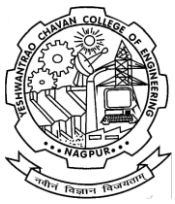

Nagar Yuwak Shikshan Sanstha's

Yeshwantrao Chavan College of Engineering

(An Autonomous Institution affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)

Hingna Road, Wanadongri, Nagpur

Department of Electronics Engineering

"Bitcoin Prize Tracker using ESP-32."

Project Group Member Names

1. (A-31) Akshat kaushik
2. (A-32) Aman Dudhpachare

IV Sem,
Section -A, IIOT

Abstract:

The project "Bitcoin Price Tracker" aims to develop a system using the ESP-32 microcontroller, an OLED display (0.96"), and additional components such as red and green LEDs and a 330 Ohm resistor. This project enables users to track and monitor the price of Bitcoin using a physical device.

By utilizing these components, the Bitcoin Price Tracker project provides a physical device that allows users to track and monitor the price of Bitcoin. The ESP-32 microcontroller processes the data and controls the OLED display and LED indicators to present the information to the user in a clear and intuitive manner. This project is suitable for individuals who prefer a tangible and visual representation of Bitcoin price movements and enables them to stay informed about cryptocurrency market trends.

Circuit and Working:

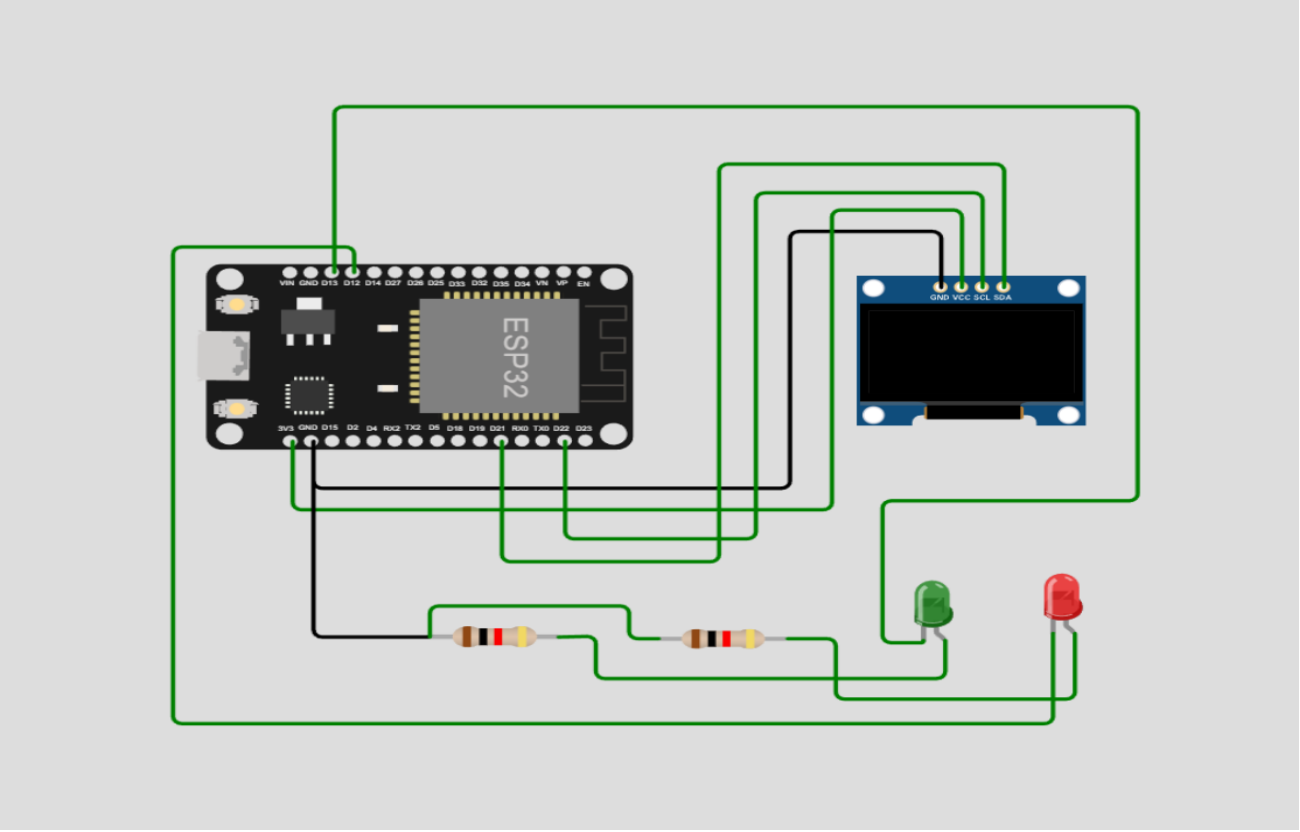

Fig. 1: project circuit diagram

Working:

To create a working Bitcoin Price Tracker project using the components mentioned (ESP-32, OLED Display 0.96", red and green LEDs, 330 Ohm resistor), you will need to follow the steps outlined below:

1. **Set up the Development Environment:**
   - Install the Arduino IDE (Integrated Development Environment) on your computer.
   - Add the ESP-32 board support to the Arduino IDE.
2. **Circuit Connection:**
   - Connect the ESP-32 to the OLED Display using the appropriate pins (e.g., SDA and SCL).
   - Connect the red and green LEDs to digital pins on the ESP-32, along with the 330 Ohm resistor in series with each LED to limit the current.
3. **Code Implementation:**
   - Write the Arduino code to fetch Bitcoin price data from a suitable API or data source.
   - Implement the code to process the price data and update the OLED Display accordingly.
   - Determine the conditions for the LEDs to light up (e.g., red LED for price decrease, green LED for price increase).
   - Code the LED indicators to reflect the price changes based on the retrieved data.
4. **Upload the Code:**
   - Connect the ESP-32 to your computer via USB.
   - Compile and upload the code from the Arduino IDE to the ESP-32.
5. **Test the Bitcoin Price Tracker:**
   - Power on the circuit and ensure that the ESP-32, OLED Display, and LEDs are functioning correctly.
   - Monitor the OLED Display to verify that it shows the current Bitcoin price and other relevant information.
   - Observe the LEDs to confirm that they indicate the price changes accurately.

With these steps, you can create a functional Bitcoin Price Tracker using the ESP-32, OLED Display 0.96", red and green LEDs, and the 330 Ohm resistor. Keep in mind that you may need to adapt the code and make adjustments to the circuit connections based on the specific pin configurations and libraries you use.

Part List:

1. **ESP -32 Development Board.**
2. **I2C OLED Display.**
3. **5mm Red & Green Display.**
4. **330 Ohm Resisters.**

Table 1: Components

Components	Part Name	Unit Price	Quantity	Total Price
ESP-32	ESP-32 Development board	450	1	450
OLED DISPLAY	0.96’ OLED DISPLAY	300	1	300
Resistors	330 Ohm resistor	5	2	10
LED’s	Red & Green Led	5	2	10
Total				770

Construction :

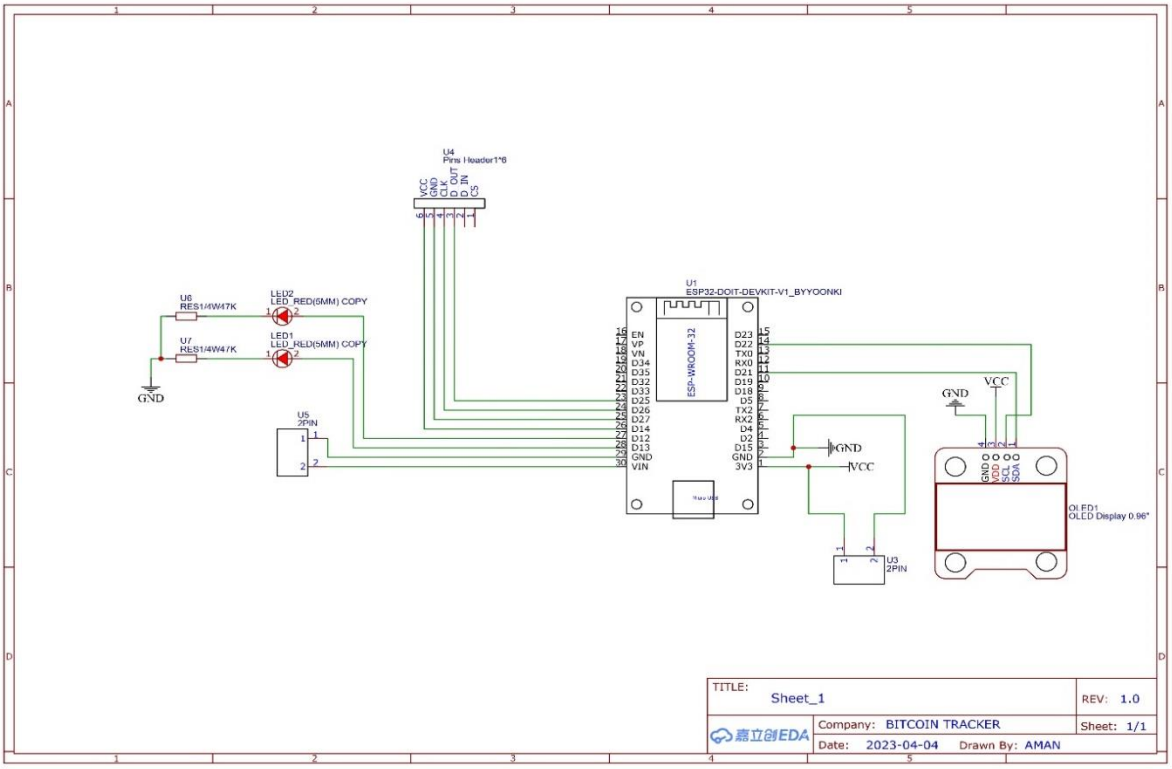

Fig. 2: Schematic

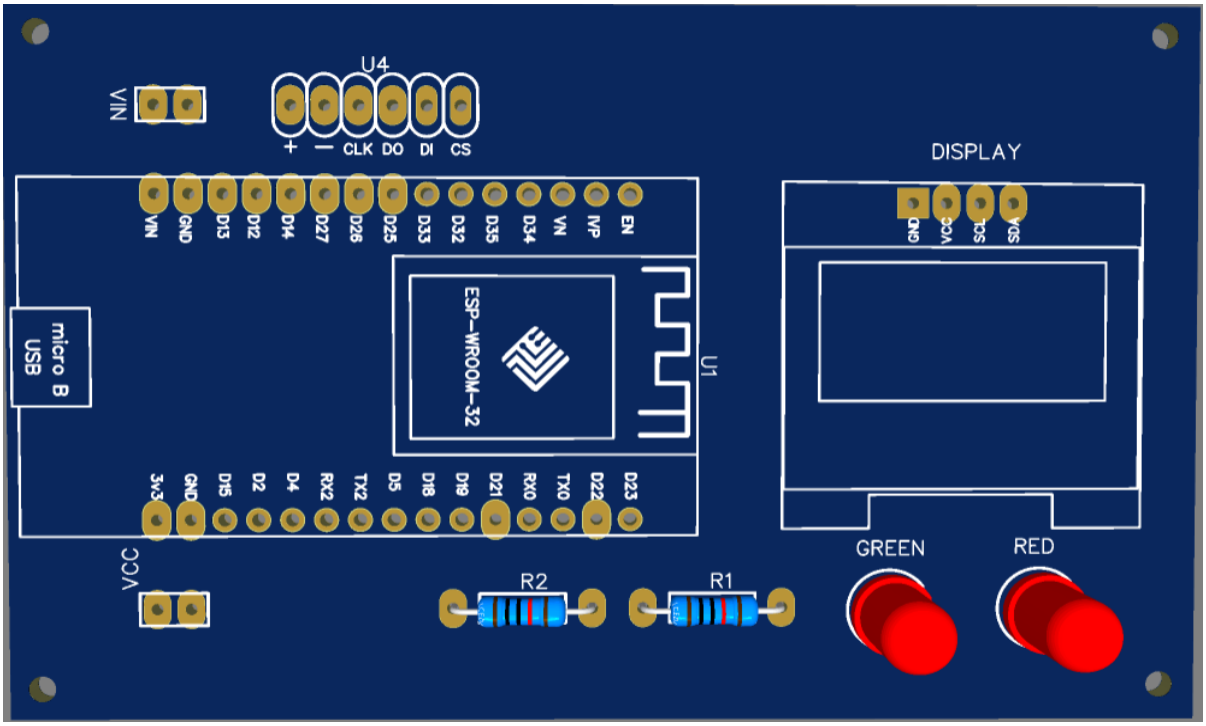

Table 2: footprints

S.No.	Componets	Footprint Name
1.	ESP-32	ESP-WROOM-32
2.	OLED	DISPLAY

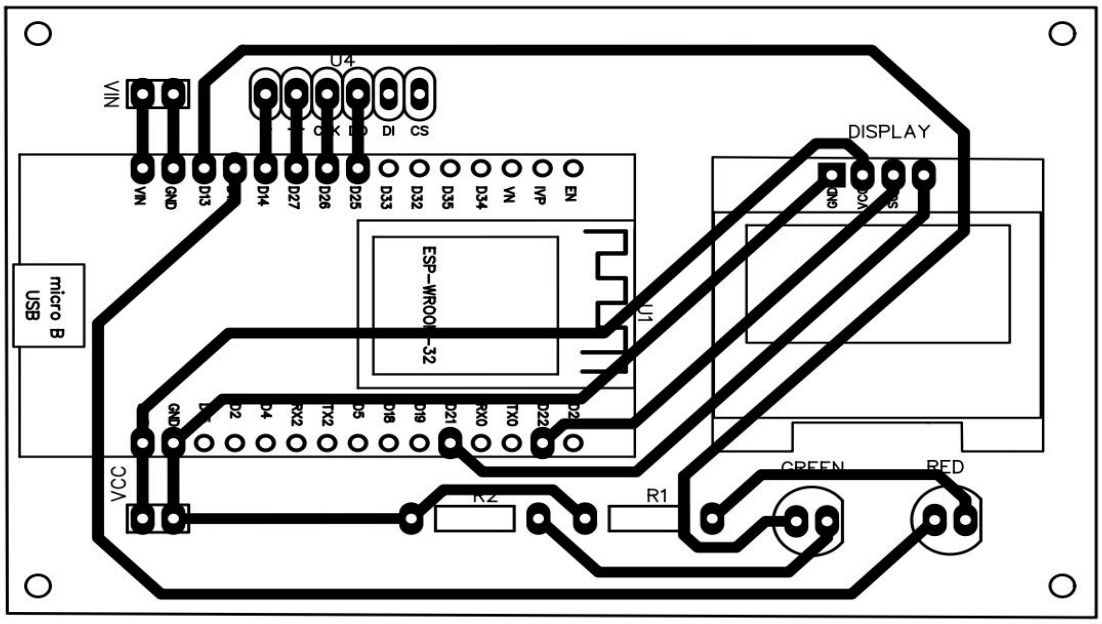

Fig. 3: PCB Layout

Program: - (if any)

```
#include <Adafruit_SSD1306.h>                                              //Include the required libraries
#include <WiFi.h>
#include <Wire.h>
#include <HTTPClient.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <ArduinoJson.h>

#define SCREEN_WIDTH 128                                                   //Define the OLED display width
and height
#define SCREEN_HEIGHT 64
#define OLED_RESET     -1                                                  // Reset pin # (or -1 if sharing
Arduino reset pin)
#define SCREEN_ADDRESS 0x3C                                                //I2C address for display
#define upLED 13
#define downLED 12
Adafruit_SSD1306 display (SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);    //Create the display object

const char* ssid = "WiFi Name";                                            //Set your WiFi network name and
password
const char* password = "Password";

const int httpsPort = 443;                                                 //Bitcoin price API powered by
CoinGecko
const String url =
"https://api.coingecko.com/api/v3/simple/price?ids=bitcoin&vs_currencies=USD&include_24hr_change=true";

WiFiClient client;                                                         //Create a new WiFi client
HTTPClient http;

String formattedDate;                                                      //Create variables to store the
date and time
String dayStamp;
String timeStamp;

const unsigned char bitcoinLogo [] PROGMEM =                               // 'Bitcoin Logo', 128x64px
{
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xc0, 0x3f, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x00, 0x03, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf0, 0x00, 0x00, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xc0, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x1e, 0x03, 0xe7, 0xc0, 0x07, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x03, 0xe7, 0xc0, 0x01, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x02, 0x24, 0x40, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x02, 0x24, 0x40, 0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0xc0, 0x02, 0x24, 0x40, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x02, 0x24, 0x40, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x03, 0x80, 0x02, 0x3c, 0x40, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x07, 0x03, 0xfe, 0x3c, 0x7c, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x06, 0x03, 0x00, 0x00, 0x1f, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x0e, 0x03, 0x00, 0x00, 0x01, 0xc0, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x0c, 0x03, 0x00, 0x00, 0x00, 0xc0, 0x03, 0x80, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x1c, 0x03, 0xf0, 0x3f, 0x80, 0x60, 0x03, 0x80, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0xf0, 0x3f, 0xe0, 0x20, 0x01, 0x80, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x18, 0x30, 0x70, 0x30, 0x01, 0xc0, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x18, 0x30, 0x10, 0x30, 0x00, 0xc0, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x18, 0x30, 0x10, 0x30, 0x00, 0xc0, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x18, 0x30, 0x10, 0x30, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x18, 0x30, 0x30, 0x20, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x18, 0x30, 0xe0, 0x60, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x60, 0x00, 0x18, 0x3f, 0xc0, 0xc0, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x60, 0x00, 0x18, 0x00, 0x01, 0xc0, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x60, 0x00, 0x18, 0x00, 0x00, 0xf0, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x60, 0x00, 0x18, 0x00, 0x00, 0x38, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x60, 0x00, 0x18, 0x3f, 0xe0, 0x18, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x60, 0x00, 0x18, 0x30, 0xf8, 0x0c, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x18, 0x30, 0x1c, 0x0c, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x18, 0x30, 0x0c, 0x0c, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x18, 0x30, 0x04, 0x0c, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x18, 0x30, 0x0c, 0x0c, 0x00, 0xc0, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x18, 0x30, 0x0c, 0x0c, 0x00, 0xc0, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x18, 0x30, 0x38, 0x0c, 0x01, 0xc0, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x18, 0x01, 0xf0, 0x3f, 0xf0, 0x08, 0x01, 0xc0, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x1c, 0x01, 0xe0, 0x1f, 0x00, 0x18, 0x01, 0x80, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x1c, 0x01, 0x00, 0x00, 0x00, 0x30, 0x03, 0x80, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x0e, 0x03, 0x00, 0x00, 0x00, 0xe0, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00,
```

```
  0x00, 0x00, 0x00, 0x00, 0x06, 0x03, 0x00, 0x00, 0x0f, 0xc0, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x07, 0x03, 0xfe, 0x3c, 0x7e, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x03, 0x80, 0x02, 0x3c, 0x40, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x02, 0x24, 0x40, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0xc0, 0x02, 0x24, 0x40, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x02, 0x24, 0x40, 0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x02, 0x24, 0x40, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x03, 0xe7, 0xc0, 0x01, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x1e, 0x03, 0xe7, 0xc0, 0x07, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xc0, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf0, 0x00, 0x00, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x03, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xc0, 0x3f, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

void setup()
{
  Serial.begin(115200);                                                    //Start the serial monitor

  pinMode(upLED, OUTPUT);                                                  //Define the LED pin outputs
  pinMode(downLED, OUTPUT);

  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS))                //Connect to the display
  {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;); // Don't proceed, loop forever
  }

  display.clearDisplay();                                                  //Clear the display
  display.setTextColor(SSD1306_WHITE);                                     //Set the text colour to white
  display.drawBitmap(0, 0, bitcoinLogo, 128, 64, WHITE);                    //Display bitmap from array
  display.display();
  delay(2000);

  display.clearDisplay();                                                  //Clear the display
  display.setTextSize(1);                                                  //Set display parameters
  display.setTextColor(WHITE);
  display.println("Connecting to WiFi...");
  display.display();

  WiFi.begin(ssid, password);

  Serial.print("Connecting to WiFi...");
  while (WiFi.status() != WL_CONNECTED)                                     //Connect to the WiFi network
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  display.println("Connected to: ");                                      //Display message once connected
  display.print(ssid);
  display.display();
  delay(1500);
  display.clearDisplay();
  display.display();
}

void loop()
{


  Serial.print("Connecting to ");                                 //Display url on Serial monitor for debugging
  Serial.println(url);

  http.begin(url);
  int httpCode = http.GET();                                                        //Get crypto price from API
  StaticJsonDocument<2000> doc;
  DeserializationError error = deserializeJson(doc, http.getString());

  if (error)                                                     //Display error message if unsuccessful
  {
    Serial.print(F("deserializeJson Failed"));
    Serial.println(error.f_str());
    delay(2500);
    return;
  }

  Serial.print("HTTP Status Code: ");
  Serial.println(httpCode);

  String BTCUSDPrice = doc["bitcoin"]["usd"].as<String>();
                                          //Store crypto price and update date in local variables
```

```
  float USD24hrFloat = doc["bitcoin"]["usd_24h_change"].as<float>();
  String USDChange = String(USD24hrFloat, 3);
  http.end();

  Serial.print("BTCUSD Price: ");                                    //Display current price on serial monitor
  Serial.println(BTCUSDPrice.toDouble());

  Serial.print("Yesterday's Price: ");                            //Display yesterday's price on serial monitor
  Serial.println(USDChange);

  bool isUp = USD24hrFloat > 0;                                 //Check whether price has increased or decreased
  double percentChange;
  String dayChangeString = "24hr. Change: ";
  if (isUp)                                                          //If price has increased from yesterday
  {
    digitalWrite(upLED, HIGH);
    digitalWrite(downLED, LOW);
  }
  else                                                             //If price has decreased from yesterday
  {
    digitalWrite(downLED, HIGH);
    digitalWrite(upLED, LOW);
  }

  Serial.print("Percent Change: ");                       //Display the percentage change on the serial monitor
  Serial.println(USDChange);

  display.clearDisplay();                                                  //Clear the OLED display
  display.setTextSize(1);
  printCenter("BTC/USD", 0, 0);                                      //Display the comparison header

  display.setTextSize(2);
  printCenter("$" + BTCUSDPrice, 0, 25);                                    //Display the current price

  display.setTextSize(1);                                                 //Display the change percentage
  dayChangeString = dayChangeString + USDChange + "%";
  printCenter(dayChangeString, 0, 55);
  display.display();                                                        //Execute the new display

  http.end();                                                              //End the WiFi connection
 delay(15000);
}
void printCenter(const String buf, int x, int y)  //Function to centre the current price in the display width
{
  int16_t x1, y1;
  uint16_t w, h;
  display.getTextBounds(buf, x, y, &x1, &y1, &w, &h);                   //Calculate string width
  display.setCursor((x - w / 2) + (128 / 2), y);                         //Set cursor to print string in
centre
  display.print(buf);                                                     //Display string
}
```

Testing:

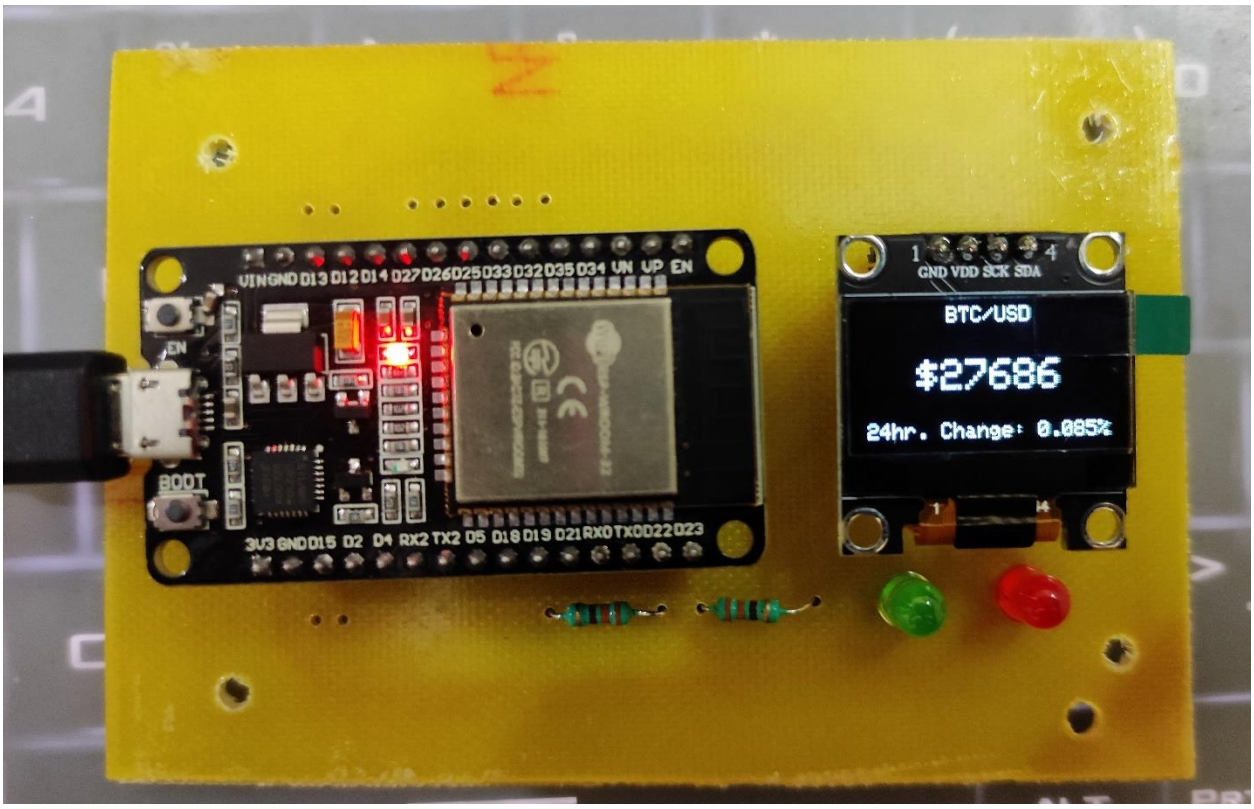

Fig. 4: Photograph of your project with output

Applications:

Personal Bitcoin Price Monitoring: Individuals interested in Bitcoin investments or trading can use the Bitcoin Price Tracker to monitor the price fluctuations of Bitcoin. It provides them with real-time updates, allowing them to make informed decisions regarding buying or selling Bitcoin.

Cryptocurrency Enthusiasts: This project is also suitable for cryptocurrency enthusiasts who want to stay up to date with the latest Bitcoin price movements. They can use the physical device as a convenient way to have a quick glance at the current Bitcoin price without relying on online platforms or mobile applications.

Learning Tool: The Bitcoin Price Tracker project can serve as a learning tool for those interested in understanding how microcontrollers like ESP-32 can be utilized to interact with external components such as OLED displays and LEDs. By building this project, enthusiasts can gain practical experience in hardware integration and programming.

Educational Demonstrations: The project can be used as an educational demonstration in schools or workshops to introduce the concept of cryptocurrency and showcase how technology can be used to track and visualize real-time data.

Promotional Item: Companies or organizations related to cryptocurrency or Bitcoin can use this project as a promotional item. They can distribute it to their clients or attendees at conferences, seminars, or promotional events. It serves as a functional and engaging item that reinforces their branding and promotes awareness about Bitcoin.