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**

[Содержание](#_heading=h.7mt9x4b37ce) 2

[Введение](#_heading=h.1fob9te) 2

[Теоретическая и практическая главы](#_heading=h.hst8x8f1r8tz) 5

[Заключение](#_heading=h.dhwmmf3wq1kk) 15

[Список используемой литературы](#_heading=h.m8jvbwhv7ujg) 16

[Приложения](#_heading=h.s6k7t2lwcp0k) 17

# 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 128х64 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):

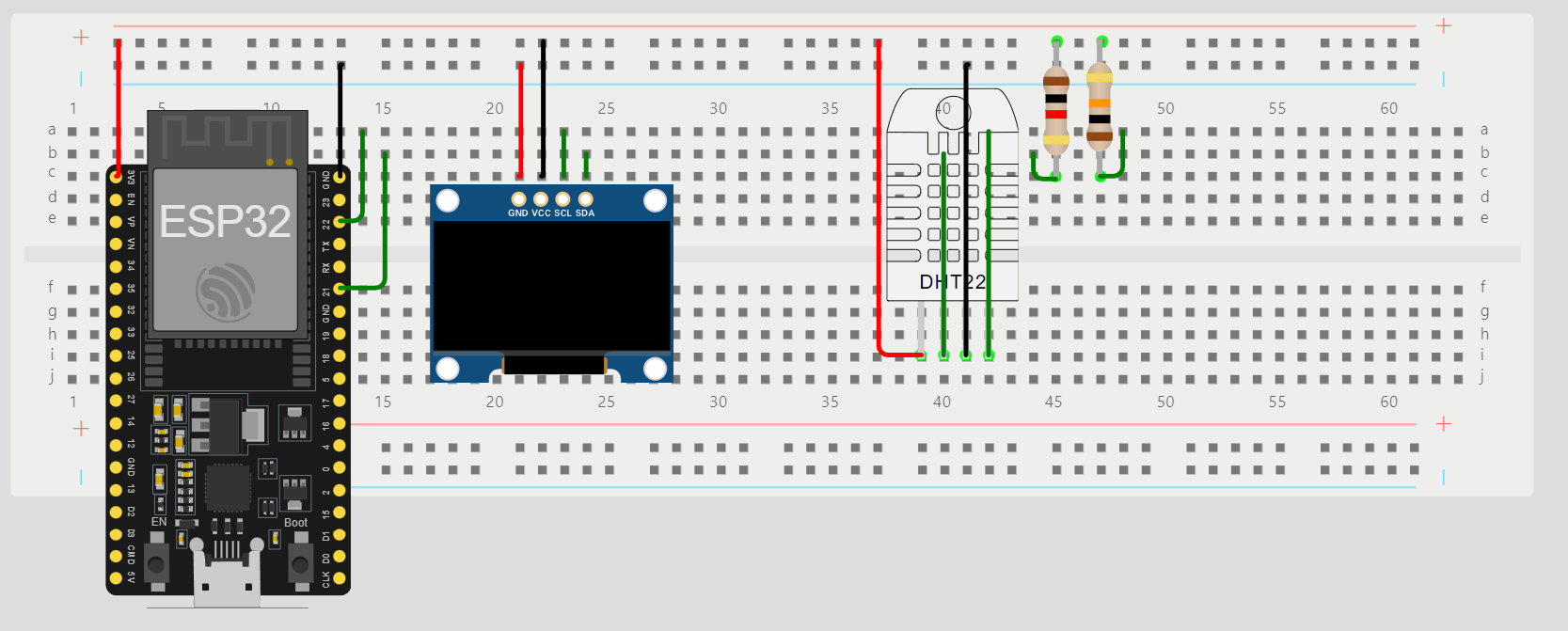
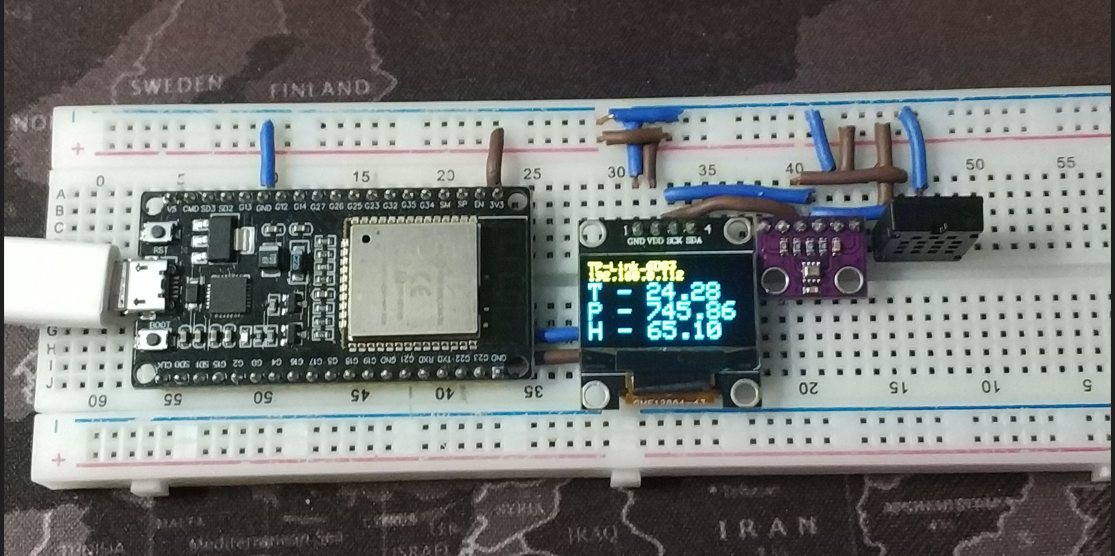


Фото собранного макета представлено на следующем рисунке:



1. **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 х 26 мм.  
- Вес: 9,8 г

* 1. OLED дисплей SSD1306 128х64

- Цвет – монохромный;

- Разрешение – 128 х 64;

- Графический чип – SSD1306;

- Интерфейс – I2C;

- Цвет дисплея – синий/желтый;

- Угол обзора> 160 °;

- Напряжение питания – от 3 до 5 В;

- Размер: 27х27х4 мм.

* 1. 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°С … +85°С (±0.01°С);

- Максимальная частота работы интерфейса I2C: 3.4MHz;

- Максимальная частота работы интерфейса SPI: 10 МГц;

- Размер модуля: 21 х 18 мм;

* 1. 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%;

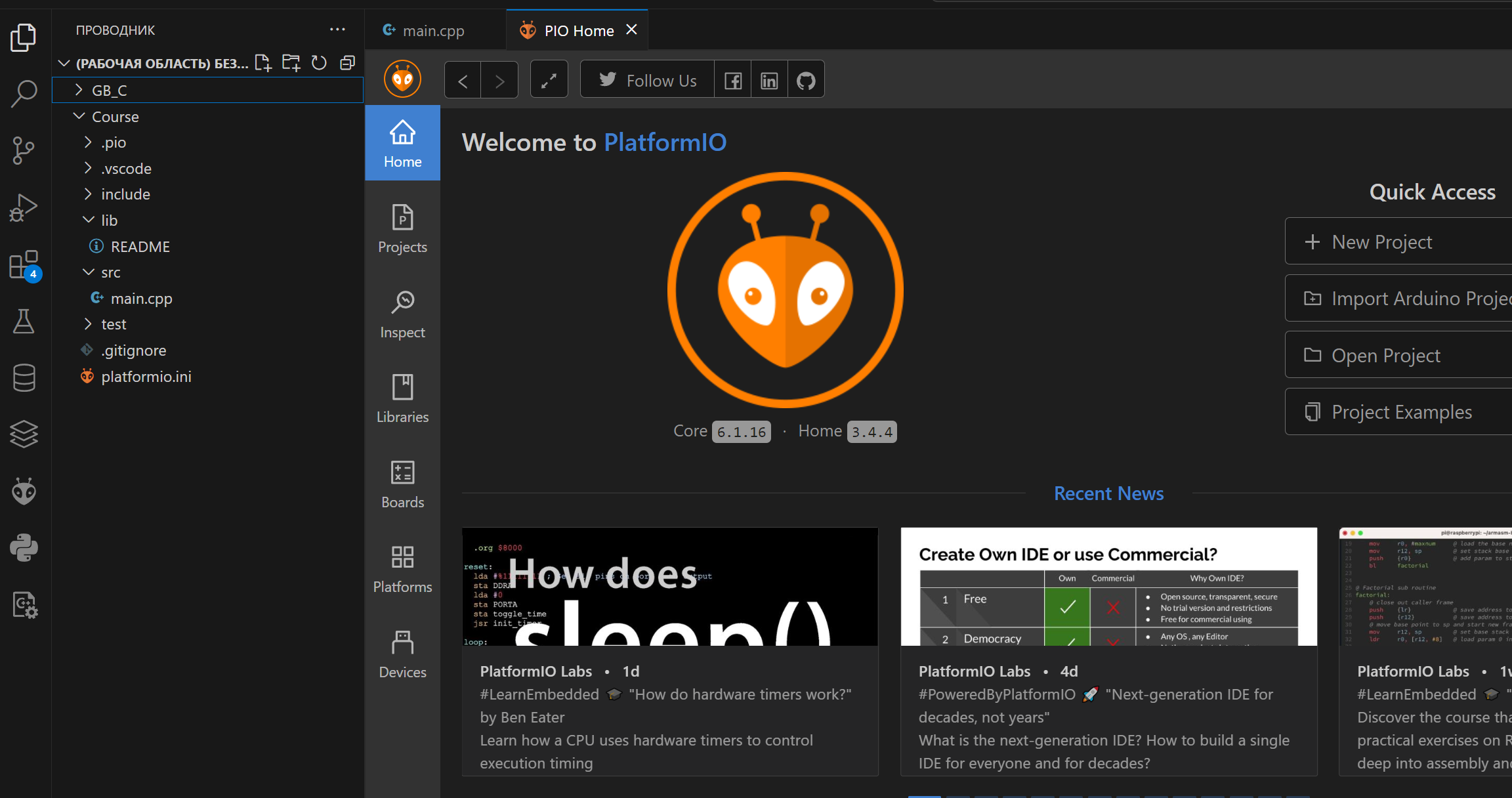
- Погрешность измерений влажности воздуха – 3%;

- Время между считыванием данных – 2 сек;

- Габариты – 15х12,1х4,5 мм.

1. **Программирование микроконтроллера**
   1. Подготовка среды программирования.

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

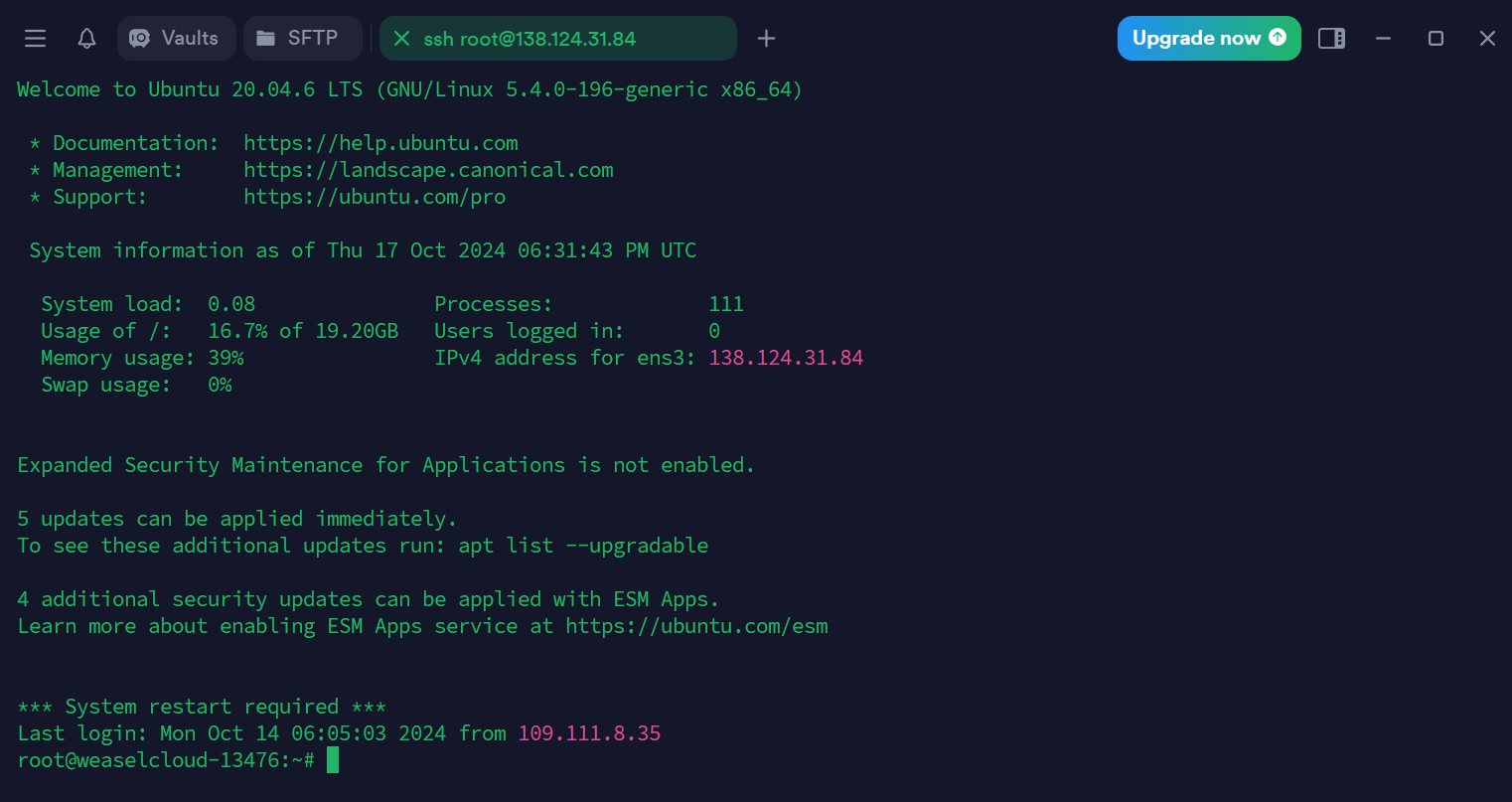


* 1. Подготовка библиотек для работы с периферией и подключаемыми устройствами.

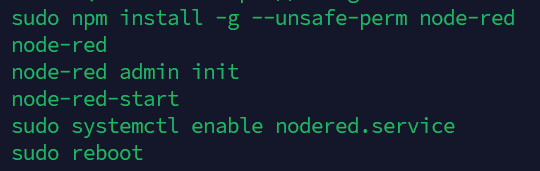
1. #include <WiFi.h> //работа с Wifi
2. #include <Wire.h> //работа с протоколом I2C
3. #include <PubSubClient.h> //работа с протоколом MQTT
4. #include <Adafruit\_Sensor.h> //работа с датчиками
5. #include <Adafruit\_BMP280.h> //работа с датчиком BMP280
6. #include <Adafruit\_AM2320.h> //работа с датчиком AM2320.h
7. #include <Adafruit\_SSD1306.h> //работа с дисплеем SSD1306
   1. Создание переменных и объектов для работы с оборудованием.
8. #define I2C\_display 0x3C
9. #define I2C\_BMP280 0x76
10. const char\* ssid = "TP-Link\_6D83";
11. const char\* password = "PASSWORD";
12. const char\* mqtt\_server = "138.124.31.84";
13. WiFiClient espClient;
14. PubSubClient client(espClient);
15. long lastMsg = 0;
16. char msg[50];
17. int value = 0;
18. float temperature = 0;
19. float pressure = 0;
20. float humidity =0;
21. Adafruit\_SSD1306 display = Adafruit\_SSD1306(128, 64, &Wire, -1);
22. Adafruit\_BMP280 bmp;
23. Adafruit\_AM2320 am2320 = Adafruit\_AM2320();
    1. Функция подключения Wifi с выводом сообщений в терминал.
24. void setup\_wifi() {
25. delay(10);
26. Serial.println();
27. Serial.print("Connecting to ");
28. Serial.println(ssid);
29. WiFi.begin(ssid, password);
30. while (WiFi.status() != WL\_CONNECTED) {
31. delay(500);
32. Serial.print(".");
33. }
34. Serial.println("");
35. Serial.println("WiFi connected");
36. Serial.println("IP address: ");
37. Serial.println(WiFi.localIP());
38. }
    1. Функция подключения к брокеру MQTT с выводом сообщений в терминал.
39. void reconnect() {
40. while (!client.connected()) {
41. Serial.print("Attempting MQTT connection...");
42. if (client.connect("ESP8266Client")) {
43. Serial.println("connected");
44. client.subscribe("esp32/output");
45. } else {
46. Serial.print("failed, rc=");
47. Serial.print(client.state());
48. Serial.println(" try again in 5 seconds");
49. delay(5000);
50. }
51. }
52. }
    1. Функция инициализации переменных, определения режимов работы датчиков, дисплея, wifi.
53. void setup() {
54. Serial.begin(9600);
55. Wire.begin();
56. am2320.begin();
57. display.begin(SSD1306\_SWITCHCAPVCC, I2C\_display);
58. display.clearDisplay();
59. display.setTextColor(WHITE);
60. bmp.begin(I2C\_BMP280);
61. bmp.setSampling(Adafruit\_BMP280::MODE\_NORMAL,     /\* Operating Mode. \*/
62. Adafruit\_BMP280::SAMPLING\_X2,     /\* Temp. oversampling \*/
63. Adafruit\_BMP280::SAMPLING\_X16,    /\* Pressure oversampling \*/
64. Adafruit\_BMP280::FILTER\_X16,      /\* Filtering. \*/
65. Adafruit\_BMP280::STANDBY\_MS\_500); /\* Standby time. \*/
66. setup\_wifi();
67. client.setServer(mqtt\_server, 1883);
68. }
    1. Основной цикл программы. Опрос датчиков, вывод на дисплей, проверка соединения с MQTT брокером и восстановление связи в случае обрыва, передача данных на удаленный сервер.
69. void loop() {
70. temperature = bmp.readTemperature();
71. pressure = bmp.readPressure()\*0.00750062;
72. humidity = am2320.readHumidity();
73. display.clearDisplay();
74. display.setCursor(0,0);
75. display.setTextSize(1);
76. display.println(ssid);
77. display.println(WiFi.localIP());
78. display.setTextSize(2);
79. display.print("T - ");
80. display.println(temperature);
81. display.print("P - ");
82. display.println(pressure);
83. display.print("H - ");
84. display.println(humidity);
85. display.display();
86. if (!client.connected()) {
87. reconnect();
88. }
89. client.loop();
90. long now = millis();
91. if (now - lastMsg > 5000) {
92. lastMsg = now;
93. char tempString[8];
94. dtostrf(temperature, 1, 2, tempString);
95. client.publish("esp32/temperature", tempString);
96. char pressString[8];
97. dtostrf(pressure, 1, 2, pressString);
98. client.publish("esp32/pressure", pressString);
99. char humidString[8];
100. dtostrf(humidity, 1, 2, humidString);
101. client.publish("esp32/humidity", humidString);
102. }
103. }

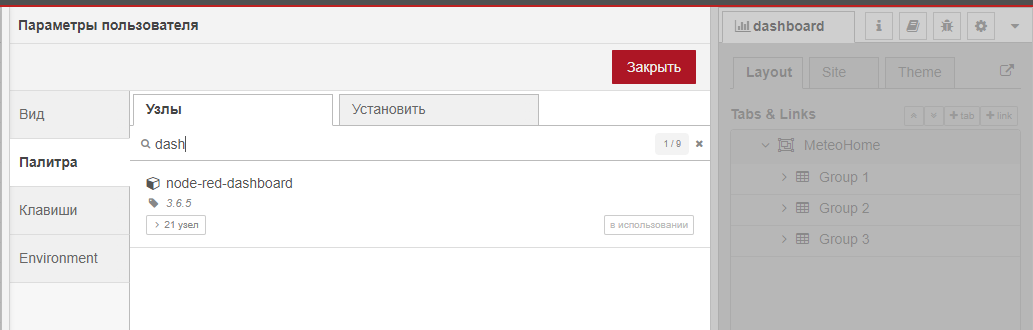
Полный листинг программы приведен в приложении 1.

1. **Подготовка серверной части**
   1. Подключение к удаленной VDS машине по протоколу SSH.

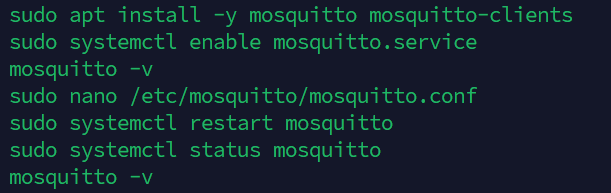


* 1. Подготовка Node-RED Dashboard .

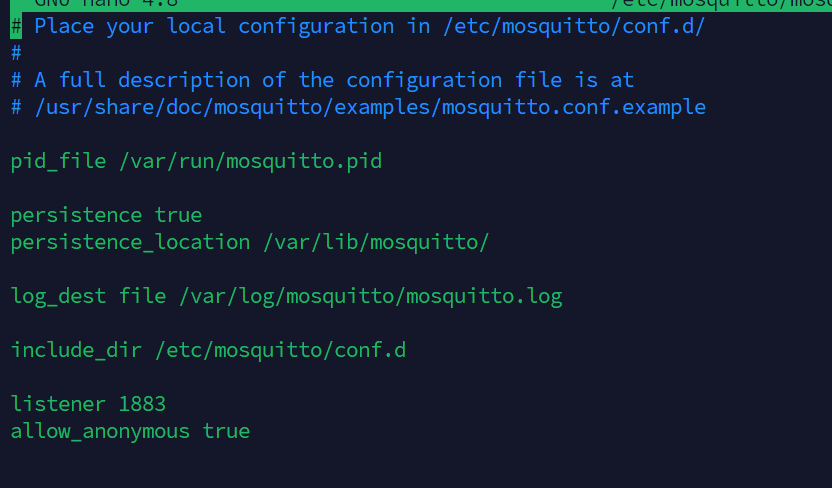


Установка Dashboard

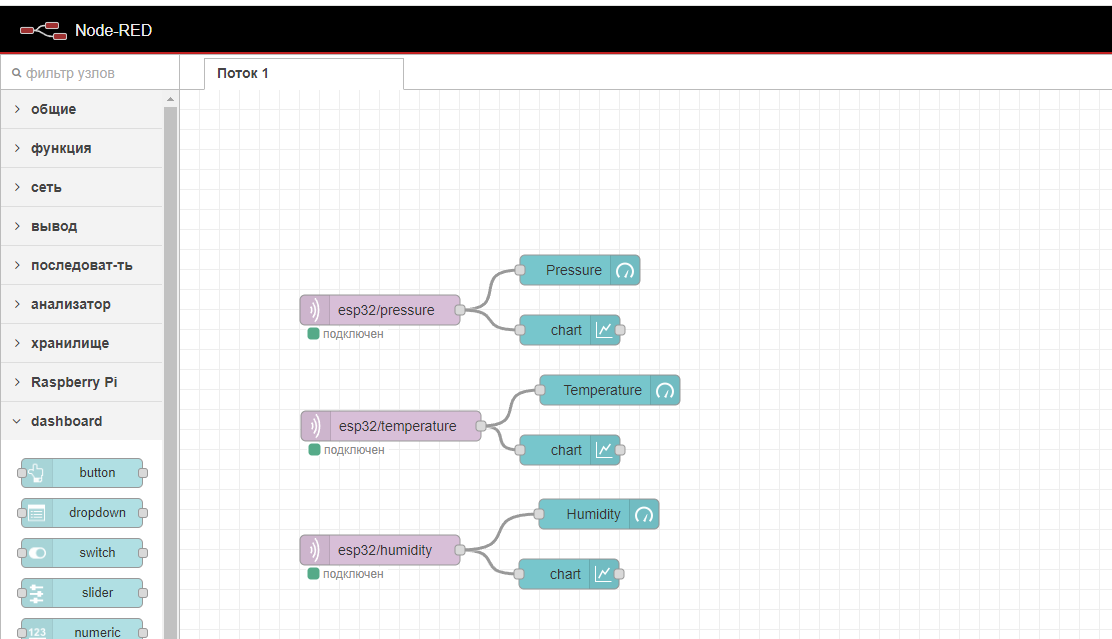
* 1. Подготовка MQTT-брокера Mosquitto .



Конфигурация Mosquitto доступ без пароля.

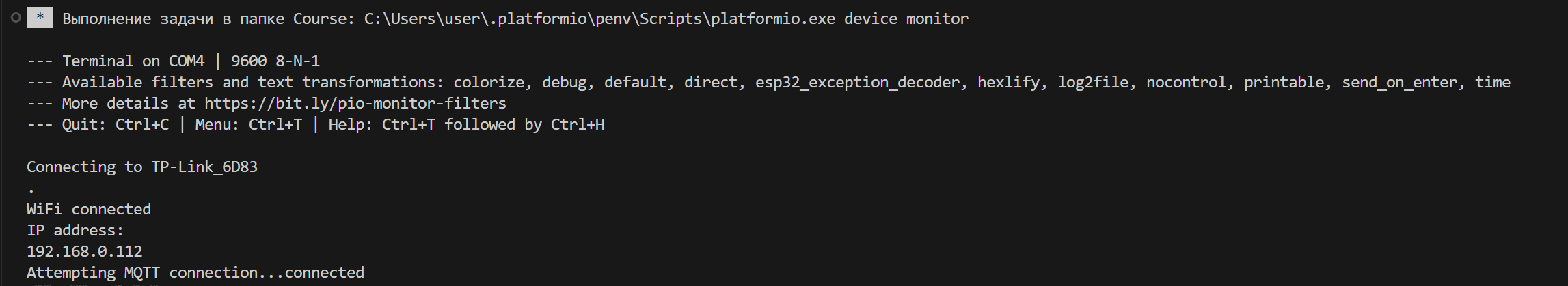


Создание потоков, подписка на топики ESP32.



1. **Описание работы системы**

После включения питания в терминале отображается следующая информация:

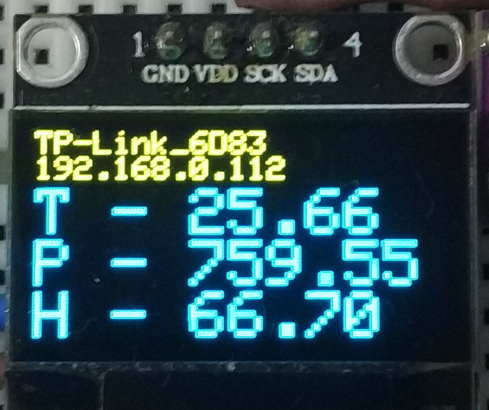


Служебные сообщения от котроллера ESP32 – номер СОМ порта, скорость СОМ порта и т.д.

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

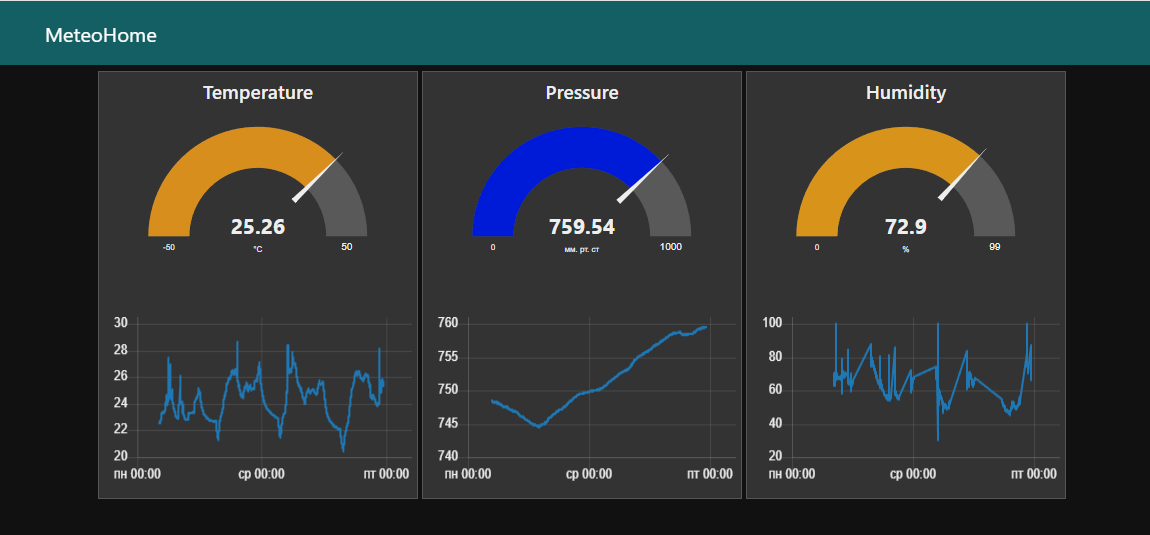
Текущая температура °С

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/*](http://138.124.31.84:1880/ui/) *ссылка на web сервер проекта.*
* [*https://github.com/JonLED1/GB\_C\_IOT/tree/main/Final\_work*](https://github.com/JonLED1/GB_C_IOT/tree/main/Final_work) *ссылка на Github проекта*
* [*https://wokwi.com*](https://wokwi.com) *Эмулятор ESP32*
* [*https://randomnerdtutorials.com/*](https://randomnerdtutorials.com/) *Уроки и примеры с микроконтроллером ESP32*
* [*https://nodered.org/*](https://nodered.org/) *официальный сайт Node-RED*
* [*https://3d-diy.ru/*](https://3d-diy.ru/) *описание технической части оборудования*
* [*https://younglinux.info/*](https://younglinux.info/) *Уроки Linux, Bash, C/C++*
* [*https://alexgyver.ru/*](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,          "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"              ]          ]      }, | "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": "",    {          "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      }  ] |