School of Electronics Engineering

ECE6001 Wireless Sensor Networks and IoT J-Component

SMART CONTROLLING OF TEMPERATURE USING ARDUINO WITH NRF COMMUNICATION

A project report submitted as a part of J-component of

ECE6001 Wireless Sensor Networks and IoT

By

Mohan Kumar R (21MTS0014)

Prakash P (21MTS0017)

Under the guidance of

Prof. SivaCoumar R

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

WINTER SEMESTER 2021-22

DECLARATION

We hereby declare that the report entitled ‘SMART CONTROLLING OF TEMPERATURE USING ARDUINO WITH NRF COMMUNICATION’ submitted by us for the credit of the j-component of ECE6001 Wireless Sensor Networks and IoT to Prof. Sivacoumar R, SENSE, VIT is a bonafide work carried out by us and has not been submitted to any other course or institute.

Place: VIT- VELLORE

Date: 10/06/2022

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

There are various wireless communication technologies used in building IoT applications and RF (Radio Frequency) is one of them. nRF24L01 is a single-chip radio transceiver module that operates on 2.4 - 2.5 GHz (ISM band). This transceiver module consists of a fully integrated frequency synthesizer, a power amplifier, a crystal oscillator, a demodulator, a modulator, and Enhanced Shock Burs protocol engine. Output power, frequency channels, and protocol setup are easily programmable through an SPI interface. The operating voltage range of this Transceiver module is 1.9V to 3.6V. It has Built-in Power Down and Standby modes that make it power-saving and easily realizable.

Arduino Uno doesn't have any inbuilt wireless communication support like Bluetooth or Wi-Fi, but it can be easily interfaced with other Wi-Fi or wireless modules like LoRa, nRF, Bluetooth and can be used to build IoT based Applications. Here we have used Arduino with ESP8266, with LoRa, with 433 MHz RF Module, with nRF24L01 module and with Bluetooth to establish wireless communication for IoT projects.

2. Hardware Requirements

2.1 ESP8266

The ESP8266 is a low-cost Wi-Fi chip with full TCP/IP stack and MCU (Micro Controller Unit) capability produced by Shanghai-based manufacturer, Espressif Systems. You can use it to provide WiFi access for your Raspberry-PI boards very inexpensively; or you can simply use it by itself. The low cost of ESP8266 makes it ideal for Internet of Things. NodeMCU is an open-source LUA based firmware developed for the ESP8266 wifi chip. By exploring functionality with the ESP8266 chip, NodeMCU firmware comes with the ESP8266 Development board/kit i.e. NodeMCU Development board. Since NodeMCU is an open-source platform, its hardware design is open for edit/modify/build. NodeMCU Dev Kit/board consist of ESP8266 wifi enabled chip. The ESP8266 is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol. For more information about ESP8266, you can refer to the ESP8266 WiFi Module. There is Version2 (V2) available for NodeMCU Dev Kit i.e. NodeMCU Development Board v1.0 (Version2)**,** which usually comes in black colored PCB. NodeMCU Development board is featured with wifi capability, analog pin, digital pins, and serial communication protocols. To get started with using NodeMCU for IoT applications first we need to know about how to write/download NodeMCU firmware in NodeMCU Development Boards. And before that where this NodeMCU firmware will get as

per our requirement. There are online NodeMCU custom builds available using which we can easily get our custom NodeMCU firmware as per our requirement. After setting up ESP8266 with Node-MCU firmware, let's see the IDE (Integrated Development Environment) required for the development of NodeMCU.

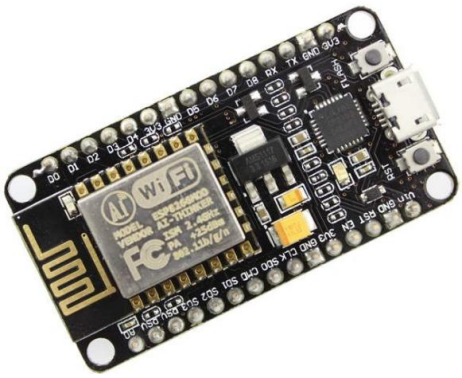

Fig. 1 NodeMCU

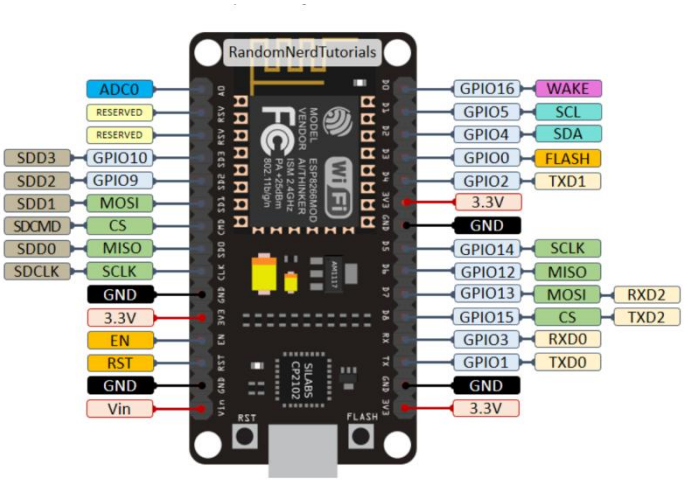

Fig. 2 NodeMCU Pin Diagram

2.2 Arduino Uno

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and

professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike. Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments.There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than \$50

- **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

- **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

- **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

- **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

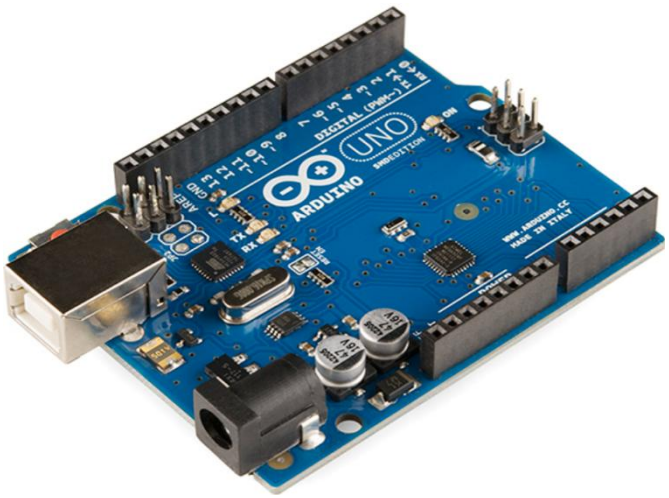

Fig. 3 Arduino Uno

2.3 Relay Module

A power relay module is an electrical switch that is operated by an electromagnet. The electromagnet is activated by a separate low-power signal from a micro controller. When activated, the electromagnet pulls to either open or close an electrical circuit. A simple relay consists of wire coil wrapped around a soft iron core, or solenoid, an iron yoke that delivers a low reluctance path for magnetic flux, a movable iron armature and one or more sets of contacts. The movable armature is hinged to the yoke and linked to one or more set of the moving contacts. Held in place by a spring, the armature leaves a gap in the magnetic circuit when the relay is de-energized. While in this position, one of the two sets of contacts is closed while the other set remains open. When electrical current is passed through a coil, it generates a magnetic field that in turn activates the armature. This movement of the movable contacts makes or breaks a connection with the fixed contact. When the relay is de-energized, the sets of contacts that were closed, open and breaks the connection and vice versa if the contacts were open. When switching off the current to the coil, the armature is returned, by force, to its relaxed position. This force is usually provided by a spring, but gravity can also be used in certain applications. Most power relays are manufactured to operate in a quick

manner. For distribution of power in high current applications, GEP Power Products is the industry leader in high power relay module design and manufacturing. Rated up to 70 amps, GEP's power relay modules are designed for seamless integration in high power distribution applications. The convenient integral mounting brackets provide easy installation and accessibility. With endless options such as terminal position assurance available for wire retention, GEP Power Products' power distribution solutions and off-road industry knowledge are second to none.

Fig. 4 Single Channel 5V Relay Module

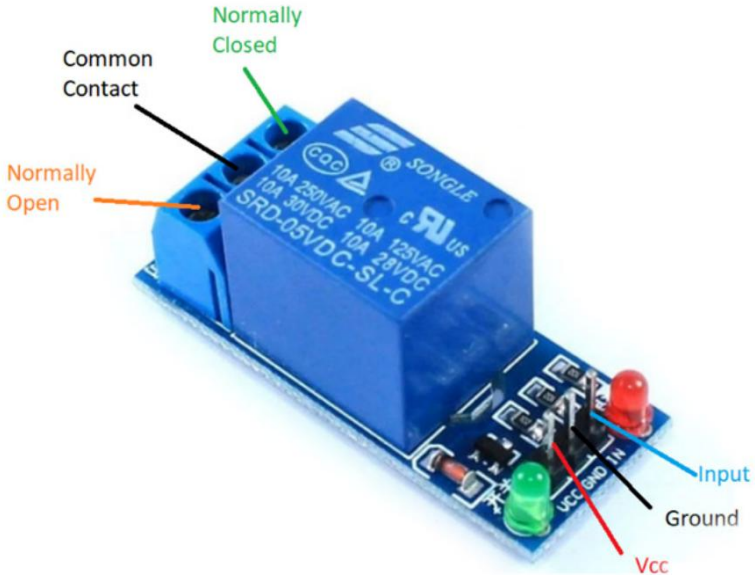

Fig. 5 Pin Diagram of 5V Channel Relay Module

2.4 LM35 Temperature Sensor

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ˚ Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±1⁄4˚C at room temperature and ±3⁄4˚C over a full −55 to +150˚C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1˚C in still air. The LM35 is rated to operate over a −55˚ to +150˚C temperature range, while the LM35C is rated for a −40˚ to +110˚C range (−10˚ with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package. The LM35 series are precision integrated-circuit 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. Lower cost is assured by trimming and calibration at the wafer level. The low-output impedance, linear output and precise inherent calibration of the LM35 device makes interfacing to readout or control circuitry especially easy. The device is used with single power supplies, or with plus and minus supplies. As the LM35 device draws only 60 µA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 device is rated to operate over a −55°C to 150°C temperature range, while the LM35C device is rated for a −40°C to 110°C range (−10° with improved accuracy). The LM35-series devices are available packaged in hermetic TO transistor packages, while the LM35C, LM35CA, and LM35D devices are available in the plastic TO-92 transistor package. The LM35D device is available in an 8-lead surface-mount small-outline package and a plastic TO-220 package.

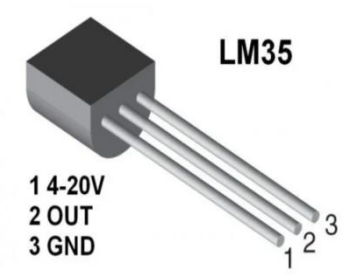

Fig. 6 LM35 Temperature Sensor

2.5 LCD Display

16×2 LCD is a 32 digits display screen for all kinds of CMOS/TTL devices. This word comes from the liquid crystal and 16X2 represents its screen size. In Liquid crystal display 16×2, there are 2 rows and 16 columns. Besides, 5×8 pixel makes a single digit. Any digit from ASCII code is viewable on the module. It supports the custom signs and designs but those require some specific methods and have some limitations. This display module has too much use in most of the commercial projects and there is almost a library in every programming language about it. The premade libraries made it easy to interface with other devices. It will be a very complicated task to handle everything with the help of a microcontroller. So an Interface IC like HD44780 is used, which is mounted on the backside of the LCD Module. The function of this IC is to get the Commands and Data from the microcontroller and process them to display meaningful information onto the LCD Screen. Command register stores various commands given to the display. Data register stores data to be displayed. The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. Printing a character to LCD16x2, we need to send the ASCII code of that character to LCD16x2. Suppose we want to print the character 'H' on the LCD, then we need to send 0x48 data (ASCII code of 'H') to LCD16x2. The LCD16x2 has its own controller, which takes over the printing work on the LCD16x2.LCDs can either be normally on (positive) or off (negative), depending on the polarizer arrangement. For example, a character positive LCD with a backlight will have black lettering on a background that is the colour of the backlight, and a character negative LCD will have a black background with the letters being of the same colour as the backlight. Optical filters are added to white on blue LCDs to give them their characteristic appearance.

Fig. 7 Pin Diagram of LCD Display

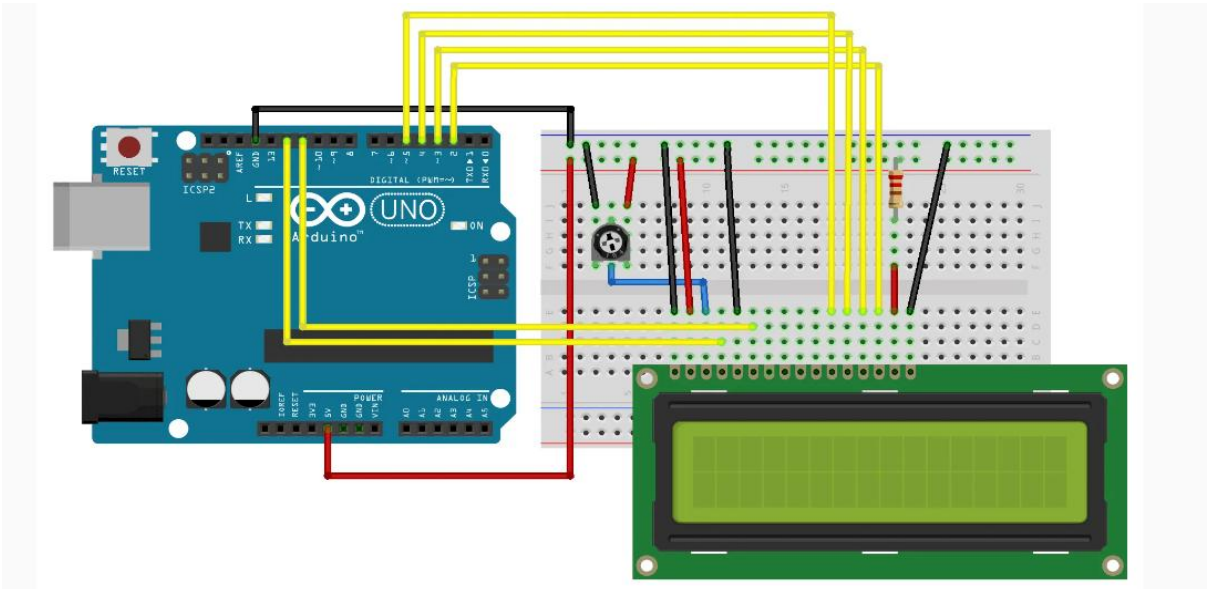

Fig. 8 Circuit diagram of LCD Display and Arduino Uno

2.6 NRF

nRF24L01 is a single chip radio transceiver for the world wide 2.4 - 2.5 GHz ISM band. The transceiver consists of a fully integrated frequency synthesizer, a power amplifier, a crystal oscillator, a demodulator, modulator and Enhanced Shock Burst protocol engine. Output power, frequency channels, and protocol setup are easily programmable through a SPI interface. Current consumption is very low, only 9.0mA at an output power of -6dBm and 12.3mA in RX mode. Built-in Power Down and Standby modes makes power saving easily realizable. The radio transmitters and receivers include frequency generator, enhanced Shock Burst mode controller, power amplifier, crystal oscillator modulator and demodulator. You

can select the output power channel and protocol by setting through the SPI port. The current consumption for the nRF24L01+ is extremely low - under the transmitter mode, when the transmitting power is 0dBm, the current consumption is only 11.3mA; under the receiving mode, it is 13.5mA; under the power down and idle mode, the consumption is even lower. As for application, it's widely used in many devices such as wireless mouse and keyboard, game handle, remote control set, industry sensor, toys, etc. The nRF24L01 is a wireless transceiver module, meaning each module can both send as well as receive data. They operate in the frequency of 2.4GHz, which falls under the ISM band and hence it is legal to use in almost all countries for engineering applications. The modules when operated efficiently can cover a distance of 100 meters (200 feet) which makes it a great choice for all wireless remote controlled projects. The module operates at 3.3V hence can be easily used with 3.2V systems or 5V systems. Each module has an address range of 125 and each module can communicate with 6 other modules hence it is possible to have multiple wireless units communicating with each other in a particular area. Hence mesh networks or other types of networks are possible using this module. The NRF24L01 module works with the help of SPI communications. These modules can either be used with a 3.3V microcontroller or a 5V microcontroller but it should have an SPI port. The complete details on how to use the module through SPI is given the data sheet below. The circuit diagram shows how the module should be interfaced with a microcontroller. If you are interfacing the module with Arduino , then there are ready made libraries available like the R24Library. With the help of these libraries you can easily interface the nRF24L01 with Arduino with few lines of code. If you are using for some other microcontroller then you have to read through the datasheet to understand how to establish the SPI communication. The nRF24L01 module is a bit tricky to use especially since there are many cloned versions in the market. If you are having any problem with getting it work, try adding a 10uF and 0.1uF capacitor in parallel to the Vcc and Ground pins. Also make sure the 3.3V supply is clean and does not have any noise coupled in it.

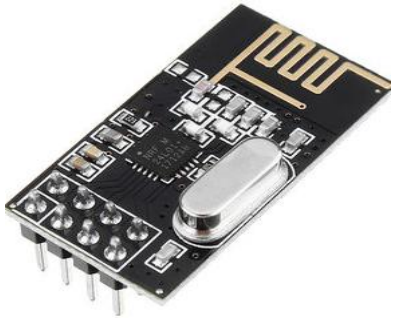

Fig. 9 nRF24L01

3.Working

3.1 Interfacing nRF24L01 with Arduino UNO - Transmitting Side

The transmitter side consists of an Arduino UNO, nRF24L01 module, 5v Relay Module, SMPS power supply, Bulb, Fan. Arduino continuously gets data from the Lm35 sensor and sends it to the Arduino board for checking the condition for threshold temperature, if it higher than the threshold then relay will trigger the fan and turn off the bulb in opposite case bulb will turn on and fan will turn off the switching operation is controlled by relay and finally all the data values are sent to the nRF24L01 Transmitter.

3.2 Interfacing nRF24L01 with NodeMCU - Receiving Side

The receiver side consists of nRF24L01 module, and 16*2 LCD module and Node MCU. The receiver side nRF module receives the data from the transmitter and sends it to Node MCU which sends the data to the webserver.

4.Code Implementation

4.1 Transmitting Side

```
#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

int msg[1];

RF24 radio(7, 8); // CE, CSN

const byte address[6] = "00001";

const char text[10];

const int sensor=A0; // Assigning analog pin A5 to variable 'sensor'

float tempc; //variable to store temperature in degree Celsius

float tempf; //variable to store temperature in Fahreinheit

float vout; //temporary variable to hold sensor reading

int fan=2;

int lamp=3;
```

```
void setup() {
  pinMode(sensor,INPUT);
  pinMode(fan,OUTPUT);
  pinMode(lamp,OUTPUT);
  Serial.begin(9600);
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening();
}
void loop() {
  vout=analogRead(sensor); //Reading the value from sensor
  vout=(vout*500)/1023;
  tempc=vout; // Storing value in Degree Celsius
  tempf=(vout*1.8)+32; // Converting to Fahrenheit
  Serial.print("In Degree C = ");
  Serial.print(tempc);
  Serial.print(" ");
  Serial.print(" In Fahrenheit = ");
  Serial.print(tempf);
  Serial.println();
  dtostrf(tempc,2,2,text);
  radio.write(&text, sizeof(text));
if(tempc<36.0){
    digitalWrite(lamp,HIGH);
    digitalWrite(fan,LOW);
```

```
    Serial.println("condition1");
    }
  if(tempc>=36.0){
    digitalWrite(lamp,LOW);
    digitalWrite(fan,HIGH);
    Serial.println("condition2");
    }
    delay(1000);
}
```

4.2 Receiving Side

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <LiquidCrystal.h>
const int RS = D0, EN = D1, d4 = D8, d5 = D3, d6 = 3, d7 = 1;
LiquidCrystal lcd(RS, EN, d4, d5, d6, d7);
int msg[1];
RF24 radio(2, 4);
const byte address[6] = "00001";
const char* ssid = "******"; //ssid of your wifi
const char* password = "******"; //password of your wifi
const char* host = "******"; //replace it with your webhost url
void setup() {
  Serial.begin(9600);
```

```
  lcd.begin(16, 2);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    lcd.clear();
    lcd.print("WiFi connecting");
  }
  lcd.clear();
  lcd.print("WiFi connected");
  delay(2000);
}
void loop() {
  lcd.clear();
  if (radio.available()) {
    char text[10] = "";
    radio.read(&text, sizeof(text));
    Serial.print("Temperature = ");
    Serial.println(text);
    Serial.println(" C..");
    lcd.clear();
    lcd.setCursor(0, 0);
```

```
lcd.print("Temperature in C");
lcd.setCursor(0, 1);
lcd.print(text);
Serial.print("Connecting to ");
Serial.println(host);
WiFiClient client;
const int httpPort = 80;
if(!client.connect(host, httpPort)){
  Serial.println("Connection Failed");
  return;
}
String url = "/insert.php?temperature_c=" + String(text);
Serial.println("Requesting URL: ");
Serial.println(url);
client.print(String("GET ") + url + " HTTP/1.1\r\n" +
             "Host: " + host + "\r\n" +
             "Connection: close\r\n\r\n");
delay(500);
while(client.available()){
  String line = client.readStringUntil('\r');
  Serial.print(line);
}
Serial.println();
Serial.println("closing connection");
delay(2000);
}
```

5. Practical Analysis

5.1 Transmitter Side

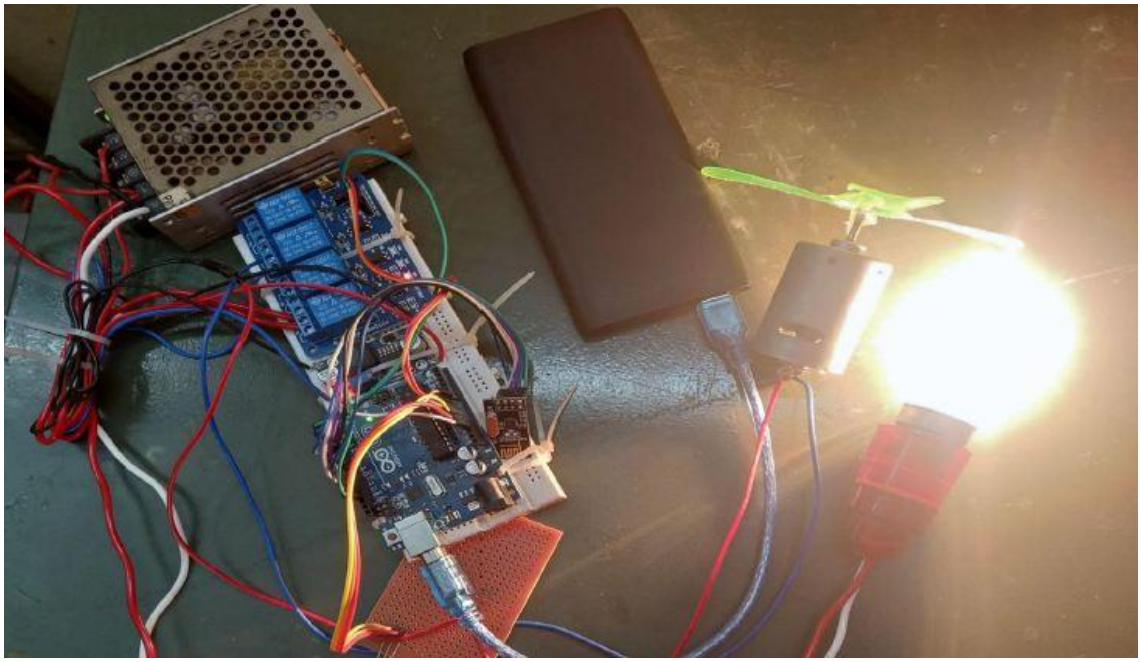

Fig. 10 Interfacing nRF24L01 with Arduino UNO - Transmitting Side

5.2 Receiving Side

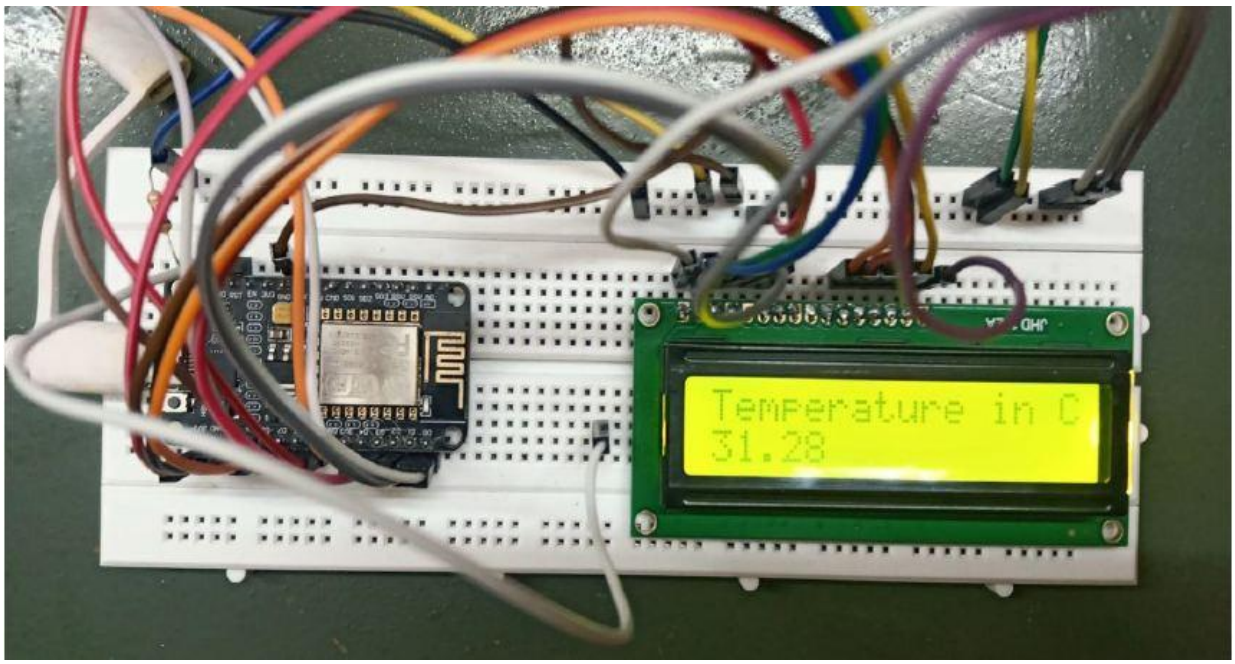

Fig. 11 Interfacing nRF24L01 with NodeMCU - Receiving Side

5.3 Website

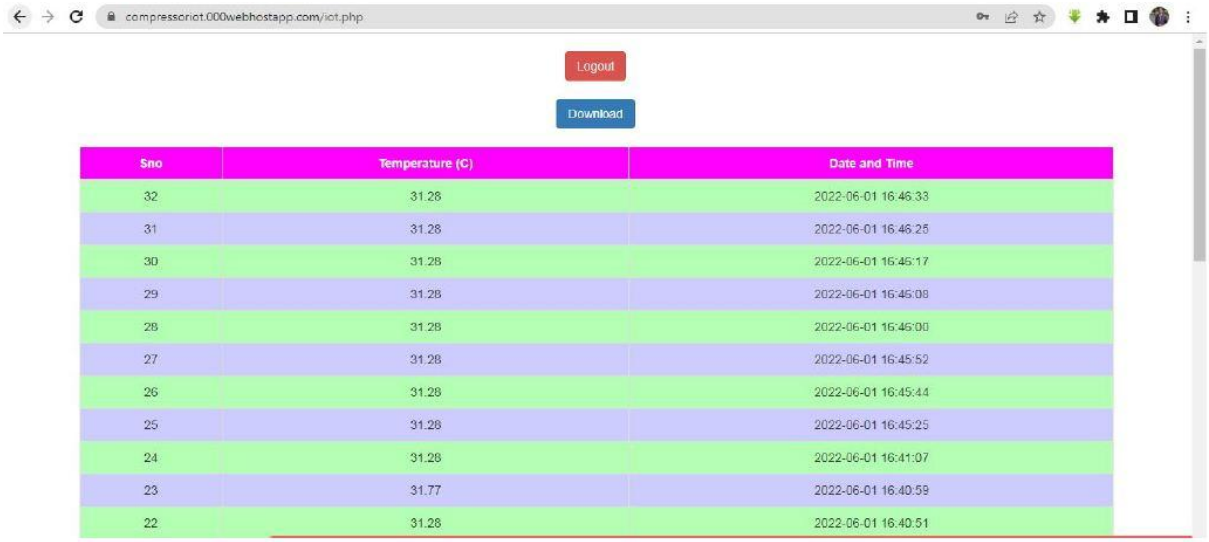

Fig. 12 Website

5.4 Mobile Application

Fig. 13 Mobile Application

6. Conclusion

In this project we have implemented a relay which controls the switch of bulb and fan according to the threshold value of temperature in LM35 which is programmed in Arduino the data is passed to NRF transmitter to send it to the receiver side which sends the data to Node MCU and also display the value in LCD display .The displayed values are also stored in webserver via Node MCU.