1306 Weatherman

-BEWARE OF WEATHER

EMBEDDED SYSTEMS AND INTERNET OF THINGS

Ch. Sai Gowtham-319126512078
E. Rajkamal-319126512079
J. Sri Ram Prateek-319126512086
K. Bharath Kumar-319126512088

TABLE OF CONTENTS

1. INTRODUCTION

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

When using an online application, application connects to server and sends data. The server collects this data, serves according to the request by the client or user. The server retrieves the data, interprets it and performs necessary actions and sends it back to the client. The application again interprets the data and presents the information in a readable way that user wanted to.

To understand in a better way, let us consider an example.

Imagine you are sitting in a restaurant with a menu of choices to order from. The kitchen is a part of system that will prepare your order. But the problem is you cannot directly go into the kitchen and cook the food. It is the waiter who accepts your request and pass it to the kitchen and gets food from kitchen and serve it to you. This is the same thing API does. API gets request from user and pass it to the end-points and gets the data from those end-points and passes it to the user. When we are requesting for a data, API uses GET method for requesting to that endpoint. If the request is valid, then we will get a response in return and generally the data from the web will be in JSON (JavaScript Object Notation) format. To get the required information, we need to parse it. This is how an API works.

OPENWEATHERMAP API:

Openweather API helps working with the weather data in an easier way.

Openweather provides essential weather parameters such as temperature, precipitation, probability of precipitation, humidity, feels like, pressure, cloudiness, wind, etc. The full list of weather parameters for every weather API you can find on the product pages in the API documentation.

History for 40 years with hour granularity

Current state with every 10 min update

Forecast with minute granularity for the next one hour

Forecast for 4 days with hour granularity

Forecast for 16 days that give you data four times a day at night, day, evening, and morning

Forecast for 30 days

Global Precipitation Maps based on radar data, satellite images and powered by ML

Global map layers such as current, forecasted, and historical Temperature, Pressure, Wind speed, Clouds, and others; 15 layers in total.

2. HARDWARE USED

1. Node MCU.

2. 0.96” OLED Display.

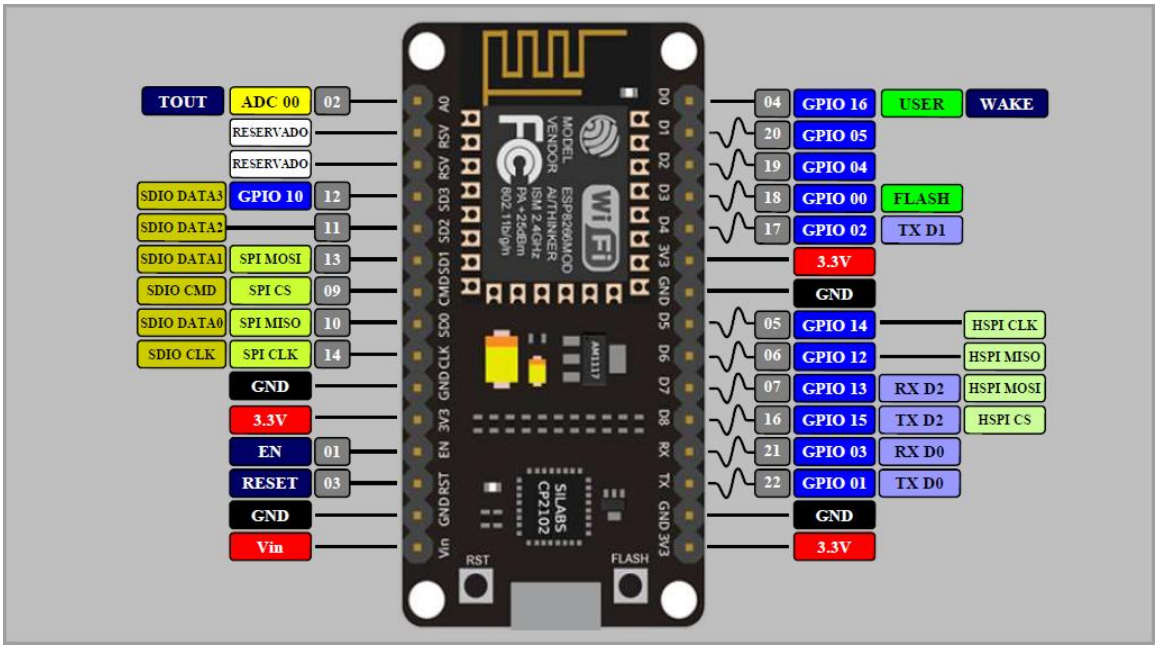
3. Breadboard and Connecting Wires.

4. Power Supply.

Hardware Description:

Node MCU:

NodeMCU is an open-source firmware for which open-source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits. Both the firmware and prototyping board designs are open source.



OLED Display:

An organic light-emitting diode (OLED or organic LED), also known as organic electroluminescent (organic EL) diode, is a light-emitting diode (LED) in which the emissive electroluminescent layer is a film of organic compound that emits light in response to an electric current. This organic layer is situated between two electrodes; typically, at least one of these electrodes is transparent. OLEDs are used to create digital displays in devices such as television screens, computer monitors, and portable systems such as smartphones and handheld game consoles. A major area of research is the development of white OLED devices for use in solid-state lighting applications.

This 0.96” I2C OLED Display is an OLED monochrome 128×64 dot matrix display module with I2C Interface. It is perfect when you need an ultra-small display. Comparing to LCD, OLED screens are way more competitive, which has several advantages such as high brightness, self-emission, high contrast ratio, slim outline, wide viewing angle, wide temperature range, and low power consumption. It is compatible with any 3.3V-5V microcontroller, such as Arduino.

Specifications:

Display Size: 2.44 cm (0.96 inch)

Pixel Color: Blue

Resolution: 128 x 64 Pixels

Driving Voltage: 3.3-5V

Operating Temperature: -40~70 Celsius

Interface Type: I2C IIC

Length (cm): 2.75 cm

Width (cm): 2.75 cm

Height (cm): 1.1 cm

Weight (gm): 7 gm

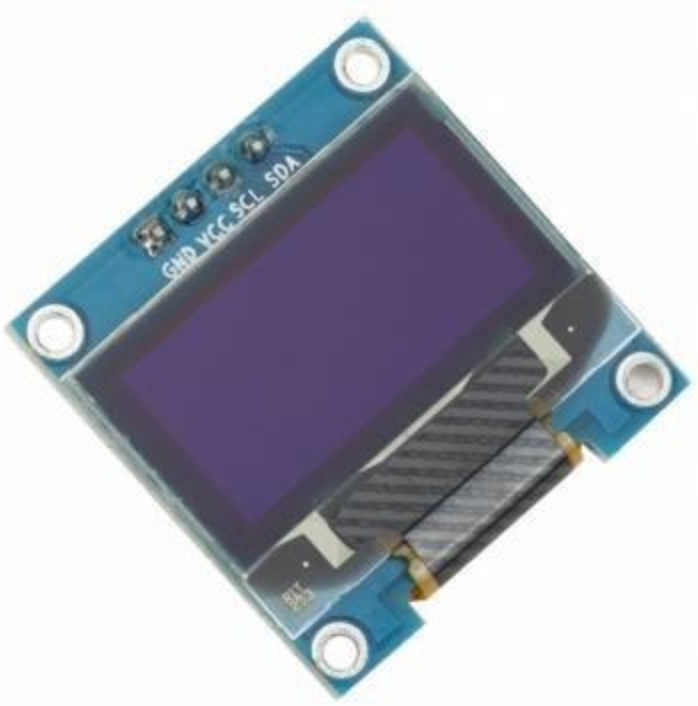
Interface pin for 4 PIN 128x64 OLED:

VCC: 3.3 V-5 V

GND: Ground

SCL: Serial Clock

SDA: Serial Data



3. PROGRAM

```
//---------------------------------------Include Library

#include <ESP8266WiFi.h>#include <ESP8266HTTPClient.h>

#include <WiFiClient.h>

#include <NTPClient.h>

#include <WiFiUdp.h>

#include "TimeLib.h"

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>

#include <ArduinoJson.h>

//----------------------------------------

WiFiClient wifiClient;

#define ON_Board_LED 2  //--> Defining an On Board LED (GPIO2 = D4), used for indicators when the process of connecting to a wifi router and when there is a client request.

//---------------------------------------OLED screen size configuration

#define SCREEN_WIDTH 128 //--> OLED display width, in pixels

#define SCREEN_HEIGHT 64 //--> OLED display height, in pixels

//----------------------------------------

//---------------------------------------Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)

#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

//----------------------------------------

//---------------------------------------Define NTP Client to get time

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP, "pool.ntp.org");

//----------------------------------------
```

```
//----------------------------------------SSID and Password of your WiFi router
const char* ssid = "airtel fiber";  //--> Your wifi name
const char* password = "Pass#123"; //--> Your wifi password
//----------------------------------------
//----------------------------------------TimeLib configuration
struct timelib_tm tinfo; //--> Structure that holds human readable time information;
timelib_t now, initialt;
unsigned long previousMillisGetTimeDate = 0; //--> will store last time was updated
const long intervalGetTimeDate = 1000; //--> The interval for updating the time and date
//----------------------------------------
//----------------------------------------Declaration of time and date variables
//Week Days
String weekDays[7]={"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"};
String weekDays3Dgt[7]={"SUN", "MON", "TUE", "WED", "THU", "FRI", "SAT"};
//Month names
String months[12]={"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct",
"Nov", "Dec"};
unsigned long epochTime;
int currentHour,currentMinute,currentSecond;
String weekDay;
int monthDay,currentMonth;
String currentMonthName;
int currentYear,currentYearforSet;
String _DateTime;
//----------------------------------------
//----------------------------------------Variable declarations to hold weather data
```

```
String current_weather_Description;

String current_weather;

int current_temperature, current_pressure, current_humidity, current_wind_deg,
current_visibility;

float current_wind_speed, current_feels_like, current_uv, current_dew_point;

String current_weather_sym;

String forecast_weather[4];

String forecast_weather_sym[4];

float forecast_temp_min[4];

float forecast_temp_max[4];

//---------------------------------------

//---------------------------------------openweathermap API configuration

String openWeatherMapApiKey = "05cc6d6791dc672dec45dae53fb53ffc";

// Please choose one.

String city = "Visakhapatnam";

String countryCode = "IN";

String units = "metric";

//---------------------------------------

//---------------------------------------Variable declaration for the json data and defining the
ArduinoJson(DynamicJsonBuffer)

String strjsonBuffer;

String strjsonBufferCF;

DynamicJsonBuffer jsonBuffer;

//---------------------------------------


//---------------------------------------Variable declaration for the weather data update indicator

int Upadate_Trig = 1;
```

```
//---------------------------------------

//---------------------------------------Variable declarations for weather and loading symbols / icons

// 'Process1_Sym', 8x8px

const unsigned char Process1_Sym [] PROGMEM = {

  0x00, 0x18, 0x00, 0x01, 0x99, 0x80, 0x01, 0x81, 0x80, 0x08, 0x00, 0x10, 0x18, 0x00, 0x18,
0x00,

  0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0x00, 0x06, 0x60, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00,
0x00,

  0x00, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0x00,
0x06,

  0x60, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x18, 0x08, 0x00, 0x10,
0x01,

  0x81, 0x80, 0x01, 0x99, 0x80, 0x00, 0x18, 0x00

};

// 'Process2_Sym', 8x8px

const unsigned char Process2_Sym [] PROGMEM = {

  0x00, 0x66, 0x00, 0x00, 0x66, 0x00, 0x06, 0x00, 0x60, 0x06, 0x00, 0x60, 0x00, 0x00, 0x00,
0x30,

  0x00, 0x0c, 0x30, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x00, 0x03, 0xc0,
0x00,

  0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0x00, 0x00,
0x00,

  0x00, 0x00, 0x00, 0x30, 0x00, 0x0c, 0x30, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x06, 0x00, 0x60,
0x06,

  0x00, 0x60, 0x00, 0x66, 0x00, 0x00, 0x66, 0x00

};


// 'Clear_Daylight_Sym', 24x24px

const unsigned char Clear_Daylight_Sym [] PROGMEM = {
```

```
  0x00, 0x18, 0x00, 0x00, 0x18, 0x00, 0x10, 0x18, 0x08, 0x38, 0x00, 0x1c, 0x1c, 0x3c, 0x38,
0x08,

  0xff, 0x10, 0x01, 0xc1, 0x80, 0x03, 0x00, 0xc0, 0x06, 0x00, 0x60, 0x06, 0x00, 0x60, 0x0c,
0x00,

  0x30, 0xec, 0x00, 0x37, 0xec, 0x00, 0x37, 0x0c, 0x00, 0x30, 0x06, 0x00, 0x60, 0x06, 0x00,
0x60,

  0x03, 0x00, 0xc0, 0x01, 0xc3, 0x80, 0x08, 0xff, 0x10, 0x1c, 0x3c, 0x38, 0x38, 0x00, 0x1c,
0x10,

  0x18, 0x08, 0x00, 0x18, 0x00, 0x00, 0x18, 0x00

};

// 'Clear_Night_Sym', 24x24px

const unsigned char Clear_Night_Sym [] PROGMEM = {

  0x08, 0x08, 0x80, 0x08, 0x1c, 0xc0, 0x1c, 0x08, 0xe0, 0x3e, 0x00, 0xf0, 0xff, 0x80, 0xd8,
0x3e,

  0x00, 0xcc, 0x1c, 0x00, 0xc6, 0x08, 0x01, 0x86, 0x08, 0x01, 0x83, 0x00, 0x03, 0x03, 0x00,
0x03,

  0x03, 0x40, 0x06, 0x03, 0xe0, 0x06, 0x03, 0x40, 0x1c, 0x03, 0x00, 0x78, 0x03, 0x01, 0xe0,
0x03,

  0xff, 0x80, 0x06, 0x7e, 0x00, 0x06, 0x30, 0x00, 0x0c, 0x18, 0x00, 0x18, 0x0c, 0x00, 0x30,
0x07,

  0x00, 0xe0, 0x03, 0xff, 0xc0, 0x00, 0xff, 0x00

};

// 'Thunderstorm', 24x24px

const unsigned char Thunderstorm_Sym [] PROGMEM = {

  0x01, 0xf0, 0x00, 0x03, 0xf8, 0x70, 0x06, 0x0c, 0xf8, 0x1c, 0x07, 0x8c, 0x38, 0x03, 0x06,
0x60,

  0x00, 0x03, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0xe0, 0x00, 0x07, 0x7f,
0xff,

  0xfe, 0x3f, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x10, 0x1e, 0x02, 0x20, 0x24, 0x04, 0x40, 0x48, 0x08,

  0xf8, 0x9f, 0x9f, 0x11, 0x00, 0x82, 0x21, 0xfd, 0x04, 0x40, 0x0a, 0x08, 0x80, 0x14, 0x10,
0x00,
```

```
  0x28, 0x00, 0x00, 0x50, 0x00, 0x00, 0x60, 0x00

};

// 'Drizzle', 24x24px

const unsigned char Drizzle_Sym [] PROGMEM = {

  0x01, 0xf0, 0x00, 0x03, 0x18, 0x70, 0x06, 0x0c, 0xf8, 0x1c, 0x07, 0x8c, 0x38, 0x03, 0x06,
0x60,

  0x00, 0x03, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0xe0, 0x00, 0x07, 0x7f,
0xff,

  0xfe, 0x3f, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0xc3, 0x0c, 0x30, 0xc3, 0x0c,

  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x86, 0x18, 0x61, 0x86, 0x18, 0x61, 0x00, 0x00, 0x00,
0x00,

  0x00, 0x00, 0x30, 0xc3, 0x0c, 0x30, 0xc3, 0x0c

};

// 'Rain', 24x24px

const unsigned char Rain_Sym [] PROGMEM = {

  0x01, 0xf0, 0x00, 0x03, 0xf8, 0x70, 0x06, 0x0c, 0xf8, 0x1c, 0x07, 0x8c, 0x38, 0x03, 0x06,
0x60,

  0x00, 0x03, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0xe0, 0x00, 0x07, 0x7f,
0xff,

  0xfe, 0x3f, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x84, 0x21, 0x21, 0x08,
0x42,

  0x42, 0x10, 0x84, 0x84, 0x21, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x84, 0x21,
0x21,

  0x08, 0x42, 0x42, 0x10, 0x84, 0x84, 0x21, 0x08

};

// 'Snow', 24x24px

const unsigned char Snow_Sym [] PROGMEM = {

  0x01, 0xf0, 0x00, 0x03, 0x18, 0x70, 0x06, 0x0c, 0xf8, 0x1c, 0x07, 0x8c, 0x38, 0x03, 0x06,
0x60,
```

```
  0x00, 0x03, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0xe0, 0x00, 0x07, 0x7f,
0xff,

  0xfe, 0x3f, 0xff, 0xfc, 0x00, 0x00, 0x00, 0xa0, 0x50, 0x20, 0x40, 0x20, 0xa8, 0xa0, 0x50, 0x70,

  0x00, 0x01, 0xfc, 0x04, 0x00, 0x70, 0x15, 0x00, 0xa8, 0x0e, 0x00, 0x20, 0x3f, 0x80, 0x00,
0x0e,

  0x0a, 0x05, 0x15, 0x04, 0x02, 0x04, 0x0a, 0x05

};

// 'Mist', 24x24px

const unsigned char Mist_Sym [] PROGMEM = {

  0x01, 0xf0, 0x00, 0x03, 0x18, 0x70, 0x06, 0x0c, 0xf8, 0x1c, 0x07, 0x8c, 0x38, 0x03, 0x06,
0x60,

  0x00, 0x03, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0xe0, 0x00, 0x07, 0x7f,
0xff,

  0xfe, 0x3f, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0xc3, 0x0c, 0x79, 0xe7, 0x9e,

  0xcf, 0x3c, 0xf3, 0x86, 0x18, 0x61, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0xc3, 0x0c,
0x79,

  0xe7, 0x9e, 0xcf, 0x3c, 0xf3, 0x86, 0x18, 0x61

};

// 'Clouds_Daylight', 24x24px

const unsigned char Clouds_Daylight_Sym [] PROGMEM = {

  0x00, 0x18, 0x00, 0x00, 0x18, 0x00, 0x10, 0x18, 0x08, 0x38, 0x00, 0x1c, 0x1c, 0x3c, 0x38,
0x08,

  0xff, 0x10, 0x01, 0xc3, 0x80, 0x03, 0x00, 0xc0, 0x06, 0x00, 0x60, 0x06, 0x00, 0x60, 0x0c,
0x00,

  0x30, 0xec, 0x00, 0x37, 0xed, 0xf0, 0x37, 0x0f, 0xf8, 0x70, 0x06, 0x0c, 0xf8, 0x1c, 0x07, 0x8c,

  0x38, 0x03, 0x06, 0x60, 0x00, 0x03, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03, 0xc0, 0x00, 0x03,
0xe0,

  0x00, 0x07, 0x7f, 0xff, 0xfe, 0x3f, 0xff, 0xfc

};

// 'Clouds_Night_Sym', 24x24px
```

```
const unsigned char Clouds_Night_Sym [] PROGMEM = {

  0x00, 0x00, 0x02, 0x00, 0x01, 0x07, 0x04, 0x01, 0xc2, 0x04, 0x01, 0xe0, 0x0e, 0x01, 0xb0,
0x3f,

  0x81, 0x98, 0x0e, 0x01, 0x8c, 0x04, 0x03, 0x0c, 0x04, 0x03, 0x06, 0x00, 0x03, 0x06, 0x00,
0x06,

  0x03, 0x00, 0x06, 0x03, 0x00, 0x0c, 0x03, 0x00, 0x38, 0x03, 0x01, 0xf0, 0x06, 0x7f, 0xc0,
0x06,

  0x3e, 0x00, 0x0c, 0x30, 0x00, 0x0c, 0x18, 0x00, 0x18, 0x0c, 0x00, 0x30, 0x07, 0x00, 0xe0,
0x43,

  0xc3, 0xc2, 0xe0, 0xff, 0x07, 0x40, 0x3c, 0x02

};

// 'Cloudy_Sym', 24x24px

const unsigned char Cloudy_Sym [] PROGMEM = {

  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x00, 0x01, 0xf0, 0xe0, 0x03, 0x19, 0xf0,
0x0e,

  0x0f, 0x1c, 0x1c, 0x06, 0x0e, 0x30, 0x00, 0x03, 0x60, 0x00, 0x03, 0x61, 0xc0, 0x03, 0x63,
0xe1,

  0xc3, 0x36, 0x33, 0xe6, 0x3c, 0x1e, 0x3c, 0x78, 0x0c, 0x1c, 0xe0, 0x00, 0x06, 0xc0, 0x00,
0x06,

  0xc0, 0x00, 0x06, 0xc0, 0x00, 0x06, 0xe0, 0x00, 0x0c, 0x7f, 0xff, 0xf8, 0x3f, 0xff, 0xf0, 0x00,

  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

char *mist[] = {"Mist", "Smoke", "Haze","Dust", "Fog",
"Sand","Dust","Ash","Squall","Tornado"};

//---------------------------------------

//---------------------------------------Variable declaration for calculating the connection time

int cnt_con = 0;

//---------------------------------------


int timezone_offset;
```

```
//=================================================================
=================Subroutines for the connection process
void Logo(){
  display.clearDisplay(); //--> for Clearing the display
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  display.println("   1306");
  display.setCursor(0, 33);
  display.println("weatherman");
  display.setTextSize(1);
  display.setCursor(0, 52);
  display.println("  Beware of weather");
  display.display();
  delay(5000);
  }
void Connect() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password); //--> Connect to your WiFi router
  Serial.println("");
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");


 //---------------------------------------Displays the text and icon of the connection process on OLED
```

```
    digitalWrite(ON_Board_LED, LOW); //--> Make the On Board Flashing LED on the process of connecting to the wifi router.

    display.clearDisplay(); //--> for Clearing the display

    display.setTextSize(1);

    display.setTextColor(WHITE);

    display.setCursor(20, 15);

    display.println("Connect to WiFi");

    display.drawBitmap(52, 25, Process1_Sym, 24, 24, WHITE); //--> display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

    display.drawRect(0, 0, 128, 64, WHITE); //--> Display frame

    display.display();

    delay(250);

       digitalWrite(ON_Board_LED, HIGH); //--> Make the Off Board Flashing LED on the process of connecting to the wifi router.

    display.clearDisplay(); //--> for Clearing the display

    display.setTextSize(1);

    display.setTextColor(WHITE);

    display.setCursor(20, 15);

    display.println("Connect to WiFi");

    display.drawBitmap(52, 25, Process2_Sym, 24, 24, WHITE); //--> display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

    display.drawRect(0, 0, 128, 64, WHITE); //--> Display frame

    display.display();

    delay(250);

    cnt_con++;

    if(cnt_con > 59) {

      cnt_con = 0;

      Serial.println("Failed to connect !");
```

```
    display.clearDisplay(); //--> for Clearing the display
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0, 28);
    display.println(" Failed to connect !");
    display.drawRect(0, 0, 128, 64, WHITE); //--> Display frame
    display.display();
    delay(5000);
    Waiting_to_be_connected_again();
   }
   //----------------------------------------
  }
  //----------------------------------------
  digitalWrite(ON_Board_LED, HIGH); //--> Turn off the On Board LED when it is connected to the wifi router.
  //----------------------------------------If successfully connected to the wifi router, the IP Address that will be visited is displayed in the serial monitor
  Serial.println("");
  Serial.print("Successfully connected to : ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  display.clearDisplay(); //--> for Clearing the display
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 24);
```

```
  display.println("     Connection");
  display.setCursor(0, 33);
  display.println("     Successful");
  display.drawRect(0, 0, 128, 64, WHITE); //--> Display frame
  display.display();
  delay(2000);
  //---------------------------------------
}
//=====================================================================
=================
//=====================================================================
=================Subroutine if connection is lost
void Connection_lost() {
  Serial.println("WiFi Disconnected");
  display.clearDisplay(); //--> for Clearing the display
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 28);
  display.println("  Connection lost !");
  display.drawRect(0, 0, 128, 64, WHITE); //--> Display frame
  display.display();
  delay(5000);
  Connect();
}
//=====================================================================
=================
```

```
//================================================================================Subroutines waiting to be reconnected
void Waiting_to_be_connected_again() {
  for(int i = 30; i > -1; i--) {
    Serial.print("Will be connected again:");
    Serial.println(i);
    display.clearDisplay(); //--> for Clearing the display
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0, 24);
    display.println("       Will be");
    display.setCursor(0, 33);
    display.print("  connected again:");
    display.println(i);
    display.drawRect(0, 0, 128, 64, WHITE); //--> Display frame
    display.display();
    delay(1000);
  }
  Connect();
}
//================================================================================

//================================================================================setup()
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  delay(500);
```

```
  //----------------------------------------SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
  // Address 0x3C for 128x32 and Address 0x3D for 128x64.
  // But on my 128x64 module the 0x3D address doesn't work. What works is the 0x3C address.
  // So please try which address works on your module.
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); //--> Don't proceed, loop forever
  }
  //----------------------------------------
    //----------------------------------------Show initial display buffer contents on the screen
  // the library initializes this with an Adafruit splash screen.
  // Show the display buffer on the screen(Update screen). You MUST call display() after
  // drawing commands to make them visible on screen!
  display.display();
  delay(2000);
  //----------------------------------------
     pinMode(ON_Board_LED,OUTPUT); //--> On Board LED port Direction output
  digitalWrite(ON_Board_LED, HIGH); //--> Turn off Led On Board
  Logo();
  Connect();
  timeClient.begin(); //--> Initialize a NTPClient to get time
  display.clearDisplay(); //--> for Clearing the display
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(3, 19);
```

```
  display.println("  Get weather data");
  display.setCursor(3, 28);
  display.println("        from");
  display.setCursor(3, 37);
  display.println("   openweathermap");
  display.drawRect(0, 0, 128, 64, WHITE); //--> Display frame
  display.display();
  delay(1000);
  Get_Weather_Data();
   Serial.println("Data will be updated every 5 minute.");
}
void loop() {
  if(WiFi.status()== WL_CONNECTED){
    //_____________________________________Millis to display time, date and weather data
    unsigned long currentMillisGetTimeDate = millis();
    if (currentMillisGetTimeDate - previousMillisGetTimeDate >= intervalGetTimeDate) {
      previousMillisGetTimeDate = currentMillisGetTimeDate;
           now = timelib_get(); //--> Get the timestamp from the library
           timelib_break(now, &tinfo); //--> Convert to human readable format
      //---------------------------------------Processing time and date data
      char timeProcess[8];
      sprintf(timeProcess, "%02u:%02u:%02u", tinfo.tm_hour, tinfo.tm_min, tinfo.tm_sec);
      char _time[8];
      strcpy(_time,timeProcess);
      char dateDayProcess[2];
      sprintf(dateDayProcess, "%02u", tinfo.tm_mday);
```

```
 char _dateDay[2];
 strcpy(_dateDay,dateDayProcess);
 _DateTime = _dateDay;
 _DateTime += "-" + currentMonthName + "-" + String(currentYear) + " " + _time;
 //----------------------------------------
//----------------------------------------Conditions for updating weather data, time and date.
 if(Upadate_Trig == 1 && tinfo.tm_min % 15 == 0) {
   display.clearDisplay(); //--> for Clearing the display
  display.drawRect(0, 0, 128, 64, WHITE); //--> Display frame
 Display_current_weather_data();
 display.drawRect(0, 0, 128, 64, WHITE);
   display.setTextSize(1);
   display.setTextColor(WHITE);
   display.setCursor(4, 52); //--> (x position, y position)
   display.println("  Updating Data...");
   display.drawLine(0, 47, 127, 47, WHITE);
   display.display();
   Get_Weather_Data();
   Serial.println("Set the Date and Time from the Internet...");
   Set_Time_and_Date();
  Upadate_Trig = 0;
 }
  if(tinfo.tm_min % 15 != 0) {
   Upadate_Trig = 1;
 }
 //----------------------------------------
```

```
      //---------------------------------------Conditions for displaying more detailed weather data
     if(tinfo.tm_sec > 20 && tinfo.tm_sec < 26) {
      Display_weather_description();
     } else if(tinfo.tm_sec > 25 && tinfo.tm_sec < 36) {
      Display_other_information_1();
     } else if(tinfo.tm_sec > 35 && tinfo.tm_sec < 46) {
      Display_other_information_2();
     } else if(tinfo.tm_sec > 45 && tinfo.tm_sec < 56) {
      Display_weather_forecast_data();
     } else {
      display.clearDisplay(); //--> for Clearing the display
      display.drawRect(0, 0, 128, 64, WHITE); //--> Display frame
      Display_current_weather_data();
      Display_current_DateTime();
      display.drawLine(0, 47, 127, 47, WHITE);
     display.display();
     }
     //---------------------------------------
   }
   //_____________________________________
  } else {
   Connection_lost();
  }
}
void Get_Weather_Data() {
  if(WiFi.status()== WL_CONNECTED){
```

```
  //_____________________________________Get longitude, latitude and timezone offset

  //---------------------------------------If using a city name

  String current_serverPath = "http://api.openweathermap.org/data/2.5/weather?q=" + city + "," + countryCode + "&units=" + units + "&APPID=" + openWeatherMapApiKey;

  //---------------------------------------

  //---------------------------------------If using a city ID

  //String current_serverPath = "http://api.openweathermap.org/data/2.5/weather?id=" + city + "&units=" + units + "&APPID=" + openWeatherMapApiKey;

  //---------------------------------------

  Serial.println("--------------------");

  Serial.println("Get weather data from openweathermap...");

  Serial.println();

  strjsonBuffer = httpGETRequest(current_serverPath.c_str());

  //Serial.println(strjsonBuffer);

  JsonObject& root = jsonBuffer.parseObject(strjsonBuffer);

   // Test if parsing succeeds.

  if (!root.success()) {

    Serial.println("parseObject() failed");

    display.clearDisplay(); //--> for Clearing the display

    display.setTextSize(1);

    display.setTextColor(WHITE);

    display.setCursor(0, 19);

    display.println("    Failed to get");

    display.setCursor(0, 28);

    display.println("    weather data !");

    display.setCursor(4, 37);

    display.println("parseObject() failed");
```

```
    display.drawRect(0, 0, 128, 64, WHITE); //--> Display frame
    display.display();
    delay(5000);
    return;
  }
  String lon = root["coord"]["lon"];
  String lat = root["coord"]["lat"];
  timezone_offset = root["timezone"];
  Serial.println();
  Serial.print("Lon : ");
  Serial.println(lon);
  Serial.print("Lat : ");
  Serial.println(lat);
  Serial.print("Timezone offset : ");
  Serial.println(timezone_offset);
  Serial.println();
  jsonBuffer.clear();
  Serial.println("Set the Date and Time from the Internet...");
  Set_Time_and_Date();
  //_____________________________________
  //_____________________________________Get current weather data and weather forecast data for the following days
  //---------------------------------------Get current weather data
  String current_forecast_serverPath = "http://api.openweathermap.org/data/2.5/onecall?lat=" + lat + "&lon=" + lon + "&units=" + units + "&exclude=hourly&APPID=" + openWeatherMapApiKey;
 strjsonBufferCF = httpGETRequest(current_forecast_serverPath.c_str());
  //Serial.println(strjsonBuffer);
```

```
JsonObject& rootCF = jsonBuffer.parseObject(strjsonBufferCF);
// Test if parsing succeeds.
if (!rootCF.success()) {
  Serial.println("parseObject() failed");
  display.clearDisplay(); //--> for Clearing the display
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 19);
  display.println("    Failed to get");
  display.setCursor(0, 28);
  display.println("    weather data !");
  display.setCursor(4, 37);
  display.println("parseObject() failed");
  display.drawRect(0, 0, 128, 64, WHITE); //--> Display frame
  display.display();
  delay(5000);
  return;
}
String str_current_weather = rootCF["current"]["weather"][0]["main"];
current_weather = str_current_weather;
String str_current_weather_Description = rootCF["current"]["weather"][0]["description"];
current_weather_Description = str_current_weather_Description;
String str_current_weather_sym = rootCF["current"]["weather"][0]["icon"];
current_weather_sym = str_current_weather_sym;
current_temperature = rootCF["current"]["temp"];
current_feels_like = rootCF["current"]["feels_like"];
```

```
current_uv = rootCF["current"]["uvi"];
current_dew_point = rootCF["current"]["dew_point"];
current_pressure = rootCF["current"]["pressure"];
current_humidity = rootCF["current"]["humidity"];
current_visibility = rootCF["current"]["visibility"];
current_wind_speed = rootCF["current"]["wind_speed"];
current_wind_deg = rootCF["current"]["wind_deg"];
//---------------------------------------
//---------------------------------------Get weather forecast data for the following days
String str_forecast_weather1 = rootCF["daily"][0]["weather"][0]["main"];
forecast_weather[0] = str_forecast_weather1;
String str_forecast_weather_sym1 = rootCF["daily"][0]["weather"][0]["icon"];
forecast_weather_sym[0] = str_forecast_weather_sym1;
forecast_temp_max[0] = rootCF["daily"][0]["temp"]["max"];
forecast_temp_min[0] = rootCF["daily"][0]["temp"]["min"];
String str_forecast_weather2 = rootCF["daily"][1]["weather"][0]["main"];
forecast_weather[1] = str_forecast_weather2;
String str_forecast_weather_sym2 = rootCF["daily"][1]["weather"][0]["icon"];
forecast_weather_sym[1] = str_forecast_weather_sym2;
forecast_temp_max[1] = rootCF["daily"][1]["temp"]["max"];
forecast_temp_min[1] = rootCF["daily"][1]["temp"]["min"];
String str_forecast_weather3 = rootCF["daily"][2]["weather"][0]["main"];
forecast_weather[2] = str_forecast_weather3;
String str_forecast_weather_sym3 = rootCF["daily"][2]["weather"][0]["icon"];
forecast_weather_sym[2] = str_forecast_weather_sym3;
forecast_temp_max[2] = rootCF["daily"][2]["temp"]["max"];
```

```
forecast_temp_min[2] = rootCF["daily"][2]["temp"]["min"];
String str_forecast_weather4 = rootCF["daily"][3]["weather"][0]["main"];
forecast_weather[3] = str_forecast_weather4;
String str_forecast_weather_sym4 = rootCF["daily"][3]["weather"][0]["icon"];
forecast_weather_sym[3] = str_forecast_weather_sym4;
forecast_temp_max[3] = rootCF["daily"][3]["temp"]["max"];
forecast_temp_min[3] = rootCF["daily"][3]["temp"]["min"];
//---------------------------------------
//---------------------------------------
Serial.println();
Serial.println("Current weather data");
Serial.print("weather : ");
Serial.println(current_weather);
Serial.print("weather description : ");
Serial.println(current_weather_Description);
Serial.print("weather symbol : ");
Serial.println(current_weather_sym);
Serial.print("temperature : ");
Serial.print(current_temperature);
Serial.println(" °C");
Serial.print("feels_like : ");
Serial.print(current_feels_like);
Serial.print(" °C");
Serial.print(" (");
Serial.print(round(current_feels_like));
Serial.print(" °C");
```

```
Serial.println(")");
Serial.print("UV : ");
Serial.print(current_uv);
Serial.print(" (");
Serial.print(round(current_uv));
Serial.println(")");
Serial.print("Dew point : ");
Serial.print(current_dew_point);
Serial.print(" °C");
Serial.print(" (");
Serial.print(round(current_dew_point));
Serial.print(" °C");
Serial.println(")");
Serial.print("pressure : ");
Serial.print(current_pressure);
Serial.println(" hPa");
Serial.print("humidity : ");
Serial.print(current_humidity);
Serial.println(" %");
Serial.print("visibility : ");
Serial.print(current_visibility);
Serial.print(" m");
Serial.print(" (");
float current_visibility_to_km = current_visibility / 1000;
Serial.print(current_visibility_to_km);
Serial.print(" km");
```

```
  Serial.println(")");
  Serial.print("wind_speed : ");
  Serial.print(current_wind_speed);
  Serial.println(" m/s");
  Serial.print("wind_deg : ");
  Serial.print(current_wind_deg);
  Serial.println("°");
  //----------------------------------------
  //----------------------------------------
  int wds[4] =
{timeClient.getDay(),timeClient.getDay()+1,timeClient.getDay()+2,timeClient.getDay()+3};
  for(int i = 0; i < 4; i++) {
    if(wds[i] == 7) {
      wds[i] = 0;
    }
    if(wds[i] == 8) {
      wds[i] = 1;
    }
    if(wds[i] == 9) {
      wds[i] = 2;
    }
  }
  String wd;
  //----------------------------------------
  //----------------------------------------
  wd = weekDays[wds[0]];
  Serial.println();
```

```
Serial.print("Weather forecast data : ");
Serial.println(wd);
Serial.print("weather : ");
Serial.println(forecast_weather[0]);
Serial.print("weather_sym : ");
Serial.println(forecast_weather_sym[0]);
Serial.print("temp_max : ");
Serial.print(forecast_temp_max[0]);
Serial.print(" °C");
Serial.print(" (");
Serial.print(round(forecast_temp_max[0]));
Serial.print(" °C");
Serial.println(")");
Serial.print("temp_min : ");
Serial.print(forecast_temp_min[0]);
Serial.print(" °C");
Serial.print(" (");
Serial.print(round(forecast_temp_min[0]));
Serial.print(" °C");
Serial.println(")");
Serial.println();
wd = weekDays[wds[1]];
Serial.println();
Serial.print("Weather forecast data : ");
Serial.println(wd);
Serial.print("weather : ");
```

```
Serial.println(forecast_weather[1]);
Serial.print("weather_sym : ");
Serial.println(forecast_weather_sym[1]);
Serial.print("temp_max : ");
Serial.print(forecast_temp_max[1]);
Serial.print(" °C");
Serial.print(" (");
Serial.print(round(forecast_temp_max[1]));
Serial.print(" °C");
Serial.println(")");
Serial.print("temp_min : ");
Serial.print(forecast_temp_min[1]);
Serial.print(" °C");
Serial.print(" (");
Serial.print(round(forecast_temp_min[1]));
Serial.print(" °C");
Serial.println(")");
Serial.println();
wd = weekDays[wds[2]];
Serial.println();
Serial.print("Weather forecast data : ");
Serial.println(wd);
Serial.print("weather : ");
Serial.println(forecast_weather[2]);
Serial.print("weather_sym : ");
Serial.println(forecast_weather_sym[2]);
```

```
Serial.print("temp_max : ");
Serial.print(forecast_temp_max[2]);
Serial.print(" °C");
Serial.print(" (");
Serial.print(round(forecast_temp_max[2]));
Serial.print(" °C");
Serial.println(")");
Serial.print("temp_min : ");
Serial.print(forecast_temp_min[2]);
Serial.print(" °C");
Serial.print(" (");
Serial.print(round(forecast_temp_min[2]));
Serial.print(" °C");
Serial.println(")");
wd = weekDays[wds[3]];
Serial.println();
Serial.print("Weather forecast data : ");
Serial.println(wd);
Serial.print("weather : ");
Serial.println(forecast_weather[3]);
Serial.print("weather_sym : ");
Serial.println(forecast_weather_sym[3]);
Serial.print("temp_max : ");
Serial.print(forecast_temp_max[3]);
Serial.print(" °C");
Serial.print(" (");
```

```
    Serial.print(round(forecast_temp_max[3]));
    Serial.print(" °C");
    Serial.println(")");
    Serial.print("temp_min : ");
    Serial.print(forecast_temp_min[3]);
    Serial.print(" °C");
    Serial.print(" (");
    Serial.print(round(forecast_temp_min[3]));
    Serial.print(" °C");
    Serial.println(")");
    Serial.println("---------------------");
    Serial.println();
    //---------------------------------------
    jsonBuffer.clear();
    Serial.println("Set the Date and Time again from the Internet...");
    Set_Time_and_Date();
    //_____________________________________
  }
  else {
    Serial.println("WiFi Disconnected");
  }
}
//====================================================================
=================

//====================================================================
=================httpGETRequest
String httpGETRequest(const char* serverName) {
```

```
HTTPClient http;
// Your IP address with path or Domain name with URL path
http.begin(wifiClient,serverName);
// Send HTTP POST request
int httpResponseCode = http.GET();
String payload = "{}";
if (httpResponseCode == 200) {
  Serial.print("HTTP Response code: ");
  Serial.println(httpResponseCode);
  payload = http.getString();
}
else {
  Serial.print("Error code: ");
  Serial.println(httpResponseCode);
  display.clearDisplay(); //--> for Clearing the display
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 19);
  display.println("    Failed to get");
  display.setCursor(0, 28);
  display.println("    weather data !");
  display.setCursor(0, 37);
  display.print("  HTTP Response: ");
  display.println(httpResponseCode);
  display.drawRect(0, 0, 128, 64, WHITE); //--> Display frame
  display.display();
```

```
   delay(5000);
  }
  // Free resources
  http.end();
  return payload;
}
//=====================================================================
=================
//=====================================================================
=================Subroutines for setting the time and date from the internet
void Set_Time_and_Date() {
  timeClient.setTimeOffset(timezone_offset);
  // Send an HTTP GET request
  timeClient.update();
  epochTime = timeClient.getEpochTime();
  currentHour = timeClient.getHours();
  currentMinute = timeClient.getMinutes();
  currentSecond = timeClient.getSeconds();
  weekDay = weekDays[timeClient.getDay()];
  //Get a time structure
  struct tm *ptm = gmtime ((time_t *)&epochTime);
  monthDay = ptm->tm_mday;
  currentMonth = ptm->tm_mon+1;
  currentMonthName = months[currentMonth-1];
  currentYear = ptm->tm_year+1900;
  currentYearforSet = currentYear - 2000;
```

```
  // Set time manually to 13:55:30 Jan 1st 2014
  // YOU CAN SET THE TIME FOR THIS EXAMPLE HERE
  tinfo.tm_year = currentYearforSet;
  tinfo.tm_mon = currentMonth;
  tinfo.tm_mday = monthDay;
  tinfo.tm_hour = currentHour;
  tinfo.tm_min = currentMinute;
  tinfo.tm_sec = currentSecond;
  // Convert time structure to timestamp
  initialt = timelib_make(&tinfo);
  // Set system time counter
  timelib_set(initialt);
  // Configure the library to update / sync every day (for hardware RTC)
  //timelib_set_provider(time_provider, TIMELIB_SECS_PER_DAY);
  Serial.println("Setting Date and Time from Internet is successful.");
}
//================================================================================
void Display_current_DateTime() {
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(4, 52); //--> (x position, y position)
  display.println(_DateTime);
}
//================================================================================
```

```
//=================================================================================Subroutines for displaying weather data

void Display_current_weather_data() {

  //---------------------------------------Weather icon selection conditions

  if (current_weather == "Thunderstorm") {

    display.drawBitmap(10, 5, Thunderstorm_Sym, 24, 24, WHITE); //--> display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

  }

  else if (current_weather == "Drizzle") {

    display.drawBitmap(10, 5, Drizzle_Sym, 24, 24, WHITE); //--> display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

  }

  else if (current_weather == "Rain") {

    display.drawBitmap(10, 5, Rain_Sym, 24, 24, WHITE); //--> display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

  }

  else if (current_weather == "Snow") {

    display.drawBitmap(10, 5, Snow_Sym, 24, 24, WHITE); //--> display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

  }

  else if (current_weather == "Clear") {

    if (current_weather_sym == "01d") {

      display.drawBitmap(10, 5, Clear_Daylight_Sym, 24, 24, WHITE); //--> display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

    } else if (current_weather_sym == "01n") {

      display.drawBitmap(10, 5, Clear_Night_Sym, 24, 24, WHITE); //--> display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

    }
```

```
  }

  else if (current_weather == "Clouds") {

    if (current_weather_sym == "02d") {

      display.drawBitmap(10, 5, Clouds_Daylight_Sym, 24, 24, WHITE); //-->
display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

    } else if (current_weather_sym == "02d") {

      display.drawBitmap(10, 5, Clouds_Night_Sym, 24, 24, WHITE); //--> display.drawBitmap(x
position, y position, bitmap data, bitmap width, bitmap height, color);

    } else {

      display.drawBitmap(10, 5, Cloudy_Sym, 24, 24, WHITE); //--> display.drawBitmap(x
position, y position, bitmap data, bitmap width, bitmap height, color);

    }

  }

  for (int i = 0; i < 10; i++) {

    if (current_weather == mist[i]) {

      display.drawBitmap(10, 5, Mist_Sym, 24, 24, WHITE); //--> display.drawBitmap(x position,
y position, bitmap data, bitmap width, bitmap height, color);

    }

  }

  //----------------------------------------

  //----------------------------------------Displays weather data, temperature, humidity and wind speed

  display.setTextSize(1);

  display.setTextColor(WHITE);

  display.setCursor(50, 5); //--> (x position, y position)

  display.println(current_weather);

  display.setCursor(50, 15); //--> (x position, y position)

  String Hum = "H:" + String(current_humidity) + "%";

  display.println(Hum);
```

```
  display.setCursor(50, 25); //--> (x position, y position)
  String WS = "WS:" + String(current_wind_speed) + "m/s";
  display.println(WS);
  display.setCursor(50, 35); //--> (x position, y position)
  display.println(weekDay);
  display.setCursor(4, 35); //--> (x position, y position)
  display.print("T:");
  display.print(current_temperature);
  display.print((char)247); //--> ASCII degree symbol
  display.print("C");
  //---------------------------------------
}
//================================================================================
//================================================================================Subroutines to display data for the next few days
void Display_weather_forecast_data() {
  int forecast_X1[] = {20,90};
  int forecast_X2[] = {17,87};
  int forecast_X3[] = {8,79};
  int wds[4] =
{timeClient.getDay(),timeClient.getDay()+1,timeClient.getDay()+2,timeClient.getDay()+3};
  int round_forecast_temp_max[2];
  int round_forecast_temp_min[2];
  for(int i = 0; i < 4; i++) {
    if(wds[i] == 7) {
      wds[i] = 0;
```

```
  }
  if(wds[i] == 8) {
    wds[i] = 1;
  }
  if(wds[i] == 9) {
    wds[i] = 2;
  }
}
String forecast_weather_Prcs[2];
String forecast_weather_sym_Prcs[2];
int wds_Prcs[2];
if(tinfo.tm_sec > 45 && tinfo.tm_sec < 51) {
  forecast_weather_Prcs[0] = forecast_weather[0];
  forecast_weather_Prcs[1] = forecast_weather[1];
  forecast_weather_sym_Prcs[0] = forecast_weather_sym[0];
  forecast_weather_sym_Prcs[1] = forecast_weather_sym[1];
  wds_Prcs[0] = wds[0];
  wds_Prcs[1] = wds[1];
  round_forecast_temp_max[0] = round(forecast_temp_max[0]);
  round_forecast_temp_min[0] = round(forecast_temp_min[0]);
  round_forecast_temp_max[1] = round(forecast_temp_max[1]);
  round_forecast_temp_min[1] = round(forecast_temp_min[1]);
} else {
  forecast_weather_Prcs[0] = forecast_weather[2];
  forecast_weather_Prcs[1] = forecast_weather[3];
  forecast_weather_sym_Prcs[0] = forecast_weather_sym[2];
```

```
  forecast_weather_sym_Prcs[1] = forecast_weather_sym[3];
  wds_Prcs[0] = wds[2];
  wds_Prcs[1] = wds[3];
  round_forecast_temp_max[0] = round(forecast_temp_max[2]);
  round_forecast_temp_min[0] = round(forecast_temp_min[2]);
  round_forecast_temp_max[1] = round(forecast_temp_max[3]);
  round_forecast_temp_min[1] = round(forecast_temp_min[3]);
 }
 String wd;
 String WF, WSF;
 display.clearDisplay();
 display.setTextSize(1);
 display.setTextColor(WHITE);
 //---------------------------------------Process data for display
 for(int i = 0; i < 2; i++) {
  wd = weekDays3Dgt[wds_Prcs[i]];
  display.setCursor(forecast_X1[i], 0);
  display.print(wd);
  WF = forecast_weather_Prcs[i];
  WSF = forecast_weather_sym_Prcs[i];
  if (WF == "Thunderstorm") {
    display.drawBitmap(forecast_X2[i], 9, Thunderstorm_Sym, 24, 24, WHITE); //-->
display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);
  }
  else if (WF == "Drizzle") {
    display.drawBitmap(forecast_X2[i], 9, Drizzle_Sym, 24, 24, WHITE); //-->
display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);
```

```
    }

    else if (WF == "Rain") {

      display.drawBitmap(forecast_X2[i], 9, Rain_Sym, 24, 24, WHITE); //-->
display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

    }

    else if (WF == "Snow") {

      display.drawBitmap(forecast_X2[i], 9, Snow_Sym, 24, 24, WHITE); //-->
display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

    }

    else if (WF == "Clear") {

     if (WSF == "01d") {

       display.drawBitmap(forecast_X2[i], 9, Clear_Daylight_Sym, 24, 24, WHITE); //-->
display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

     } else if (WSF == "01n") {

       display.drawBitmap(forecast_X2[i], 9, Clear_Night_Sym, 24, 24, WHITE); //-->
display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

     }

    }

    else if (WF == "Clouds") {

     if (WSF == "02d") {

       display.drawBitmap(forecast_X2[i], 9, Clouds_Daylight_Sym, 24, 24, WHITE); //-->
display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

     } else if (WSF == "02d") {

       display.drawBitmap(forecast_X2[i], 9, Clouds_Night_Sym, 24, 24, WHITE); //-->
display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

     } else {

       display.drawBitmap(forecast_X2[i], 9, Cloudy_Sym, 24, 24, WHITE); //-->
display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

     }
```

```
    }

    for (int i = 0; i < 10; i++) {

      if (WF == mist[i]) {

        display.drawBitmap(forecast_X2[i], 9, Mist_Sym, 24, 24, WHITE); //-->
display.drawBitmap(x position, y position, bitmap data, bitmap width, bitmap height, color);

      }

    }

    String forecast_temp_max_min = String(round_forecast_temp_max[i]) + "/" +
round_forecast_temp_min[i];

    if(forecast_temp_max_min.length() == 6) {

      forecast_X3[i] += 3;

    } else if(forecast_temp_max_min.length() == 5) {

      forecast_X3[i] += 6;

    } else if(forecast_temp_max_min.length() == 4) {

      forecast_X3[i] += 9;

    } else if(forecast_temp_max_min.length() == 3) {

      forecast_X3[i] += 12;

    }

    display.setCursor(forecast_X3[i], 36);

    display.print(forecast_temp_max_min);

  }

  //---------------------------------------

  display.drawRect(0, 47, 128, 17, WHITE); //--> Display frame

  Display_current_DateTime();

 display.display();

}

//================================================================
================
```

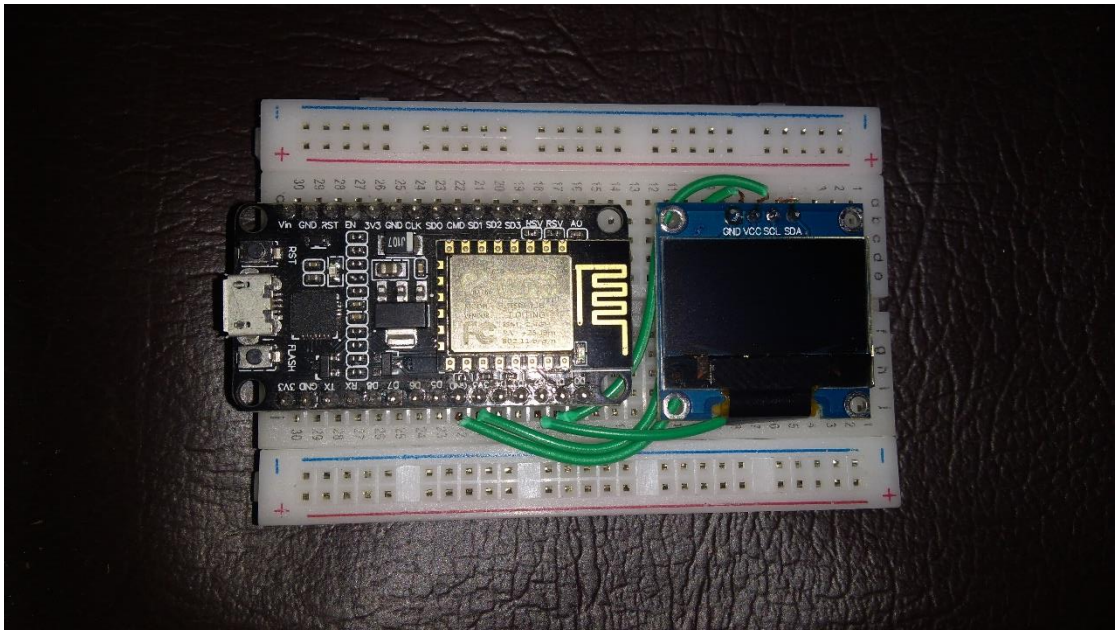
```
//=====================================================================
=================Show_weather_description()
void Display_weather_description() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0); //--> (x position, y position)
  display.println("Weather description :");
  display.setTextSize(2);
  display.setCursor(0, 15); //--> (x position, y position)
  display.println(current_weather_Description);
  display.drawRect(0, 47, 128, 17, WHITE); //--> Display frame
 Display_current_DateTime();
  display.display();
}
//=====================================================================
================

//=====================================================================
=================Displays_other_information_1()
void Display_other_information_1() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  int round_current_feels_like = round(current_feels_like);
  String Show_current_feels_like = "Feels Like :" + String(round_current_feels_like);
  display.setCursor(0, 0); //--> (x position, y position)
  display.print(Show_current_feels_like);
  display.print((char)247); //--> ASCII degree symbol
```

```
  display.print("C");
  int round_current_uv = round(current_uv);
  String Show_current_uv = "UV         :" + String(round_current_uv);
  display.setCursor(0, 10); //--> (x position, y position)
  display.println(Show_current_uv);
  int round_current_dew_point = round(current_dew_point);
  String Show_current_dew_point = "Dew Point  :" + String(round_current_dew_point);
  display.setCursor(0, 20); //--> (x position, y position)
  display.print(Show_current_dew_point);
  display.print((char)247); //--> ASCII degree symbol
  display.print("C");
  display.drawRect(0, 47, 128, 17, WHITE); //--> Display frame
  Display_current_DateTime();
 display.display();
}
//=================================================================
=================
//=================================================================
================Displays_other_information_2()
void Display_other_information_2() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  String Show_current_pressure = "Pressure    :" + String(current_pressure) + "hPa";
  display.setCursor(0, 0); //--> (x position, y position)
  display.println(Show_current_pressure);
```

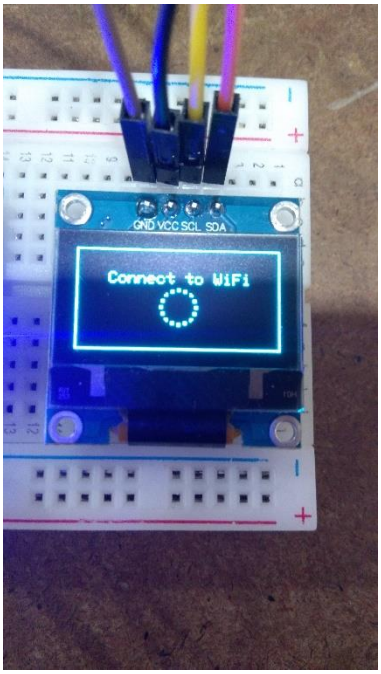
```
float current_visibility_to_km = current_visibility/1000;
String Show_current_visibility = "visibility  :" + String(current_visibility_to_km) + "km";
display.setCursor(0, 10); //--> (x position, y position)
display.println(Show_current_visibility);
String Cardinal_Direction;
if(current_wind_deg > 347 || current_wind_deg < 12) {
  Cardinal_Direction = "N";
} else if(current_wind_deg > 11 && current_wind_deg < 34) {
  Cardinal_Direction = "NNE";
} else if(current_wind_deg > 33 && current_wind_deg < 57) {
  Cardinal_Direction = "NE";
} else if(current_wind_deg > 56 && current_wind_deg < 79) {
  Cardinal_Direction = "ENE";
} else if(current_wind_deg > 78 && current_wind_deg < 102) {
  Cardinal_Direction = "E";
} else if(current_wind_deg > 101 && current_wind_deg < 124) {
  Cardinal_Direction = "ESE";
} else if(current_wind_deg > 123 && current_wind_deg < 147) {
  Cardinal_Direction = "SE";
} else if(current_wind_deg > 146 && current_wind_deg < 169) {
  Cardinal_Direction = "SSE";
} else if(current_wind_deg > 168 && current_wind_deg < 192) {
  Cardinal_Direction = "S";
} else if(current_wind_deg > 191 && current_wind_deg < 214) {
  Cardinal_Direction = "SSW";
} else if(current_wind_deg > 213 && current_wind_deg < 237) {
```

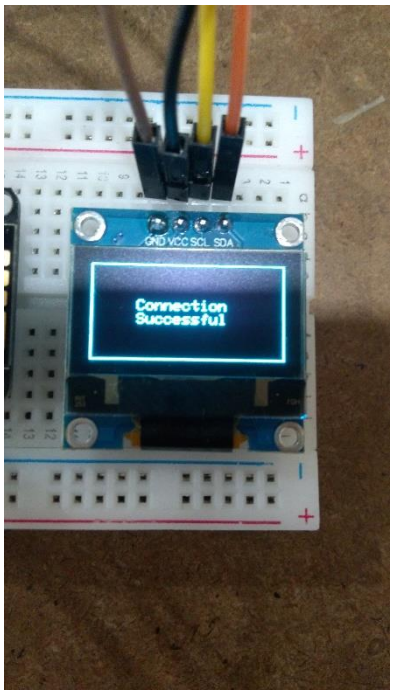
```
   Cardinal_Direction = "SW";
  } else if(current_wind_deg > 236 && current_wind_deg < 259) {
   Cardinal_Direction = "WSW";
  } else if(current_wind_deg > 258 && current_wind_deg < 282) {
   Cardinal_Direction = "W";
  } else if(current_wind_deg > 281 && current_wind_deg < 304) {
   Cardinal_Direction = "WNW";
  } else if(current_wind_deg > 303 && current_wind_deg < 327) {
   Cardinal_Direction = "NW";
  } else if(current_wind_deg > 326 && current_wind_deg < 347) {
   Cardinal_Direction = "NNW";
  }
  display.setCursor(0, 20); //--> (x position, y position)
  display.print("Wind Degree :");
  display.print(current_wind_deg);
  display.print((char)247); //--> ASCII degree symbol
  display.println(Cardinal_Direction);
  display.drawRect(0, 47, 128, 17, WHITE); //--> Display frame
  Display_current_DateTime();
 display.display();
}
```

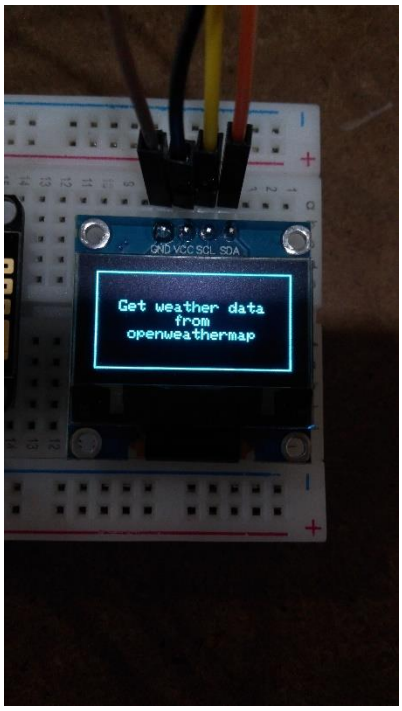
4. CIRCUIT AND WORKING MODULE

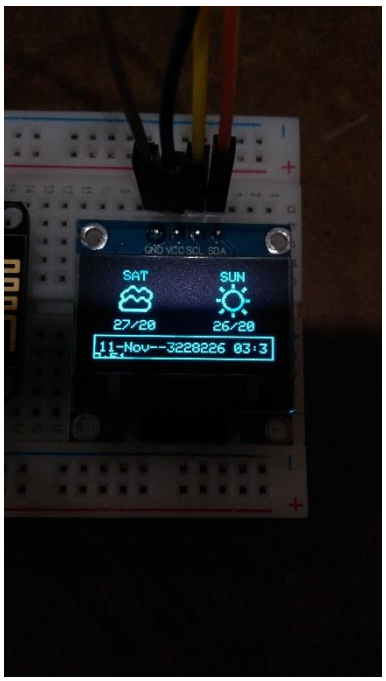


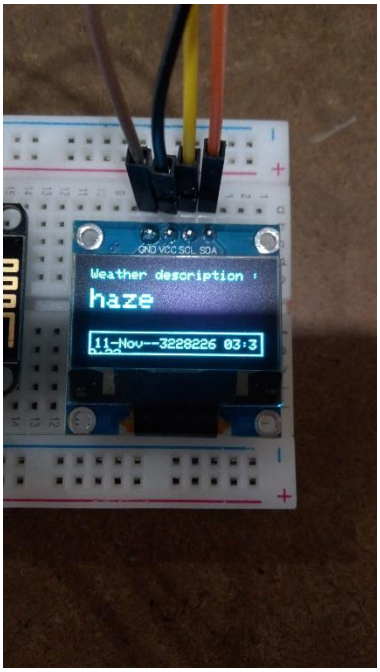
CIRCUIT DIAGRAM

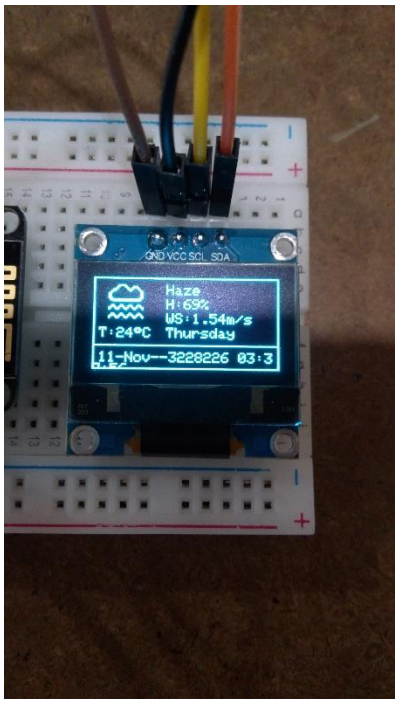


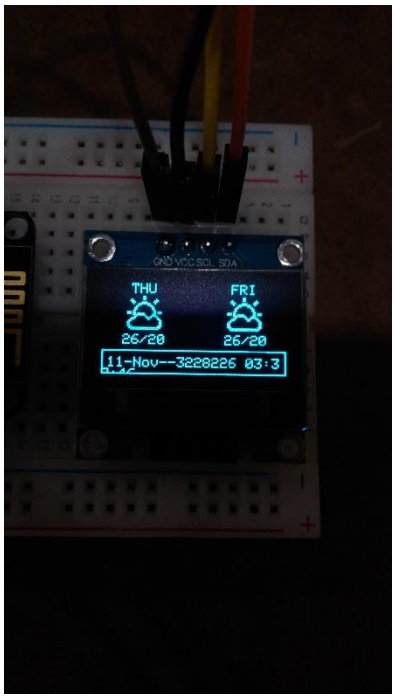

GND VCC SCL SDA
Connection
Successful


GND VCC SCL SDA
Get weather data
from
openweathermap


GND VCC SCL SDA
SAT
SUN
27/20
26/20
11-Nov--3228226 03:3


GND VCC SCL SDA
Weather description :
haze
11-Nov--3228226 03:3


GND VCC SCL SDA
Haze
H:69%
WS:1.54m/s
T:24°C Thursday
11-Nov--3228226 03:3


GND VCC SCL SDA
THU
FRI
26/20
26/20
11-Nov--3228226 03:3


GND VCC SCL SDA
Pressure :1018hPa
visibility :2.00km
Wind Degree :330°NNW
11-Nov--3228226 03:3


GND VCC SCL SDA
Feels Like :25°C
UV :6
Dew Point :19°C
11-Nov--3228226 03:3

5. RESULT

This presented system is configured for home and office uses. Here, we use OLED Display and Wi-Fi Module to get information about weather data. This tells us about the temperature, 4-day weather forecast and wind speed, visibility, dew point, UV radiation, pressure, wind direction, weather description. So, this can alert people about the weather condition and helps them to take precautionary measures according to the weather condition.