

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590018



A
Course Practical Report

Submitted in partial fulfillment for the course

18CS81-Internet Of Things
Submitted by

MOHAMMED JABIR
1VJ19CS034

Department of Computer Science & Engineering

2022 – 2023



Department of Computer Science and Engineering
VIJAYA VITTALA INSTITUTE OF TECHNOLOGY

Hennur – Bagalur Road, Doddagubbi post
Bengaluru -560077

VIJAYA VITTALA INSTITUTE OF TECHNOLOGY

Hennur – Bagalur Road, Doddagubbi post, Bengaluru -560 077

Department of Computer Science and Engineering



CERTIFICATE

Certified that the course practical carried out by **Mohammed Jabir (1VJ19CS034)** is a bonafide student of Department of Computer Science and Engineering **Vijaya Vittala Institute of Technology**, in partial fulfilment for the course **18CS81 – Internet of Things** during the year 2022–23.

Marks Obtained: _____/20

Course Faculty

HOD

VIJAYA VITTALA INSTITUTE OF TECHNOLOGY

Hennur – Bagalur Road, Doddagubbi post, Bengaluru -560 077



INDEX

Sl.No	Date	Name of Experiment	Marks Obtained	Faculty Sign
1.		Transmit A String using UART		
2.		Point to Point Communication of 2 nodes over radio frequency		
3.		Multi-Point to Single Point Communication of nodes over radio frequency. LAN(Subnetting)		
4.		I2C Protocol Study		
5.		Reading Temperature And Relative Humidity Value from the Sensor		
6.		Motion Detector		

Total =-----/20

HARDWARE LIST

SL.NO	COMPONENTS
1	Arduino UNO Boards
2	RF Transmitters/Receivers
3	DHT11 Temperature & Humidity Sensors
4	LCD Display
5	LED Lights
6	Bread Board
7	Connecting Wires

Experiment No 1: Transmit a string using UART.

Aim:

A UART's (Universal Asynchronous Receiver/Transmitter) main purpose is to transmit and receive serial String data using serial communication.

Objective:

To learn how serial communication is achieved using UART serial string data).

Components Required:

- 1.Arduino UNO
- 2.Aduino UNO USB cable.

Connections:

Arduino UNO 'to be connect with computer through USB cable.

Procedure:

1. Open the Arduino UNO IDE and create and save a sketch source code as follows.
2. Compile sketch.
3. Upload sketch on to Arduino UNO.

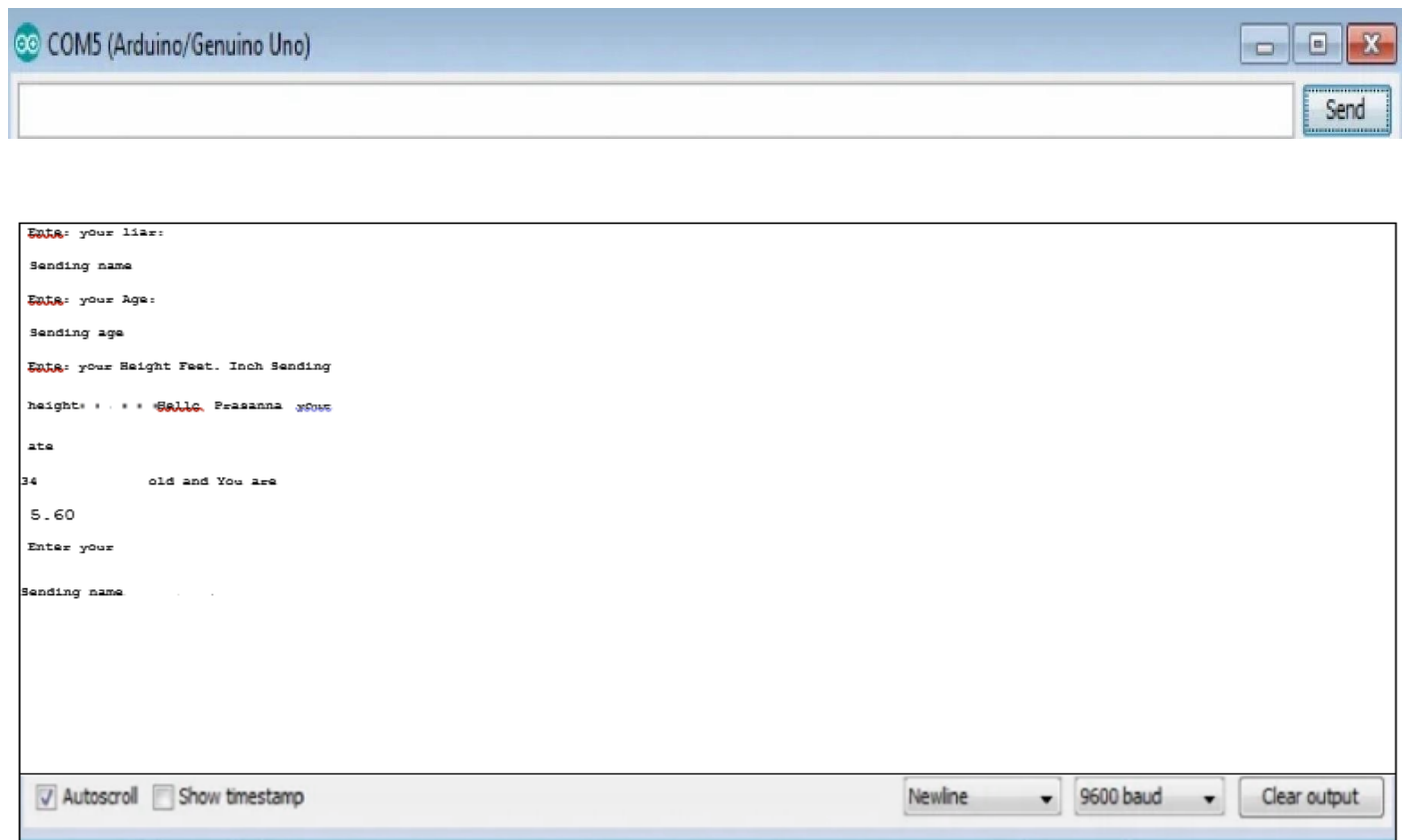
Source Code:

```
String name; int age; float height;
void setup() {
// put your setup code here, 'to run once: Serial.begin(9600);
void loop() {
// put your main code here, to run repeatedly: Serial. println ("\nEnter your Name: I' ) ;
while(Serial.available()==0)
name=Serial.readString(); Serial.println("Sending name. delay(10000);
Serial. println ("Enter your Age: while(Serialavailable()==0)
age=Serial.parseInt(); Serial.printlnC'Sending age delay(10000);
Serial. your Height Feetilnch :"); while(Serialavailable()==0)
height=Serial.parseFloat(); Serial.println("Sending height delay(5000); display(); void display()
Serial.
Serial. print(name);
Serial. println(" your are
Serial. print(age)
Serial. print(" years old and");
Serial. println(" You are ");
```

```
Serial. print(height);
```

```
Serial. print(t' tall");
```

OUTPUT:



Applications:

UARTs are used for devices including GPS units, modems, wireless communication and Bluetooth modules, amongst many other applications.

Conclusion:

Serial data communication is achieved by transmitting and receiving string data using UART.

Experiment No 2. Point-to-Point communication of two Motes over the radio frequency.

Aim:

To communicate two Motes node over 'the radio frequency.

Objective:

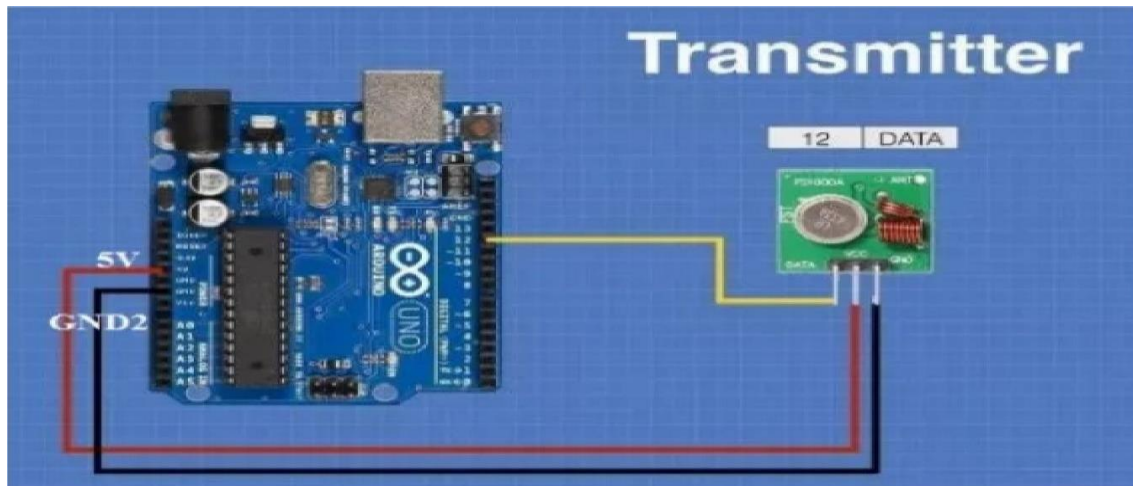
To learn how wireless communication is achieved via point to point communication of 'two motes using RF Transmitter/ Receiver.

Components Required:

1. Arduino UNO's (Two).
2. RF transmitter 433MHz.
3. RF receiver 433MHz
4. Connecting wires

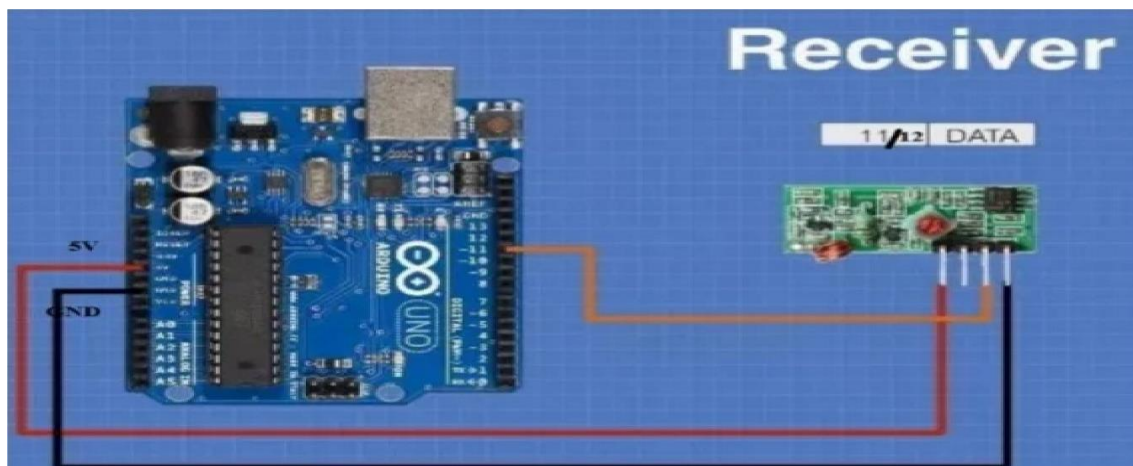
Arduino Sender Connections:

Arduino Pins	RF Transmitter
12	Data
	VCC
GND2	GND



Arduino Receiver Connections:

Arduino Pins	RF Reciever
11	Data
	VCC
GND2	GND



Procedure:

1. Open the Arduino UNO IDE and create and save a sketch source code of 'transmitter and receiver.
2. Compile the sketches.
3. Upload sketches on to Arduino UNffs through connected COM ports.

Source Code of Receiver:

```
#include <RH ASK.h>
```

```
#include <SPI.h> // Not actually used but needed 'to compile
```

```

RH ASK driver; void setup()

Serial. begin(9600); // Debugging only if (! driver.init()) failed");

void loop()

uint8_t bun 12]; uint8_t buflen = sizeof(buf); if (driver.recv(bufj &buflen)) // Non-blocking
int i;

// Message with a good checksum received, dump it.

Serial. print("Message:

Serial. println((char*)buf);

```

Source Code of Transmitter:

```

#include <RH ASK.h>

#include <SPI.h> // Not actually used but needed to compile

RH ASK driver; void setup()

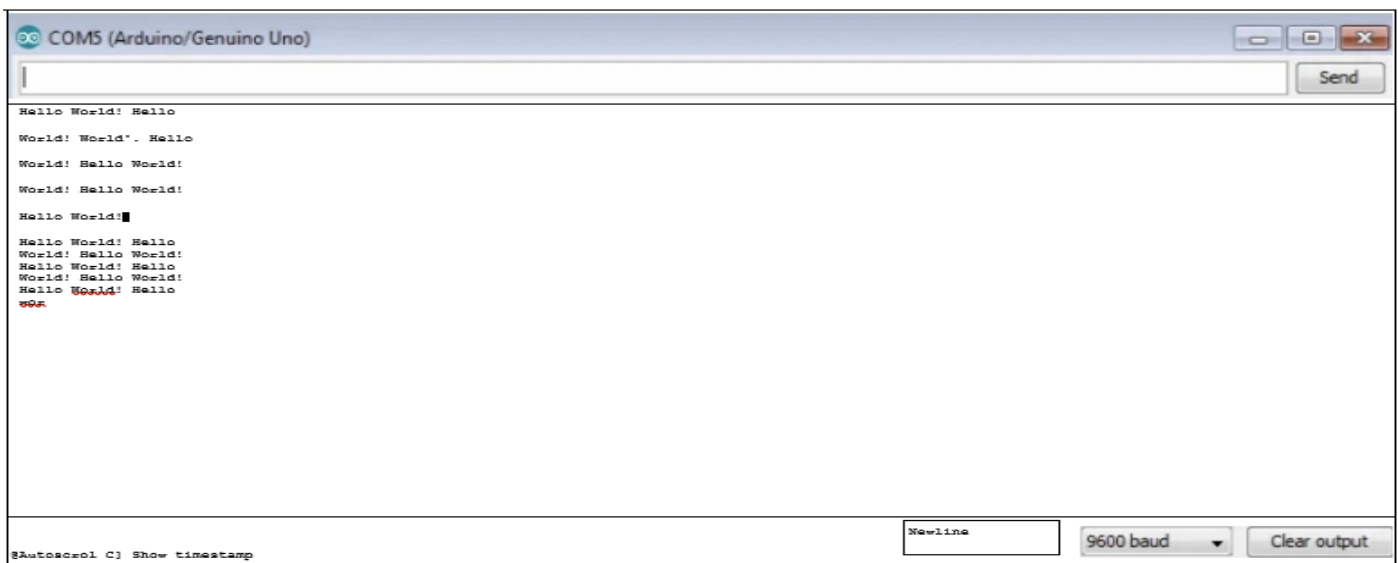
Serial. begin(9600); // Debugging only if (!driver.init()) failed");

void loop()

const char *msg = "Hello World!"; driver.send ((uint8_t *) msg, strlen(msg)); driver.waitPacketSent();
delay(1000);

```

OUTPUT:



Applications:

RF technology, 'the information can be transmitted through 'the air without requiring any cable or wires or other electronic conductors, by using electromagnetic waves like IR, RF, satellite, etc. a variety of wireless communication devices and technologies ranging from smart phones to computers, tabs, laptops, Bluetooth Technology, printers.

Conclusion: Wireless communication is achieved by RF transmitter and RF receiver message passed in the frequency range of 433MHZ

Experiment No 3. Multi-point to single point communication of Motes over the radio frequency. LAN (Subnetting).

Aim:

To study and analysis of the Wireless Communication of Motes over the radio frequency 'through multi-point 'to single point in a network.

Objective:

Point-to-multipoint (PMP) communication refers to communication that is accomplished through a distinct and specific form of one-to-many vice versa connections, offering several paths from one single 'location to various locations. Single transmitter broad casts message across the multiple receiver using wireless communication of motes through RF transmitter and receiver 433 MHZ in a network. Wireless communication is achieved.

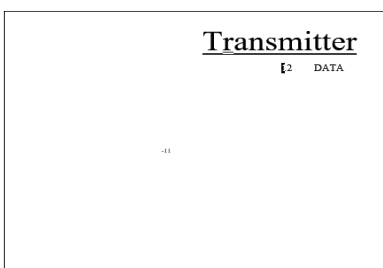
Components Required:

<u>Sl.No</u>	Components Required	QTY
1	Arduino UNO Transmitter	1
2	Arduino UNO Receiver 1	1
3	Arduino UNO Receiver 2	1
4	<u>Ad</u> duino UNO USB cables	3
5	RF Receiver 433 MHZ	2
6	RF Transmitter 433 MHZ	1
7	Connecting Wires	9

Arduino Sender Connections:

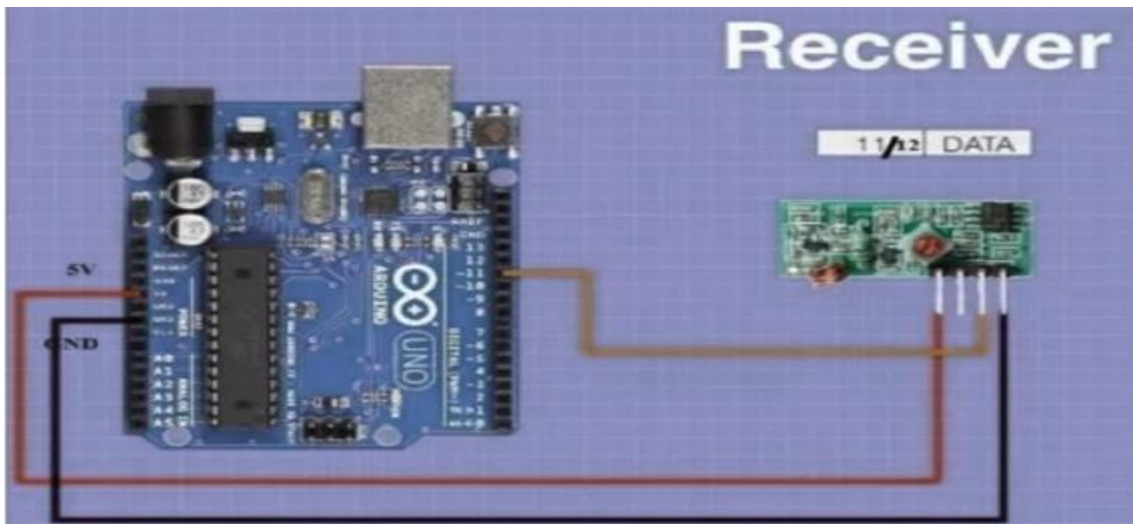
Arduino Pins	RF Transmitter
12	Data
	<u>VCC</u>
GND2	GND

Arduino Receiver 1 & 2 Connections:



Arduino Pins RF Reciever

Arduino Pins	RF <u>Reciever</u>
11	Data
	<u>VCC</u>
GND2	GND



Procedure:

1. Open the Arduino UNO IDE and create and save a sketch of Master 'transmitter and receiver 1 and receiver 2 source code.
2. Compile sketch of transmitter and receivers.
3. Upload sketch through COM COM5).
4. See the output message on to the Serial monitor by using COM ports.

Source Code Sender:

```
#include <RH ASK.h>

#include // Not actually used but needed to compile RH ASK driver; void setup()
Serial. begin(9600); // Debugging only if (! driver.init()) failed");
void loop()
const char *msg = "Welcome to 10T Laboratory"; driver.send ((uint8_t *) msg, strlen(msg));
driver.waitPacketSent();
Serial. : Sending Data....");
delay(1000);
```

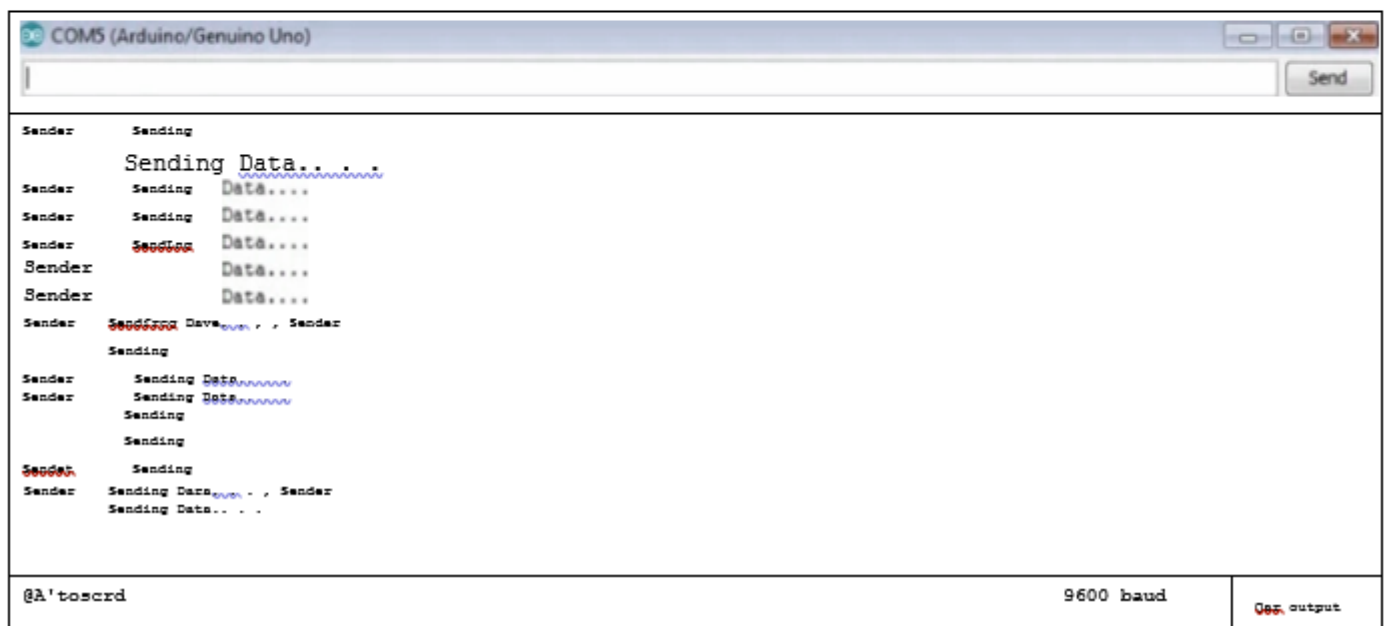
Source Code Multiple Receivers:

```
#include ASK.h>
#include <SPI.h> // Not actually used but needed to compile RH ASK driver; void setup()
Serial.begin(9600); // Debugging only if (! driver.init()) failed");
void loop()
uint8_t 'bun2 0]; uint8_t buflen = sizeof(buf); if (driver.recv(buf, &buflen)) // Non-blocking
int i;
// Message with a good checksum received, dump it.
Serial.print("Message:
Serial. println((char*)buf);
```

RECEIVER OUTPUT:



SENDER OUTPUT:



Applications:

PMP wireless networks are employed in distribution amenities, huge corporate campuses, school districts, public safety applications, etc.

Conclusion:

Wireless Network communication is achieved through multi point to single point communication nodes over radio frequency by RF transmitter and receiver. Multiple receiver and a transmitter communicate each other by broadcasting message in a network via RF transmitter and receiver nodes in the frequency 433 MHz range.

Experiment No 4. 12C protocol study

Aim:

The study of I2C protocol to communicate (in turn) with many devices, or other boards, sensors networks.

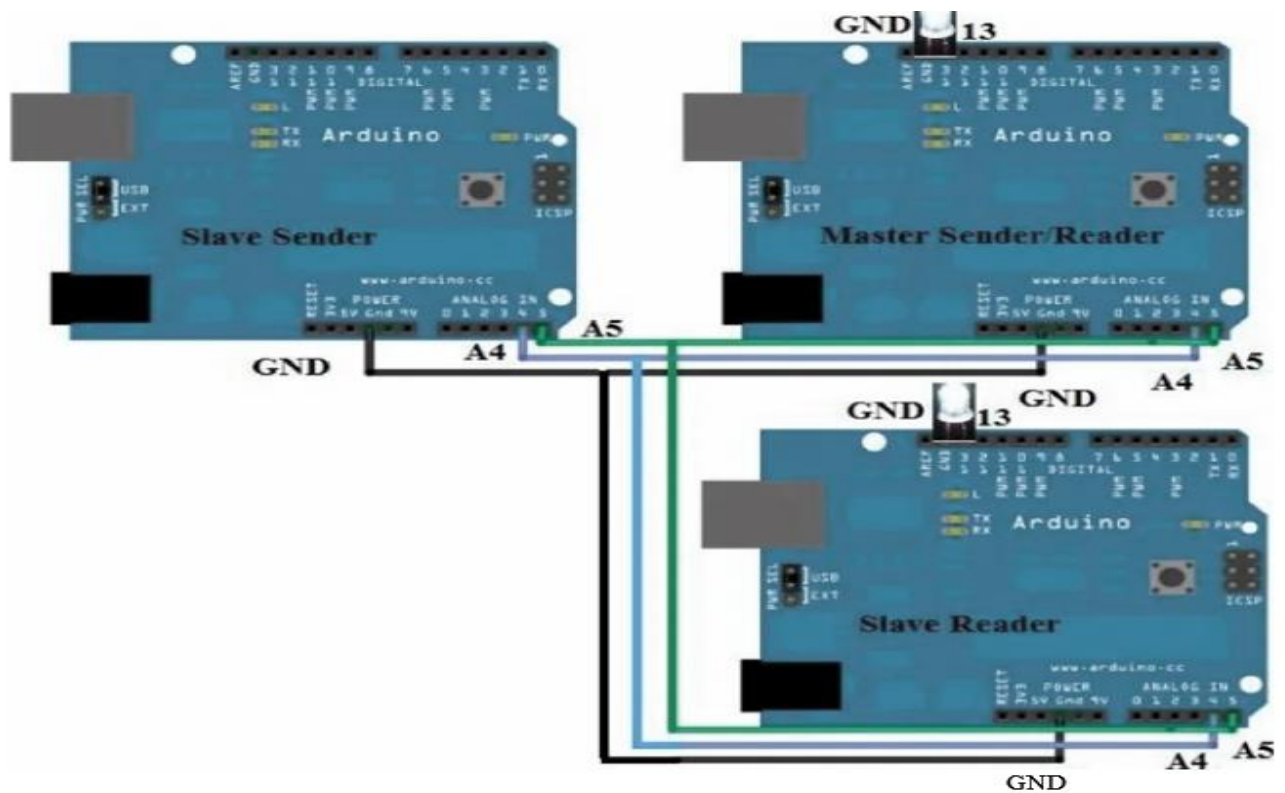
Objective:

In some situations, it can be helpful to set up two (or more!) Arduino and Genuino boards to share information with each other. In this example, two boards are programmed to communicate with one another in a Master Reader/ Slave Sender configuration via the I2C synchronous serial protocol. Several functions of Arduino's Wire Library (<http://www.arduino.cc/en/Reference/Wire>) are used to accomplish this. Arduino 1, the Master, is programmed to request, and then read 6 bytes of data sent from the uniquely addressed Slave Arduino. Once that message / a character is received, it can then be viewed in the Arduino Software (IDE) serial monitor window and LED will glow in both boards.

The I2C protocol involves using two lines to send and receive data: a serial clock pin (SCL) that the Arduino or Genuino Master board pulses at a regular interval, and a serial data pin (SDA) over which data is sent between the two devices. As the clock line changes from low to high (known as the rising edge of the clock pulse), a single bit of information - that will form in sequence the address of a specific device and a command or data - is transferred from the board to the I2C device over the SDA line. When this information is sent - bit after bit - the called upon device executes the request and transmits its data back - if required - to the board over the same line using the clock signal still generated by the Master on SCL as timing.

Components Required:

<u>Sl.No</u>	Components Required	QTY
1	Arduino UNO Master and PORT(COM4)	1
2	Arduino UNO Slave Sender and PORT(COM3)	1
3	Arduino UNO Slave Sender and PORT(COM3)	1
4	<u>Arduino</u> UNO USB cables	3
5	LED lights	2
6	Bread board	1
7	Connecting Wires	9



Connections:

Arduino Master Pins	Arduino Slaves Pins
GND	GND
Arduino Master Pins	LED
13	
GND	<u>-V_e</u>

Procedure:

1. Set up 'the connection as per diagram through bread board and connect LEDs to Master and Slave Reader.
2. Compile and upload sketch Arduino UNO Master Sender/Reader 'through COM4.
3. Compile and upload sketch Arduino UNO Slave Sender 'through COM3.
4. Compile and upload sketch Arduino UNO Slave Reader 'through COM5.
5. See the Output status message on Serial Monitor and LED light status ON (LED ONS) of Master Reader and Slave Reader UNO boards.

Master Source Code:

```
// Wire Master Reader
```

```
// Demonstrates use of the Wire 'library
```

```
// Reads data from an 12C/TWI slave device
// Refer 'to the "Wire Slave Sender/Reciever' example for use with this
#include <Wire.h> void setup()
Wire. begin(); // join i2c bus (address optional for master) Serial.begin(9600); // start serial for output
digitalWrite(13,LOW);
Wire.begin(5); // join i2c bus with address #8 Wire.onRequest(requestEvent); // register event void loop()
Wire.requestFrom(8, 6); // request 6 bytes from slave device #8 while (Wire.available())
// slave may send less 'than requested char c = Wire.read(); // receive a byte as character // Serial.print(c); //
print the character if(c
Serial.printlnC'Command Accepted, Master- LED ON"); else if(c
digitalWrite(13,LOW);
Serial.println("Command Accepted LED OFF");
delay(500); void requestEvent()
Wire.writeCH t); // respond with message of 6 bytes
// as expected by master
```

Slave Sender Source Code:

```
// Wire Slave Sender
// Demonstrates use of the Wire library
// Sends data as an 12C/TWI slave device
// Refer to the "Wire Master Readed' example for use with this #include <Wire.h> char c; void setup()
Wire.begin(8); // join i2c bus with address #8 Wire.onRequest(requestEvent); // register event void loop()
delay(100);
// function that executes whenever data is requested by master // this function is registered as an event, see
setup() void requestEvent()
Wire.write(t H'); // respond with message of 6 bytes
// as expected by master
```

Slave Reader Code:

```
// Wire Slave Reader
// Demonstrates use of the Wire library
// Reads data from an 12C/TWI master device
#include <Wire.h> void setup()
Wire. begin(); // join i2c bus (address optional for master)
Serial.begin(9600); // start serial for output
digitalWrite(13,LOW); void loop()
Wire.requestFrom(5, 6); // request 6 bytes from slave device #5 while (Wire.available())
```

```
// slave may send less than requested char c = Wire.read(); // receive a byte as character if(c 'H I)
Serial.println("Command Accepted, Slave Reader LED ON");
else if(c == I L')
digitalWrite(13,LOW);
Serial.println("Command Accepted, Slave Reader LED OFF");
delay(500);
```

MASTER SENDER READER OUTPUT:

SLAVE READER OUTPUT:

Applications :The 12C protocol used to connect a maximum of 128 devices that are all connected to communicate with 'the SCL and SDL lines of the master unit as well as the slave devices. It supports Multi master communication, which means two masters are used 'to communicate 'the external devices.

Conclusion:

The 12C protocol allows for each enabled device to have its own unique address, and as both master and slave devices to take turns communicating over a single line, it is possible for your Arduino or Genuino board to communicate (in turn) with many devices, or other boards, while using just two pins of microcontroller.

Experiment No 5 Reading Temperature and Relative Humidity value from the sensor.

Aim:

Read the sensor data 'to measure Temperature and Relative Humidity value.

Objective:

To study the Temperature and Relative Humidity value of sensor data.

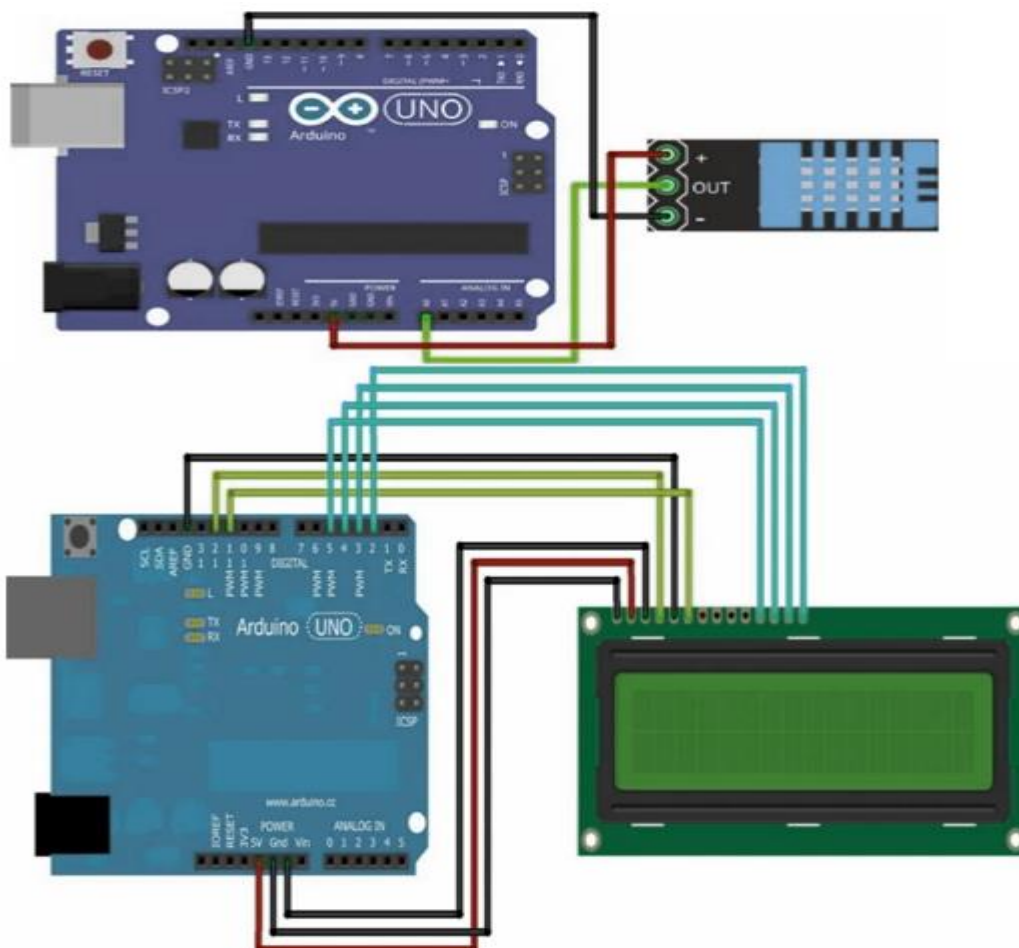
Components Required:

1. Arduino UNO.
2. DHTII Temperature and Humidity sensor data .

Connections:

Arduino Pins	DHTII Sensor
	OUT
GND	-VE

CONNECTIONS



Procedure:

1. Set up pin connection DHTII Humidity Temperature sensor and Arduino UNO as shown in figure.

2. Open the Arduino UNO IDE and create and save a sketch source code as follows.

3. Compile sketch.

4. Upload sketch on to Arduino UNO.

Source Code:

```
#include <dht.h>

#include <LiquidCrystal.h>

#define dht_apin AO // Analog Pin sensor is connected to char chi int Contrast=15;

// initialize the library with the numbers of the interface pins LiquidCrystal lcd(12, 11, 5, 4, 3, 2); dht DHT;

void setup()

Serial.begin(9600); to let system boot

Humidity & 'temperature Sensor\n \n");

analogWrite(6,Contrast);

// set up the LCD's number of columns and rows: lcd.begin(16, 2); delay(1000); //Wait before accessing Sensor

lend

void loop(){

//Start of Program

DHT.read11 (dht_apin);

//Print on Serial Monitor

Serial.printC'Current humidity = Serial. print(DHT.humidity);

Serial.

Serial

Serial. print(DHT.temperature);

Serial. println("C I");

lcd.setCursor(0,0); //goto first column (column 0) and first line (Line 0)

C = //Print at cursor Location

lcd.print(DHT.temperature);

lcd.setCursor(0,1); //goto first column (column 0) and second line (line 1)

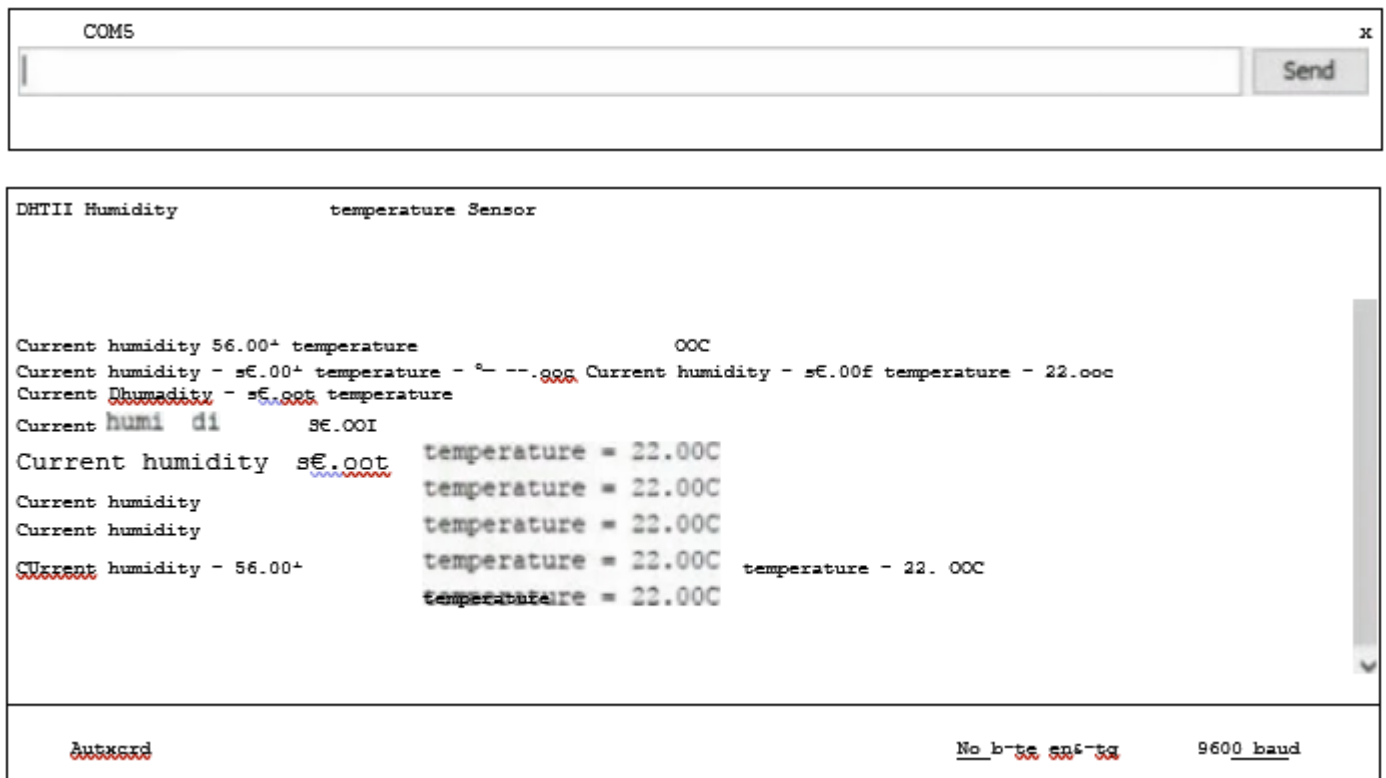
lcd.print("Hum % = "); //Print at cursor location lcd.print(DHT.humidity); delay(5000); //Wait 5 seconds

before accessing sensor again.

/[Fastest should be once every two seconds.

} // end loop()
```

OUTPUT:



The screenshot shows a serial monitor window titled 'COM5' with a 'Send' button. The main area displays the output of a DHT11 Humidity and temperature sensor. The text is as follows:

```
DHT11 Humidity      temperature Sensor

Current humidity 56.00+ temperature      00C
Current humidity - sE.00+ temperature - ^- --.000 Current humidity - sE.00f temperature - 22.00c
Current Dhumadity - sE.00t temperature
Current humi di      SE.00I
Current humidity sE.00t      temperature = 22.00C
Current humidity      temperature = 22.00C
Current humidity      temperature = 22.00C
Current humidity - 56.00+      temperature = 22.00C      temperature - 22. 00C
      temperature = 22.00C
```

At the bottom, the status bar shows 'Autocrd', 'No b-te en5-tg', and '9600 baud'.

Applications :

A humidity sensor senses, measures and reports both moisture and air temperature. The ratio of moisture in the air to the highest amount of moisture at a particular air temperature is called relative humidity. Relative humidity becomes an important factor, when looking for comfort Automated humidity control systems use humidity sensors to test humidity levels and can control processes, such as chemical dryers, required to maintain a specific relative humidity in industries.

Conclusion:

Air temperature and humidity sensors provide the benefit of obtaining measurement data for two parameters humidity and temperature using a single sensor. Sensor data help us to study and measure the humidity and temperature.

MINI PROJECT:

Home automation system using the Nodemcu ESP8266 board and the new Blynk application.

Aim: To make a home automation system using the Nodemcu ESP8266 board and the new Blynk app.

Objective: The appliances in your home can be connected to these relays. These include lamps, fans, televisions, water pumps, etc. After that, we can easily control these appliances using our smartphones. For that, this article will guide you. Also, I used two lamps for example. You can use any other appliances. But, you have to be careful about the AC voltage.

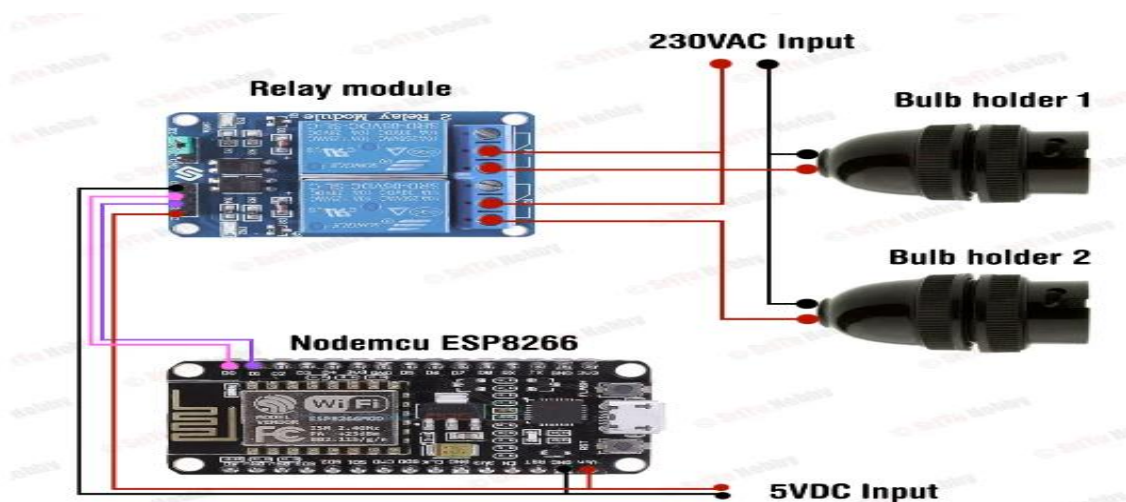
Components Required:

- Nodemcu board x 1
- Relay module x
- Bulb holder x 2
- Bulb x 2
- Plug top x 1
- Wires
- Breadboard x 1
- Jumper wires

Procedure:

1. connect the Nodemcu board to the breadboard.
2. connect the bulb holders to the relay module. For that, use the circuit diagram.
3. connect the plug top to the 230VAC input point.
4. connect the relay module to the Nodemcu board using the jumper wires. For that, use the circuit diagram above. Next, connect it to the computer.
5. set up the Blynk web application and mobile application.
6. create the program for this project.
7. connect the 5 VDC external power supply to the Nodemcu board.
8. put the bulbs to the holders and connect the plug top to the AC voltage.

Circuit Diagram:



Code:

*New Blynk app with Home Automation

<https://srituhobby.com>

*/

//Include the library files

#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

//Define the relay pins

#define relay1 D0

#define relay2 D1

#define BLYNK_AUTH_TOKEN "" //Enter your blynk auth token

char auth[] = BLYNK_AUTH_TOKEN;

char ssid[] = ""; //Enter your WIFI name

char pass[] = ""; //Enter your WIFI password

//Get the button values

BLYNK_WRITE(V0) {

bool value1 = param.asInt();

// Check these values and turn the relay1 ON and OFF

if (value1 == 1) {

digitalWrite(relay1, LOW);

} else {

digitalWrite(relay1, HIGH);

}

}

//Get the button values

BLYNK_WRITE(V1) {

bool value2 = param.asInt();

// Check these values and turn the relay2 ON and OFF

if (value2 == 1) {

digitalWrite(relay2, LOW);

} else {

digitalWrite(relay2, HIGH);

}

}

void setup() {

//Set the relay pins as output pins

pinMode(relay1, OUTPUT);

pinMode(relay2, OUTPUT);

// Turn OFF the relay

digitalWrite(relay1, HIGH);

digitalWrite(relay2, HIGH);

//Initialize the Blynk library

Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);

}

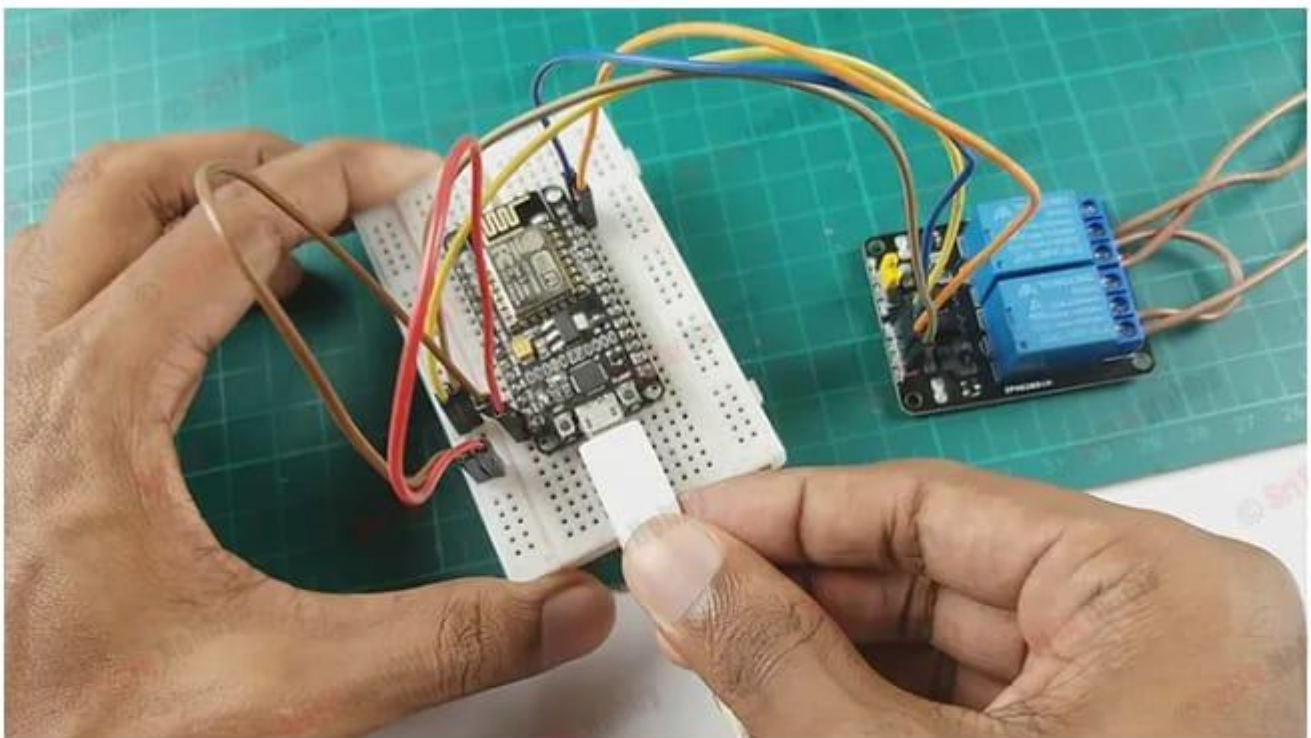
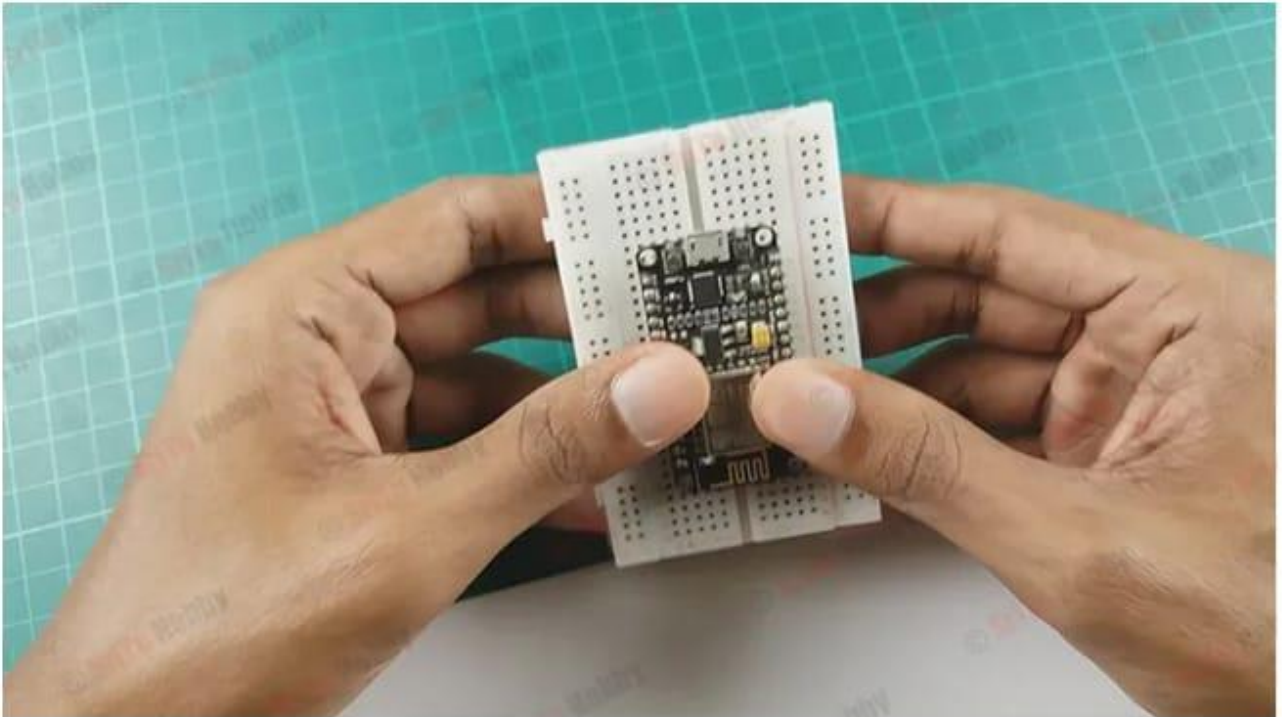
void loop() {

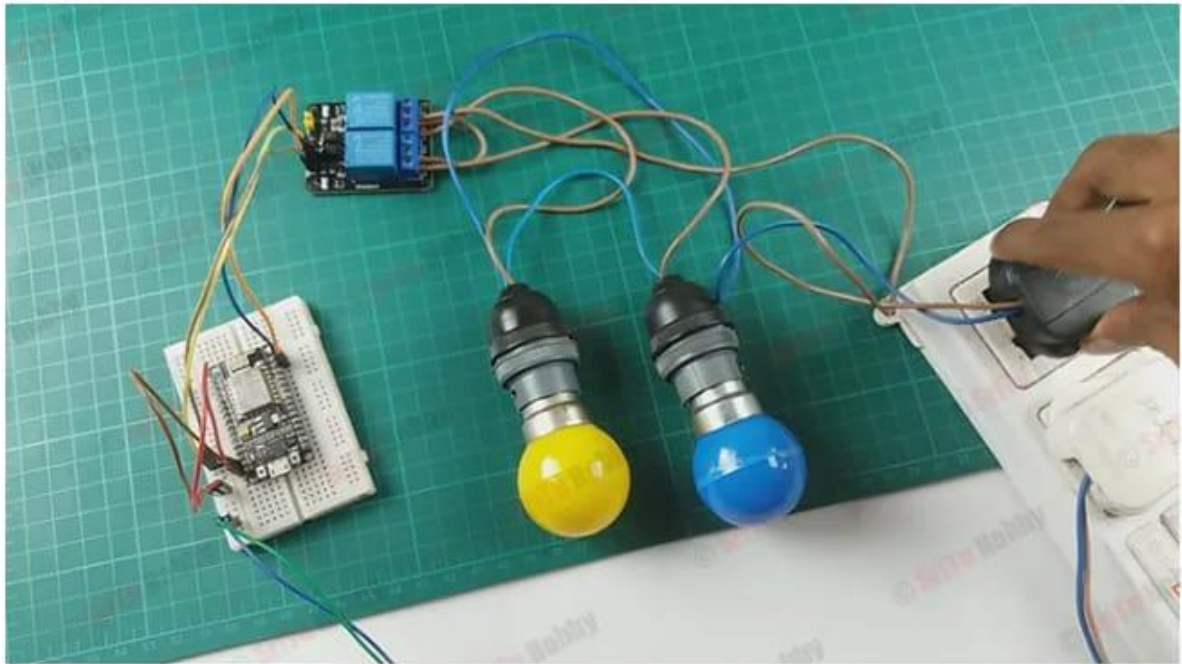
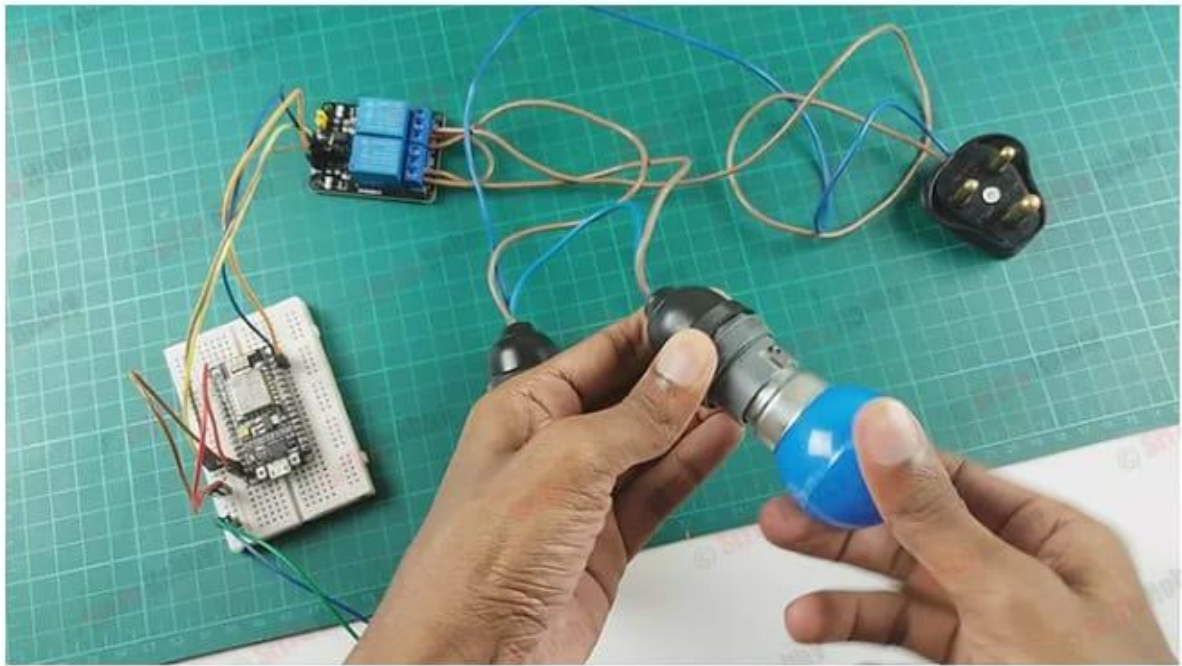
//Run the Blynk library

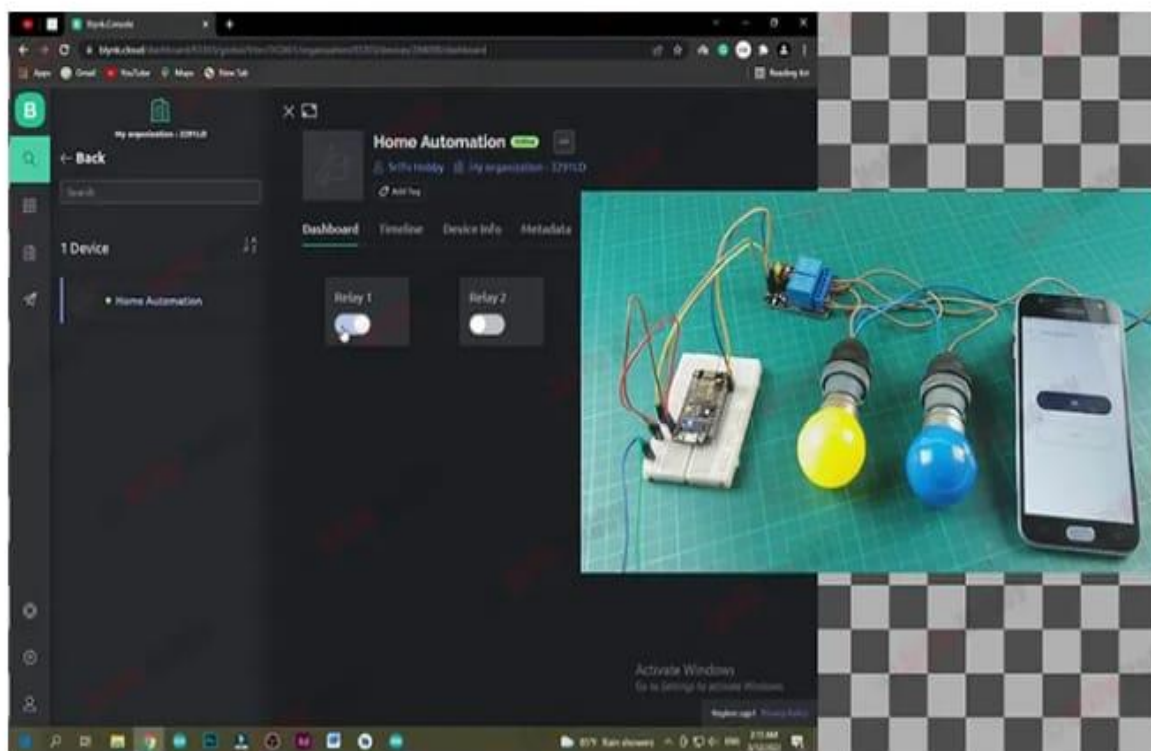
Blynk.run();

}

Screenshots:







Conclusion

The operating system of the smart mobile phone in android we develop remote control program. The program connected with wi-fi to communicate with the robot. Wireless control is the most important basic needs for all the people. Wireless network controlled robots use wi-fi modules.

Blynk android application will transmit command using wi-fi to the car so that it can move in the required direction like moving forward, reverse, turning left, turning right and stop.

References

- [1] Weimei Zhang,"Study about IOT's application in Digital Agriculture construction", Electrical and Control Engineering (ICECE), 2011 International Conference, Yichang, IEEE, 2011, pp. 2578-2581.
- [2] Mircea Murar and Stelian Brad,"Monitoring and controlling of smart equipments using Android compatible devices towards IoT applications and services in manufacturing industry", Automation, Quality and Testing, Robotics, 2014 IEEE International Conference, Cluj-Napoca, pp. 1-5.
- [3] Takeshi Yashiro, Shinsuke Kobayashi, Noboru Koshizuka and Ken Sakamura, "An Internet of Things (IoT) Architecture for Embedded Appliances", Electrical and Control Engineering (ICECE), 2011 International Conference, Yichang, IEEE, 2011, pp. 2578-2581.
- [4] Keertikumar M. J, Shubham M. and R.M. Banakar, "Evolution of IoT in Smart Vehicles: An Overview", IEEE Computer Society, pp. 804-809.
- [5] Takayuki Suyama, Yasue Kishino and Futoshi Naya "Abstracting IoT devices using virtual machine for wireless sensor nodes", Internet of Things (WF-IoT), 2014 IEEE World Forum, Seoul,2014, pp. 367-368. 1–16
- [6] Seung-Chul Son, Nak-Woo Kim, Byung-Tak Lee, Chae Ho Cho, and Jo Woon Chong "A Time Synchronization Technique for CoAP-based Home Automation Systems", IEEE Transactions on Consumer Electronics,February 2016, Vol. 62, No. 1, pp. 10-16.