Practical No. 01

Aim :

Study and Install IDE of Arduino and different types of Arduino.

Theory :

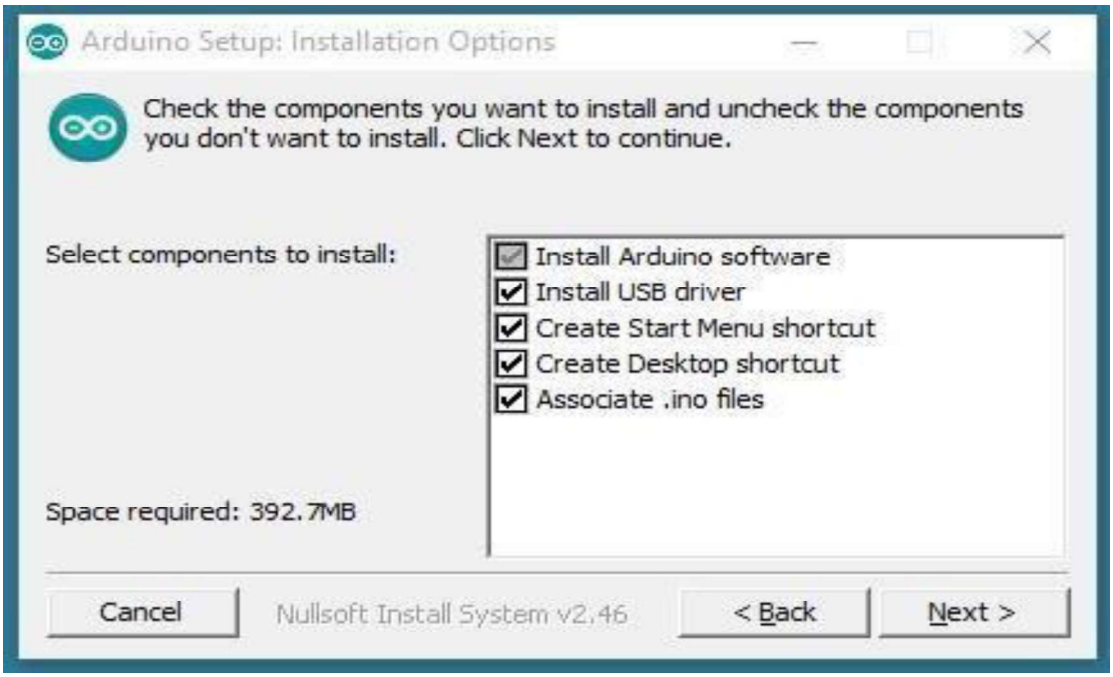
The Arduino Software (IDE) allows you to write programs and upload them to your board. In the Arduino Software Page you will find two options:

1. If you have a reliable Internet connection, you should use the online IDE (Arduino Web Editor). It will allow you to save your sketches in the cloud, having them available from any device and backed up. You will always have the most up-to-date version of the IDE without the need to install updates or community generated libraries.

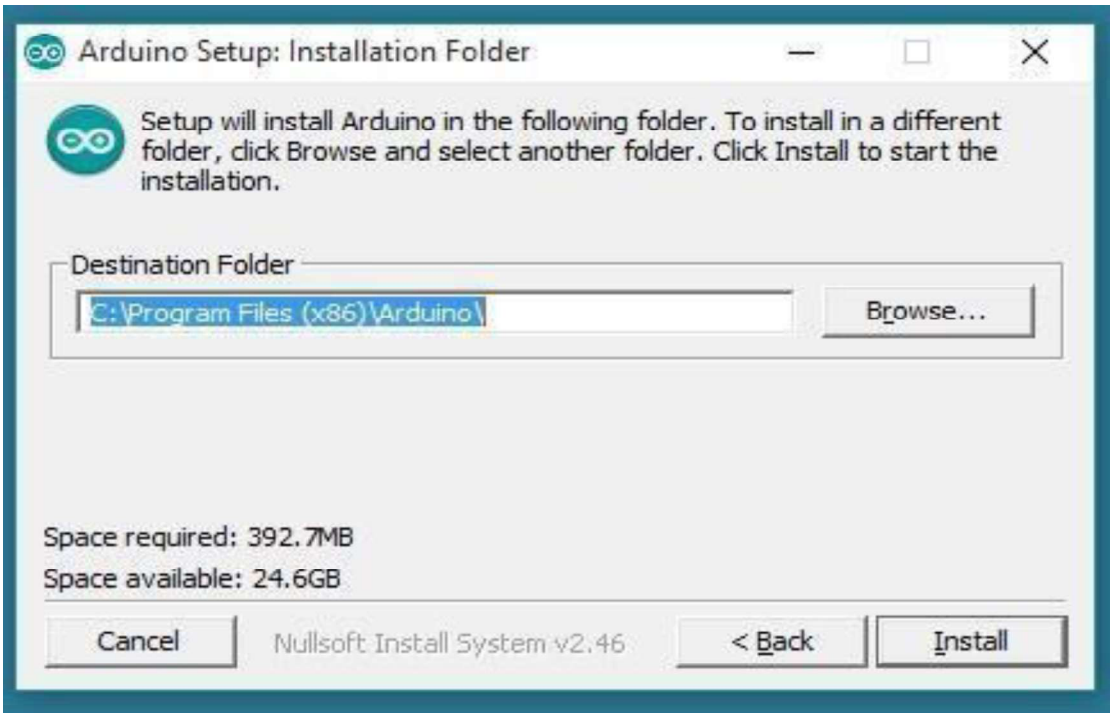
2. If you would rather work offline, you should use the latest version of the desktop IDE.

To install on windows, go to the official Arduino site and download the latest version.

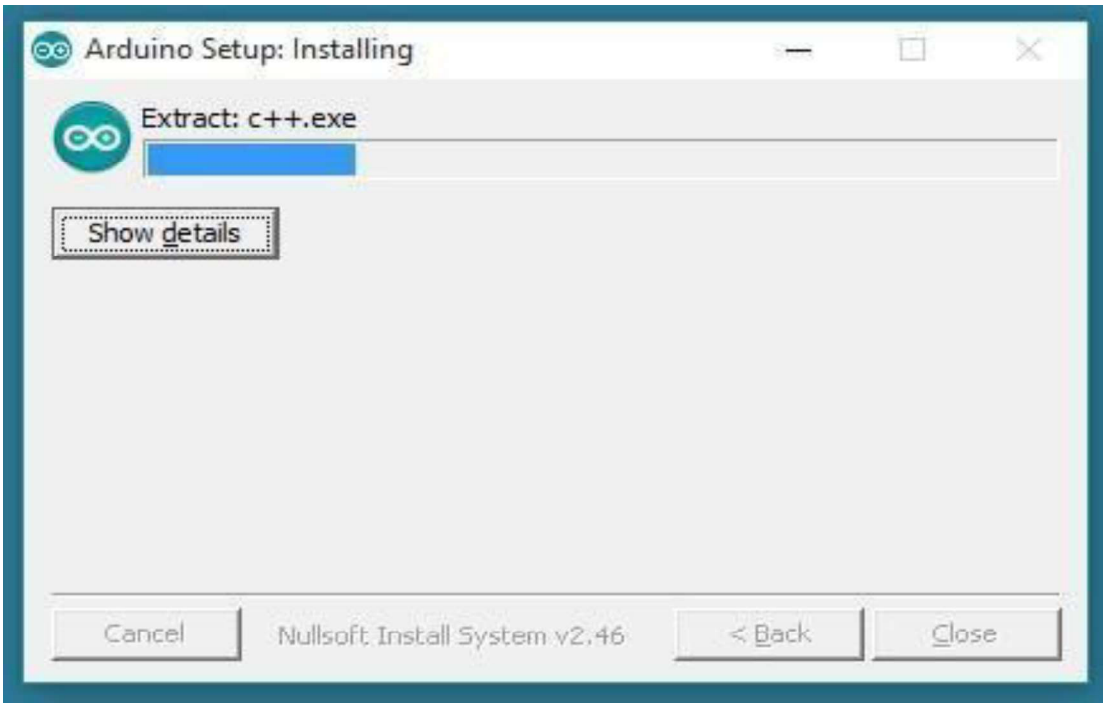
When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.



Choose the components to install –



Choose the installation directory –

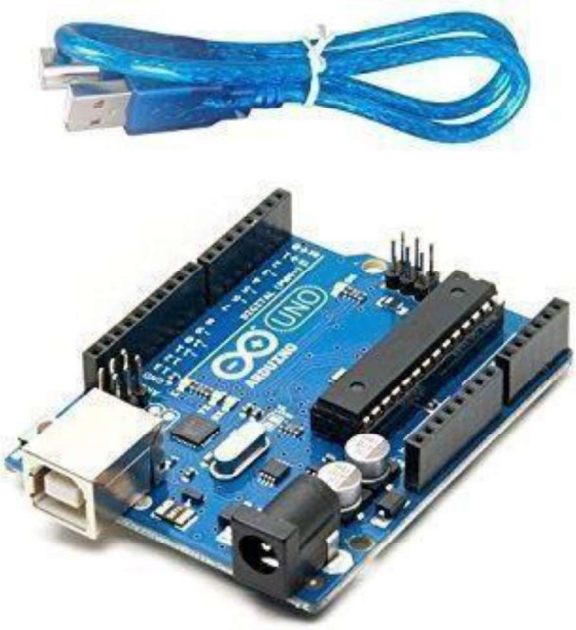


The process will extract and install all the required files to execute properly the Arduino Software (IDE).

Different Types of Arduino Boards

1)Arduino UNO (R3)

The Arduino UNO R3 is a new board and by comparing with the previous Arduino boards it has some additional features. The Arduino UNO uses the Atmega16U2 instead of 8U2 and it allows faster transfer rate & more memory. There is no need of extra devices for the Linux & Mac and the ability to have the UNO show up as a keyboard, mouse, joystick, etc.



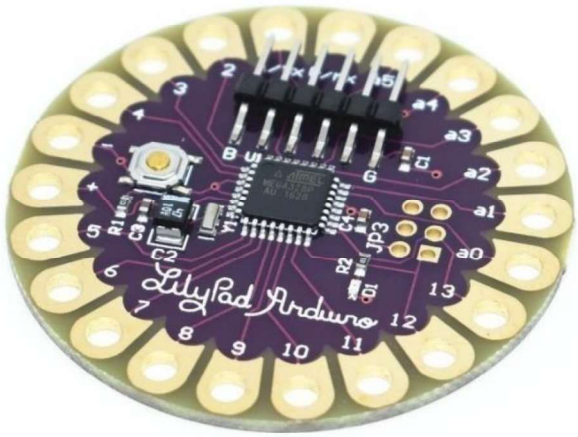
Arduino UNO

The Arduino R3 adds SDA & SCL pins which are next to the AREF and in addition, there are two pins which are placed near the RESET pin. The first pin is IOREF, it will allow the shields to adapt to the voltage from the board.

The other pin is not connected and it is reserved for the future purpose. The working of Arduino R3 is by all existing shields and it will adapt new shields which use these additional pins.

2)Lilypad Arduino

This board is an Arduino Programmable Microcontroller and it is designed to integrate easily into an e-textiles & wearable projects. The other Arduino boards have the same functionality like lightweight, round package designed to minimize snagging and profile, with wide tabs that can be sewn down and connected with conductive thread.



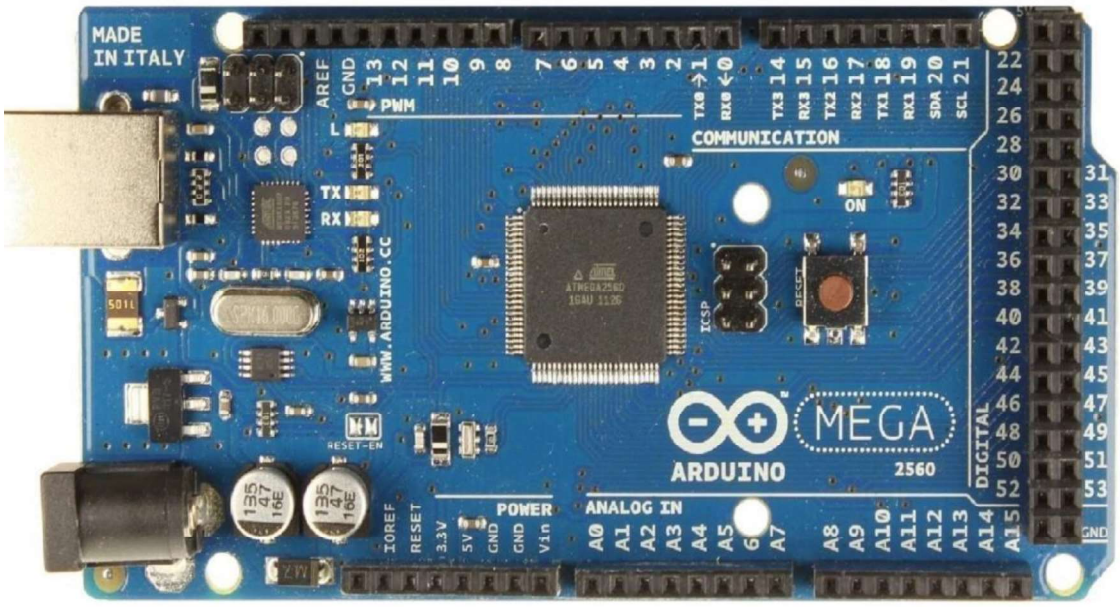
Lilypad Arduino Boards

This Arduino board consist of an Atmega 328 with the Arduino bootloader and to keep it as a small minimum external component are required. The power supply of this board is 2V to 5V and offers large pin-out holes that make it easy to sew and connect. Each pin is connected to positive and negative terminals and to control the input & output devices like light, motor, and switch.

This Arduino technology was designed and developed by Leah Buechley and each LilyPad was creatively designed to have large connecting pads to allow them to be sewn into clothing. There is an available of various input, output, and sensor boards and they are washable.

3)Arduino Mega (R3)

The Arduino Mega is a type of Microcontroller and it is based on the ATmega2560. It consists of 54 digital input/output pins and from the total pins 14 pins are used for the PWM output, 16 pins are used for the analog inputs, 4 pins are used for the hardware serial port of the UART. There are pins like crystal oscillator of 16 MHz, USB connection, RESET pin, ICSP header, and a power jack.



Arduino Mega R3

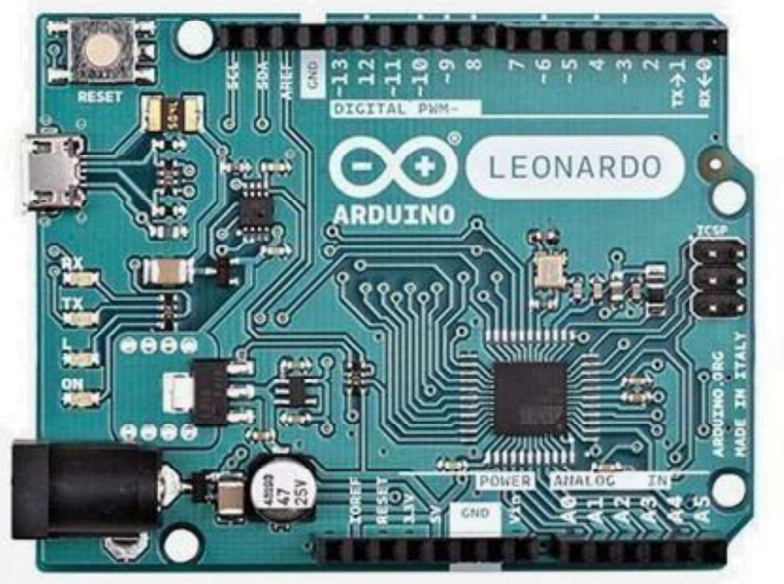
This Arduino Mega is also having SDA and SCL pins which are next to the AREF. There are two new pins near the RESET pin which are IOREF that allow the shields to adapt to the voltage provided by the board. The other is a not connected and is reserved for future purposes.

Features of the Arduino Mega (R3)

- ATmega2560 Microcontroller
- Input voltage – 7-12V
- 54 Digital I/O Pins (14 PWM outputs)
- 16 Analog Inputs
- 256k Flash Memory
- 16Mhz Clock Speed

4)Arduino Leonardo

The Leonardo Arduino board is a Microcontroller board and it is based on the ATmega32u4 data sheet. This Arduino board has 20 digital input/out pins and from the total number of pins, seven pins are used for the pulse width modulation output and 12 pins are used as an analog input and there are the 16MHz crystal oscillator, a micro USB connection, RESET pin and power jack.



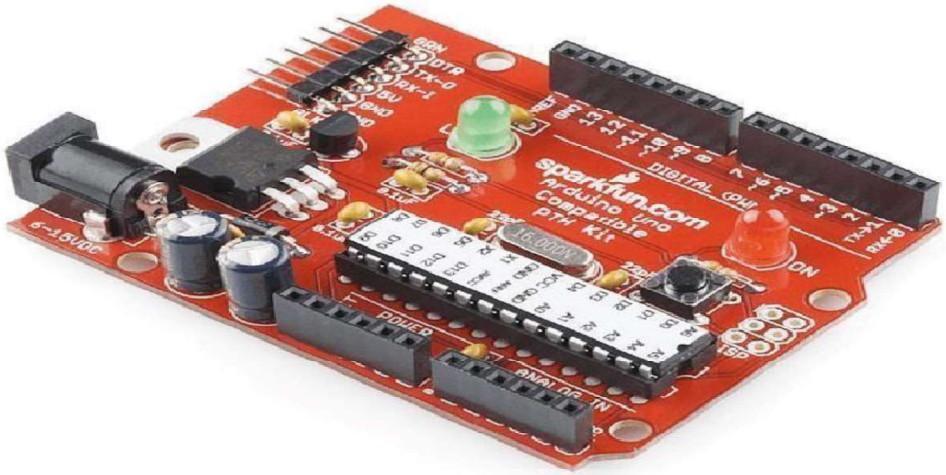
Arduino Leonardo

It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Leonardo differs from all preceding boards in that the ATmega32u4 has built-in USB communication, eliminating the need for a secondary processor.

This allows the Leonardo to appear to a connected computer as a mouse and keyboard, in addition to a virtual (CDC) serial COM port. It also has other implications for the behaviour of the board; these are detailed on the getting started page.

5)Arduino Red Board

The Arduino red board is programmed by using the USB cable of mini-B with the help of Arduino IDE software.



Arduino Red Board

Without any modifications in the security system there, it will work in Windows8 OS. The Arduino red board is more constant because USB and FTDI chips are used and they are flat on the back.

Creating it is very simple to utilize in the project design. Just plug the board, select the menu option to choose an Arduino UNO and you are ready to upload the program. You can control the Red Board over USB cable using the barrel jack.

•

Conclusion :

Hence , we studied installation of Arduino software and different types of Arduino.

Practical No. 02

Aim :

Write program using Arduino IDE for blink LED.

Theory :

LEDs are small, powerful lights that are used in many different applications. To start, we will work on blinking an LED, the Hello World of microcontrollers. It is as simple as turning a light on and off. Establishing this important baseline will give you a solid foundation as we work towards experiments that are more complex.

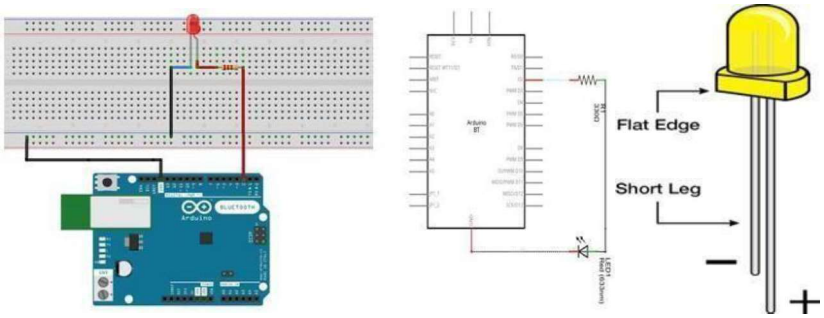
Components Required

You will need the following components −

- 1 × Breadboard
- 1 × Arduino Uno R3
- 1 × LED
- 1 × 330Ω Resistor
- 2 × Jumper

Procedure

Follow the circuit diagram and hook up the components on the breadboard as shown in the image given below.

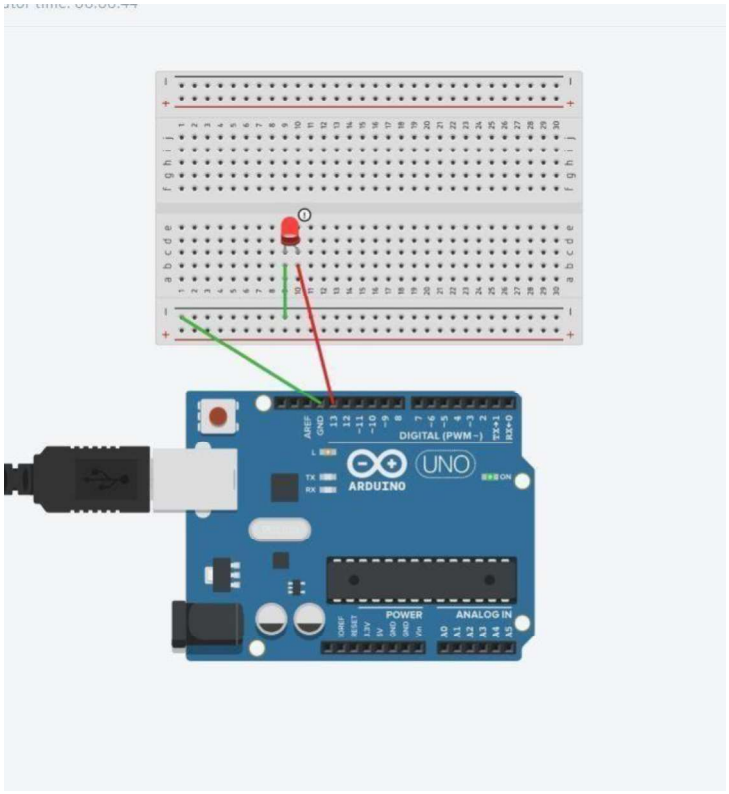


To find out the polarity of an LED, look at it closely. The shorter of the two legs, towards the flat edge of the bulb indicates the negative terminal.

Input :

```
void setup() {
   // initialize digital pin 13 as an output.
  pinMode(2, OUTPUT);
}

// the loop function runs over and over again forever

void loop() {
  digitalWrite(2, HIGH);
  // turn the LED on (HIGH is the voltage level)    delay(1000);
  // wait for a second    digitalWrite(2,
LOW);
  // turn the LED off by making the voltage LOW    delay(1000);
  // wait for a second
}
```

Output : -



Conclusion :

Hence , we executed program for blinking LED using Arduino IDE.

Practical No. 03

Aim :

Write program for RGB LED using Arduino.

Theory :

This project uses an Arduino and some LEDs to replicate a traffic light. It uses code as an internal timer and continues to run until you cut the Arduino's power supply. The components required are as follows:

1. Breadboard
2. Jumping Wires
3. Red ,Yellow and Green LED lights
4. Arduino USB 2.0 cable

Steps

1. Supply power to the breadboard

Insert one side of the jumper wire into GND on the board. Lead the other side to the breadboard. Put it on the far right column on the breadboard, at the top. This is the ground column. All the way to the right. Take a look at the picture and/or the schematics if you don't understand.

2. Adding the LEDs

Take out your LEDs and resistors. Place one end of the resistor in the column on the right, the same column we connected our jumper wire to. Extend the other end of the breadboard into the

main part of the breadboard. Attach the resistor to any row you like. Our LEDs will go on the same row. We will stick one end of the LED on one side of the breadboard, and the other end on the other side of the breadboard. The short end of the LED will go on the side your resistors are on, the right side.
Extend the other end of the LED to the right side of the breadboard.

3. Completing the circuit

Take another jumper wire, put it on the same row that you have an LED on. This is where the wires will go: Green LED: Port 2, Digital PWM section

Yellow LED, Port 3, Digital PWM section

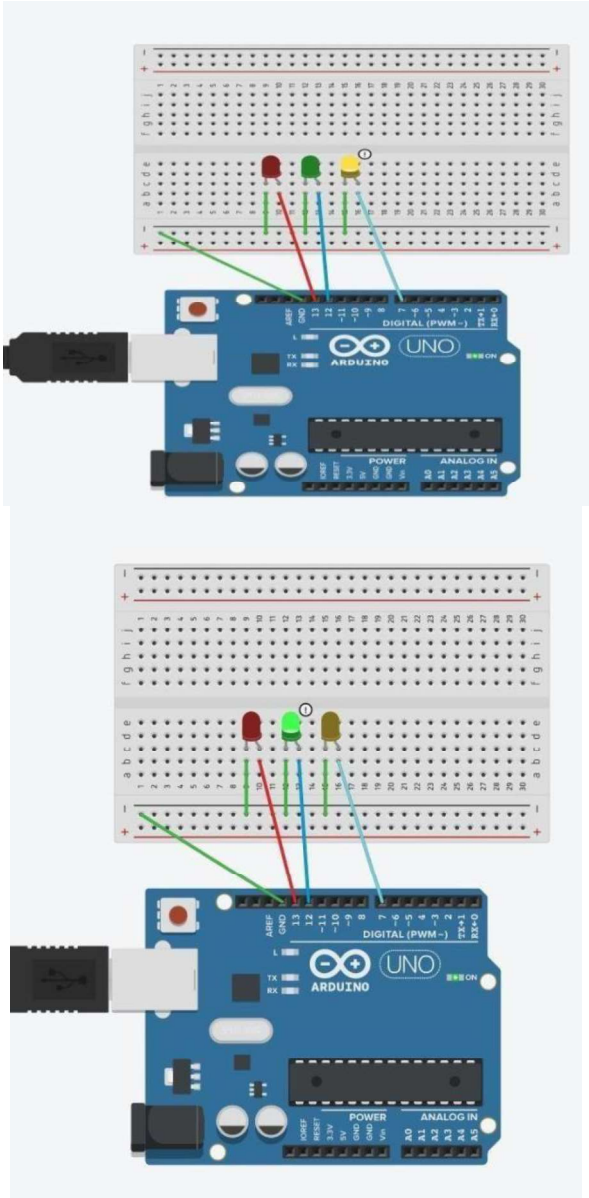
Red LED, Port 4, Digital PWM section

CODE :

```
// variables int
GREEN = 2; int
YELLOW = 3;  int
RED = 4; int
DELAY_GREEN = 5000; int
DELAY_YELLOW = 2000; int
DELAY_RED = 5000;


// basic functions
void setup()
{
  pinMode(GREEN, OUTPUT);   pinMode(YELLOW, OUTPUT);
  pinMode(RED, OUTPUT);
}

void loop()
{
  green_light();   delay(DELAY_GREEN);
```

```
yellow_light();   delay(DELAY_YELLOW);
red_light();
  delay(DELAY_RED);
}

void green_light()
{
  digitalWrite(GREEN, HIGH);    digitalWrite(YELLOW,
LOW);
  digitalWrite(RED, LOW);
}

void yellow_light()
{
  digitalWrite(GREEN, LOW);    digitalWrite(YELLOW,
HIGH);
  digitalWrite(RED, LOW);
}

void red_light()
{
  digitalWrite(GREEN, LOW);    digitalWrite(YELLOW,
LOW);
  digitalWrite(RED, HIGH);
}
```

Output:-



Conclusion: Hence , we executed program for RGB LED using Arduino.

Practical No. 04

Aim :

To study different types of sensors and implement ultrasonic sensor.

Theory :

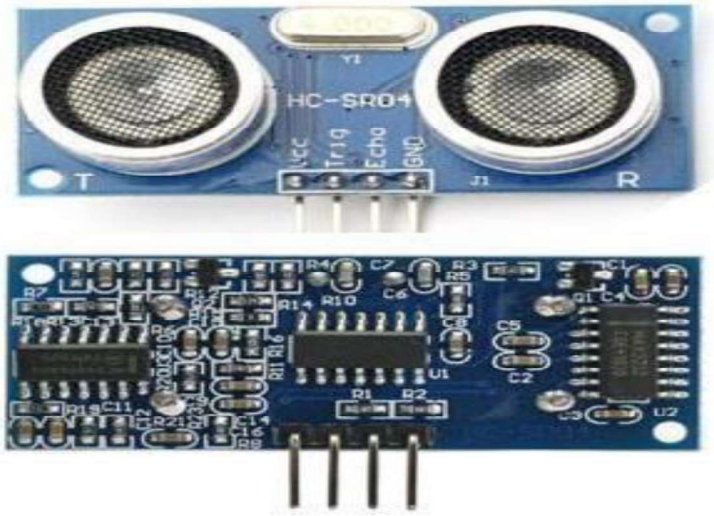
The sensors help Arduino to interact with the surroundings and for designing new applications.

Some of them are :

- Soil Moisture Sensor
- Microphone sensor
- Speed sensor
- Temperature and Humidity sensor
- IR Infrared Obstacle Avoidance sensor
- Humidity and Rain Detection sensor

Ultrasonic Sensor :

The HC-SR04 ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It offers excellent noncontact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1” to 13 feet.



Technical Specifications

- Power Supply − +5V DC
- Quiescent Current − <2mA
- Working Current − 15mA
- Effectual Angle − <15°
- Ranging Distance − 2cm – 400 cm/1″ – 13ft
- Resolution − 0.3 cm
- Measuring Angle − 30 degree

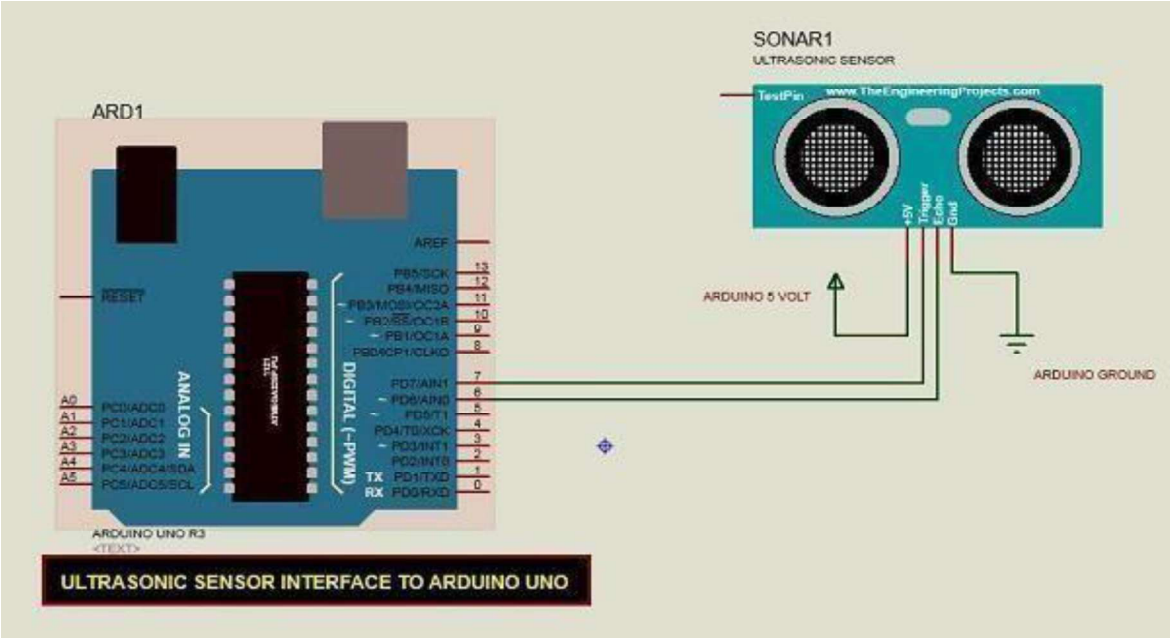
Components Required

You will need the following components −

- 1 × Breadboard
- 1 × Arduino Uno R3
- 1 × ULTRASONIC Sensor (HC-SR04)

Procedure

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

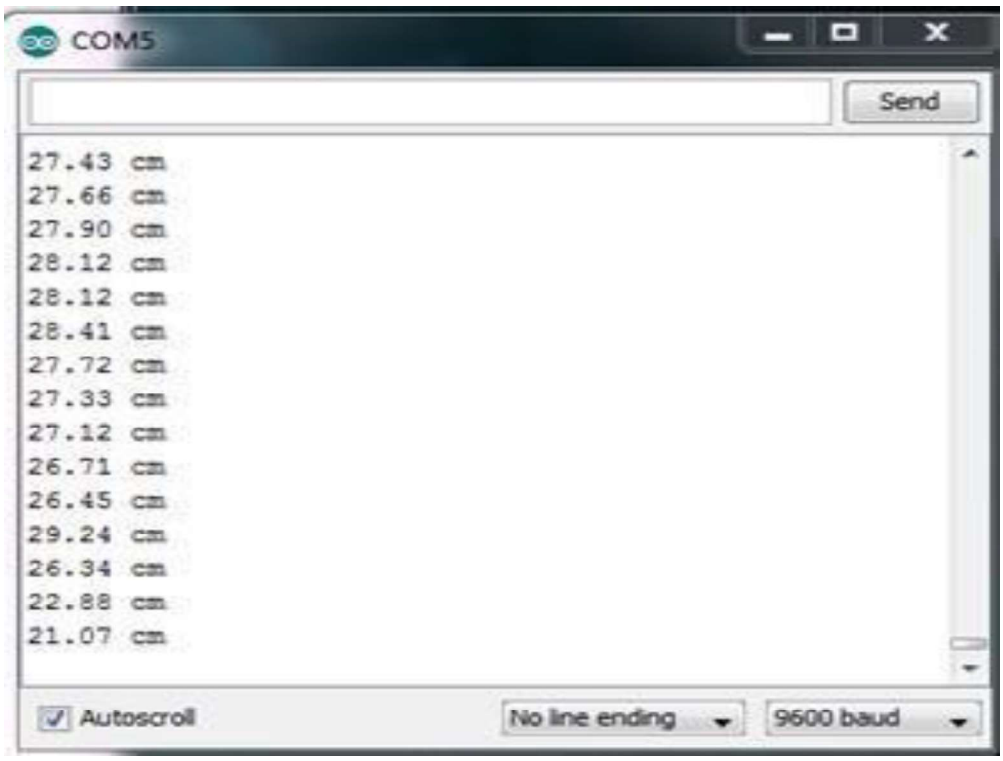


INPUT :

```
#define ECHOPIN 3       // Pin to receive echo pulse   #define
TRIGPIN 2       // Pin to send trigger pulse

void setup()
{
  Serial.begin(9600);   pinMode(ECHOPIN,
INPUT);    pinMode(TRIGPIN, OUTPUT);
}

void loop()
{
  // Start Ranging -Generating a trigger of 10us burst
digitalWrite(TRIGPIN, LOW);
delayMicroseconds(2);   digitalWrite(TRIGPIN, HIGH);
delayMicroseconds(10);   digitalWrite(TRIGPIN,
LOW);

  // Distance Calculation
```

```
  float distance = pulseIn(ECHOPIN, HIGH);   distance=
distance/58;

  Serial.print(distance);

Serial.println(" cm");

delay(200);

}
```

OUTPUT:



Conclusion :

Hence , we studied different types of sensor and performed ultrasonic sensor in Arduino.

Practical No. 05

Aim :

To study temperature sensor and write program for measuring temperature using arduino.

Theory :

The Temperature Sensor LM35 series are precision integratedcircuit temperature devices with an output voltage linearly proportional to the Centigrade temperature.

The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range.

Technical Specifications

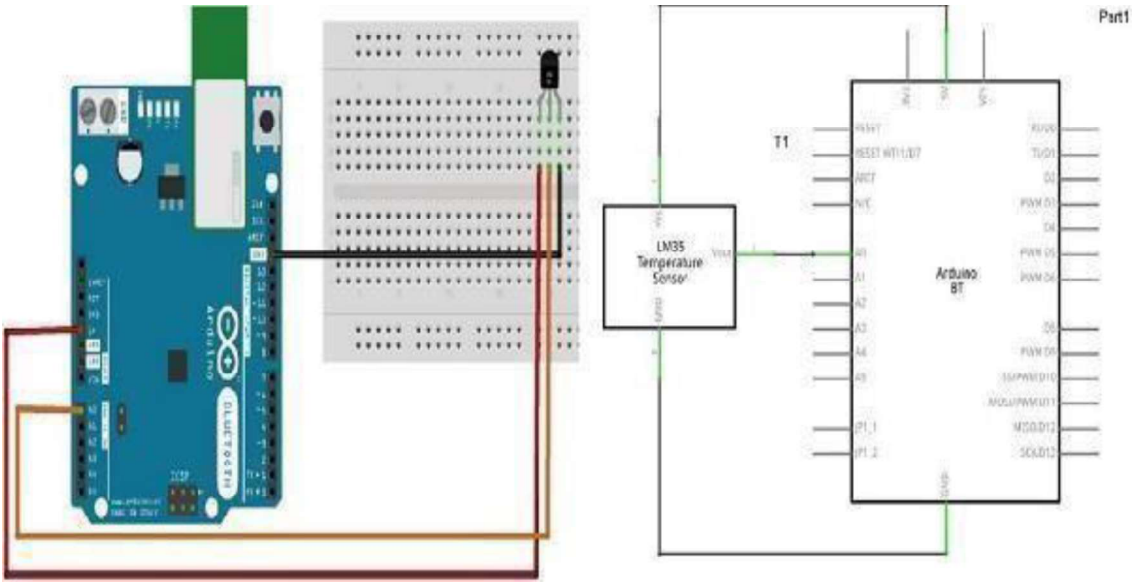
- Calibrated directly in Celsius (Centigrade)
- Linear + 10-mV/°C scale factor
- 0.5°C ensured accuracy (at 25°C)
- Rated for full −55°C to 150°C range
- Suitable for remote applications Components Required

You will need the following components −

- 1 × Breadboard

- 1 × Arduino Uno R3
- 1 × LM35 sensor Procedure

Follow the circuit diagram and hook up the components on the breadboard as shown in the image given below.

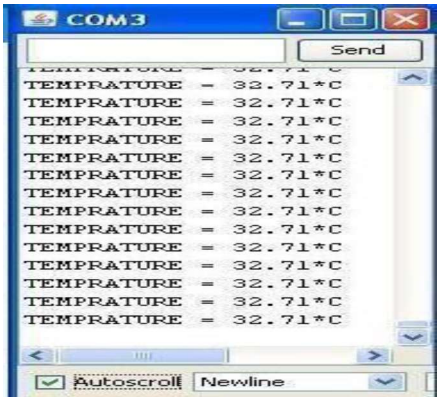


INPUT :

```
int val; int
tempPin = 1;

void setup()
{
Serial.begin(9600);
}
void loop()
{
val = analogRead(tempPin); float  mv
= ( val/1024.0)*5000; float cel
= mv/10;
float farh = (cel*9)/5 + 32;

Serial.print("TEMPRATURE = ");
```

```
Serial.print(cel);
Serial.print("*C");
Serial.println(); delay(1000);  }
```

OUTPUT :



Conclusion:- Hence we have successfully studied temperature sensor and executed program for measuring temperature using arduino.

PRACTICAL NO. 06

Aim :

Study different shield of Arduino and implement WiFi shield using Arduino.

Theory :

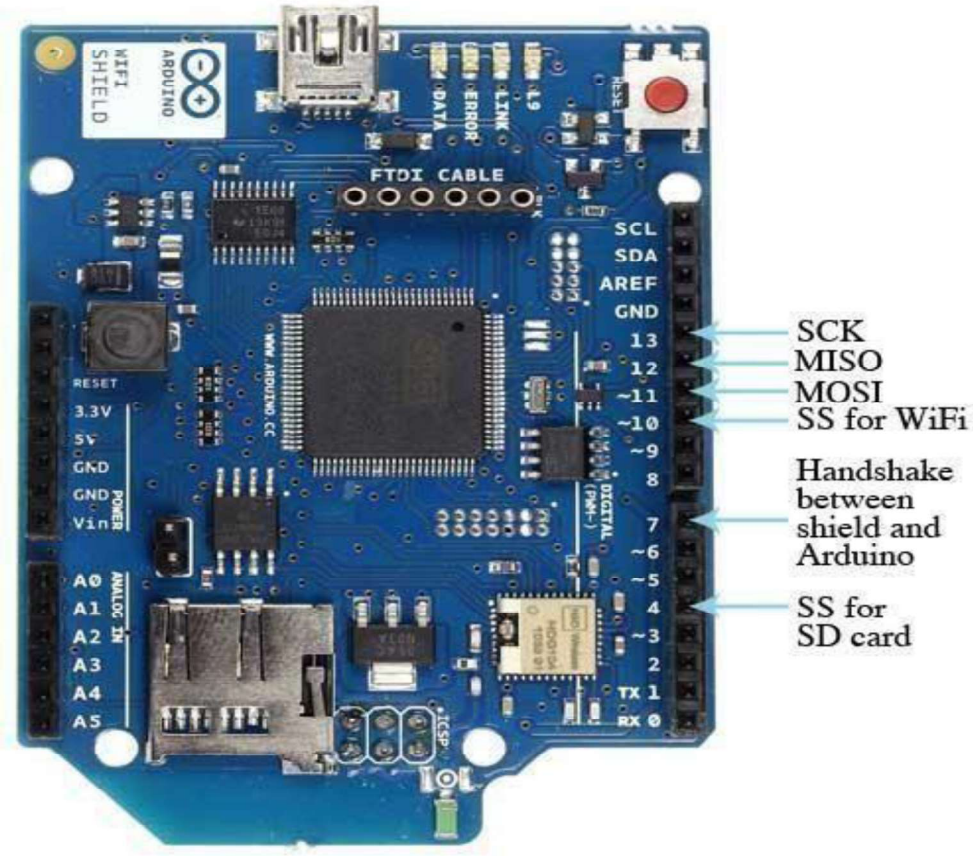
Different shields of Arduino are :

- Ethernet shield
- Relay shield
- Motor shield
- LCD shield
- Capacitive Touchpad shield
- Smoke detector shield
- RFID/NFC shield
- Camera shield and many more.

WiFi shield using Arduino :

The Arduino WiFi Shield allows an Arduino board to connect to the internet using the 802.11 wireless specification (WiFi). It is based on the HDG204 Wireless LAN 802.11b/g System in-Package. An AT32UC3 provides a network (IP) stack capable of both TCP and UDP. Use the WiFI library to write sketches which connect to the internet using the shield. The WiFI shield connects to an Arduino board using long wire-wrap headers which extend through the shield. This keeps the pin layout intact and allows another shield to be stacked on top.

The WiFi Shield can connect to wireless networks which operate according to the 802.11b and 802.11g specifications.
There is an onboard micro-SD card slot, which can be used to store files for serving over the network. It is compatible with the Arduino Uno and Mega. The onboard microSD card reader is accessible through the SD Library. When working with this library, SS is on Pin 4. Arduino communicates with both the Wifi shield's processor and SD card using the SPI bus (through the ICSP header). This is on digital pins 11, 12, and 13 on the Uno and pins 50, 51, and 52 on the Mega. On both boards, pin 10 is used to select the HDG204 and pin 4 for the SD card. These pins cannot be used for general I/O. On the Mega, the hardware SS pin, 53, is not used to select either the HDG204 or the SD card, but it must be kept as an output or the SPI interface won't work.
Digital pin 7 is used as a handshake pin between the WiFi shield and the Arduino, and should not be used.



There is an onboard Mini-USB connector. This is not for programming an attached Arduino, it is for updating the AT32UC3 using the Atmel DFU protocol. The programming jumper adjacent to the power bus and analog inputs should be left unconnected for typical use. It is only used for DFU programming mode.

On board indicators

The shield contains a number of informational LEDs:

- L9 (yellow) : this is tied to digital pin 9
- LINK (green) : indicates a connection to a network
- ERROR (red) : indicates when there is a communication error
- DATA (blue) : indicates data being transmitted/received

INPUT :

```
#include<WiFi.h>
char ssid[] = " HOME_WIFI ";     //  your network SSID (name)
char pass[] = "12345678";    // your network password   int
status = WL_IDLE_STATUS;     // the Wifi radio's status          void
setup() {
// initialize serial:
Serial.begin(9600);
// attempt to connect using WPA2 encryption:
Serial.println("Attempting to connect to WPA network...");
status = WiFi.begin(ssid, pass);   // if you're not connected,
stop here:
if ( status != WL_CONNECTED) {
Serial.println("Couldn't get a wifi connection");      while(true);
}
// if you are connected, print out info about the connection:
```

```
else {
Serial.println("Connected to network");
}
}
```

Output: - Our Arduino board can access internet with the help of wifi or in this case “HOME_WIFI” and hence we can successfully access the Arduino from over the internet also

Conclusion :

Hence , we implemented WiFi shield using Arduino.

PRACTICAL NO. 07

Aim :

Study and implement RFID/NFC using Arduino.

Theory :

Near field communication are protocols that electronic devices use to communicate and transfer data between each other. Near field communication devices have to be very near to each other, usually between 10cm, but the range can vary depending on the device that is transmitting and the size of the tag. NFC tags require no power input whatsoever. They use magnetic induction between two between two small loop antennas. The tags these days carry between 96 and 4,096 bytes of information.

Parts List :

- Arduino Uno R3
- Adafruit PN532 RFID/NFC Shield
- Arduino IDE (Integrated Development Environment)
- Rewritable NFC Tags

To test whether what we wrote on the tags was successful, we can test with the Arduino or with an NFC-enabled phone. Most smartphones running Android should be able to read NFC tags, and I will be testing with a Nexus 5. Unfortunately for iPhone users, the only iPhones that supports NFC are the iPhone 6 and the 6s, but they do not support NFC tag reading so just use the

Arduino to test out what your tag has written on them. iPhones only use their NFC capability for apple pay, therefore you cannot use them to read tags or anything else.

RFID :

An RFID reader is used to read RFID tags (which contain certain unique data stored in a chip). An RFID reader and an RFID tag, both have a coil surrounding them. When an RFID tag is shown near an RFID Reader, it collects the unique tag data (a combination of digits and characters) from the RFID tag.

How to Interface RFID Reader to Arduino

Lets first wire the whole thing up. You may observe the circuit diagram given below. Take note of the following stuffs.

Note 1:- Power supply requirement of RFID Readers vary from product to product. The RFID reader I used in this tutorial is a 12 Volts one. There are 5 Volts and 9 Volts versions available in the market.

Note 2:- You may ensure the RFID Reader and RFID Tags are frequency compatible. Generally they are supposed to be 125Khz. You may ensure this before purchasing them.

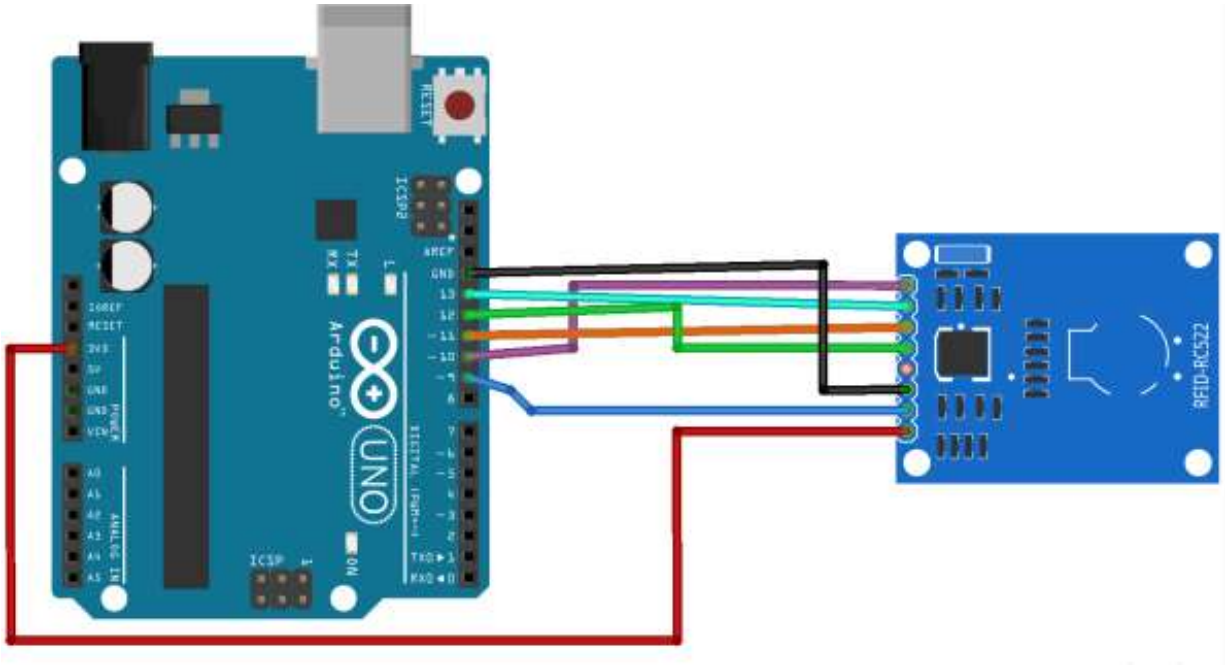
Note 3:- There are two possible outputs from an RFID Reader. One is RS232 compatible output and other one is TTL compatible output. A TTL compatible output pin can be connected directly to Arduino. Whereas an RS232 compatible output must be converted to TTL using an RS232 to TTL converter (You can design this yourself using MAX232 IC)

INPUT :

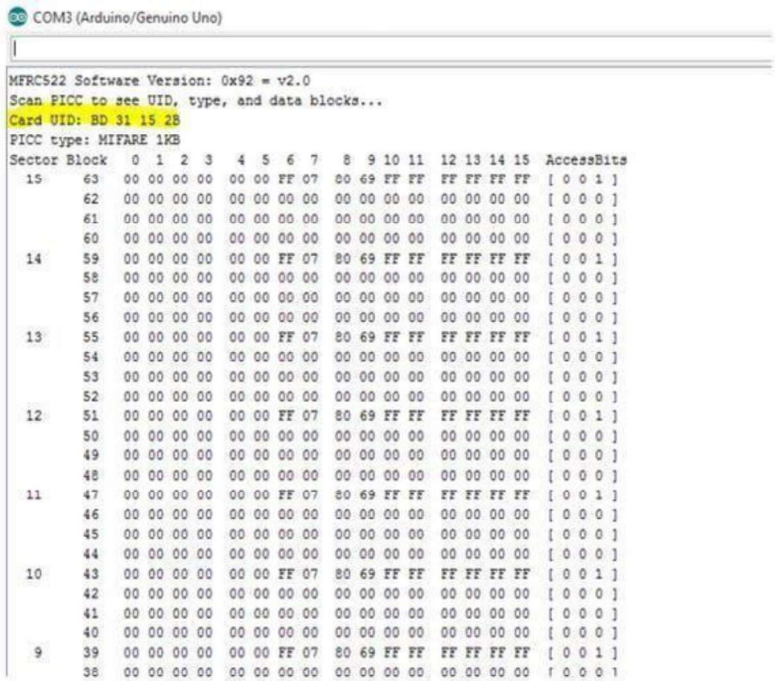
```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance.

void setup()
{
 Serial.begin(9600);   // Initiate a serial communication
SPI.begin();      // Initiate  SPI bus   mfrc522.PCD_Init();   //
Initiate MFRC522
 Serial.println("Approximate your card to the reader...");
 Serial.println();
 }
void loop()
{
 // Look for new cards
 if ( ! mfrc522.PICC_IsNewCardPresent())
 {
  return;
 }
 // Select one of the cards
 if ( ! mfrc522.PICC_ReadCardSerial())
 {
  return;
 }
 //Show UID on serial monitor
 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));
 }
```

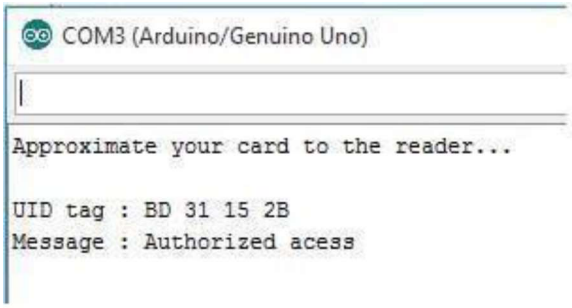
```
 Serial.println();
 Serial.print("Message : ");   content.toUpperCase();
 if (content.substring(1) == "BD 31 15 2B") //change here the UID of the card/cards  that
you want to give access
 {
   Serial.println("Authorized access");
Serial.println();     delay(3000);
 }

 else   {
   Serial.println(" Access denied");     delay(3000);
 }
}
```

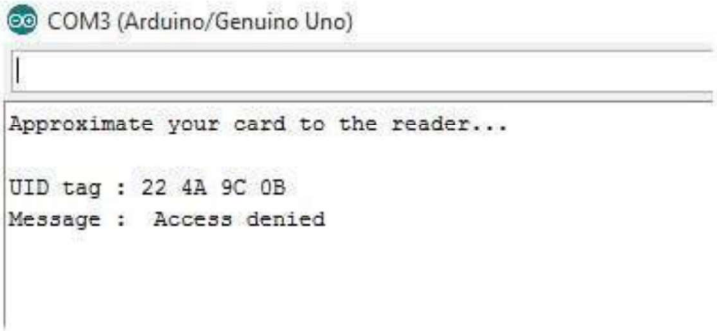
Circuit : -



Output :







-

Conclusion :

Hence we studied NFC/RFID in arduino.

PRACTICAL NO.08

Aim :

To study and implement MQTT protocol using Arduino.

Theory :

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol that was developed by IBM and first released in 1999. It uses the pub/sub pattern and translates messages between devices, servers, and applications.

MQTT architecture

The connected devices in the MQTT protocol are known as “clients,” which communicate with a server referred to as the “broker.” The broker handles the task of data transmission between clients.

Whenever a client (known as the “publisher”) wants to distribute information, it will publish to a particular topic, the broker then sends this information to any clients that have subscribed to that topic (known as “subscribers”).

The publisher does not need any data on the number or the locations of subscribers. In turn, subscribers do not need any data about the publisher. Any client can be a publisher, subscriber, or both. The clients are typically not aware of each other, only of the broker that serves as the intermediary. This setup is popularly known as the “pub/sub model.”

MQTT messages

When a client wants to send data to the broker, this is known as a “publish.” When a client wants to receive data from the broker, it will “subscribe” to a topic or topics. When a client subscribes to a

certain topic, it will receive all messages published on that topic going forward.
Along with the message itself, the publisher also sends a QoS (Quality of Service) level. This level defines the guarantee of delivery for the message. These QoS levels are as follows:

- **At most once:** When the message is published, the broker will only receive the message "at most once." This level should not be used for mission-critical information since it carries the risk that the subscribers will not receive the message.
- **At least once:** The publisher continues to resend the message until it receives an acknowledgment from the broker regarding the particular message. In other words, it's more important that the message is received than it is to ensure it is only received once. This is probably the most commonly used QoS level.
- **Exactly once:** The publisher and broker work together to ensure the broker will receive and act on a message exactly once. This requires some additional overhead in the form of a four-part handshake. Although this is the safest QoS level, it is also the slowest and therefore only used when necessary.

The Benefits of MQTT

The benefits of MQTT include:

- **Lightweight code footprint:** Devices need only a few lines of code in order to get up and running with the MQTT protocol.
- **Minimized data packets:** MQTT is very energy-efficient. This is great if a device is battery-powered or has little CPU power.
- **Speed:** MQTT operates in real time, with no delays outside of QoS.
- **Ease of implementation:** MQTT already has libraries in programming languages such as Elixir and Python.

- **Last will and testament:** If a client unexpectedly disconnects, you can set message instructions to be sent to all subscribers in order to remedy the situation.
- **Retained messages:** Each topic can have one retained message that a client automatically receives when it subscribes (like a pinned post on social media).

INPUT :

```
#include <SPI.h>
#include <Ethernet.h>
#include <PubSubClient.h> //
Function prototypes  void subscribeReceive(char* topic, byte* payload,
unsigned int length);

// Set your MAC address and IP address here byte mac[]
= { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };  IPAddress
ip(192, 168, 1, 160);

// Make sure to leave out the http and slashes!  const char*
server = "test.mosquitto.org";

// Ethernet and MQTT related objects
EthernetClient ethClient;
PubSubClient mqttClient(ethClient); void setup()
{
  // Useful for debugging purposes
Serial.begin(9600);

  //       Start       the       ethernet       connection
Ethernet.begin(mac, ip);

  // Ethernet takes some time to boot!
  delay(3000);
  //    Set    the    MQTT       server       to     the    server
        stated       above      ^   mqttClient.setServer(server, 1883);
```

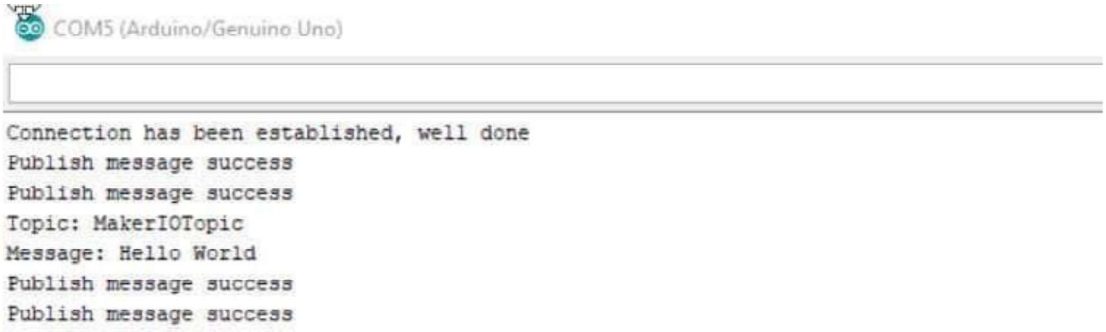
```
  // Attempt to connect to the server with the ID "myClientID"   if
(mqttClient.connect("myClientID"))
```

```
 {
  Serial.println("Connection has been established, well done");

  // Establish the subscribe event
  mqttClient.setCallback(subscribeReceive);
 }
 else
 {
  Serial.println("Looks like the server connection failed...");
 }
}
                                                    void loop()
{
 // This is needed at the top of the loop!
 mqttClient.loop();

 // Ensure that we are subscribed to the topic "MakerIOTopic"
mqttClient.subscribe("MakerIOTopic");

 // Attempt to publish a value to the topic "MakerIOTopic"
if(mqttClient.publish("MakerIOTopic", "Hello World"))
 {
  Serial.println("Publish message success");
 }
 else
 {
  Serial.println("Could not send message :(");
 }

 // Dont overload the server!    delay(4000);
}
void subscribeReceive(char* topic, byte* payload, unsigned int length)
{
 // Print the topic
 Serial.print("Topic: ");
 Serial.println(topic);
```

```
// Print the message
                                                    Serial.print("Message: ");
```

```
 for(int i = 0; i < length; i ++)
 {
  Serial.print(char(payload[i]));
 }

 // Print a newline   Serial.println("");
}
```

Output : -



Conclusion :

Hence, we studied and implemented MQTT protocol using Arduino.

PRACTICAL NO.09

Aim :

Study and configure Raspberry Pi.

Theory :

The Raspberry Pi is a credit-card sized computer designed and manufactured by the Raspberry Pi Foundation, a non-profit organization dedicated to making computers and programming instruction as accessible as possible to the widest number of people.

It included a CPU, GPU, audio/video processing, and other functionality all on a low-power chip paired with a 700Mhz single core ARM processor. Over the intervening years the foundation has released multiple revisions (switching out the Broadcom chips for improved versions and upping the CPU power with a 1.2GHz quadcore chip).

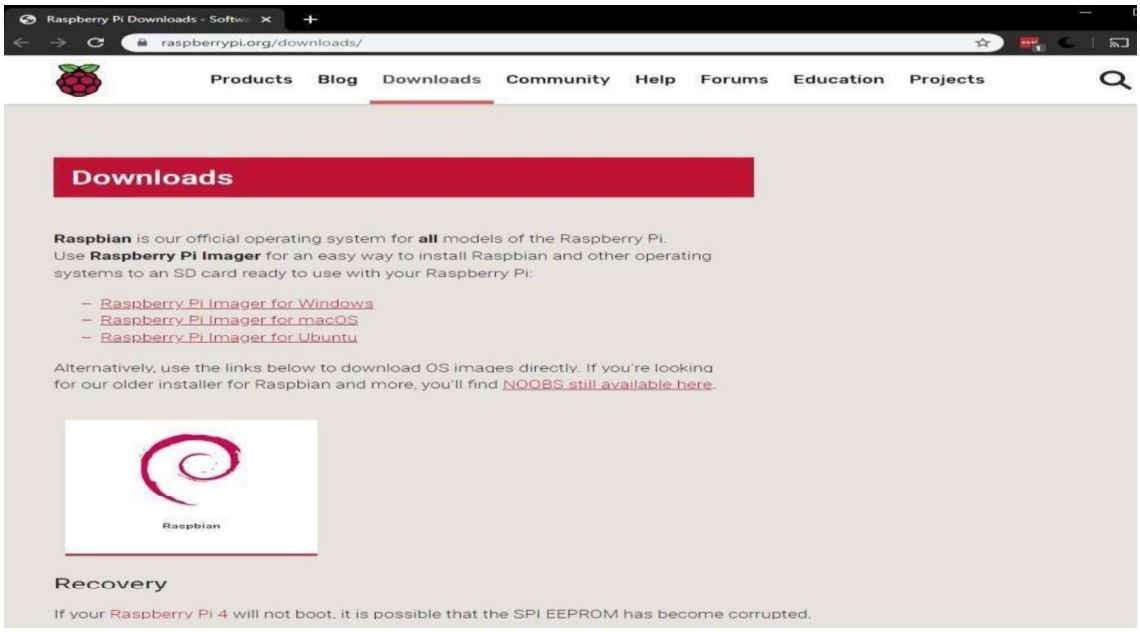
The current generation Raspberry Pi 3, seen above, supports the following hardware:

- 1.2 Ghz ARM processor Systems-On-a-Chip (SoC) with integrated 1GB RAM.
- 1 HDMI port for digital audio/video output
- 1 3.5mm jack that offers both audio and composite video out (when paired with an appropriate cable).
- 4 USB 2.0 ports for connecting input devices and peripheral addons.
- 1 microSD card reader for loading the operating system.

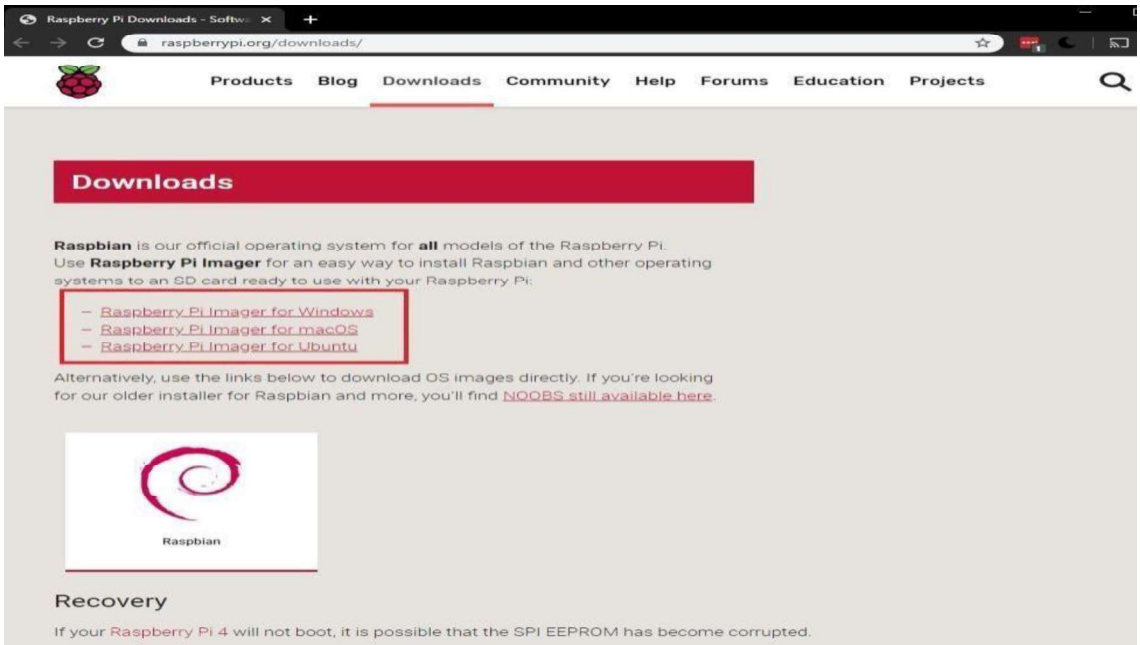
- 1 Ethernet LAN port.
- 1 Integrated Wi-Fi/Bluetooth radio antenna.
- 1 microUSB power port.
- 1 GPIO (General Purpose Input/Output) interface.

Configuring Raspberry Pi :

Download and launch the Raspberry Pi Imager.



Click on the link for the Raspberry Pi Imager that matches your operating system.

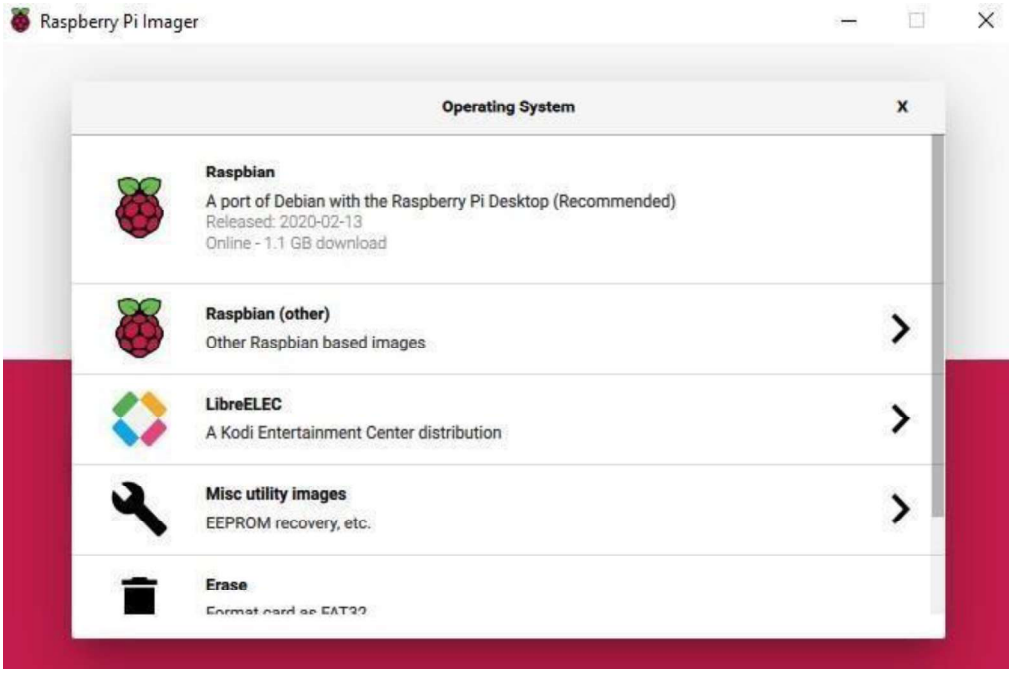


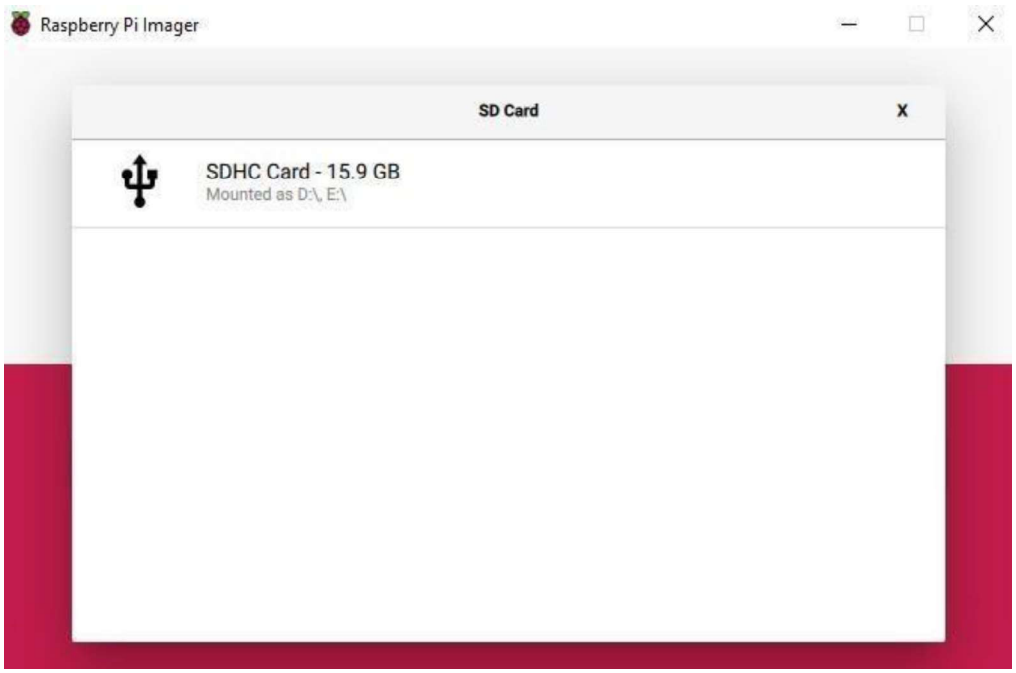
Using the Raspberry Pi Imager

Anything that's stored on the SD card will be overwritten during formatting. So if the SD card on which you want to install Raspbian currently has any files on it, e.g. from an older version of Raspbian, you may wish to back these files up first to not lose them permanently.

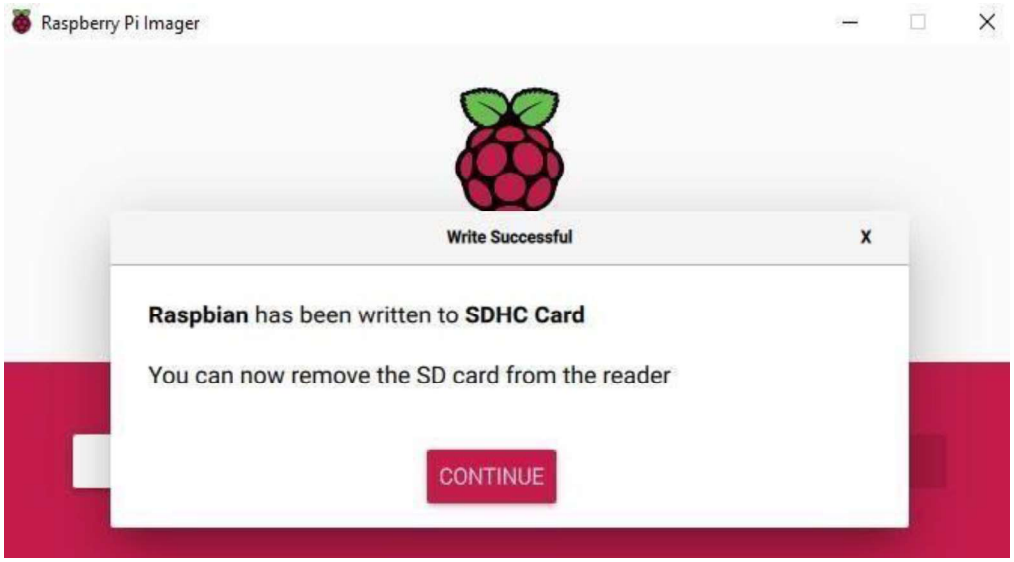
When you launch the installer, your operating system may try to block you from running it. For example on Windows I get the following:

- If you get this, click on [redacted] and then [redacted].
- Follow the instructions to install and run the Raspberry Pi Imager.
- Insert your SD card into the computer or laptop's SD card slot.
- In the Raspberry Pi Imager, select the OS that you want to install and the SD card you would like to install it on.

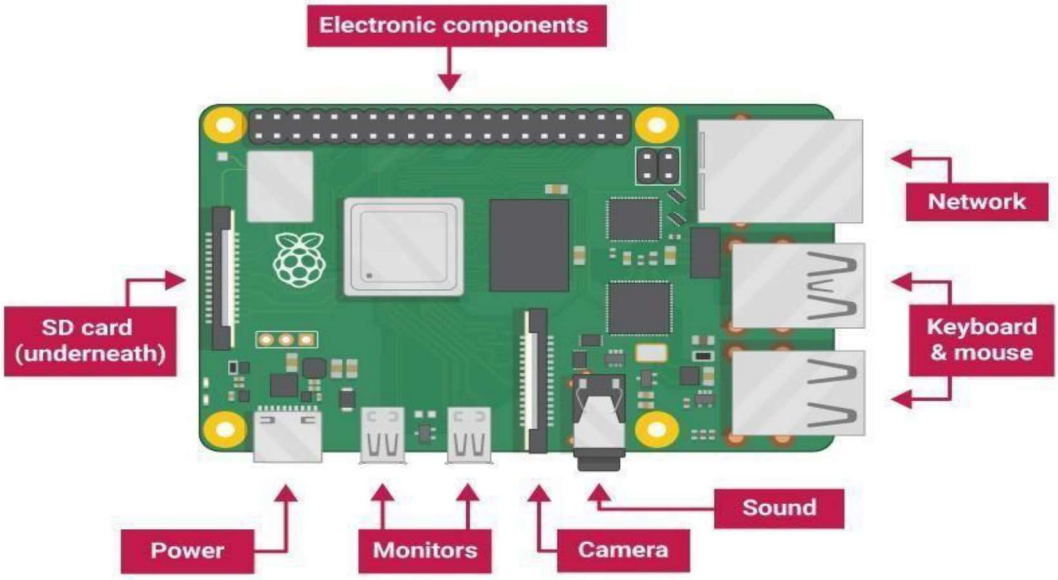




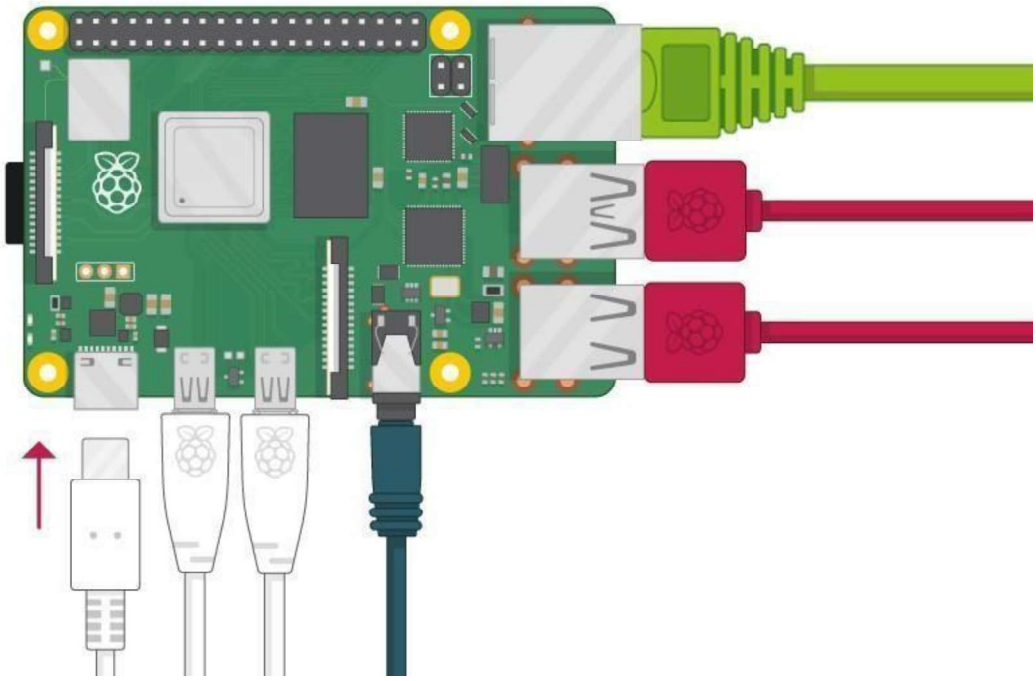
- Then simply click the button.
- Wait for the Raspberry Pi Imager to finsh writing.
- Once you get the following message, you can eject your SD card.



Now get everything connected to your Raspberry Pi. It's important to do this in the right order, so that all your components are safe.



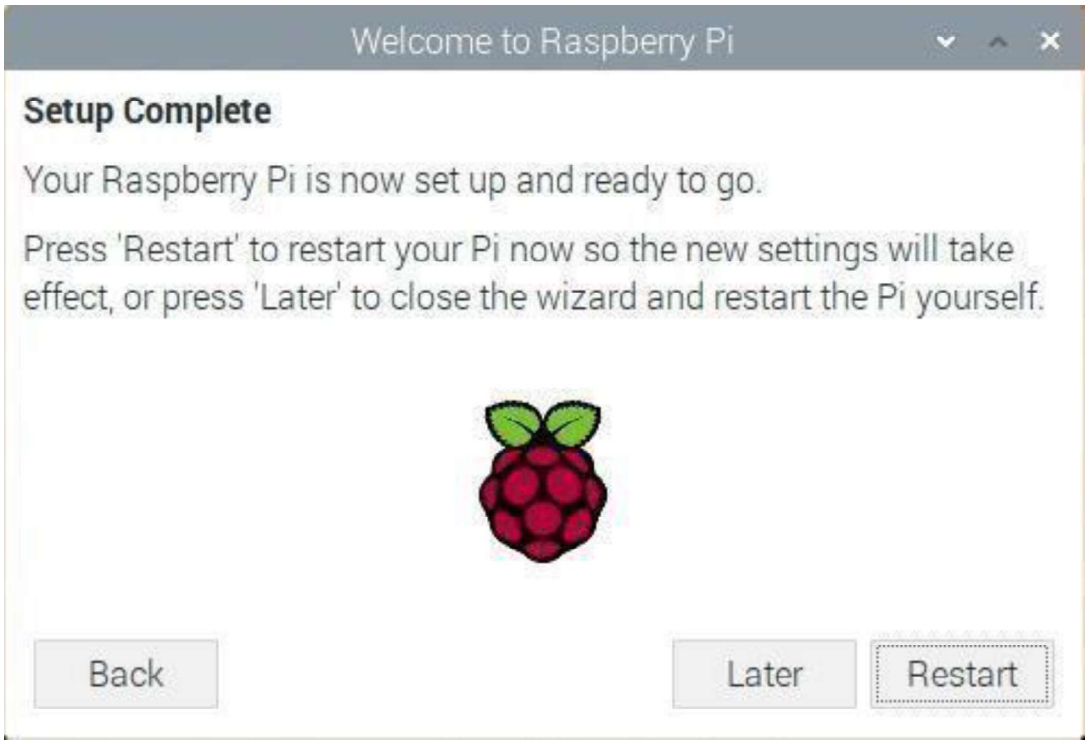
Your Raspberry Pi doesn't have a power switch: as soon as you connect it to a power outlet, it will turn on. ◦ Plug the USB power supply into a socket and connect it to your Raspberry Pi's power port.



You should see a red LED light up on the Raspberry Pi, which indicates that Raspberry Pi is connected to power. As it starts up (this is also called booting), you will see raspberries appear in the top lefthand corner of your screen.

After a few seconds the Raspbian Desktop will appear.

When you start your Raspberry Pi for the first time, the Welcome to Raspberry Pi application will pop up and guide you through the initial setup. ○ Click Next to start the setup. ○ Set your Country, Language, and Timezone, then click Next again. ○ Enter a new password for your Raspberry Pi and click Next. ○ Connect to your WiFi network by selecting its name, entering the password, and clicking Next. ○ Click Next let the wizard check for updates to Raspbian and install them (this might take a little while). ○ Click Done or Reboot to finish the setup.



Conclusion :

Hence, we studied and configured Raspberry Pi.

PRACTICAL NO. 10

Aim :

Write a program for LED blink using Raspberry Pi.

Theory :

Components required

- One led
- 100 ohm resistor
- Jumper cables

Raspberry Pi GPIO

Specifications • Output

Voltage : 3.3V

- Maximum Output Current : 16mA per pin with total current from all pins not exceeding 50mA

For controlling a Led using Raspberry Pi, both python and the GPIO library is needed.

Installing Python GPIO Library

Note: Python and GPIO library are preinstalled if you are using Raspbian.

- Make sure your Rasperry Pi is connected to the internet using either a LAN cable or a WiFi adapter.

- Open the terminal by double clicking the LXTerminal icon
- Type the following command to download the GPIO library as a tarball

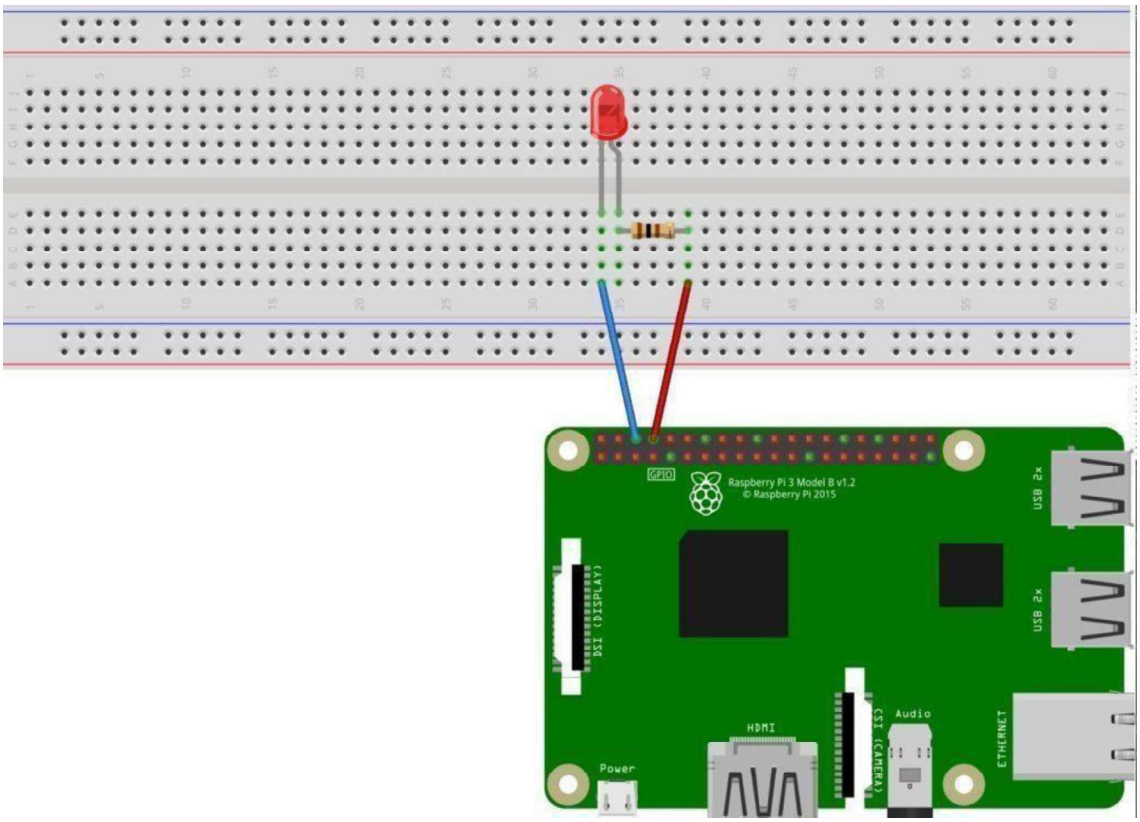
wget http://raspberry-gpio-python.googlecode.com/files/RPi.GPIO-0.4.1a.tar.gz

- Unzip the tarball tar zxvf RPi.GPIO-0.4.1a.tar.gz
- Change the directory to unzipped location cd RPi.GPIO-0.4.1a
- Install the GPIO library in python

sudo python setup.py install

Circuit Diagram

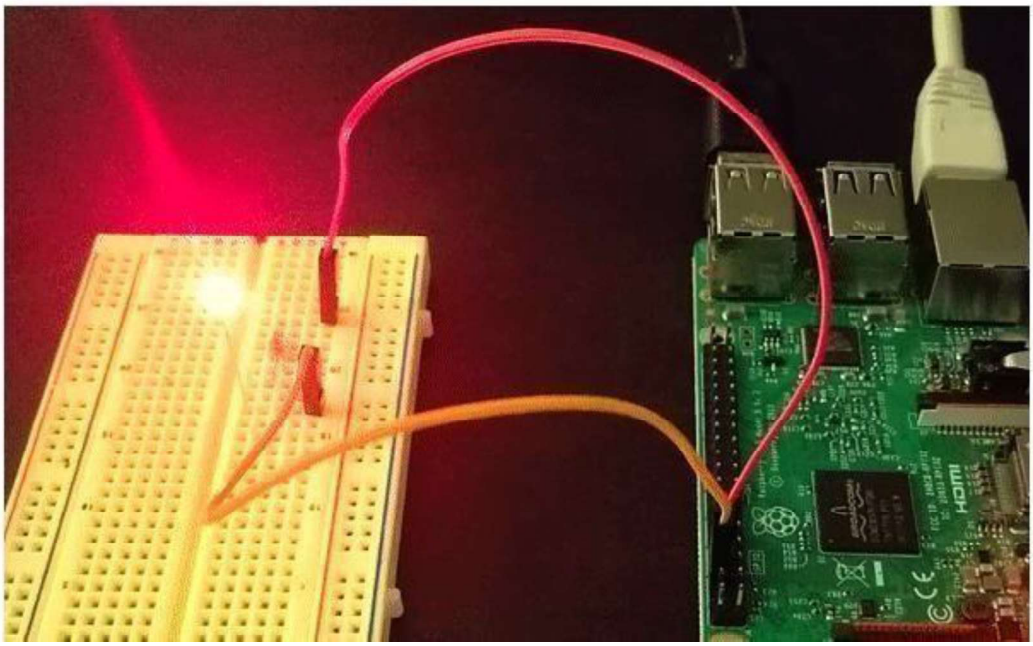
- Connect the Led to 6 (ground) and 11 (gpio) with a 100Ω resistor in series.



INPUT :

```
import time import RPi.GPIO as GPIO      ## Import
GPIO library

GPIO.setmode(GPIO.BOARD)      ## Use board pin numbering

GPIO.setup(11, GPIO.OUT)      ## Setup GPIO Pin 11 to OUT while

True:

  GPIO.output(11,True)  ## Turn on Led   time.sleep(1)

## Wait for one second

        GPIO.output(11,False) ## Turn off Led

time.sleep(1)
```

Output: -



Conclusion :

Hence , we implemented blinking LED using Raspberry Pi.