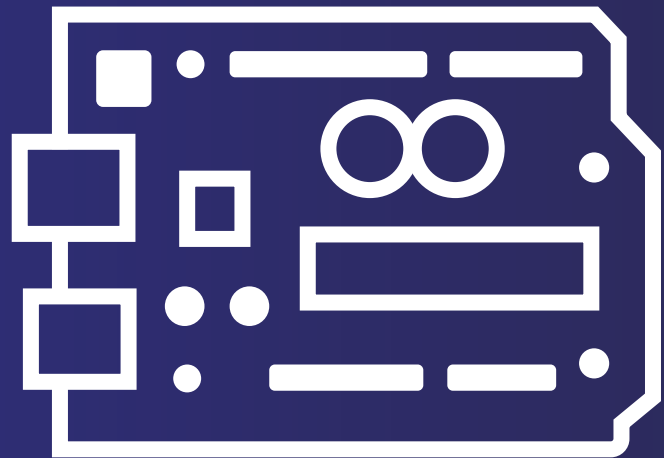


2021

Orange Intermediate Kit

For Beginners



01. Index

- 1. Arduino Introduction**
 - 1.1. What is Arduino?**
 - 1.2. Difference Between Microcontroller and Microprocessor**
 - 1.3. How to Install Arduino IDE On Windows?**
- 2. Arduino IDE?**
 - 2.1. Void Loop And Void Setup?**
 - 2.2. Analog Write And Analog Read**
 - 2.3. Digital Read And Digital Write**
- 3. I2C Backpack LCD**
 - 3.1. I2C Communication**
 - 3.2. Importance Of The I2C Communication**
 - 3.3. Interfacing I2C LCD With The Arduino**
 - 3.4. Arduino Code For I2C Backpack LCD**
- 4. Servo Motor**
 - 4.1. Interfacing Servo Motor With The Arduino.**
 - 4.2. Arduino Code For Servo Motor**
 - 4.3. Functions In the Code**
- 5. Stepper Motor**
 - 5.1. Interfacing the stepper Motor With the Arduino**
 - 5.2. Arduino Code For Stepper Motor**
- 6. DS1302 RTC Module**
 - 6.1. Features of DS1307 RTC Module**
 - 6.2. Arduino Code For DS1302 RTC Module**

02. Index

7. MFRC-522 RC522 RFID

- 7.1. RFID Reader**
- 7.2. RFID Tag**
- 7.3. Working of The RFID Reader**
- 7.4. RFID Interfacing With The Arduino**
- 7.5. Arduino Code For RFID Card**

8. Sound Sensor

- 8.1. Interfacing Of Sound Sensor With The Arduino**
- 8.2. Arduino Code For The Arduino**

9. DHT11 Sensor

- 9.1. Interfacing DHT11 Sensor With The Arduino**
- 9.2. Arduino Code For DHT11 Interfacing With The Arduino**

10. Water Level Sensor

- 10.1. Interfacing Water Level Sensor With The Arduino**
- 10.2. Arduino Code For Water Level Sensor**

11. 8*8 Matrix Display

- 11.1. Introduction To The 8*8 Matrix Display**
- 11.2. 8*8 Dot Matrix Display Interfacing With The Arduino**
- 11.3. Arduino Code For 8*8 Matrix Display**

03. Index

12. 4*4 Matrix Keypad

- 12.1. Working of The 4*4 Matrix Keypad With The Arduino
- 12.2. Interfacing 4*4 Keypad With The Arduino
- 12.3. Arduino Code For 4*4 Matrix Display With The Arduino

13. Projects

- 13.1. Student Attendance System Using Arduino
- 13.2. Room Temperature Indicator Using Arduino

Orange Intermediate 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 microprocessor.

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 Administrator' 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

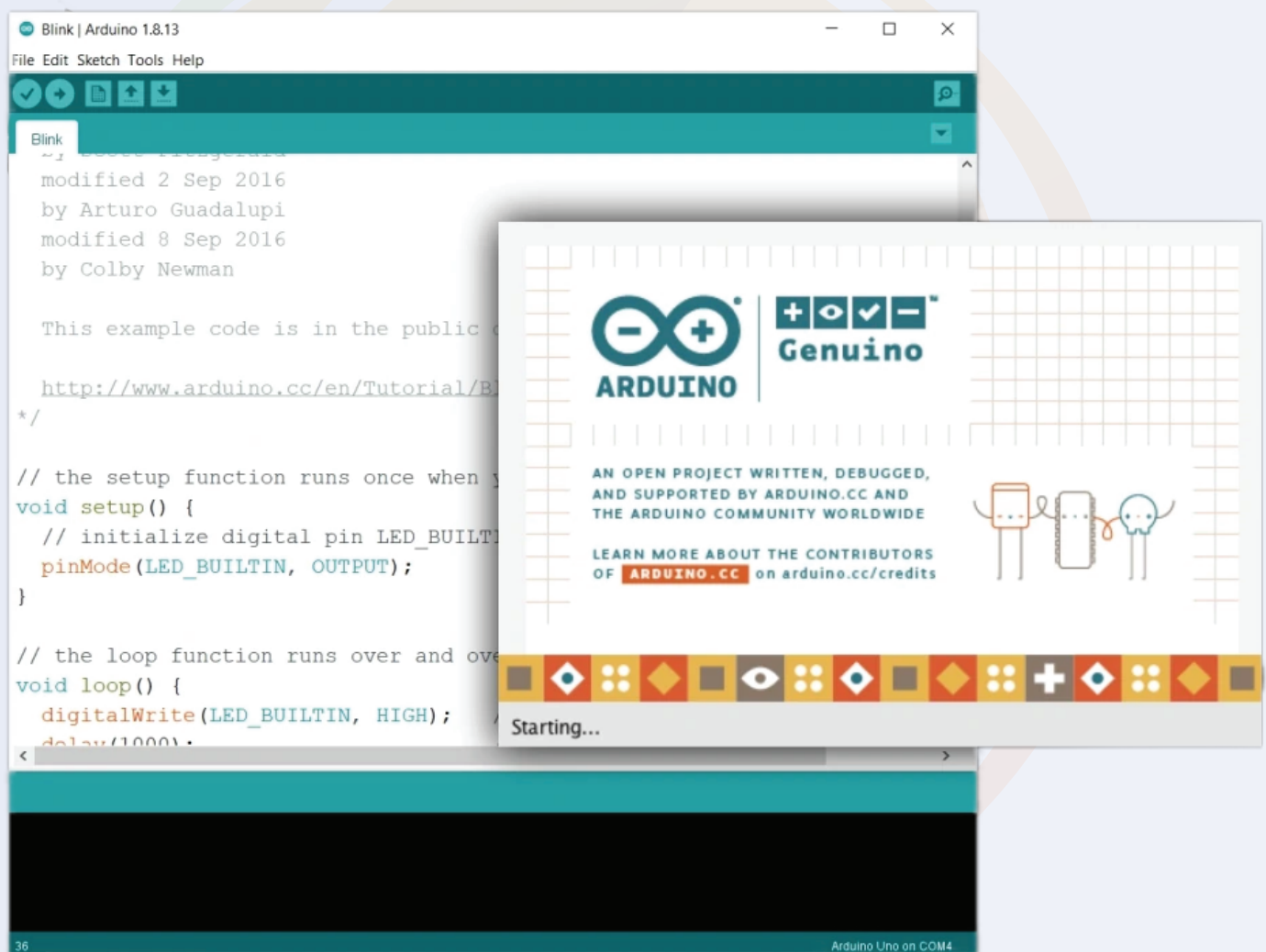
Please note - there is a misconception that boards that contain the ch340G driver are of poor quality but this is not true.

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 boards but never faced any problem.

So stop worrying about this.

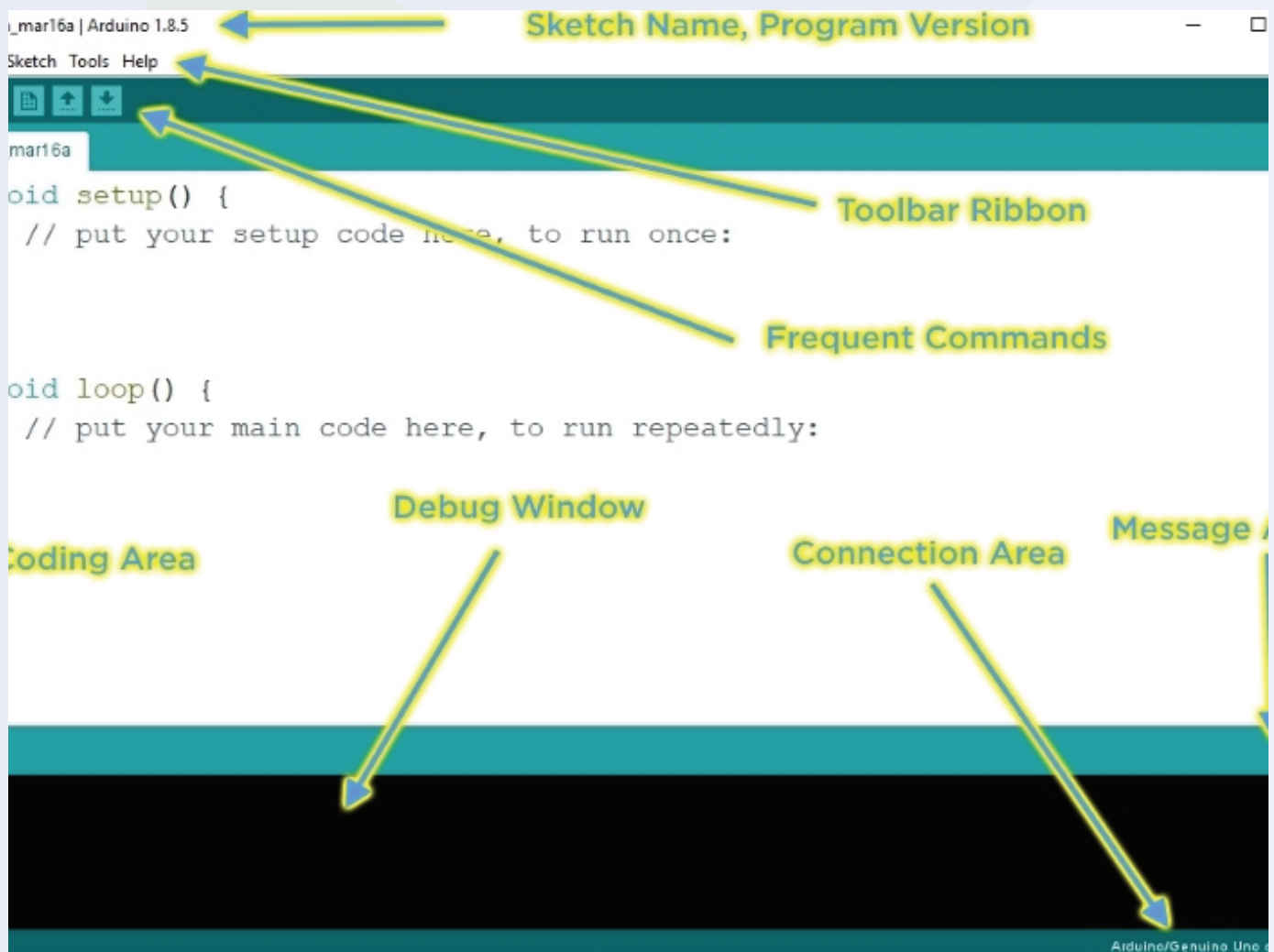
In the next part of this blog, we will learn about the UI of Arduino IDE and understand the usage of buttons available in the application.



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.



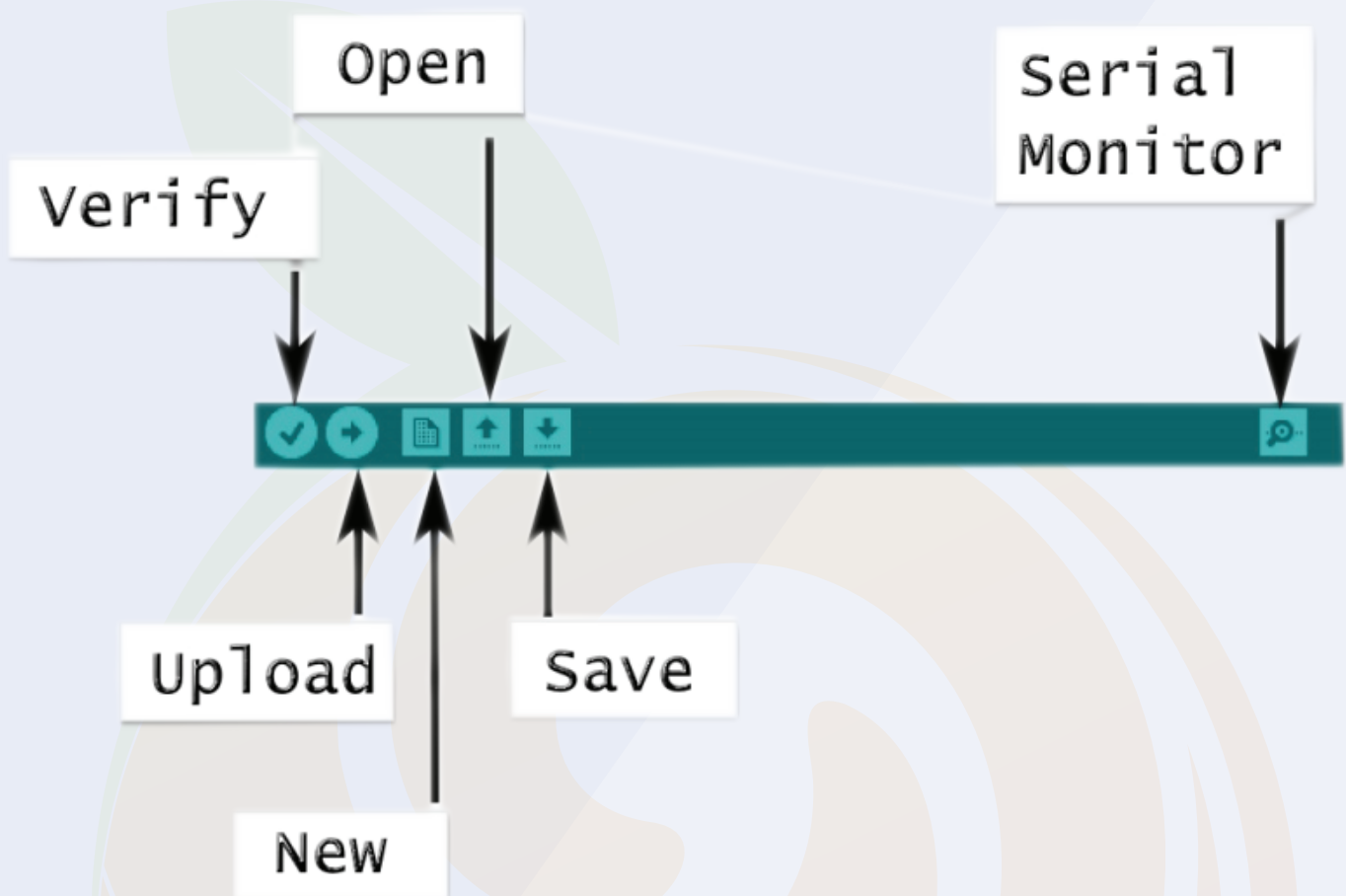
The Arduino IDE looks like this. In the above UI, you get all the functions that are useful for uploading code to your Arduino.

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.

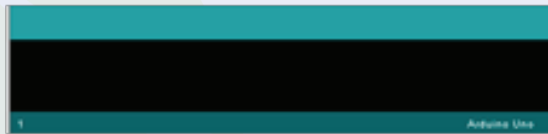
5. Search

- 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

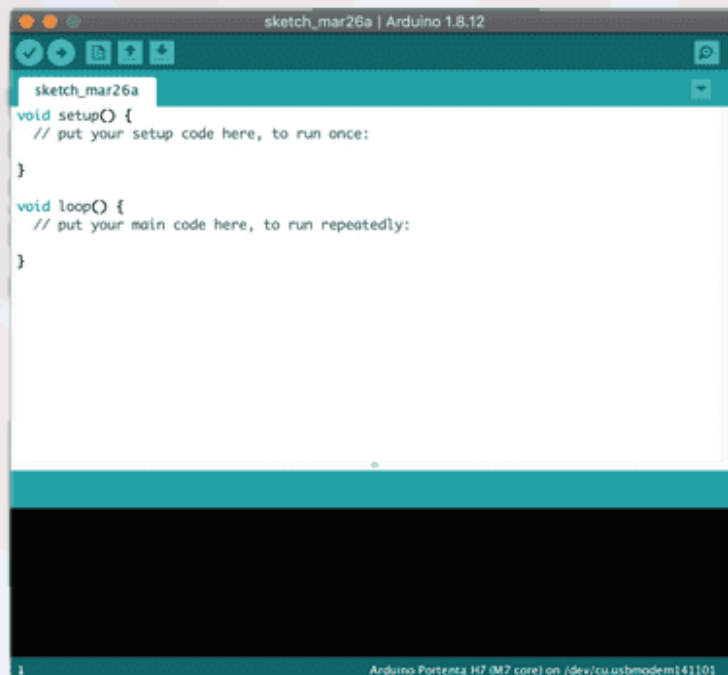
In this option you will see the update of the uploading process of the code. If there is any problem with your code then IDE will generate some error and those errors we can see in this section.



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.



```
sketch_mar26a | Arduino 1.8.12
sketch_mar26a
void setup() {
  // put your setup code here, to run once:
}

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

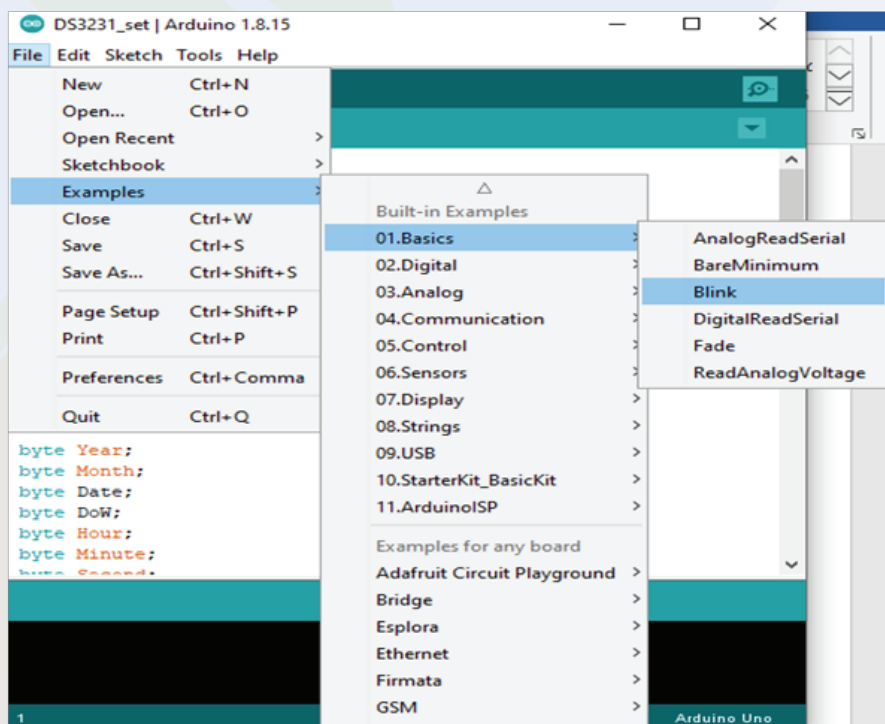
Arduino Portenta H7 (M7 core) on /dev/cu.usbmodem141101
```

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.

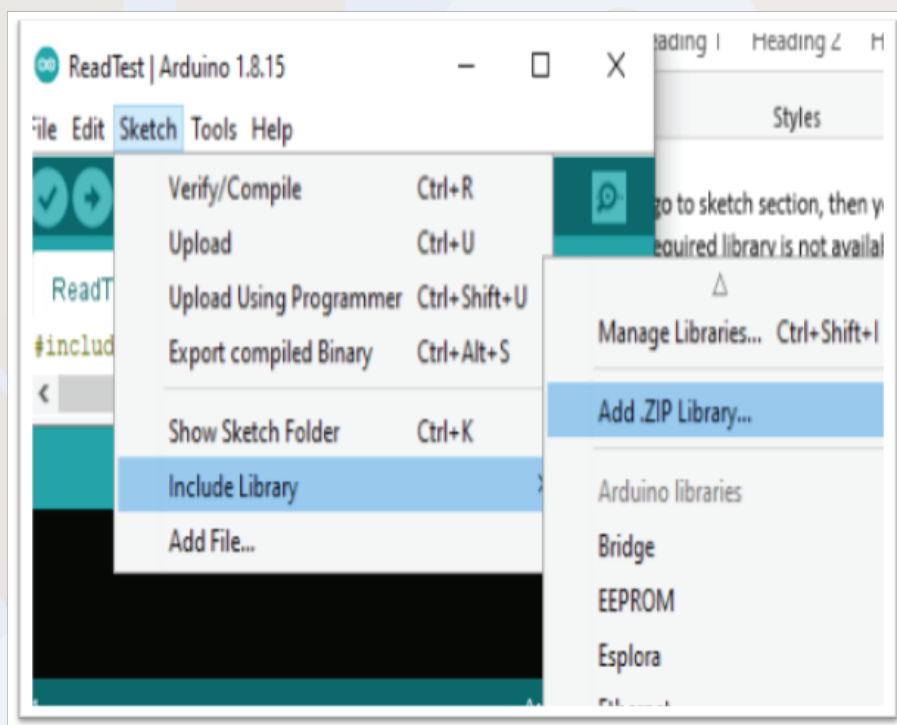
We use Arduino libraries, that means we use objects and methods of classes. So, even though we don't need to write much code, we can learn a lot by modifying existing code.

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 digitalWrite 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.

4.0. I2C LCD Backpack

You must have seen a normal LCD before this or used it in your project to fulfill the project requirement but did you know that this type of LCD with I2C backpacks are also available in the market? Now you must be wondering what is I2C.

I2C is a communication protocol. The main advantage of this communication protocol is its speed and less wiring diagram. The data transfer rate of this communication protocol is 2Mbps.

4.1. I2C Communication:

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.

4.1. I2C Communication:

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.

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.

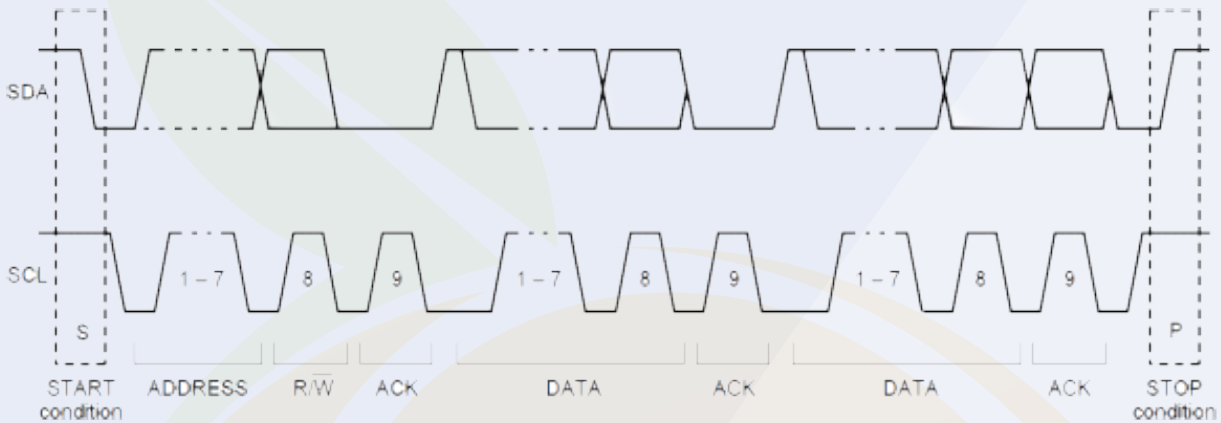
The slave devices that are connected to the master device, waits for the request from the master device and when they receive the request from the master, they either sends the data to the master or receives the data from the master.

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.

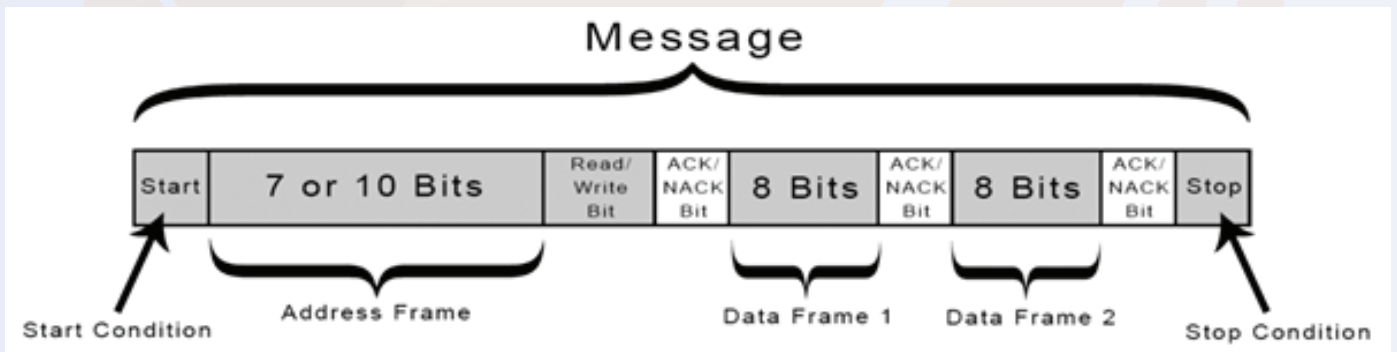
4.1. I2C Waveform – Explained

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



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.



4.1. 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.

4.1. I2C Waveform – Explained

5) ACK /NACK

After each successful transfer of the data, the master sends ACK or NACK Data. This is an acknowledge bit and this confirms the successful transfer of the data from the master to the slave device.

The master sends ack bit to the slave to confirm whether the required slave device is present on the line or not.

If the slave device is present on the line then the slave device will pull that line to low level to tell the master that, it can start the transfer of the data.

When the data frame transfer needed to end, the master will send NACK signal to the device.

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.

So, this is how the I2C communication works. If you have any doubts please let us know in the comment section.

As we have discussed earlier, interfacing process of the LCD with the Arduino using the I2C communication protocol requires less wiring. But what is importance of less wiring.

We can use wire holders and can keep the wiring of the circuit clean then how this feature of the I2C is amazing? We will talk about that part in the coming section of this blog.

4..2. Why less wiring is important and what is I2C communication protocol?

7 to 8 Arduino pins are required to interface a normal LCD display. Whereas the I2C Backpack LCD only takes up two pins and on top of that those pins are also not completely governed by the LCD. You can connect as many I2c devices as possible to those pins.

Now, you must be thinking that if we connect two or more devices to the same pin of Arduino then how the data will be sent to the particular device? Actually, this is another feature of the I2C communication protocol.

When we connect I2C devices to Arduino or any microcontroller, that microcontroller acts as a master and the devices connected to the I2C port act as a slave.

When you connect an I2C device to the microcontroller the I2C address of the device is passed back to the Arduino.

Then when communication is to take place the microcontroller checks the I2C address of each device and once the correct match is found it initiates further communication.

So, I2C communication works like this.

Coming back to our discussion, I2C display you are getting 1602 LCD display in it. The meaning of 1602 is that the display has the ability to display 16 characters on a single line and that there are two such lines.

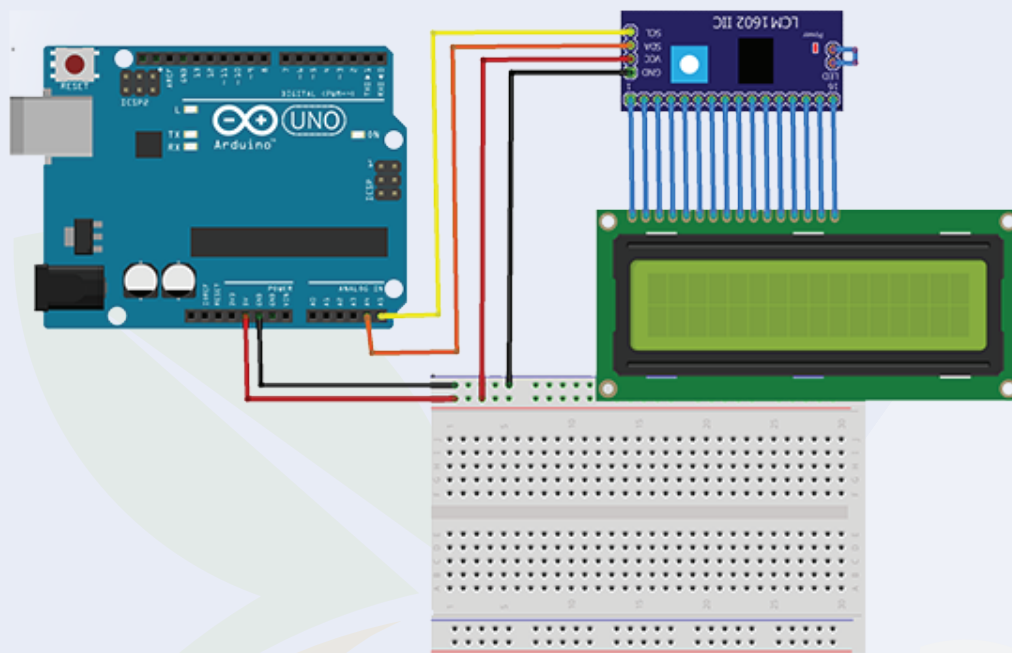
So, that was about the introduction part of I2C display. Now we will talk about its interfacing with Arduino.

4..2. Interfacing I2C LCD With the Arduino

As discussed earlier only two pins of Arduino are required for I2C display and those pins are SCL and SDA. You need to connect the SCL pin of the Arduino to the SCL pin of the LCD and the SDA pin of the Arduino to the SDA pin of the Arduino.

You can refer to the following image to understand more about the interfacing diagram.

4.2. Interfacing I2C LCD With the Arduino



4.3. Arduino Code For I2C Backpack LCD

As we are using I2C Backpack to interface LCD display with Arduino, we need to establish I2C communication between LCD device and Arduino.

You can either design your own code or you can use libraries which are easily available in the market.

In my case I have used ready-made `i2cliquid_crystal.h` library.

One more thing, there is a line in our code on which we are passing I2C address. In my case the I2C address of my LCD display is `0X16` but it may be different in your case. So, if the Arduino code doesn't work for you, don't fret. You can follow the following procedures to resolve the issue.

4.3. Arduino Code For I2C Backpack LCD

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()
{
    // initialize the LCD
    lcd.begin();

    // Turn on the backlight and print a message.
    lcd.backlight();
    lcd.print("Hello, world!");
}

void loop()
{
    // Do nothing here...
}
```

In the above code, we are using LiquidCrystal library. This library does not come preinstalled. You have to install this library manually. I have shared the link of that library below you can use that link and download the library.

<https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library>

4.4. Troubleshooting For I2C backpack

If your LCD is not showing any character on it then there can be two possible reasons for this.

1. Wiring is not correct.
2. The I2C address of the LCD is different from the set address.

To resolve these problems, you must first check the wiring diagram. Make sure you have connected everything correctly. You can refer to the connection diagram to understand the connection diagram.

If the issue is still there then there might be an issue with the I2C address. To solve this problem you must know the I2C address.

And you can use the following code to get the I2C address of your device. This code returns the I2C address of the devices that are connected to the Arduino's I2C communication port.

4.5. I2C scanner For Arduino

This was about I2C Address Scanner code now below code you can use for your application.

If everything is done correctly, the following code will display "Hello Robu!!" on the first row of the LCD display.

If it is not showing any character on the LCD then please follow the above procedure.

If the problem persists please let me know in the comment section.

It was about I2C LCD, if you have any doubt regarding this section please let us know in the comment section.

In the next part of this blog, we will understand the working of servo motor.

4..5. I2C scanner For Arduino

```
#include <Wire.h>

void setup()
{
  Wire.begin();

  Serial.begin(9600);
  while (!Serial);    // Leonardo: wait for serial monitor
  Serial.println("\nI2C Scanner");
}

void loop()
{
  byte error, address;
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
    // The i2c_scanner uses the return value of
    // the Write.endTransmission to see if
    // a device did acknowledge to the address.
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println(" !");

      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknown error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);    // wait 5 seconds for next scan
}
```

5.0. Servo Motor

In this orange arduino kit you are getting a small servo motor. These types of motors are very famous in automation fields and are being widely used in industrial applications.

The main advantage of this type of motors is that these motors are available in the market at very low cost and on top of this the shaft speed of such motors is very precise.

These motors come with the proper casing. If you open the casing of the motor you will see a DC motor and an encoder in it.

The application of the encoder used in a servo motor is to keep track of the stator.

In the below section we will learn about the interfacing of servo motor with Arduino.

5.1. Interfacing the Servo Motor With Arduino

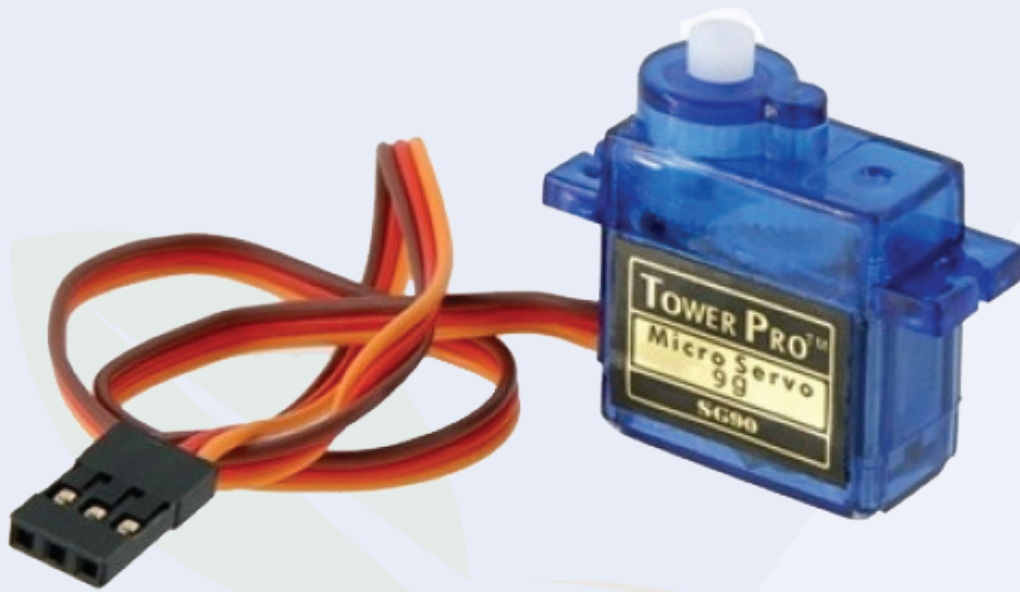
There are three pins in any servo motor I have mentioned the function of those pins below please have a look.

VCC - 4 to 9V - Here you can connect the positive pin of the external power supply to the servo motor.

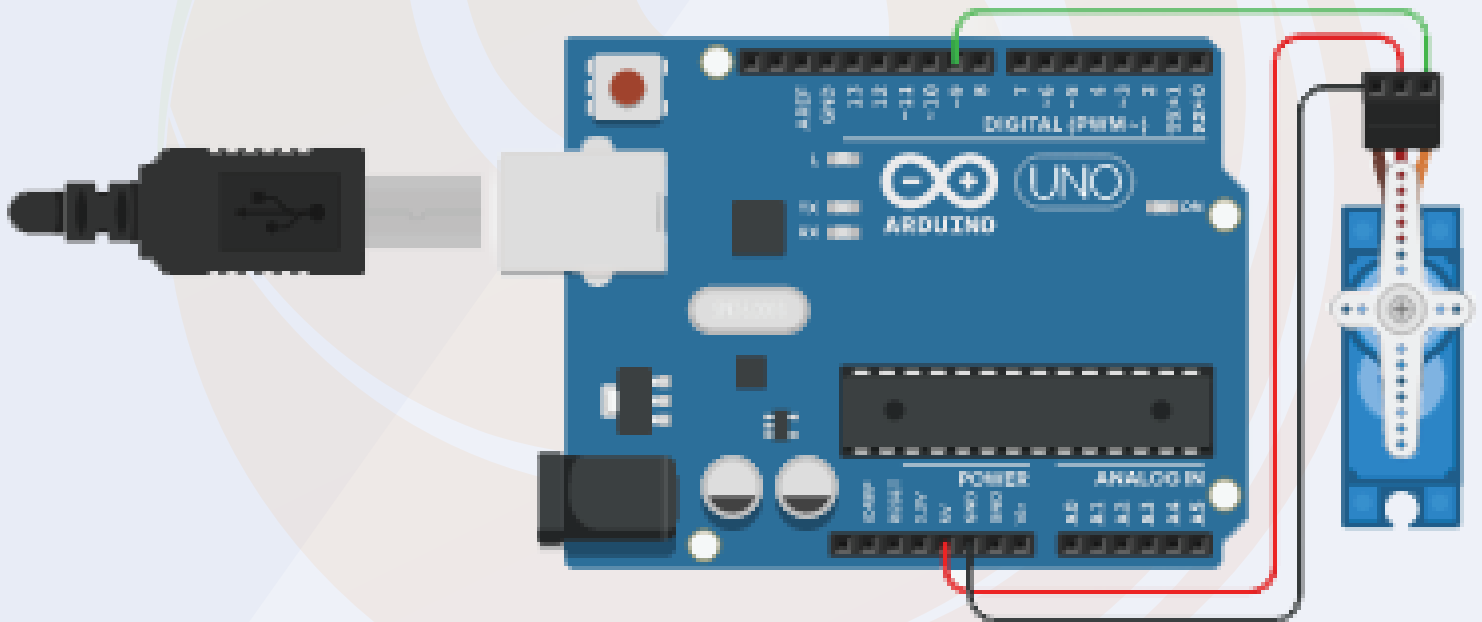
GND - Here you can connect the GND pin of the Arduino and the GND pin of the external power source.

Signal Pin - This is the signal pin of the servo motor. This pin is connected to the internal microcontroller and you need to connect this pin to the GPIO pin of the Arduino.

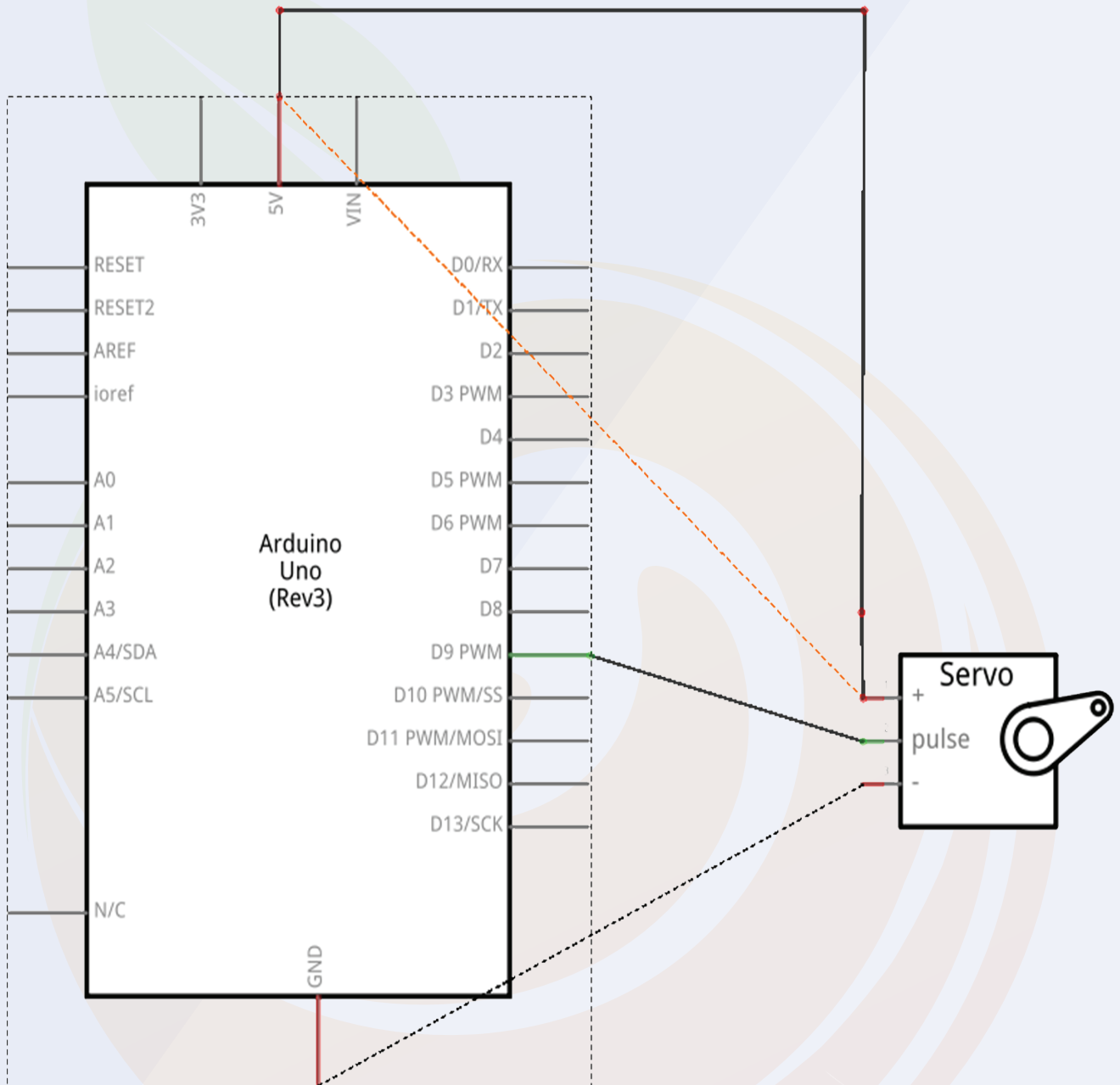
This is how you have to make the connection.



5.1. Interfacing the Servo Motor With Arduino



5.1. Interfacing the Servo Motor With Arduino



5.2. Arduino Code For The Servo Motor

In this section, I have explained the Arduino code that we will be using to control the servo motor.

Before we talk about Arduino code, I would like to tell you about the fundamentals of coding.

As discussed earlier, the servo motor has an inbuilt control unit which keeps track of the moment of the unit shaft. The signal pin is connected only to the same control unit.

When we give pwm signal to that control unit. The control unit turns on the DC motor for a specific period depending on the duration of the received PWM signals.

For example, if we give a 2ms-term PWM signal to the servo motor, the control unit will rotate the motor 180-degrees. When we give a PWM signal of 1ms duration to the servo motor, it will rotate the motor towards the 0-degree position.

Talking about the code, in this code we are using servo.h library. This library is specially designed for servo motor and BLDC motor.

5.2. Servo Motor Code Explanation

We are using two function in the code and they are as follows;

1. Servo.attach
2. Servo.write

Servo.attach This function is used to define the signal pin for the servo motor. Make sure you only connect the PWM pin to this pin. If you connect the digital pin to this pin then the servo motor will not work properly.

Servo.write This function takes one parameter and that is angle. We have to pass angle value to this pin. In the following code, we are commanding the motor to rotate 90 degrees.

5.2. Servo Motor Code Explanation

It was about the code now you can use the following code and make necessary changes.

If you face any problem please let me know.

```
#include <Servo.h>

int pos = 0;

Servo servo_9;

void setup()
{
  servo_9.attach(9, 500, 2500);
}

void loop()
{
  // sweep the servo from 0 to 180 degrees in steps
  // of 1 degrees
  for (pos = 0; pos <= 180; pos += 1) {
    // tell servo to go to position in variable 'pos'
    servo_9.write(pos);
    // wait 15 ms for servo to reach the position
    delay(15); // Wait for 15 millisecond(s)
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    // tell servo to go to position in variable 'pos'
    servo_9.write(pos);
    // wait 15 ms for servo to reach the position
    delay(15); // Wait for 15 millisecond(s)
  }
}
```

6.0. Stepper Motor

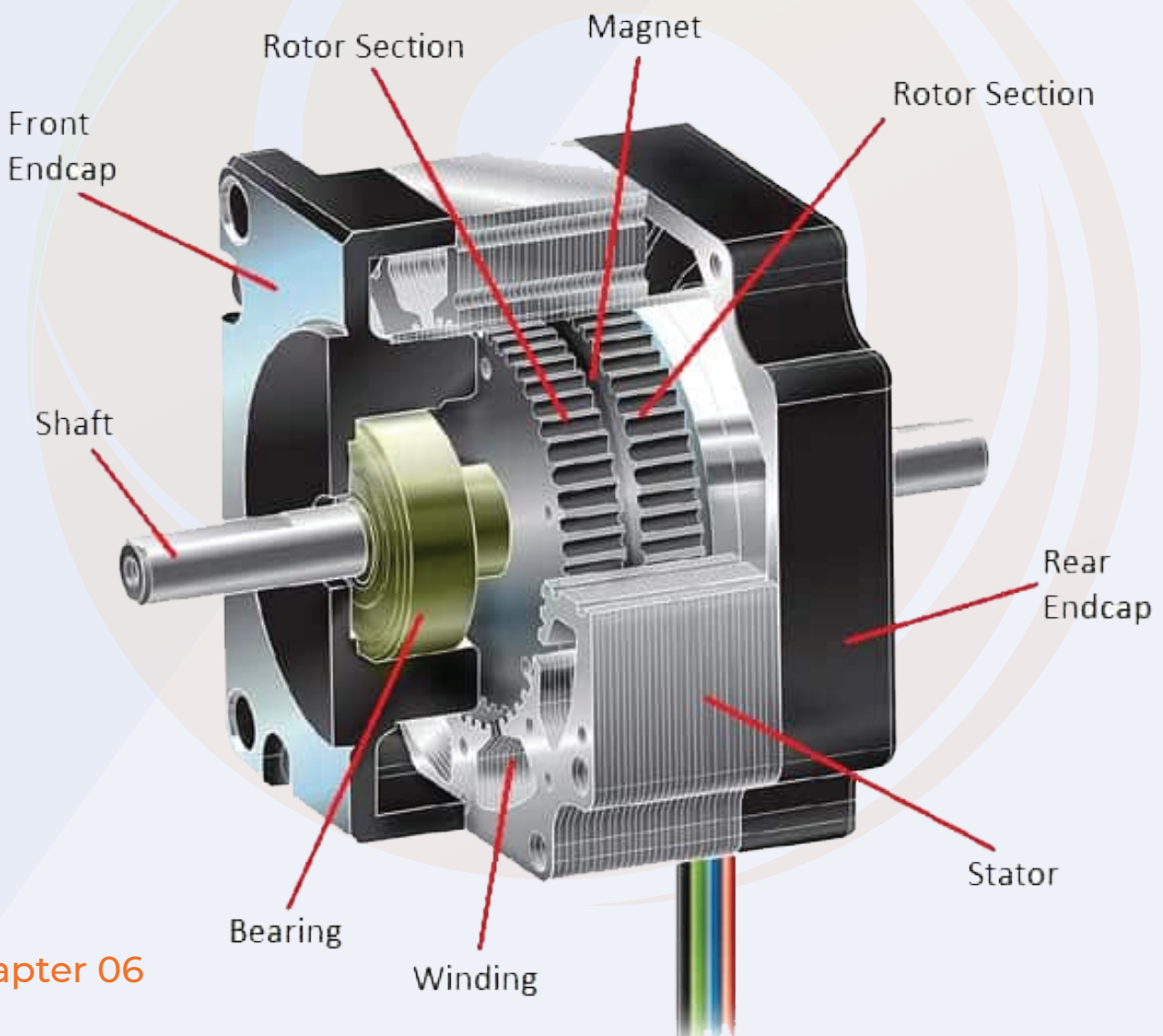
6.0. Stepper Motor

As you know, stepper motor is the type of DC motor which converts the electrical pulses into step-by-step mechanical motion of the shaft.

Unlike other DC motors, this motor has a permanent magnet rotor that operates when the stator is energized. The construction of the stator is similar to that of a normal DC motor. The only difference is that the stator of this type of motor has mechanical teeth. After providing the pulses generated by the microcontroller to the stator, these teeth align with the teeth of the rotor.

The sequence of pulses generated by the microcontroller allows the shaft of the motor to rotate in discrete steps. This motor uses a certain amount of such pulses to complete one revolution, these pulses are also responsible for controlling the speed of the motor. Increasing the frequency of the input pulses increases the speed of the motor shaft rotation.

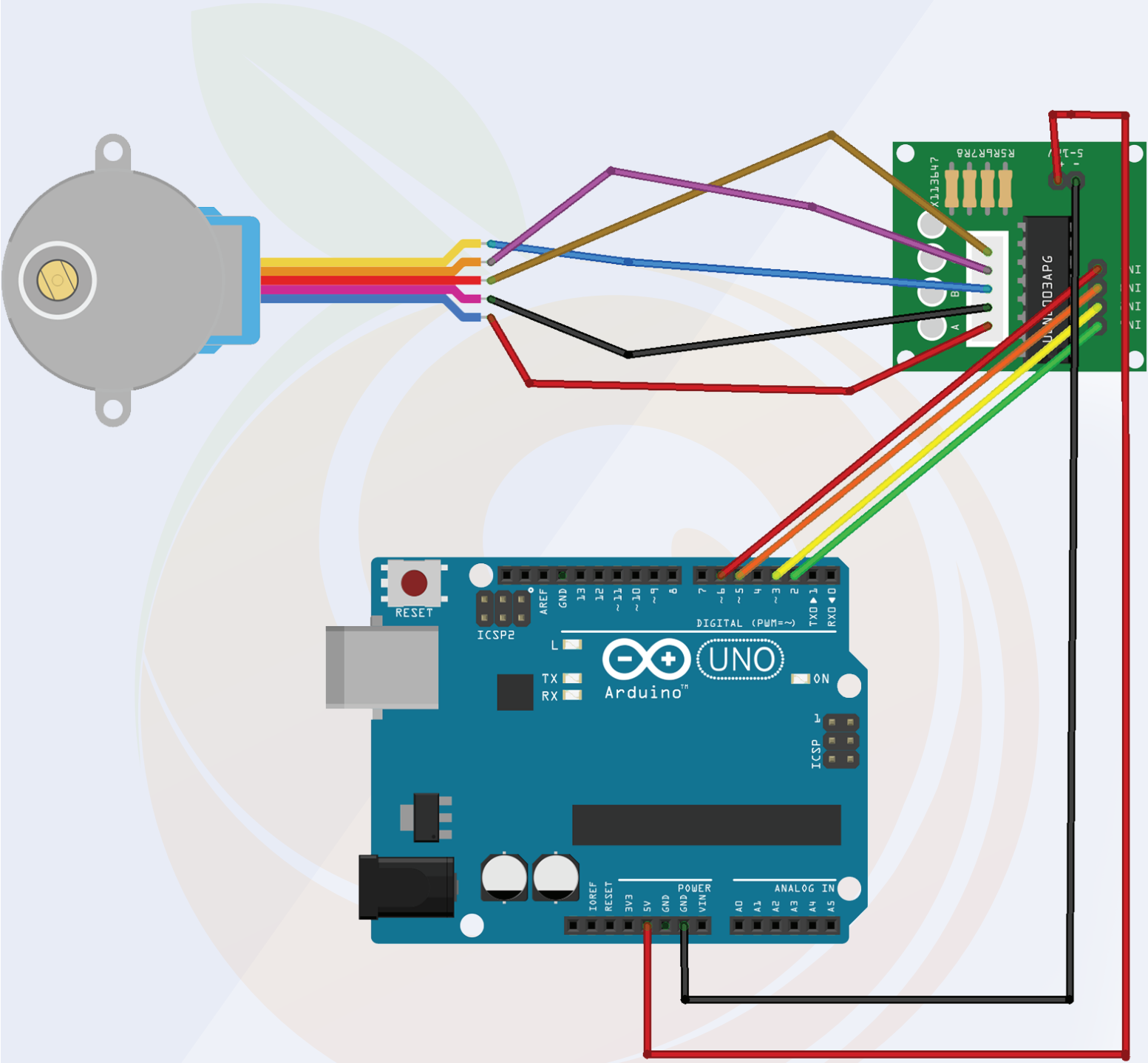
The following picture shows the internal construction of the stepper motor.



6.1. Interfacing Stepper Motor With Arduino

You are getting a 28byj-48 stepper motor and UNL-2003 stepper motor driver with this Orange Intermediate Kit. This motor has four pins and you have to connect those pins to the stepper motor driver as shown in the picture.

Please check the following image and do the connection accordingly.



frizing

6.1. Arduino Code For Controlling The Stepper Motor

In the following section I have shared the code for controlling the stepper motor. With the help of the following code, we are generating a sequence. That sequence will rotate the motor in clockwise or anticlockwise direction.

You can use the following code and see the stepper motor rotating. As far as I know I have covered everything about the stepper motor if you have any doubts please let us know in the comment section.

```
#include <Stepper.h>

const int stepsPerRevolution = 200; // change this to fit the number of steps per revolution
// for your motor

// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 2, 3, 5, 6);

int stepCount = 0; // number of steps the motor has taken

void setup() {
  // nothing to do inside the setup
}

void loop() {
  // read the sensor value:
  int sensorReading = analogRead(A0);
  // map it to a range from 0 to 100:
  int motorSpeed = map(sensorReading, 0, 1023, 0, 100);
  // set the motor speed:
  if (motorSpeed > 0) {
    myStepper.setSpeed(motorSpeed);
    // step 1/100 of a revolution:
    myStepper.step(stepsPerRevolution / 100);
  }
}
```

7.0. DS1302 RTC Module

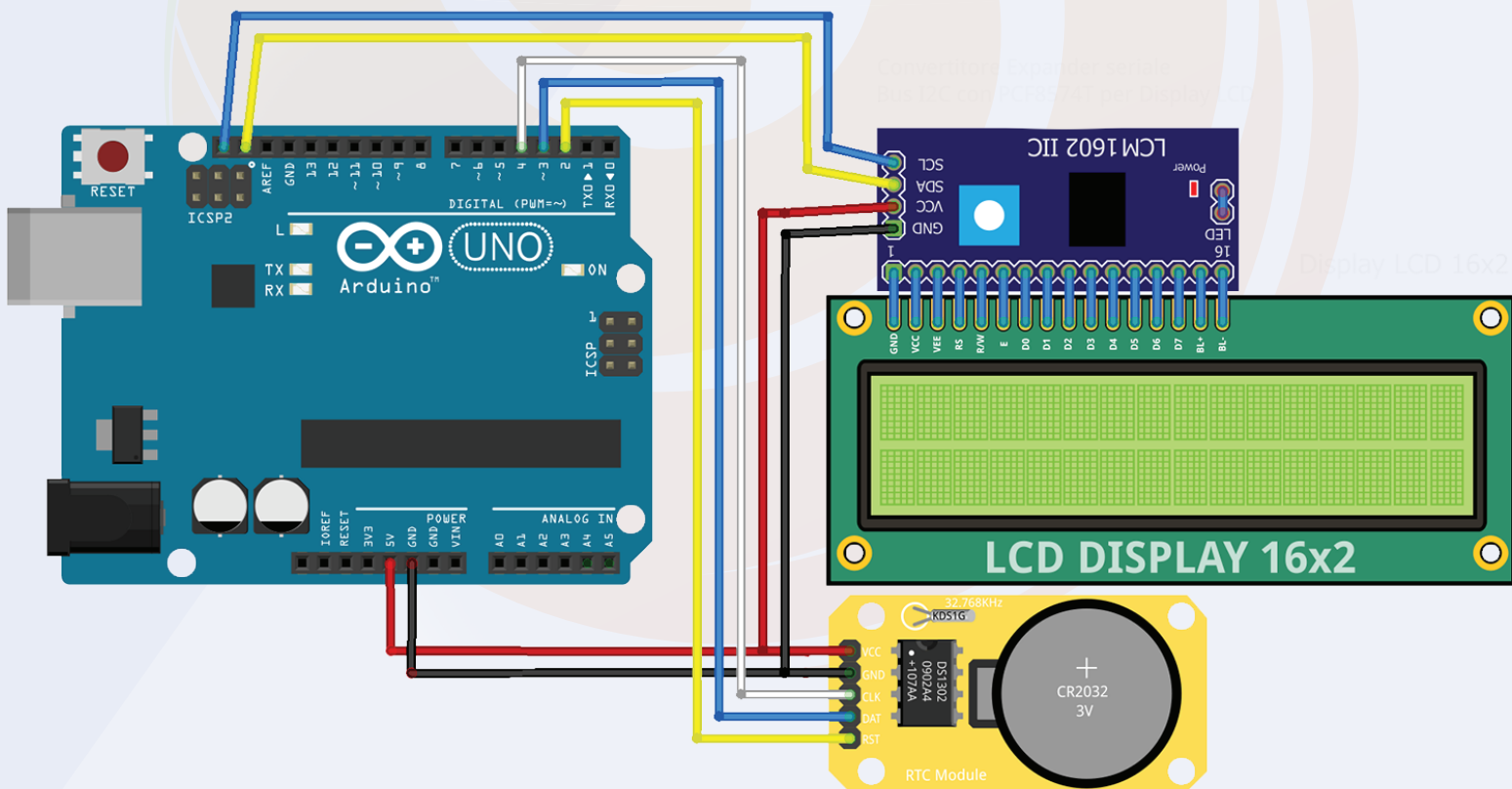
RTC – As the name suggests, the DS1307 RTC module is used as a module to remember TIME and DATE and, as I told you earlier, it has an inbuilt battery which keeps the RTC module running . With batteries involved, the module stays on and gives accurate time information whenever asked.

7.1. Features of DS1307 RTC Module

- Two-wire I2C interface
- Hour: Minutes: Seconds AM/PM
- Leap year compensation Accurate calendar up to the year 2100
- Consumes Less than 500nA in Battery-Backup
- 1Hz output pin
- 56 Bytes of Non-volatile memory available to the user 4KB of serial electrically erasable and programmable read-only memory (EEPROM)
- Embed DS18B20 temperature sensor interface with the pull-up resistor.

7.2. DS1302 RTC Module Interfacing With Arduino

As mentioned in the features, this module uses I2C communication to communicate with the master (microcontroller). So, to communicate with this module, you need to establish I2C communication between DS1302 RTC and Arduino.



7.3. Arduino Code For DS1302 RTC Module

```
#include <DS1302.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

DS1302 rtc(2, 3, 4);

LiquidCrystal_I2C lcd(0x27, 16, 2); // Here we are using 0x27 default address. If the
display is not working, then try changing the I2C address.

void setup()
{
    rtc.halt(false);
    rtc.writeProtect(false);

    Serial.begin(9600);

    lcd.init();
    lcd.backlight();

    //rtc.setDOW(SUNDAY);
    //rtc.setTime(11, 32, 0);
    //rtc.setDate(12, 2, 2017);
}

void loop()
{
    Serial.print(rtc.getDOWStr());
    Serial.print(" ");

    Serial.print(rtc.getDateStr());
    Serial.print(" -- ");

    Serial.println(rtc.getTimeStr());

    lcd.clear();
    data ();
    ora ();

    delay (1000);
}

void data ()
{
    lcd.setCursor(0, 0);
    lcd.print(rtc.getDOWStr());
    lcd.print(" ");
    lcd.print(rtc.getDateStr());
}

void ora()
{
    lcd.setCursor(0, 1);
    lcd.print(rtc.getTimeStr());
}
```


8.0. MFRC-522 RC522 RFID

RFID is a radio identification system being widely used in the automation industry.

RFID cards are used in industry to digitally tag items.

This technology uses RF signals to transfer information. It has three main components which are as follows.

8.1. RFID Reader

This is a reader that reads the information of an RFID tag. This module has inbuilt antenna and radio transmitter receiver which receives the signal transmitted by the RFID receiver.

8.2. RFID Tags

RFID tags are mounted on items that need to be counted. There are two types of RFID tags available in the market.

- Active RFID Tags
- Passive RFID Tag

Active RFID tags have an inbuilt battery which provides power to the RFID tag control unit whereas passive RFID tags do not have an inbuilt battery which receives power from the RFID readers and after receiving the power it returns its information to the RFID reader.

8.3. Working of the RFID Technology

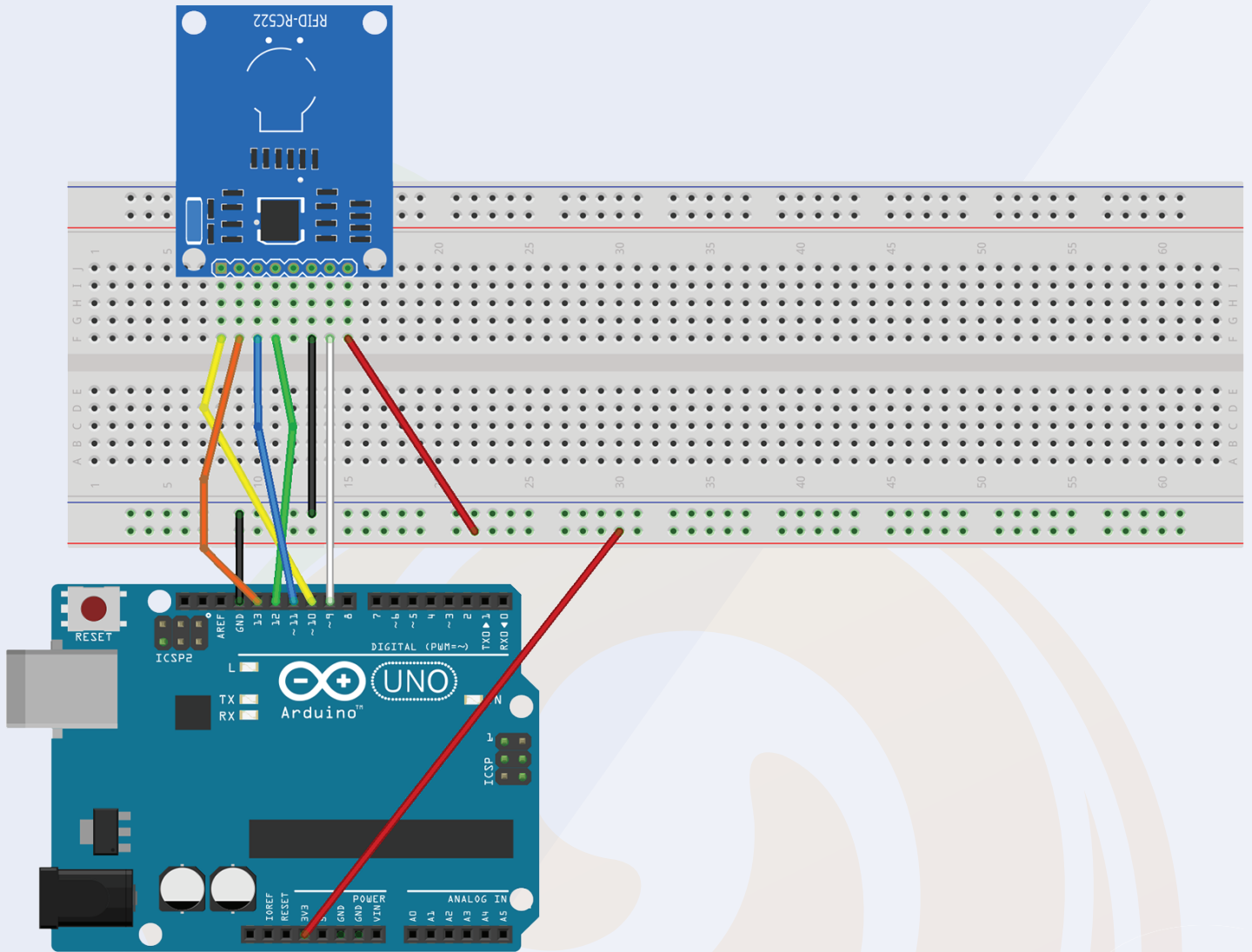
Till now we have learned about the components used in RFID technology and also learned the importance of those components.

In this part, we will learn how these components are integrated to form a complete unit and how it works.

So, we learned about RFID readers. Most of the RFID readers available in the market use SPI communication to transfer the data to the master device.

In the image below you can see that we have connected the RFID receiver to the SPI port of the Arduino.

8.3. Working of the RFID Technology



When we place the RFID card near the RFID reader, the RFID reader will detect the tag's RFID number and send that number to the Arduino. And that's how RFID works.

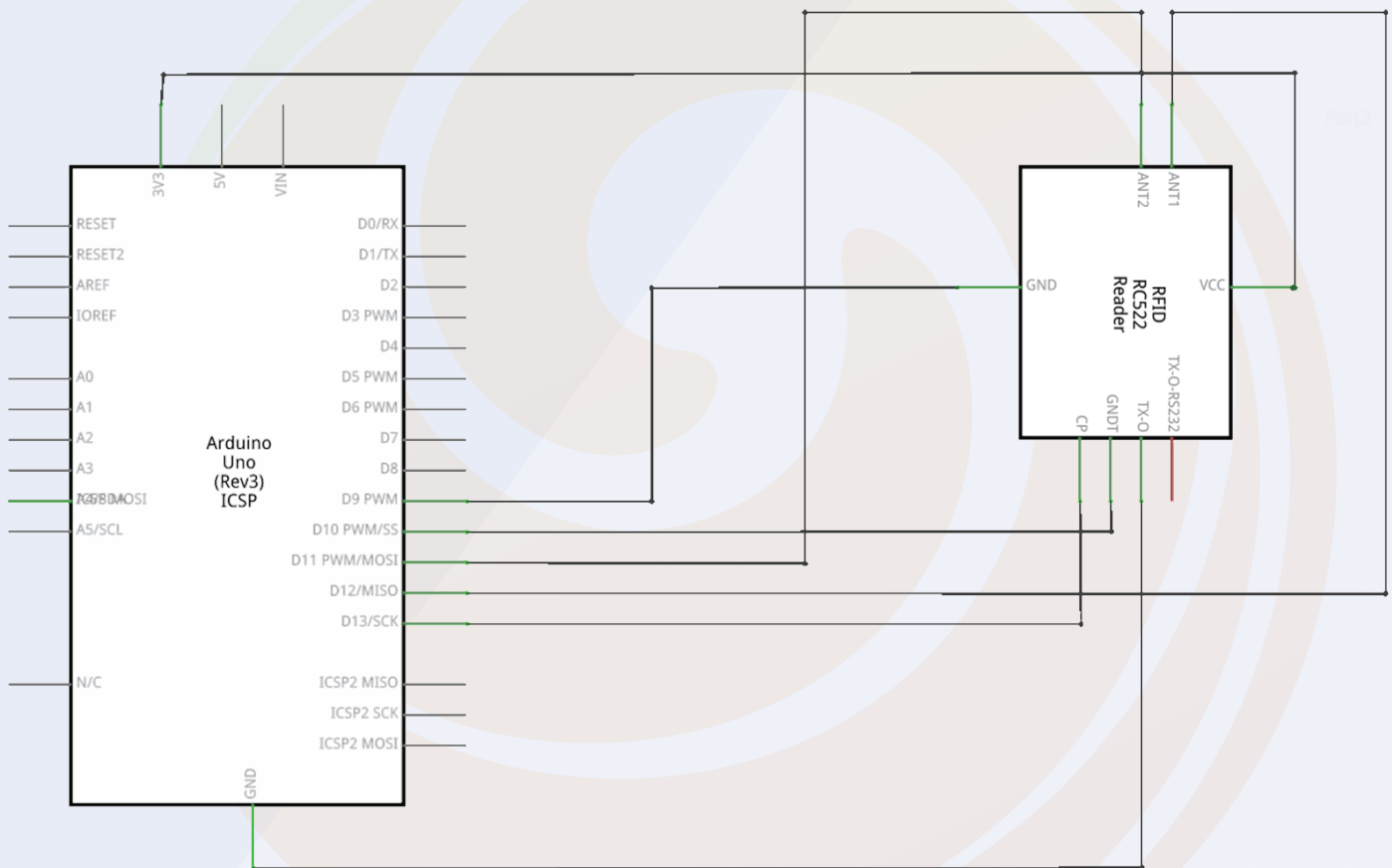
I hope you are now clear about RFID technology. If you have any doubt you can ask me in the comment section. In the next part of this blog we will learn about interfacing RFID card with Arduino.

8.4.MFRC-522 RC522 RFID Interfacing With Arduino

The RFID you get with this kit uses SPI communication. SPI communication is parallel line communication and uses four pins which are as follows.

- MOSI – Master out slave in
- Miso - Master in slave out
- SCK – Clock line
- VCC and GND - Power Pins

To interface the given RFID reader with the Arduino we will be using the Arduino's SPI communication port and the interfacing diagram is as follows.



8.5. Arduino Code For MFRC522 RC522 RFID

The following code is for the RFID reader if you have connected the RFID reader with the Arduino properly then you can use the following code and start working with the RFID reader.

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup()
{
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
  Serial.println("Approximate your card to the reader...");
  Serial.println();
}

void loop()
{
  if (!mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }

  if (!mfrc522.PICC_ReadCardSerial())
  {
    return;
  }

  Serial.print("UID tag :");
  String content = "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("Message : ");
  content.toUpperCase();
  if (content.substring(1) == "29 C2 07 5E") // Make sure you change this with your own UID number
  {
    Serial.println("Authorised access");
    Serial.println();
    delay(3000);
  }
  else {
    Serial.println(" Access denied");
    delay(3000);
  }
}
```

9.Sound Sensor

Sound sensors are the modules that are used to detect the sound sensor. It has a microphone which detects the sound waves and generates a digital output according to the received frequency of the sound.

It has inbuilt gain amplifier which is used to adjust the gain of the module. If the module is not generating the output properly then you can change the potentiometer to adjust the gain of the module.

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

9.1. Interfacing Sound Sensor With Arduino

The sound sensor we are getting with this kit has three pins which are as follows.

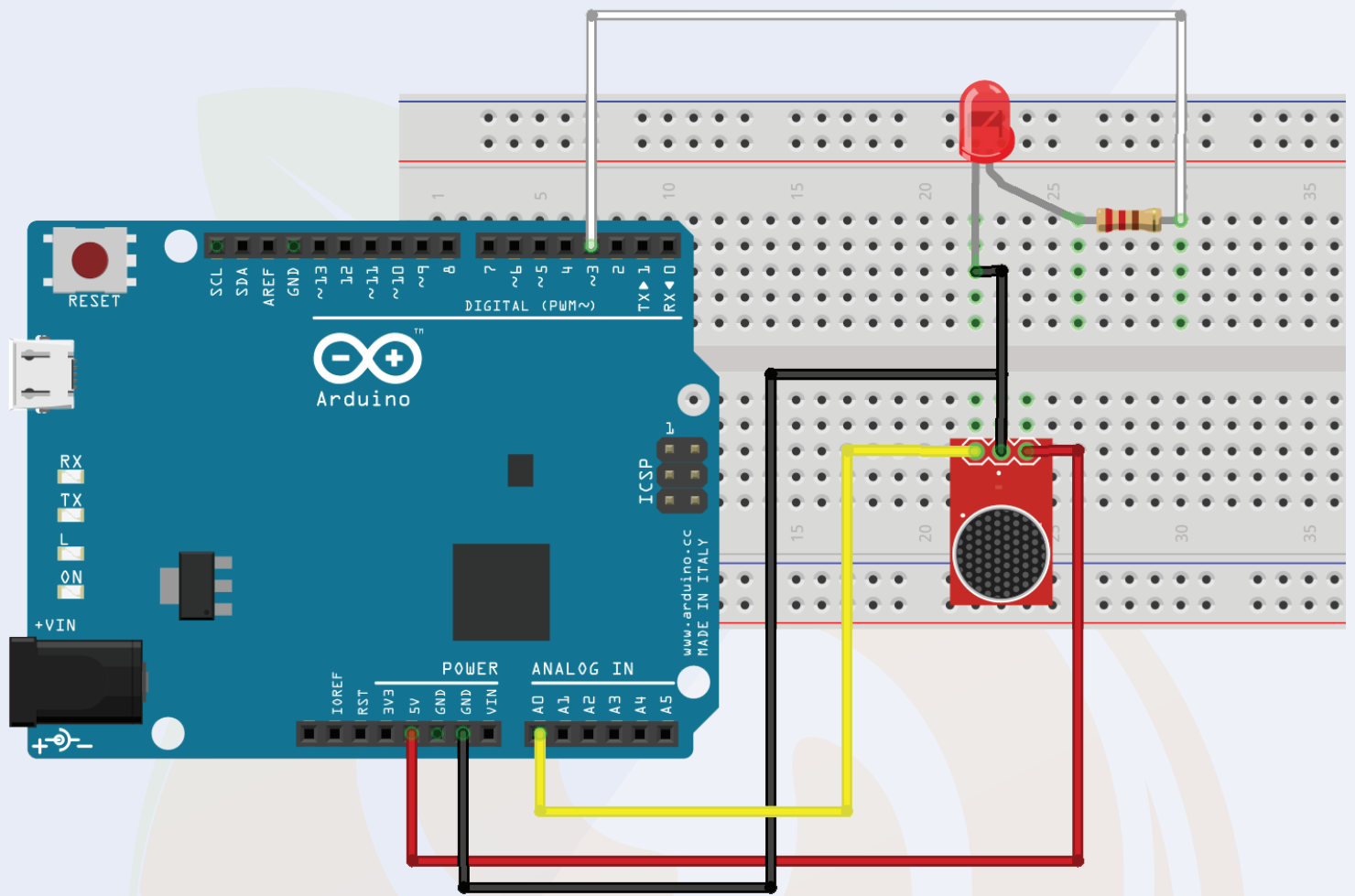
Vcc - You can connect this pin to the 5v pin of the Arduino.

GND - This is the GND pin of the module. You can connect this pin to the GND pin of the Arduino.

Signal Pin - Here you will find the output of the signal. The module will generate a high level signal if the frequency of the signal is present and the module will generate a low level signal if it is silent.

The connection diagram is so simple. I have shared the interfacing diagram below. You can see the image below to understand the connection diagram.

9.1. Interfacing Sound Sensor With Arduino



9.2. Arduino Code For Sound Sensor

As we know that sound sensor generates either high level signal or low-level signal, so to find out the output of sensor, we are using digital read function.

You can use the following code to work with this module.

9.3 . Arduino Code For Sound Sensor.

```
int soundSensor=A0;
int LED=3;
boolean LEDStatus=false;

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

void loop() {

  int SensorData=digitalRead(soundSensor);
  if(SensorData==1){

    if(LEDStatus==false){
      LEDStatus=true;
      digitalWrite(LED,HIGH);
    }
    else{
      LEDStatus=false;
      digitalWrite(LED,LOW);
    }
  }
}
```

10. DHT11 Temperature and Humidity Sensor Module

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

The DHT11 sensor is designed in such a way that it will retain moisture. When water vapor collects on subtraction of sensor. The ions will be released by subtracting. An increase in ions will decrease the resistance between the two electrodes.

10.0. DHT11 Temperature and Humidity Sensor Module

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

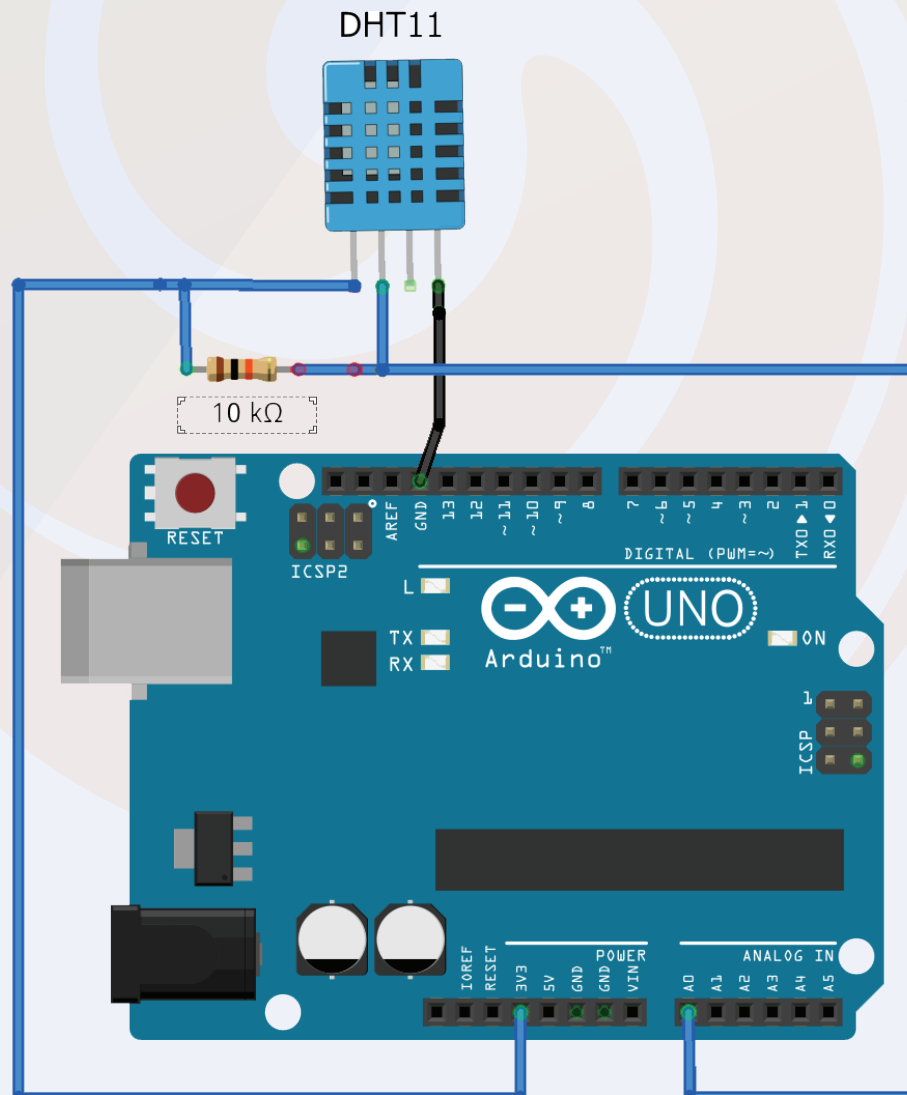
Talking about the temperature sensor, the NTC type thermistor has been given in the DHT11 sensor. This thermistor is nothing but a temperature sensor.

The output value of the thermistor changes according to the change in the surrounding temperature. So by calculating the change in the output voltage of the sensor we can find out the temperature readings.

10.1. Interfacing DHT11 Sensor With Arduino

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

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



10.2. Arduino Code DHT11 Interfacing With Arduino

In this code we are using DHT.h library. This library will take care of all the timing requirement and will return the required results only.

```
#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"));
}
```

11. Water Level Sensor

This sensor works on the principle of variable resistance. The sensor consists of a series of parallel exposed conductors. Together this series acts as a variable resistor whose resistance varies according to the water level in the water tank.

The more the sensor is submerged in water, the better the conductivity and the lower the resistance. The less the sensor is immersed in water, the worse the conductivity and the higher the resistance.

The water level sensor is designed according to the resistance of the water produced. i.e. it will produce a voltage proportional with the resistance.

11.1. Interfacing Water level Sensor With Arduino

This sensor has three output pins which are as follows.

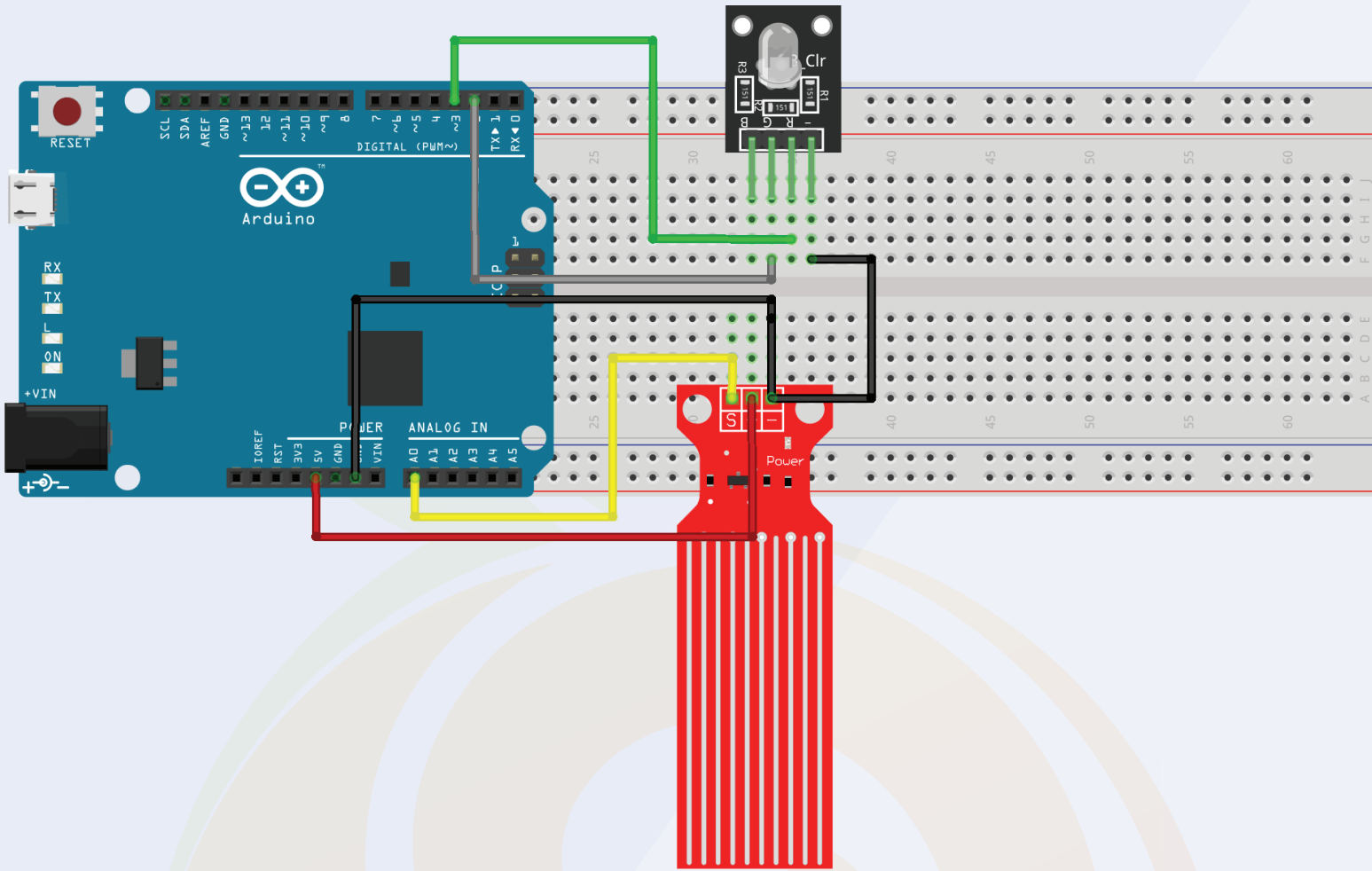
The S (Signal Pin) is an analog output that will be connected to the Arduino's analog pin.

+VCC is the sensor's powering pin. Accepted input voltage is 3.3v to 5v.

The -GND bus is a ground connection.

- In this interfacing, connect the signal pin of the sensor to A0 of the Arduino.
- Connect the sensor's VCC to digital pin 7 of the Arduino.
- Connect the sensor's GND to the Arduino's ground.
- Connect the cathodes of the red, yellow, green LEDs to digital pins 2,3,4 of the Arduino respectively.
- Connect all the anodes of the LED to the GND pins of the Arduino as usual. In this way, you can interface this sensor with Arduino.

11.1. Interfacing Water level Sensor With Arduino



fritzing

11.2. Arduino Code For Water Level Sensor With Arduino

This sensor works on the principle of variable resistance. The sensor consists of a series of parallel exposed conductors. Together this series acts as a variable resistor whose resistance varies according to the water level in the water tank.

As the more water sensor is submerged, the better the conductivity and the lower the resistance. The less water the sensor is immersed in, the worse the conductivity and the higher the resistance.

The water level sensor is designed according to the resistance of the water produced. i.e. it will produce a voltage proportional with the resistance.

12. 8*8 Matrix Display

8*8 matrix is used in many applications. People also use 8*8 matrix display as snake game display. How can we use this display for snake game? This question may be on your mind.

We will learn about that part in this section.

In the previous section, we learned about the performance of seven segments. In this section, we will learn about 8*8 matrix display and interfacing 8*8 matrix display with Arduino.

These displays are quite popular in the market. Using these displays we will not only be able to print plain text but we can also animate the text which we will print on the display.

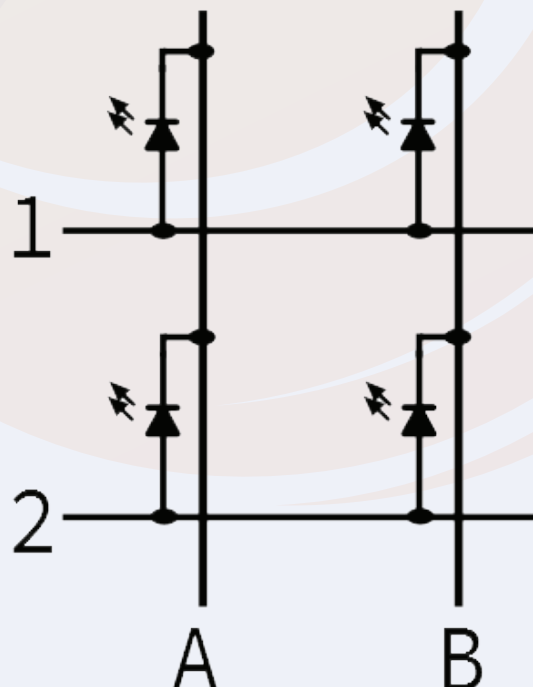
To understand how these displays can be used for animation or snake games, we first need to understand the basics of 8*8 matrix displays.

12.1. Construction of The 8*8 Matrix Display

The 8*8 matrix display is nothing but a combination of several LEDs. These LEDs are connected in a series, parallel configuration.

In series and parallel configuration, the anode and cathode terminals of the LED are connected to each other in the following manner.

Please refer to the following picture to understand the working of 8*8 matrix display.

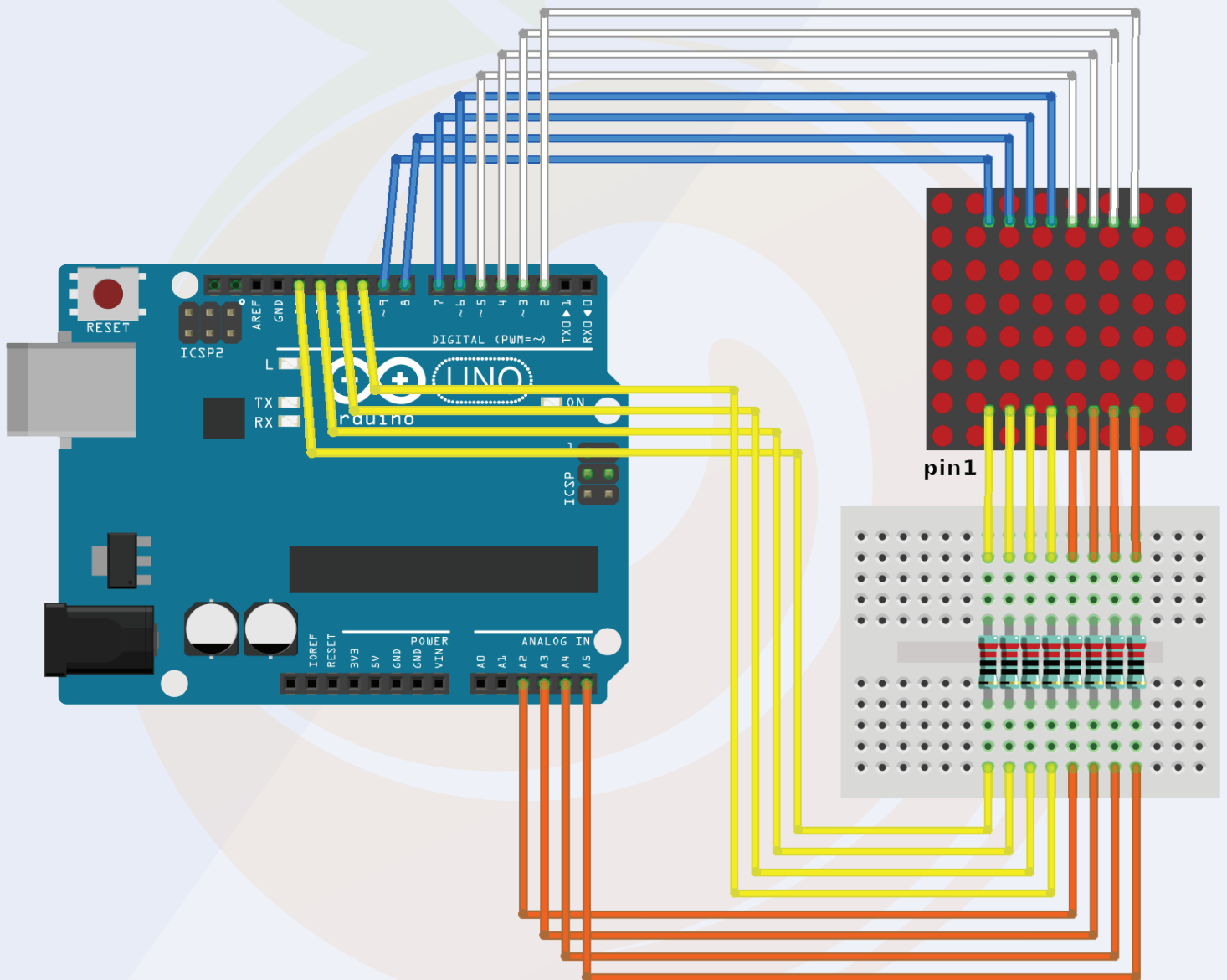


12.1. Construction of The 8*8 Matrix Display

Now that we know about building 8*8 LED matrix display, in the below section, we will learn about interfacing 8*8 LED matrix with Arduino. The dot matrix display available in the market uses Max 232 IC. But what we are getting with this kit, there is no max 232 IC.

MAX232 is a multiplexer, which is used to protect the pins required by the dot matrix display to turn on the LEDs of the dot matrix display.

But since our module doesn't have that max 232 IC, we need to connect each pin of the dot matrix display to the GPIO pin of the Arduino.



fritzing

12.2. Arduino Code For 8*8 Dot Matrix Display

```
// 2-dimensional array of row pin numbers:
int R[] = {2,7,A5,5,13,A4,12,A2};
// 2-dimensional array of column pin numbers:
int C[] = {6,11,10,3,A3,4,8,9};

unsigned char biglove[8][8] = //the big "heart"
{
  0,0,0,0,0,0,0,0,
  0,1,1,0,0,1,1,0,
  1,1,1,1,1,1,1,1,
  1,1,1,1,1,1,1,1,
  1,1,1,1,1,1,1,1,
  0,1,1,1,1,1,0,
  0,0,1,1,1,0,0,
  0,0,0,1,0,0,0,
};

unsigned char smalllove[8][8] = //the small "heart"
{
  0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,
  0,0,1,0,0,1,0,0,
  0,1,1,1,1,1,0,
  0,1,1,1,1,1,0,
  0,0,1,1,1,0,0,
  0,0,0,1,0,0,0,
  0,0,0,0,0,0,0,
};

void setup()
{
  // iterate over the pins:
  for(int i = 0;i<8;i++)
  // initialize the output pins:
  {
    pinMode(R[i],OUTPUT);
    pinMode(C[i],OUTPUT);
  }
}

void loop()
{
  for(int i = 0 ; i < 100 ; i++) //Loop display 100 times
  {
    Display(biglove); //Display the "Big Heart"
  }
  for(int i = 0 ; i < 50 ; i++) //Loop display 50 times
  {
    Display(smalllove); //Display the "small Heart"
  }
}

void Display(unsigned char dat[8][8])
{
  for(int c = 0; c<8;c++)
  {
    digitalWrite(C[c],LOW);//use thr column
    //loop
    for(int r = 0;r<8;r++)
    {
      digitalWrite(R[r],dat[r][c]);
    }
    delay(1);
    Clear(); //Remove empty display light
  }
}

void Clear()
{
  for(int i = 0;i<8;i++)
  {
    digitalWrite(R[i],LOW);
    digitalWrite(C[i],HIGH);
  }
}
```

13. 4*4 Keypad

Keypads are used in electronic devices to take input from the user. We can see the use of keypads in many devices like coffee vending machines, ATM machines.

But do you know how the keypad works?

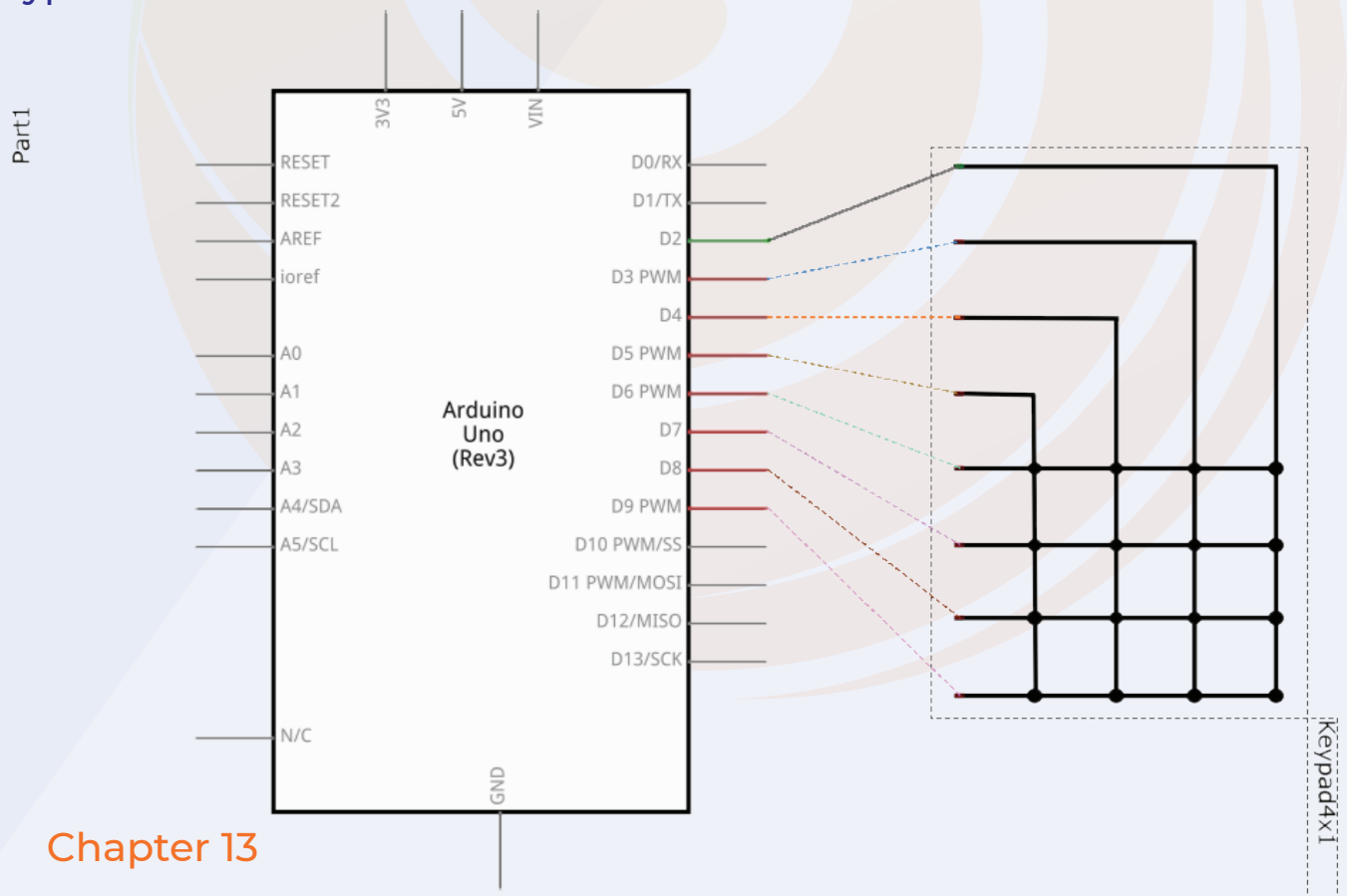
In this section, we will learn about keypads and also learn how to use keypads with Arduino.

13.1. Working of The 4*4 Matrix Keypad With The Arduino

So, to understand the working of 4*4 keypad, we must first know the working of the switch.

To know about the switch, you can check the above section. In the above section, we have mentioned the types of switches which work as switches. So, coming back to our discussion, the 4*4 keypad is a combination of 64 switches. These switches are connected in series and parallel configuration.

Please refer to the following image to understand the circuit diagram of 4*4 keypad.



13.1. Working of The 4*4 Matrix Keypad With The Arduino

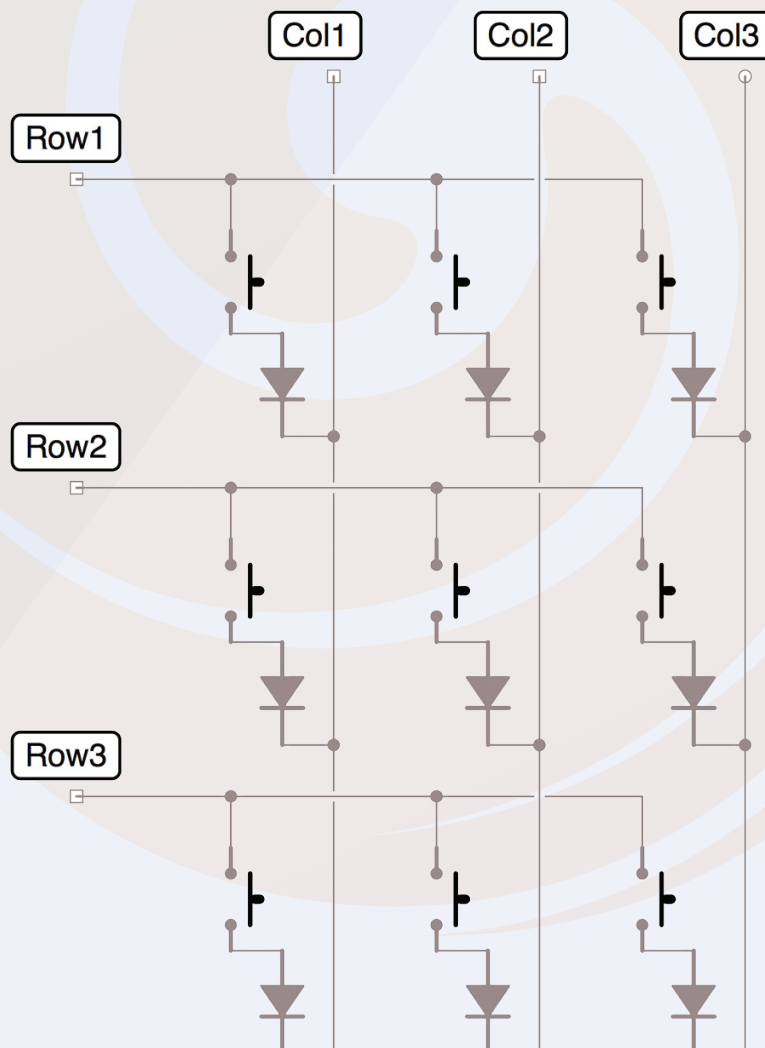
In the image above, you can see the combination of 4 rows and 4 columns. In those rows and columns, switches are connected. When we press the switch, the position of the particular column or row to which the switch is connected changes.

13.2. How to identify which key on the 4*4 keypad is pressed?

To detect the key pressed on the keypad, we will first do the following setup.

Please refer to the following setup. You do not need to apply the following setup. You just check the image and try to understand the setup.

If you have any doubts then you can mention your doubts in the comment section.



13.2. How to identify which key on the 4*4 keypad is pressed?

In the above setup, the column pins of the keyboard are connected to 5V and the row pins are connected to the GND pin of the supply.

When we press a switch on the keypad, a LOW voltage is generated at the output of the column to which the switch is connected.

When we get a low level signal on a column, we can say that the switch that has been pressed is present on that column.

Now we know the column on which the pressed switch is present but each column has a total of four switches.

We have to use the scanning method to identify which of the four switches in the column have been pressed.

In the scanning method, we put zero volts on each row pin one by one and we read the position of the column pins.

So in this way the column from which the switch is pressed will generate the high level signal and the other column will generate the low level signal. So in this way we can find out which key of the keypad is pressed.

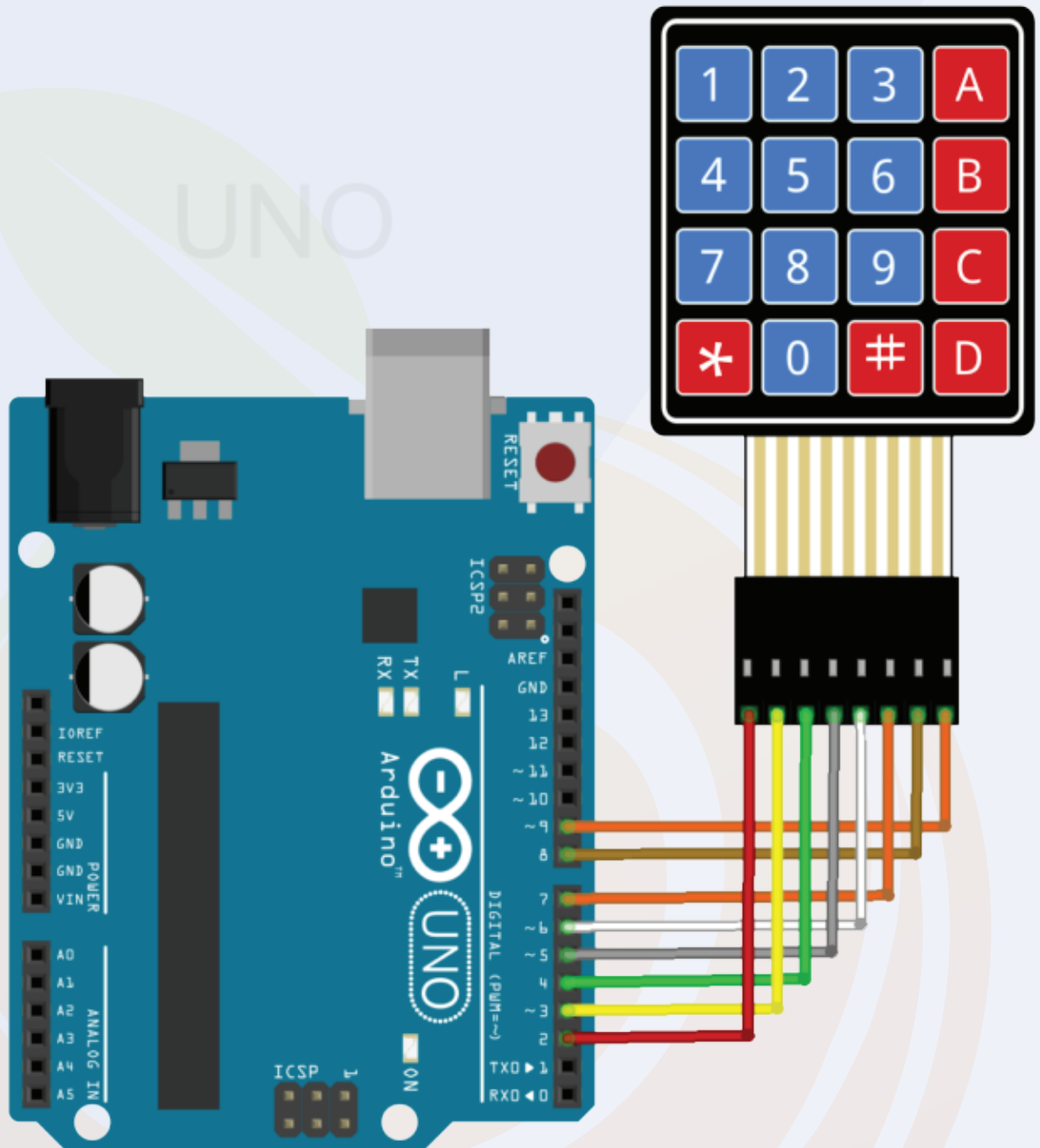
If you have any doubt please let me know in the comment section.

13.3. Interfacing 4*4 Matrix Display With Arduino

It is very easy to interface the keypad with the Arduino. There are total eight pins in 4*4 keypad and we connect those pins to any GPIO pins of Arduino.

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

13.4. Interfacing 4*4 Matrix Display With Arduino



13.5. Arduino Code For 4*4 Matrix Display with Arduino

The main purpose of this project is to read the output of the pins and write 5v on the pins. So, for this purpose, we can use digitalRead and digitalWrite functions.

This is one way to work with keypad but we can also use readymade library.

In the following code we are using keypad.h library.

You do not need to install this library because you already have this library installed on your system. You can directly use the following code without installing any library.

```
#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

byte rowPins[ROWS] = {9, 8, 7, 6}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {5, 4, 3, 2}; //connect to the column pinouts of the keypad

//Create an object of keypad
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

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

void loop(){
  char key = keypad.getKey();// Read the key

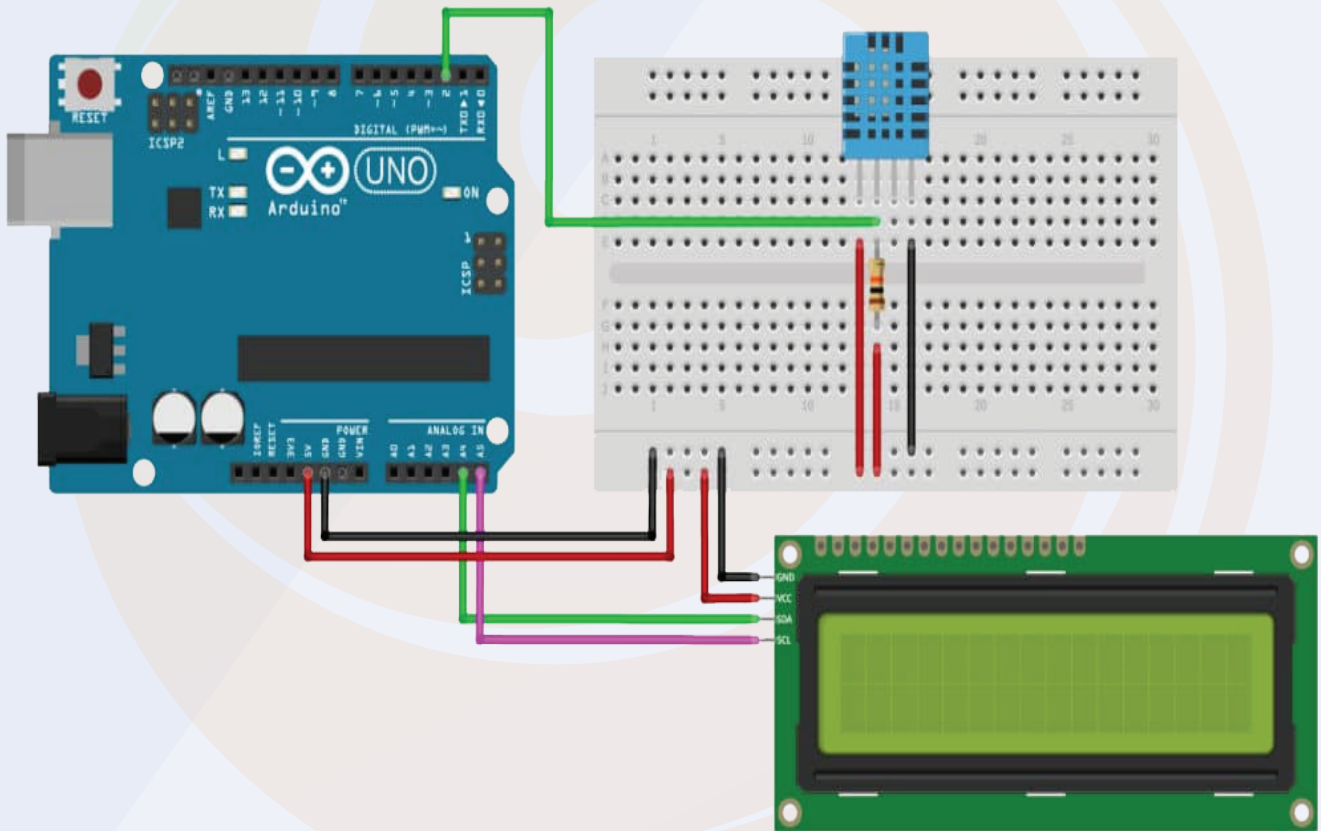
  // Print if key pressed
  if (key){
    Serial.print("Key Pressed : ");
    Serial.println(key);
  }
}
```

14.1. Projects

14.1. Room Temperature Indicator Using Arduino

Room Temperature indicator is one more interesting project that we can do using this kit.

In this project we will be using an Arduino, LCD and temperature sensor. The Arduino will be reading the output of the sensor and printing the output on the LCD display. You can use the following diagram to understand the interfacing diagram.



fritzing

14.1. Room Temperature Indicator Using Arduino

In the above code, we have integrated the code of LCD module and DHT sensor. The Arduino is reading the output of the DHT sensor and printing that out on the LCD module.

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
// Set DHT pin:
#define DHTPIN 2
// Set DHT type, uncomment whatever type you're using!
#define DHTTYPE DHT11 // DHT 11
// #define DHTTYPE DHT22 // DHT 22 (AM2302)
// #define DHTTYPE DHT21 // DHT 21 (AM2301)
// Initialize DHT sensor for normal 16mhz Arduino:
DHT dht = DHT(DHTPIN, DHTTYPE);
void setup() {
  // Begin serial communication at a baud rate of 9600:
  Serial.begin(9600);
  // Setup sensor:
  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)
  // Read the humidity in %:
  float h = dht.readHumidity();
  // Read the temperature as Celsius:
  float t = dht.readTemperature();
  // Read the temperature as Fahrenheit:
  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("Failed to read from DHT sensor!");
    return;
  }
  // Compute heat index in Fahrenheit (default):
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius:
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" % ");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" \xC2\xB0");
  Serial.print("C | ");
  Serial.print(f);
  Serial.print(" \xC2\xB0");
  Serial.print("F ");
  Serial.print("Heat index: ");
  Serial.print(hic);
  Serial.print(" \xC2\xB0");
  Serial.print("C | ");
  Serial.print(hif);
  Serial.print(" \xC2\xB0");
  Serial.println("F");
}
```

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 !!!