Geekbrains

Метеостанция с дисплеем и web-сервером для удаленного наблюдения за текущими показаниями

Программа: Разработчик
Специализация: Инженерных умных устройств
Лялин Евгений Дмитриевич

Тула
2024

Содержание

Введение

Проект «Метеостанция с дисплеем и веб-сервером» на базе ESP32 и протокола MQTT предоставляет уникальную возможность для получения актуальной информации о температуре, атмосферном давлении и влажности в режиме реального времени.

ESP32, обладая высокой производительностью и встроенными функциями Wi-Fi идеально подходит для создания IoT-устройств. Использование протокола MQTT обеспечивает надежную и эффективную передачу данных между метеостанцией и сервером, что позволяет пользователям получать информацию о температуре, влажности, атмосферном давлении и других параметрах с любой точки мира.

Данный проект не только позволит пользователям следить за текущими погодными условиями, но и станет отличной основой для дальнейших исследований в области метеорологии и интернет-технологий. Введение в проект включает описание его целей, функциональных возможностей, а также архитектуры системы, что позволит оценить его потенциал и практическую ценность.

Целью проекта является получение практических навыков работы с датчиками и устройствами подключенных к шине I2C, программирование на языке C/C++, передача данных по протоколу MQQT и создание табло визуализации на основе Node-RED.

Для реализации проекта применяется следующее оборудование:

- микроконтроллер ESP32;

- дисплей SSD1306 128x64 I2C;

- датчик давления и температуры BMP280;

-датчик влажности AM2320;

- макетная плата с набором проводников.

Для реализации web сервера будет использоваться арендованный VDS сервер на основе Ubuntu 20.04 с установленным MQTT-брокером и Node-Red Dashboard.

Node-RED — мощный инструмент для визуального программирования. Создание потоков подключения позволяет легко интегрировать различные устройства и сервисы.

В ходе выполнения проекта используются следующие инструменты:

- Visual Studio Code с установленным расширением PltformIO – для написания и отладки кода для микроконтроллера ESP32;

- Termius – SSH-клиент для доступа к удаленному серверу;

- мультиметр Mestek для прозвонки правильности собранных цепей;

- wokwi.com- онлайн эмулятор ESP32;

- Google Chrome в качестве программы для визуализации данных.

Теоретическая и практическая главы

1. Сборка макета устройства

Процесс сборки включает в себя следующие шаги:

1.1 Подключение платы ESP32 к компьютеру для питания и программирования. Модуль ESP32 имеет встроенный преобразователь USB-RS 232 для возможности прошивки микроконтроллера, а также преобразователь напряжения 5В в 3.3В. Все устройства (дисплей, датчики) питаются от напряжения 3.3В.

Обмен данными микроконтроллера со всеми устройствами ведется по протоколу I2C.

I2C (Inter-Integrated Circuit) — это протокол связи, который позволяет микроконтроллерам и периферийным устройствам обмениваться данными. Он использует всего два провода для связи: SDA (Serial Data Line) и SCL (Serial Clock Line).

1.2 Подключение дисплея SSD1306:

- VCC — подключение к VCC (3.3V).
- GND — подключение к GND.
- SDA — подключение к SDA на ESP32 (GPIO 21).
- SCL — подключение к SCL на ESP32 (GPIO 22).

1.3 Подключение датчика давления и температуры BMP280:

- VCC — подключение к VCC (3.3V).
- GND — подключение к GND.
- SDA — подключение к SDA (тот же, что и для SSD1306).
- SCL — подключение к SCL (тот же, что и для SSD1306).

1.4 Подключение датчика влажности AM2320:

- VCC — подключение к VCC (3.3V).
- GND — подключение к GND.
- SDA — подключение к SDA (тот же, что и для SSD1306).
- SCL — подключение к SCL (тот же, что и для SSD1306).

1.5 Согласно требованиям протокола, I2C к выводам SCL и SDA подключены подтягивающие к питанию 3.3В резисторы номиналом 10 кОм.

Схема устройства представлена на следующем рисунке (датчик BMP280 не показан):

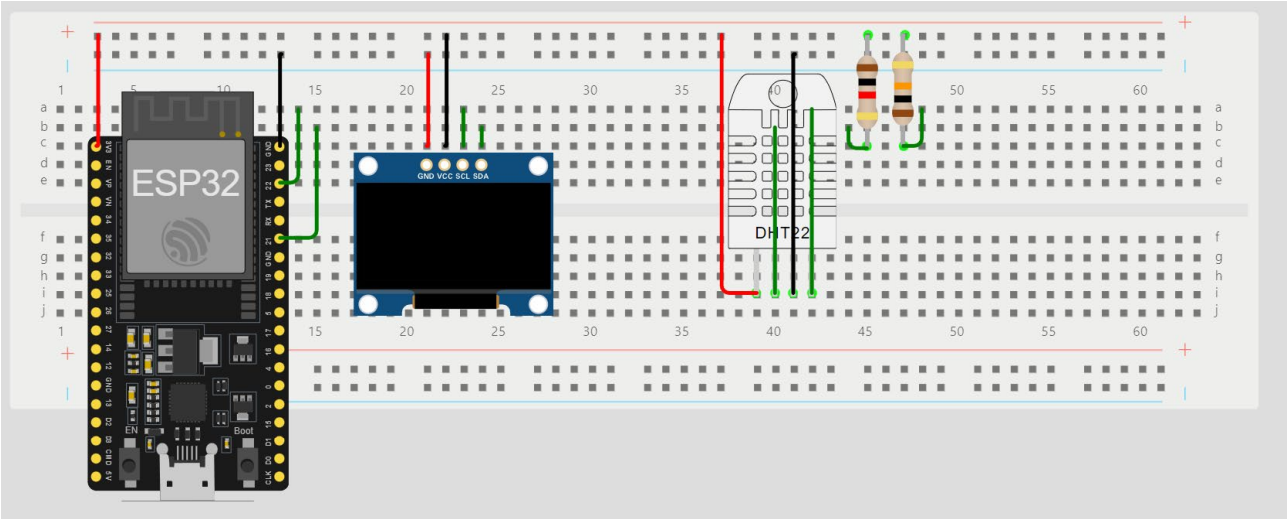
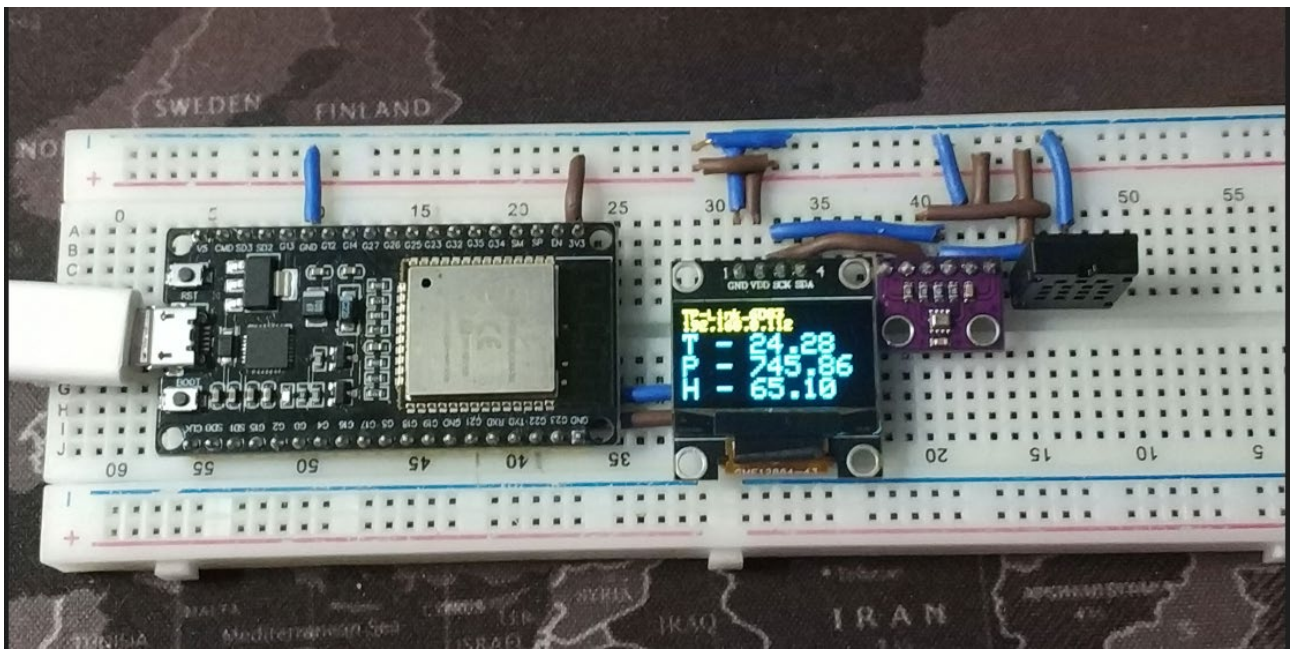


Фото собранного макета представлено на следующем рисунке:



2. Описание комплектующих

2.1 Микроконтроллер ESP32-WROOM32

- Рабочее напряжение: 3,3 В
- Входное напряжение (USB): 5 В
- GPIO: 34
- Контакты аналогового входа: 16 (макс. вход: 3,2 В)
- Аналоговые выходные контакты: 2
- Флэш-память: 4 МБ
- Тактовая частота: 80 МГц/240 МГц
- ЦП: 32-битный
- Особенность: Bluetooth 4.2, 9 контактов сенсорного датчика.
- Интерфейсы: SPI, I2C, I2S, CAN, UART
- Протоколы Wi-Fi: 802.11 b/g/n (802.11n до 150 Мбит/с)
- Частота Wi-Fi: 2,4 ГГц – 2,5 ГГц
- Размеры: 52 мм x 26 мм.
- Вес: 9,8 г

2.2 OLED дисплей SSD1306 128x64

- Цвет – монохромный;
- Разрешение – 128 x 64;
- Графический чип – SSD1306;
- Интерфейс – I2C;
- Цвет дисплея – синий/желтый;
- Угол обзора> 160 °;
- Напряжение питания – от 3 до 5 В;
- Размер: 27x27x4 мм.

2.3 Датчик давления и температуры BMP280

- Напряжение питания: 1.71V – 3.6V;
- Интерфейс обмена данными: I2C или SPI;
- Ток потребления в рабочем режиме: 2.7uA при частоте опроса 1 Гц;
- Диапазон измерения атмосферного давления: 300hPa – 1100hPa (±0.12hPa), что эквивалентно диапазону от -500 до 9000 м над уровнем моря;
- Диапазон измерения температуры: -40°C … +85°C (±0.01°C);
- Максимальная частота работы интерфейса I2C: 3.4MHz;
- Максимальная частота работы интерфейса SPI: 10 МГц;
- Размер модуля: 21 x 18 мм;

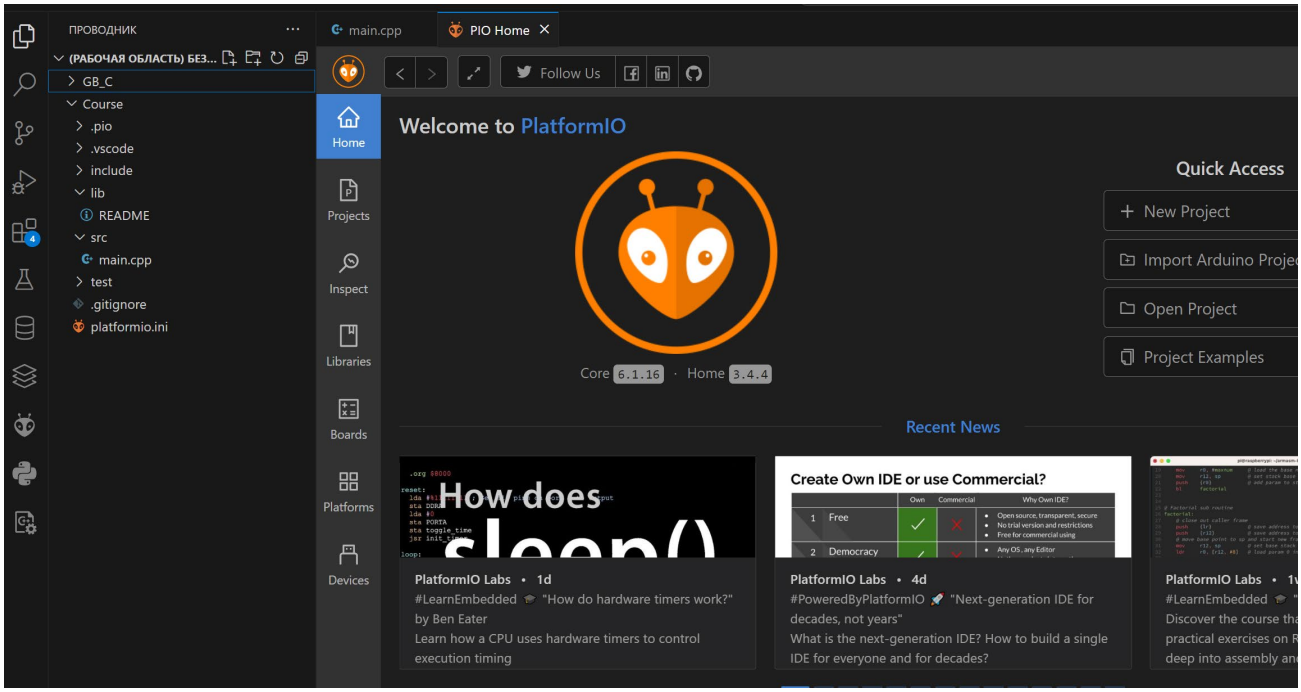
2.4 Датчик влажности AM2320

- Напряжение питания – 3,1-5,5 В;
- Потребляемый ток – 8-10 мА в режиме ожидания, до 950 мА в режиме измерений;
- Диапазон измерения температуры от -40º до +80ºС;
- Максимальная погрешность измерений – 0,5 ºС;
- Диапазон измерения влажности воздуха – 0-100%;
- Погрешность измерений влажности воздуха – 3%;
- Время между считыванием данных – 2 сек;
- Габариты – 15x12,1x4,5 мм.

3. Программирование микроконтроллера

3.1 Подготовка среды программирования.

Программирование микроконтроллера производится в среде Arduino в программе Visual Studio Code с установленным расширением PlatformIO.



3.2 Подготовка библиотек для работы с периферией и подключаемыми устройствами.

```
1. #include <WiFi.h> //работа с Wifi
2. #include <Wire.h> //работа с протоколом I2C
3. #include <PubSubClient.h> //работа с протоколом MQTT
4. #include <Adafruit_Sensor.h> //работа с датчиками
5. #include <Adafruit_BMP280.h> //работа с датчиком BMP280
6. #include <Adafruit_AM2320.h> //работа с датчиком AM2320.h
7. #include <Adafruit_SSD1306.h> //работа с дисплеем SSD1306
```

3.3 Создание переменных и объектов для работы с оборудованием.

```
8. #define I2C_display 0x3C
9. #define I2C_BMP280 0x76
10.
11.     const char* ssid = "TP-Link_6D83";
12.     const char* password = "PASSWORD";
13.     const char* mqtt_server = "138.124.31.84";
14.
15.     WiFiClient espClient;
16.     PubSubClient client(espClient);
17.     long lastMsg = 0;
18.     char msg[50];
19.     int value = 0;
20.
21.     float temperature = 0;
22.     float pressure = 0;
23.     float humidity =0;
24.
25.     Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire, -1);
26.     Adafruit_BMP280 bmp;
27.     Adafruit_AM2320 am2320 = Adafruit_AM2320();
```

3.4 Функция подключения Wifi с выводом сообщений в терминал.

```
28.     void setup_wifi() {
29.       delay(10);
30.       Serial.println();
31.       Serial.print("Connecting to ");
32.       Serial.println(ssid);
33.       WiFi.begin(ssid, password);
34.       while (WiFi.status() != WL_CONNECTED) {
35.         delay(500);
36.         Serial.print(".");
37.       }
38.
39.       Serial.println("");
40.       Serial.println("WiFi connected");
41.       Serial.println("IP address: ");
42.       Serial.println(WiFi.localIP());
43.     }
```

3.5 Функция подключения к брокеру MQTT с выводом сообщений в терминал.

```
44.     void reconnect() {
45.       while (!client.connected()) {
46.         Serial.print("Attempting MQTT connection...");
47.         if (client.connect("ESP8266Client")) {
```

```
48.           Serial.println("connected");
49.           client.subscribe("esp32/output");
50.         } else {
51.           Serial.print("failed, rc=");
52.           Serial.print(client.state());
53.           Serial.println(" try again in 5 seconds");
54.           delay(5000);
55.         }
56.       }
57.     }
```

3.7 Функция инициализации переменных, определения режимов работы датчиков, дисплея, wifi.

```
58. void setup() {
59.   Serial.begin(9600);
60.   Wire.begin();
61.   am2320.begin();
62.
63.   display.begin(SSD1306_SWITCHCAPVCC, I2C_display);
64.   display.clearDisplay();
65.   display.setTextColor(WHITE);
66.
67.   bmp.begin(I2C_BMP280);
68.   bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,     /* Operating Mode. */
69.                   Adafruit_BMP280::SAMPLING_X2,     /* Temp. oversampling */
70.                   Adafruit_BMP280::SAMPLING_X16,    /* Pressure oversampling */
71.                   Adafruit_BMP280::FILTER_X16,      /* Filtering. */
72.                   Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */
73.
74.   setup_wifi();
75.   client.setServer(mqtt_server, 1883);
76. }
```

3.8 Основной цикл программы. Опрос датчиков, вывод на дисплей, проверка соединения с MQTT брокером и восстановление связи в случае обрыва, передача данных на удаленный сервер.

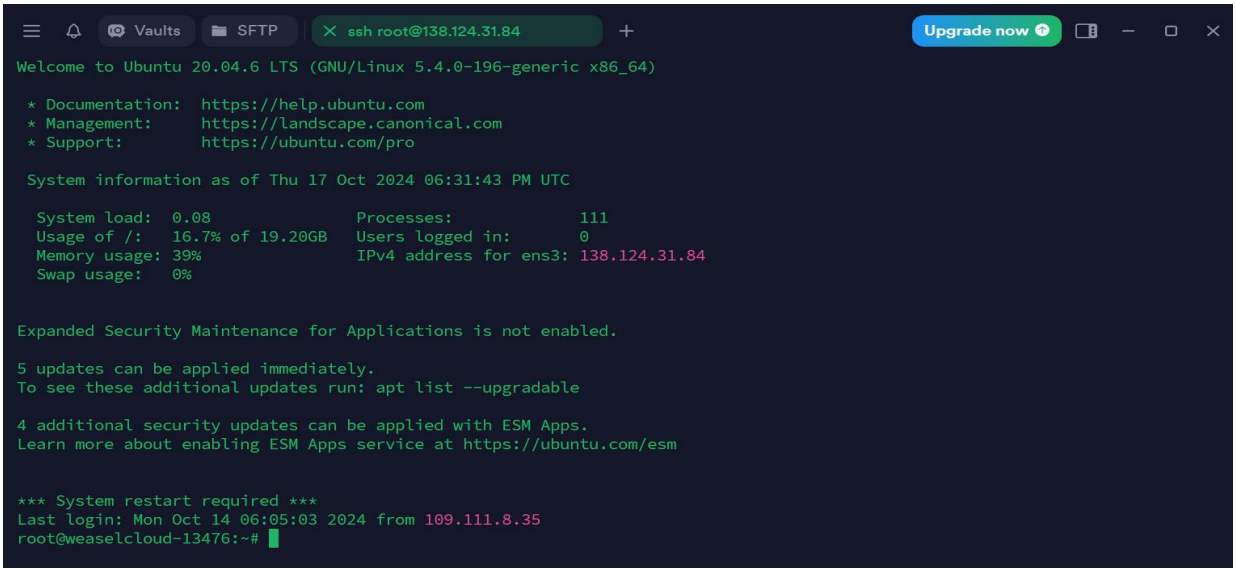
```
77.     void loop() {
78.       temperature = bmp.readTemperature();
79.       pressure = bmp.readPressure()*0.00750062;
80.       humidity = am2320.readHumidity();
81.
82.       display.clearDisplay();
83.       display.setCursor(0,0);
84.       display.setTextSize(1);
85.       display.println(ssid);
86.       display.println(WiFi.localIP());
87.       display.setTextSize(2);
```

```
88.       display.print("T - ");
89.       display.println(temperature);
90.       display.print("P - ");
91.       display.println(pressure);
92.       display.print("H - ");
93.       display.println(humidity);
94.       display.display();
95.
96.       if (!client.connected()) {
97.         reconnect();
98.       }
99.       client.loop();
100.
101.      long now = millis();
102.      if (now - lastMsg > 5000) {
103.        lastMsg = now;
104.        char tempString[8];
105.        dtostrf(temperature, 1, 2, tempString);
106.        client.publish("esp32/temperature", tempString);
107.        char pressString[8];
108.        dtostrf(pressure, 1, 2, pressString);
109.        client.publish("esp32/pressure", pressString);
110.        char humidString[8];
111.        dtostrf(humidity, 1, 2, humidString);
112.        client.publish("esp32/humidity", humidString);
113.      }
114.    }
```

Полный листинг программы приведен в приложении 1.

4. Подготовка серверной части

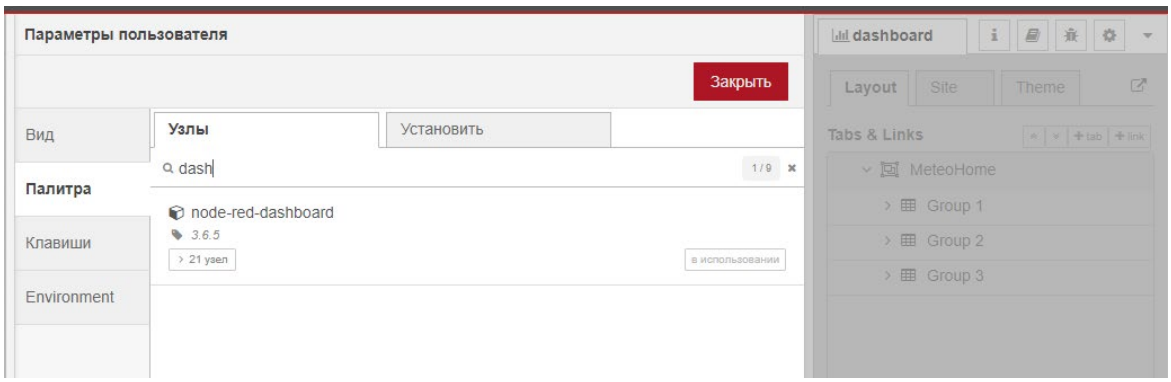
4.1 Подключение к удаленной VDS машине по протоколу SSH.



4.2 Подготовка Node-RED Dashboard .

```
sudo npm install -g --unsafe-perm node-red
node-red
node-red admin init
node-red-start
sudo systemctl enable nodered.service
sudo reboot
```

Установка Dashboard



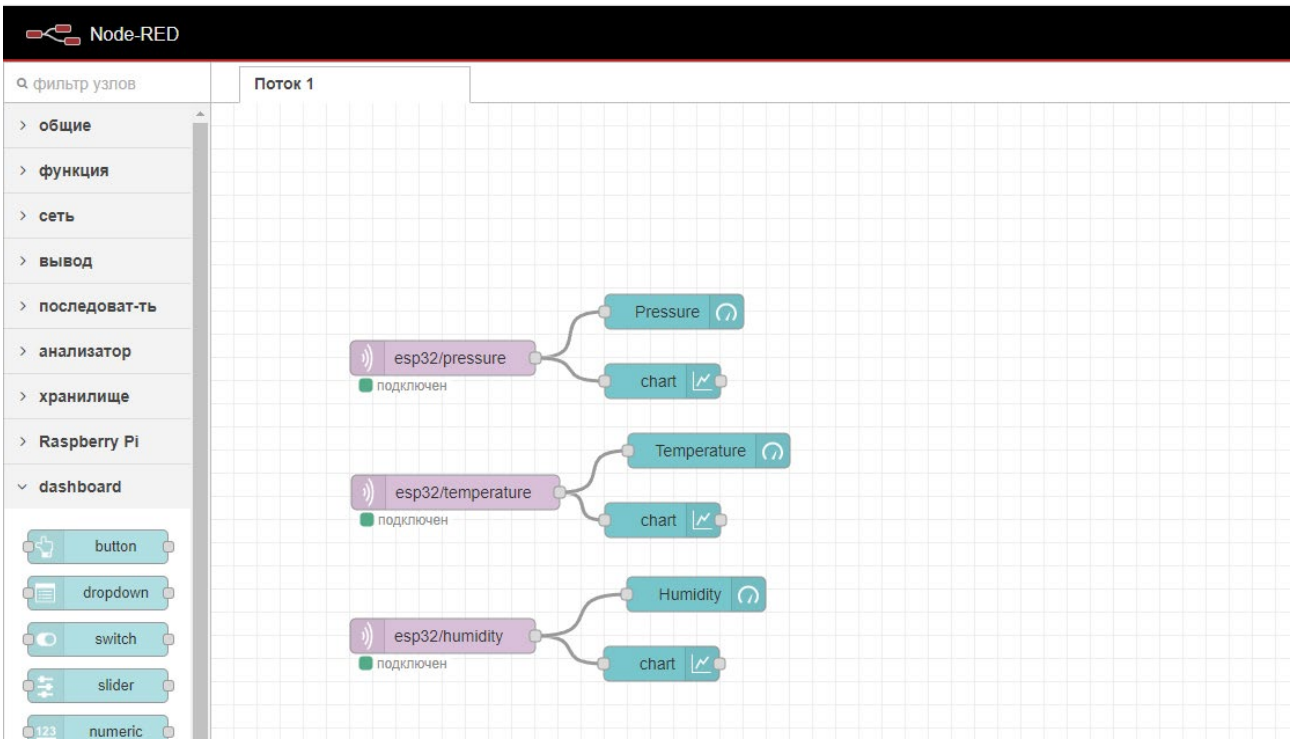
4.3Подготовка MQTT-брокера Mosquitto .

```
sudo apt install -y mosquitto mosquitto-clients
sudo systemctl enable mosquitto.service
mosquitto -v
sudo nano /etc/mosquitto/mosquitto.conf
sudo systemctl restart mosquitto
sudo systemctl status mosquitto
mosquitto -v
```

Конфигурация Mosquitto доступ без пароля.

```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

listener 1883
allow_anonymous true
```

Создание потоков, подписка на топики ESP32.



5. Описание работы системы

После включения питания в терминале отображается следующая информация:

```
Выполнение задачи в папке Course: C:\Users\user\.platformio\penv\Scripts\platformio.exe device monitor

--- Terminal on COM4 | 9600 8-N-1
--- Available filters and text transformations: colorize, debug, default, direct, esp32_exception_decoder, hexlify, log2file, nocontrol, printable, send_on_enter, time
--- More details at https://bit.ly/pio-monitor-filters
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H

Connecting to TP-Link_6D83
.
WiFi connected
IP address:
192.168.0.112
Attempting MQTT connection...connected
```

Служебные сообщения от котроллера ESP32 – номер COM порта, скорость COM порта и т.д.

После первичной инициализации в терминал выводится сообщение с названием подключаемой точки доступа. При успешном подключении выводится сообщение – WiFi connected и IP адрес ESP32 в локальной сети.

Затем производится установка связи с MQTT брокером, после успешного соединения выводится сообщение – connected.

На экране дисплея выводится следующая информация

Имя точки доступа SSID

Локальный IP адрес

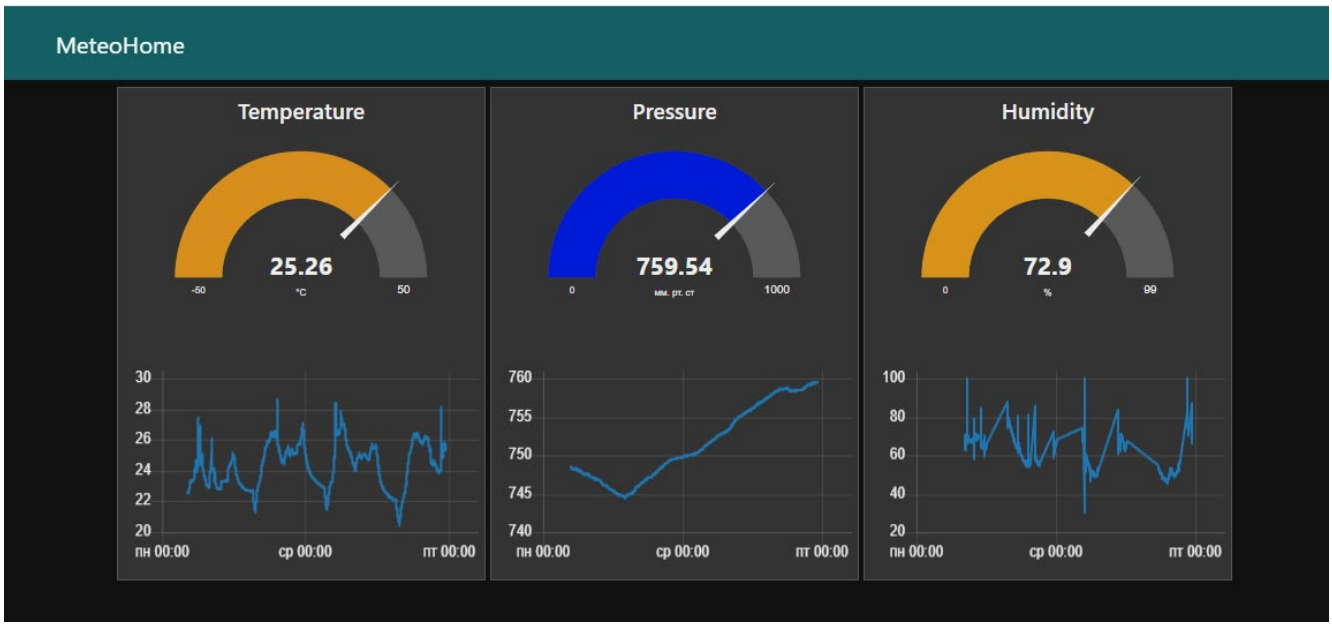
Текущая температура °C

Текущее давление мм.рт.ст

Текущая влажность %

При переходе по адресу http://138.124.31.84:1880/ui/ выводится текущее значение температуры, атмосферного давления и влажности в виде стрелочных приборов и цифровых значений.

Ниже на графиках представлены архивы значений за последнюю неделю



Заключение

Проект «Метеостанция с дисплеем и веб-сервером» на базе ESP32 и протокола MQTT стал успешной реализацией современных технологий в области интернет вещей. Он продемонстрировал, как с помощью доступного оборудования и программного обеспечения можно создать функциональное устройство, способное в реальном времени предоставлять актуальную информацию о погодных условиях и микроклимате в помещении.

Использование микроконтроллера ESP32 обеспечило высокую производительность и надежность системы, а интеграция с протоколом MQTT позволила эффективно передавать данные на удаленный сервер. Это не только обеспечивает пользователям доступ к информации о температуре, влажности и атмосферном давлении из любой точки мира, но и создает основу для дальнейшего развития проекта — например, добавления новых датчиков или расширения функционала веб-интерфейса.

Проект также стал отличной платформой для практического освоения навыков работы с датчиками, программирования на C/C++, а также визуализации данных с помощью Node-RED. Использование современных инструментов разработки, таких как Visual Studio Code и онлайн-эмуляторы, значительно упростило процесс создания и отладки кода.

В итоге, данная метеостанция не только предоставляет полезную информацию о текущих погодных условиях, но и открывает новые горизонты для исследований в области метеорологии и IoT-технологий. Проект может служить основой для будущих разработок и улучшений, что делает его ценным вкладом в изучение современных технологий и их применения в реальной жизни.

Список используемой литературы

- *http://138.124.31.84:1880/ui/ ссылка на web сервер проекта.*
- *https://github.com/JonLED1/GB_C_IOT/tree/main/Final_work ссылка на Github проекта*
- *https://wokwi.com Эмулятор ESP32*
- *https://randomnerdtutorials.com/ Уроки и примеры с микроконтроллером ESP32*
- *https://nodered.org/ официальный сайт Node-RED*
- *https://3d-diy.ru/ описание технической части оборудования*
- *https://younglinux.info/ Уроки Linux, Bash, C/C++*
- *https://alexgyver.ru/ Практические занятия ESP32, Raspberry, C/C++*

Приложение 1. Листинг кода C++ для микроконтроллера ESP32.

```
#include <WiFi.h>
#include <Wire.h>
#include <PubSubClient.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <Adafruit_AM2320.h>
#include <Adafruit_SSD1306.h>

#define I2C_display 0x3C
#define I2C_BMP280 0x76

const char* ssid = "TP-Link_6D83";
const char* password = "98519569";
const char* mqtt_server = "138.124.31.84";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

float temperature = 0;
float pressure = 0;
float humidity =0;

Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire, -1);
Adafruit_BMP280 bmp;
Adafruit_AM2320 am2320 = Adafruit_AM2320();

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
```

```
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
      client.subscribe("esp32/output");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(9600);
  Wire.begin();
  am2320.begin();

  display.begin(SSD1306_SWITCHCAPVCC, I2C_display);
  display.clearDisplay();
  display.setTextColor(WHITE);

  bmp.begin(I2C_BMP280);
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,     /* Operating Mode. */
                  Adafruit_BMP280::SAMPLING_X2,     /* Temp. oversampling */
                  Adafruit_BMP280::SAMPLING_X16,    /* Pressure oversampling */
                  Adafruit_BMP280::FILTER_X16,      /* Filtering. */
                  Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */

  setup_wifi();
  client.setServer(mqtt_server, 1883);
}

void loop() {
  temperature = bmp.readTemperature();
  pressure = bmp.readPressure()*0.00750062;
  humidity = am2320.readHumidity();

  display.clearDisplay();
  display.setCursor(0,0);
  display.setTextSize(1);
  display.println(ssid);
  display.println(WiFi.localIP());
  display.setTextSize(2);
  display.print("T - ");
  display.println(temperature);
  display.print("P - ");
  display.println(pressure);
  display.print("H - ");
  display.println(humidity);
  display.display();
```

```
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 5000) {
    lastMsg = now;
    char tempString[8];
    dtostrf(temperature, 1, 2, tempString);
    client.publish("esp32/temperature", tempString);
    char pressString[8];
    dtostrf(pressure, 1, 2, pressString);
    client.publish("esp32/pressure", pressString);
    char humidString[8];
    dtostrf(humidity, 1, 2, humidString);
    client.publish("esp32/humidity", humidString);
  }
}
```

Приложение 2. Листинг кода в формате json Node-RED.

```
[
    {
        "id": "b6932d514be1a226",
        "type": "tab",
        "label": "Поток 1",
        "disabled": false,
        "info": "",
        "env": []
    },
    {
        "id": "abf7079a.653be8",
        "type": "mqtt in",
        "z": "b6932d514be1a226",
        "name": "",
        "topic": "esp32/temperature",
        "qos": "2",
        "datatype": "auto-detect",
        "broker": "10e78a89.5b4fd5",
        "nl": false,
        "rap": false,
        "inputs": 0,
        "x": 211,
        "y": 336,
        "wires": [
            [
                "21eae8f8.2971b8",
                "b057e85acbb91c7d"
            ]
        ]
    },
    {
        "id": "4aecba01.78ce64",
        "type": "mqtt in",
        "z": "b6932d514be1a226",
        "name": "",
        "topic": "esp32/pressure",
        "qos": "2",
        "datatype": "auto-detect",
        "broker": "10e78a89.5b4fd5",
        "nl": false,
        "rap": false,
        "inputs": 0,
        "x": 200,
        "y": 220,
        "wires": [
            [
                "df37e6b7.64c1c8",
                "bb751fd803fd11d4"
            ]
        ]
    },
    {
        "id": "21eae8f8.2971b8",
        "type": "ui_chart",
        "z": "b6932d514be1a226",
        "name": "",
        "group": "61285987.c20328",
        "order": 2,
        "width": 0,
        "height": 0,
        "label": "",
        "chartType": "line",
        "legend": "false",
        "xformat": "dd HH:mm",
        "interpolate": "linear",
        "nodata": "",
        "dot": false,
        "ymin": "",
        "ymax": "",
        "removeOlder": 1,
        "removeOlderPoints": "",
        "removeOlderUnit": "604800",
        "cutout": 0,
        "diff": false,
        "useOneColor": false,
        "useUTC": false,
        "colors": [
            "#1f77b4",
            "#aec7e8",
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "outputs": 1,
        "useDifferentColor": false,
        "className": "",
        "x": 390,
        "y": 360,
        "wires": [
            []
        ]
    },

{
        "id": "df37e6b7.64c1c8",
        "type": "ui_gauge",
        "z": "b6932d514be1a226",
        "name": "",
        "group": "3897eee0be7b7e25",
        "order": 1,
        "width": 0,
        "height": 0,
        "gtype": "gage",
        "title": "Pressure",
        "label": "мм. рт. ст",
        "format": "{{value}}",
        "min": 0,
        "max": "1000",
        "colors": [
            "#00b3d9",
            "#0073e6",
            "#001bd7"
        ],
        "seg1": "33",
        "seg2": "66",
        "diff": false,
        "className": "",
        "x": 400,
        "y": 180,
        "wires": []
    },
    {
        "id": "b057e85acbb91c7d",
        "type": "ui_gauge",
        "z": "b6932d514be1a226",
        "name": "",
        "group": "61285987.c20328",
        "order": 1,
        "width": 0,
        "height": 0,
        "gtype": "gage",
        "title": "Temperature",
        "label": "°C",
        "format": "{{value}}",
        "min": "-50",
        "max": "50",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
  {
```

```
        "className": "",
        "x": 430,
        "y": 300,
        "wires": []
    },
{
        "id": "bb751fd803fd11d4",
        "type": "ui_chart",
        "z": "b6932d514be1a226",
        "name": "",
        "group": "3897eee0be7b7e25",
        "order": 1,
        "width": 0,
        "height": 0,
        "label": "",
        "chartType": "line",
        "legend": "false",
        "xformat": "dd HH:mm",
        "interpolate": "linear",
        "nodata": "",
        "dot": false,
        "ymin": "",
        "ymax": "",
        "removeOlder": 1,
        "removeOlderPoints": "",
        "removeOlderUnit": "604800",
        "cutout": 0,
        "useOneColor": false,
        "useUTC": false,
        "colors": [
            "#1f77b4",
            "#aec7e8",
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "outputs": 1,
        "useDifferentColor": false,
        "className": "",
        "x": 390,
        "y": 240,
        "wires": [
            []
        ]
    },
  {
        "id": "607b3d826136aac3",
        "type": "mqtt in",
        "z": "b6932d514be1a226",
        "name": "",
        "topic": "esp32/humidity",
        "qos": "2",
        "datatype": "auto-detect",
        "broker": "10e78a89.5b4fd5",
        "nl": false,
        "rap": false,
        "inputs": 0,
        "x": 200,
        "y": 460,
        "wires": [
            [
                "e22d3f9cf3aa8120",
                "780a9ef3df654e6e"
            ]
        ]
    },
```

```
        "id": "e22d3f9cf3aa8120",
        "type": "ui_chart",
        "z": "b6932d514be1a226",
        "name": "",
        "group": "97ec16e08db6dda4",
        "order": 2,
        "width": 0,
        "height": 0,
        "label": "",
        "chartType": "line",
        "legend": "false",
        "xformat": "dd HH:mm",
        "interpolate": "linear",
        "nodata": "",
        "dot": false,
        "ymin": "",
        "ymax": "",
        "removeOlder": 1,
        "removeOlderPoints": "",
        "removeOlderUnit": "604800",
        "cutout": 0,
        "useOneColor": false,
        "useUTC": false,
        "colors": [
            "#1f77b4",
            "#aec7e8",
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "outputs": 1,
        "useDifferentColor": false,
        "className": "",
        "x": 389,
        "y": 484,
        "wires": [
            []
        ]
    },
    {
        "id": "780a9ef3df654e6e",
        "type": "ui_gauge",
        "z": "b6932d514be1a226",
"name": "",
        "group": "97ec16e08db6dda4",
        "order": 1,
        "width": 0,
        "height": 0,
        "gtype": "gage",
        "title": "Humidity",
        "label": "%",
        "format": "{{value}}",
        "min": "0",
        "max": "99",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
        "diff": false,
        "className": "",
        "x": 419,
        "y": 424,
        "wires": []
    },
```

```
        "id": "10e78a89.5b4fd5",
        "type": "mqtt-broker",
        "name": "",
        "broker": "localhost",
        "port": "1883",
        "clientid": "",
        "usetls": false,
        "compatmode": true,
        "keepalive": "60",
        "cleansession": true,
        "birthTopic": "",
        "birthQos": "0",
        "birthPayload": "",
        "closeTopic": "",
        "closeQos": "0",
        "closePayload": "",
        "willTopic": "",
        "willQos": "0",
        "willPayload": ""
    },
    {
        "id": "61285987.c20328",
        "type": "ui_group",
        "name": "Group 1",
        "tab": "e7c46d5e.a1283",
        "order": 1,
        "disp": false,
        "width": "6",
        "collapse": false,
        "className": ""
    },
```



```
    {
    {
        "id": "3897eee0be7b7e25",
        "type": "ui_group",
        "name": "Group 2",
        "tab": "e7c46d5e.a1283",
        "order": 2,
        "disp": false,
        "width": "6",
        "collapse": false,
        "className": ""
    },



    {
        "id": "97ec16e08db6dda4",
        "type": "ui_group",
        "name": "Group 3",
        "tab": "e7c46d5e.a1283",
        "order": 3,
        "disp": false,
        "width": "6",
        "collapse": false,
        "className": ""
    },
    {
        "id": "e7c46d5e.a1283",
        "type": "ui_tab",
        "name": "MeteoHome",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    }
]
```