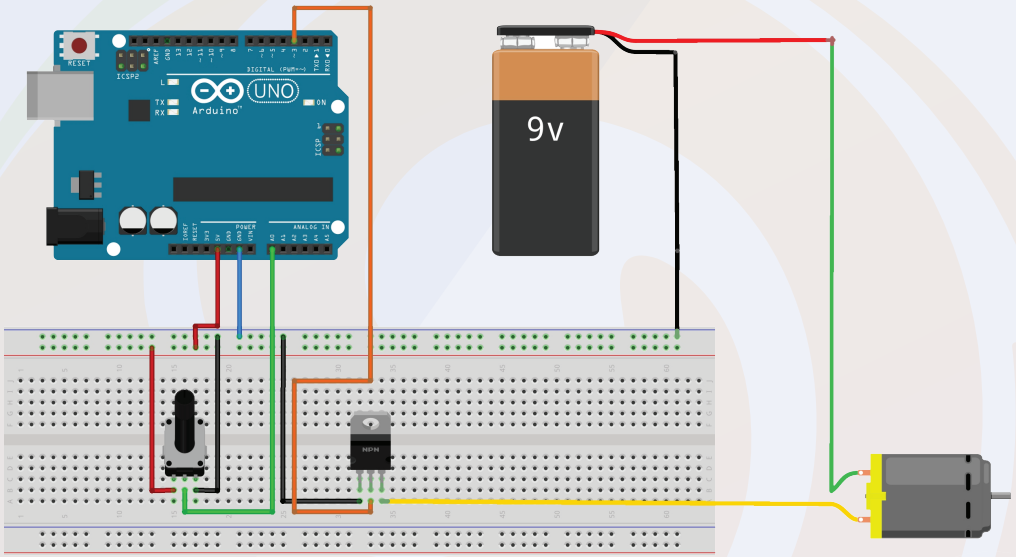
The DC motor speed can be controlled by applying varying DC voltage; whereas the direction of rotation of the motor can be changed by reversing the direction of current through it.

For applying varying voltage, we can make use of PWM technique.

For reversing the current, we can make use of NPN transistor.

You are getting an NPN transistor with this kit.

You can use that NPN and can do the connection as shown in the images below.



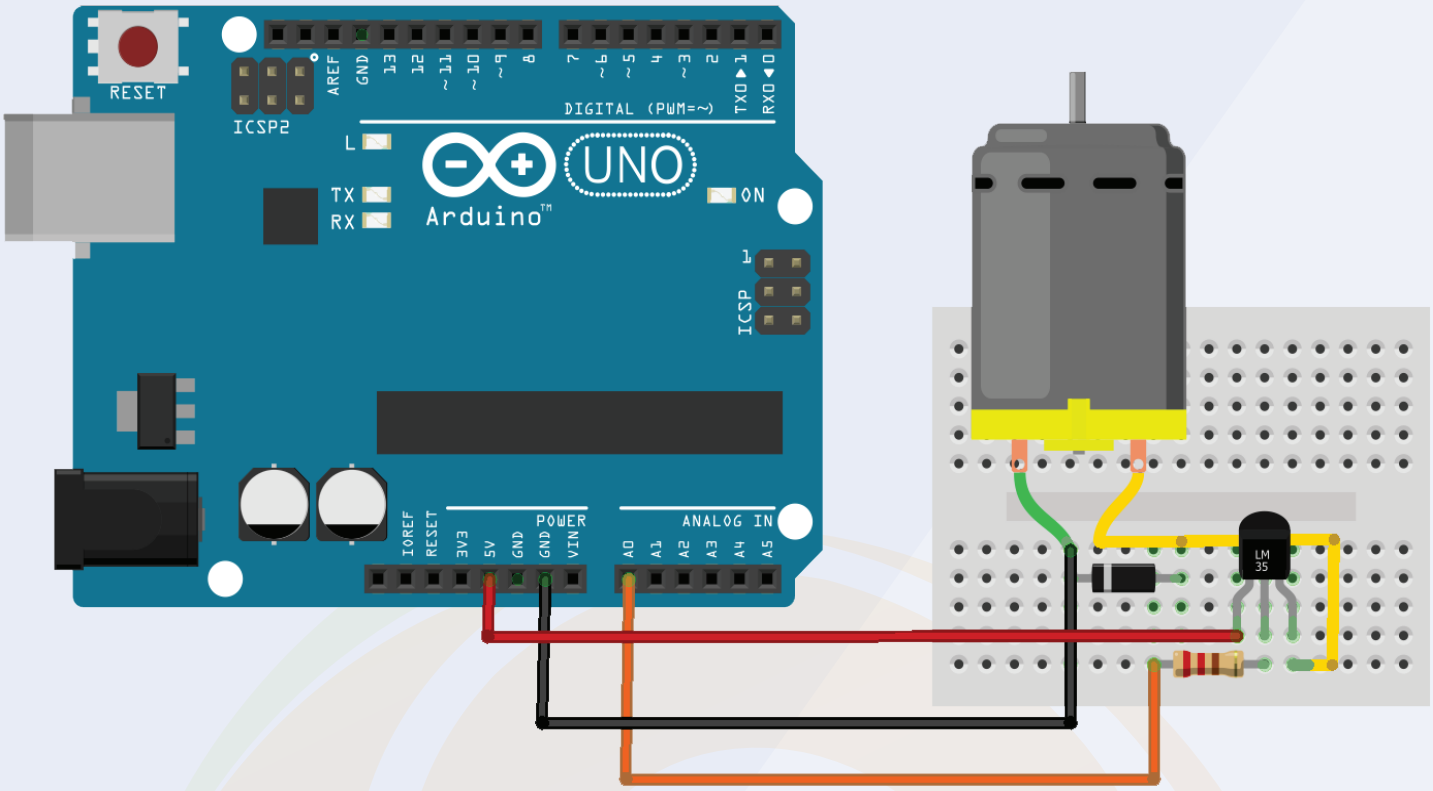
9.1. DC motor interfacing With the Arduino

Follow the circuit diagram and make the connections as shown in the image given below.

In this diagram, we have connected the base terminal of the transistor to the GPIO pin of the Arduino and the collector pin to the DC motor.

Please check the following diagram to understand the interfacing.

9.1. DC motor interfacing With the Arduino



9.2. Arduino Code For Controlling DC Motor

```
int motorPin = A0;

void setup() {
  pinMode(motorPin, OUTPUT);
  Serial.begin(9600);
  while (! Serial);
  Serial.println("Speed 0 to 255");
}

void loop() {
  if (Serial.available()) {
    int speed = Serial.parseInt();
    if (speed >= 0 && speed <= 255) {
      analogWrite(motorPin, speed);
    }
  }
}
```

10. Gyro Sensor

Nowadays, Gyro Sensor is being used in many applications such as drones, E-Cars and many more.

MPU6050 sensor has many functions over the single chip. It consists a MEMS accelerometer, a MEMS gyro, and temperature sensor.

This module is so accurate as this module uses 16 bit ADC to convert analog signals to digital.

This MPU6050 module is a compact chip having both accelerometer and gyro. This is a very useful device for many applications like drones, robots, motion sensors. It is also called Gyroscope or Triple axis accelerometer.

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

10.1. Gyro Sensor interfacing With the Arduino

This module works on I2C serial communication by default but it can be configured for SPI interface by configuring it register. For I2C this has SDA and SCL lines.

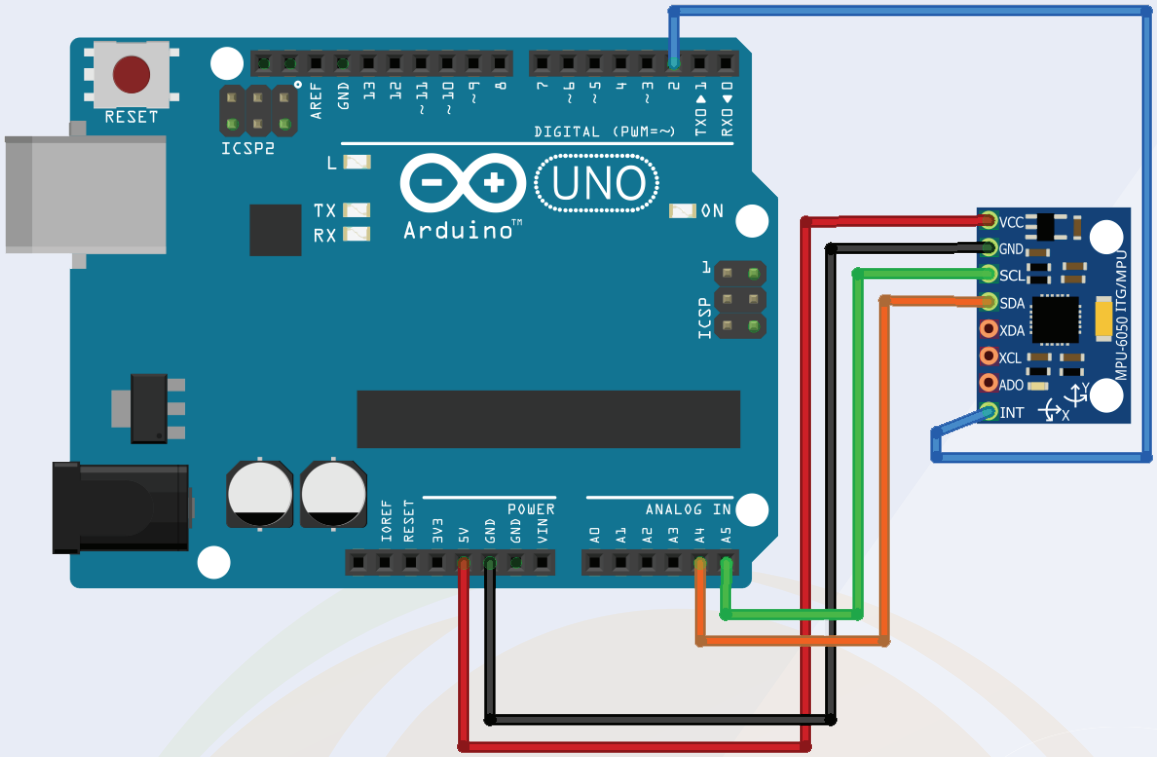
Please do connection as shown in the below image and if you face any issues please let us know in the comment section.

10.1.1 Pin Configuration:

Vcc:-
this pin is used for powering the MPU6050 module with respect to ground

GND:- this is ground pin
SDA:-SDA pin is used for data between controller and mpu6050 module
SCL:- SCL pin is used for clock input
XDA:- This is sensor I2C SDA Data line for configuring and reading from external sensors ((optional) not used in our case)
XCL:- This is sensor I2C SCL clock line for configuring and reading from external sensors ((optional) not used in our case)
ADO:- I2C Slave Address LSB (not applicable in our case)
INT:- Interrupt pin for indication of data ready

10.1. Gyro Sensor interfacing With the Arduino



10.1. Arduino Code For Gyro Sensor

Here we have used this MPU6050 library to interface it with Arduino. So first of all, we need to download the MPU6050 library from GitHub and install it in Arduino IDE.

After it, we can find example codes in the example. The user may test that code by directly uploading them to Arduino and can see values over serial monitor.

10.1. Arduino Code For Gyro Sensor

```
#include "I2Cdev.h"
#include "MPU6050.h"

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for InvenSense evaluation board)
// AD0 high = 0x69
MPU6050 accelgyro;
//MPU6050 accelgyro(0x69); // <-- use for AD0 high

int16_t ax, ay, az;
int16_t gx, gy, gz;



// uncomment "OUTPUT_READABLE_ACCELGYRO" if you want to see a tab-separated
// list of the accel X/Y/Z and then gyro X/Y/Z values in decimal. Easy to read,
// not so easy to parse, and slow(er) over UART.
#define OUTPUT_READABLE_ACCELGYRO

// uncomment "OUTPUT_BINARY_ACCELGYRO" to send all 6 axes of data as 16-bit
// binary, one right after the other. This is very fast (as fast as possible
// without compression or data loss), and easy to parse, but impossible to read
// for a human.
//#define OUTPUT_BINARY_ACCELGYRO


#define LED_PIN 13
bool blinkState = false;

void setup() {
    // join I2C bus (I2Cdev library doesn't do this automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    // initialize serial communication
    // (38400 chosen because it works as well at 8MHz as it does at 16MHz, but
    // it's really up to you depending on your project)
```

10.1. Arduino Code For Gyro Sensor

```
    Serial.begin(38400);

    // initialize device
    Serial.println("Initializing I2C devices...");
    accelgyro.initialize();

    // verify connection
    Serial.println("Testing device connections...");
    Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");



    // configure Arduino LED pin for output
    pinMode(LED_PIN, OUTPUT);
}

void loop() {
    // read raw accel/gyro measurements from device
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    // these methods (and a few others) are also available
    //accelgyro.getAcceleration(&ax, &ay, &az);
    //accelgyro.getRotation(&gx, &gy, &gz);

    #ifdef OUTPUT_READABLE_ACCELGYRO
        // display tab-separated accel/gyro x/y/z values
        Serial.print("a/g:\t");
        Serial.print(ax); Serial.print("\t");
        Serial.print(ay); Serial.print("\t");
        Serial.print(az); Serial.print("\t");
        Serial.print(gx); Serial.print("\t");
        Serial.print(gy); Serial.print("\t");
        Serial.println(gz);
    #endif

    #ifdef OUTPUT_BINARY_ACCELGYRO
        Serial.write((uint8_t)(ax >> 8)); Serial.write((uint8_t)(ax & 0xFF));
        Serial.write((uint8_t)(ay >> 8)); Serial.write((uint8_t)(ay & 0xFF));
        Serial.write((uint8_t)(az >> 8)); Serial.write((uint8_t)(az & 0xFF));
        Serial.write((uint8_t)(gx >> 8)); Serial.write((uint8_t)(gx & 0xFF));
        Serial.write((uint8_t)(gy >> 8)); Serial.write((uint8_t)(gy & 0xFF));
        Serial.write((uint8_t)(gz >> 8)); Serial.write((uint8_t)(gz & 0xFF));
    #endif

    // blink LED to indicate activity
    blinkState = !blinkState;
    digitalWrite(LED_PIN, blinkState);
}
```

11. ESP01 Module

ESP01 module is a WIFI module. This module is used to connect the microcontroller to the WIFI router.

These modules are widely used in IOT applications. With the help of this module, you will able to do so many things such as creating data network, creating an IoT project and what not.

If you are working on an IOT project then this module should be your first choice.

11.1. How to interface ESP01 with the Arduino?

Till this point we learned about the importance of the ESP-01 module in this section we will understand the interfacing of the esp01 module with the Arduino. This module uses UART communication for data transfer.

So, we will have to establish UART communication to communicate with this module.

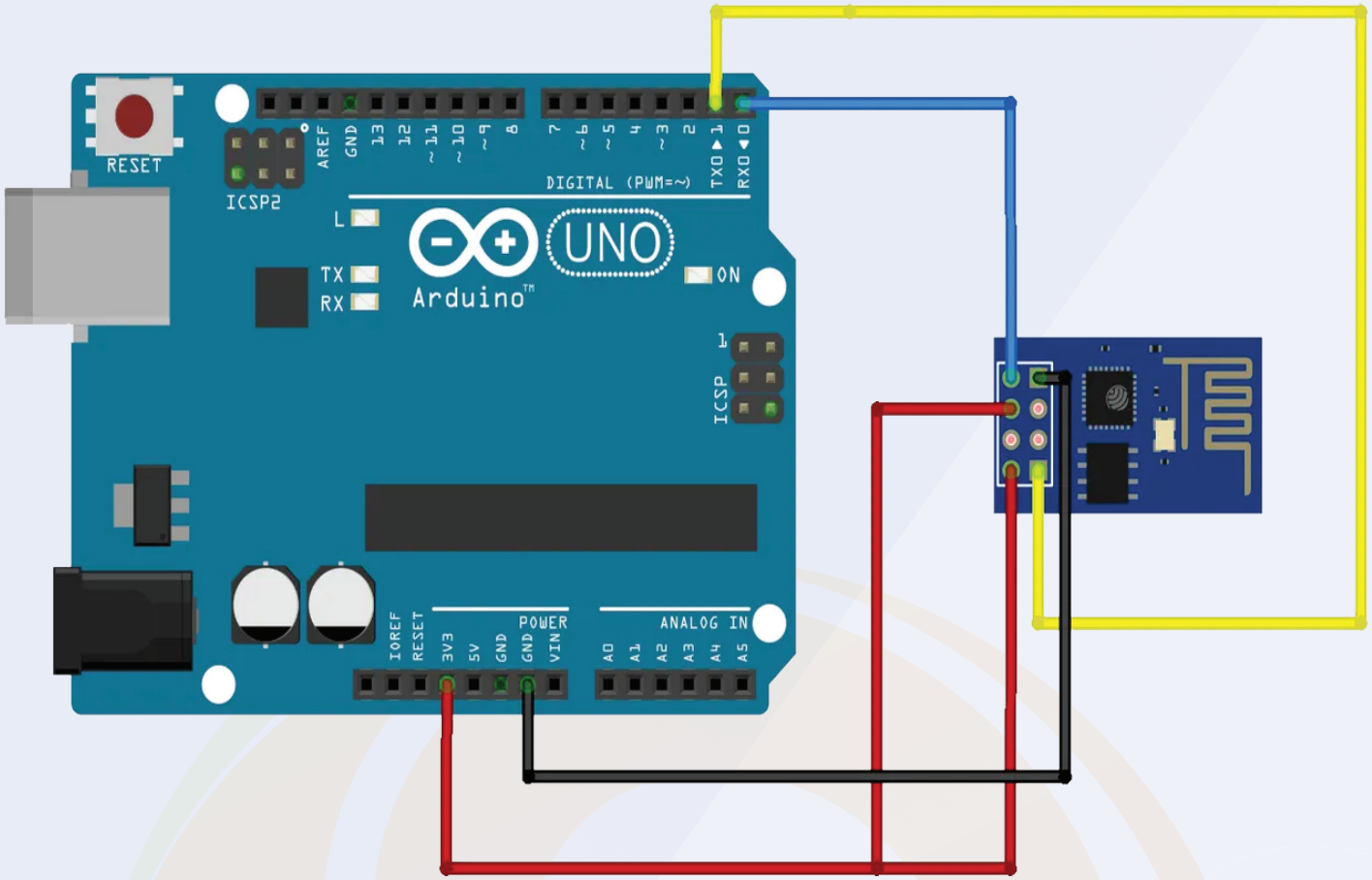
In the below section, I have explained the pin details and have shown the interfacing diagram. Please take a look.

11.2. Pin Description of ESP8266 ESP-01 Module

- VCC: It is the power pin through which 3.3V is supplied.
- GND: It is the ground pin.
- TX: This pin is used to transmit serial data to other devices.
- RX: The RX pin is used to receive serial data from other devices.
- RST: It is the Reset Pin and it is an active LOW Pin. (ESP8266 will reset if the RST pin receives LOW signal).
- CH_PD: This is the chip enable pin and it is an active HIGH Pin. It is usually connected to 3.3V.
- GPIO0: The GPIO0 (General Purpose I/O) Pin has dual functions – one for normal GPIO Operation and other for enabling the Programming Mode of ESP8266.
- GPIO2: This is GPIO Pin.

Please check the image below image to understand the interfacing of the ESP01 with the Arduino.

11.1. How to interface ESP01 with the Arduino?



11.2. Arduino Code For Controlling ESP01 Module

In order to work with this module, we will have to install the ESP8266.h library.

To install the board follow the following steps.

Open Arduino and then open the Preferences window.

Enter https://arduino.esp8266.com/stable/package_esp8266com_index.json into the File>Preferences>Additional Boards Manager URLs field of the Arduino IDE. You can add multiple URLs, separating them with commas.

Open Boards Manager from Tools > Board menu and install esp8266 platform (and don't forget to select your ESP8266 board from Tools > Board menu after installation).

11.2. Arduino Code For Controlling ESP01 Module

After this you can use the code that are written in the library.
The following code will collect the information of nearby WIFI modules and will print that information on the serial monitor.

```
#include <ESP8266WiFi.h>

const char* ssid     = "your-ssid";
const char* password = "your-password";

const char* host = "data.sparkfun.com";
const char* streamId   = "....................";
const char* privateKey = "....................";

void setup() {
  Serial.begin(115200);
  delay(10);

  // We start by connecting to a WiFi network

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

int value = 0;

void loop() {
  delay(5000);
  ++value;

  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
```

11.2. Arduino Code For Controlling ESP01 Module

After this you can use the code that are written in the library.
The following code will collect the information of nearby WIFI modules and will print that information on the serial monitor.

```
#include <ESP8266WiFi.h>

const char* ssid     = "your-ssid";
const char* password = "your-password";

const char* host = "data.sparkfun.com";
const char* streamId   = "....................";
const char* privateKey = "....................";

void setup() {
  Serial.begin(115200);
  delay(10);

  // We start by connecting to a WiFi network

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

int value = 0;
```

11.2. Arduino Code For Controlling ESP01 Module

After this you can use the code that are written in the library.
The following code will collect the information of nearby WIFI modules and will p

```
#include <ESP8266WiFi.h>

const char* ssid     = "your-ssid";
const char* password = "your-password";

const char* host = "data.sparkfun.com";
const char* streamId   = "....................";
const char* privateKey = "....................";

void setup() {
  Serial.begin(115200);
  delay(10);

  // We start by connecting to a WiFi network

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

int value = 0;
```

Project

P.1. DC Motor Speed Control Using Joystick

In this project, we are controlling the speed of the motor with the help of joystick. We have already discussed the working of the DC motor and joystick in this section we will learn how to use joystick to control the DC motor.

Talking about the interfacing details, we have already discussed about it in the above sections.

You can check the above section to understand the working of the DC motor and joystick with the Arduino.

11.2. Connection Details and code explanation.

In this project, we are reading the input from the joystick and based on the output received from the joystick we are generating the PWM signal on the pin to which the NPN transistor is connected.

You can refer to the following code to understand the working of this project

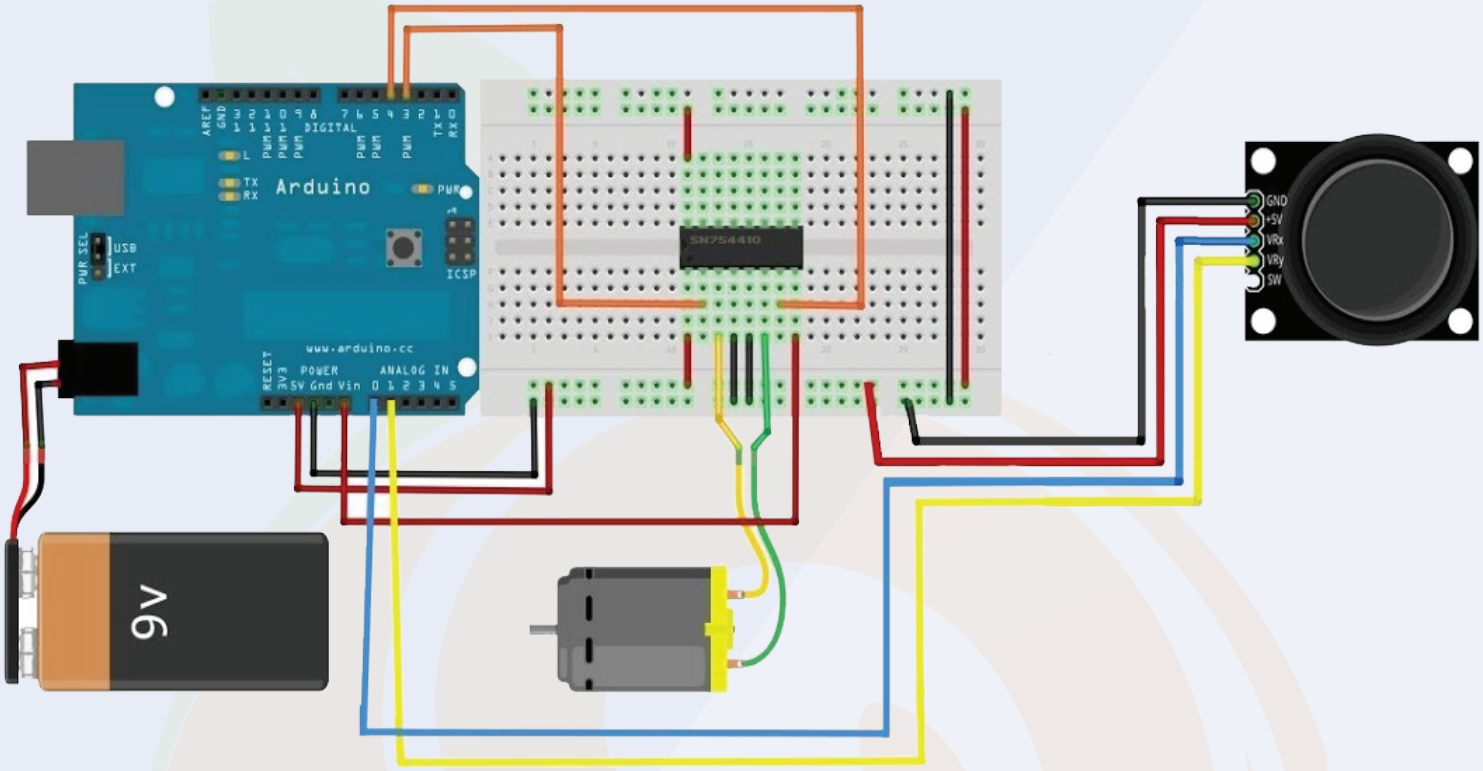
Project

P.1. Arduino Code

```
int speedPin=5;
int dir1=4;
int dir2=3;
int mSpeed;
int jPin=A1;
int jVal;

void setup() {
  // put your setup code here, to run once:
pinMode(speedPin,OUTPUT);
pinMode(dir1,OUTPUT);
pinMode(dir2,OUTPUT);
pinMode(jPin,INPUT);
Serial.begin(9600);

}

void loop() {
  // put your main code here, to run repeatedly:
jVal=analogRead(jPin);

Serial.println(jVal);
if (jVal<512){
digitalWrite(dir1,LOW);
digitalWrite(dir2,HIGH);
mSpeed=-255./512.*jVal+255.;
analogWrite(speedPin,mSpeed);
}
if(jVal>=512){
digitalWrite(dir1,HIGH);
digitalWrite(dir2,LOW);
mSpeed=(255./512.)*jVal-255.;
analogWrite(speedPin,mSpeed);
}
}
```

Project

P.1. Interfacing Diagram



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 !!!