
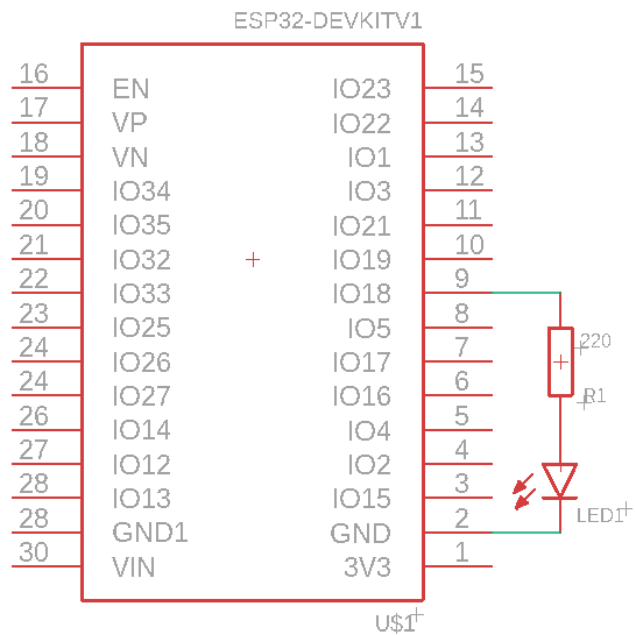
	Instytut Informatyki Politechniki Śląskiej Zespół Mikroinformatyki i Teorii Automatów Cyfrowych			
Rok akademicki:	Rodzaj studiów:	Przedmiot:	Grupa	Sekcja
2022/2023	SSI	SMIW	3	2
Imię:	Bartłomiej	Prowadzący:	OA	
Nazwisko:	Gordon			
<h2><i>Raport końcowy</i></h2>				
Temat projektu: <div style="text-align: center; margin-top: 20px;"> <h1>Rozproszony system pomiaru temperatury w domu z wykorzystaniem lokalnej sieci.</h1> </div>				
Data oddania: dd/mm/rrrr		<h2>10.02.2023r</h2>		

1.Opis tematu

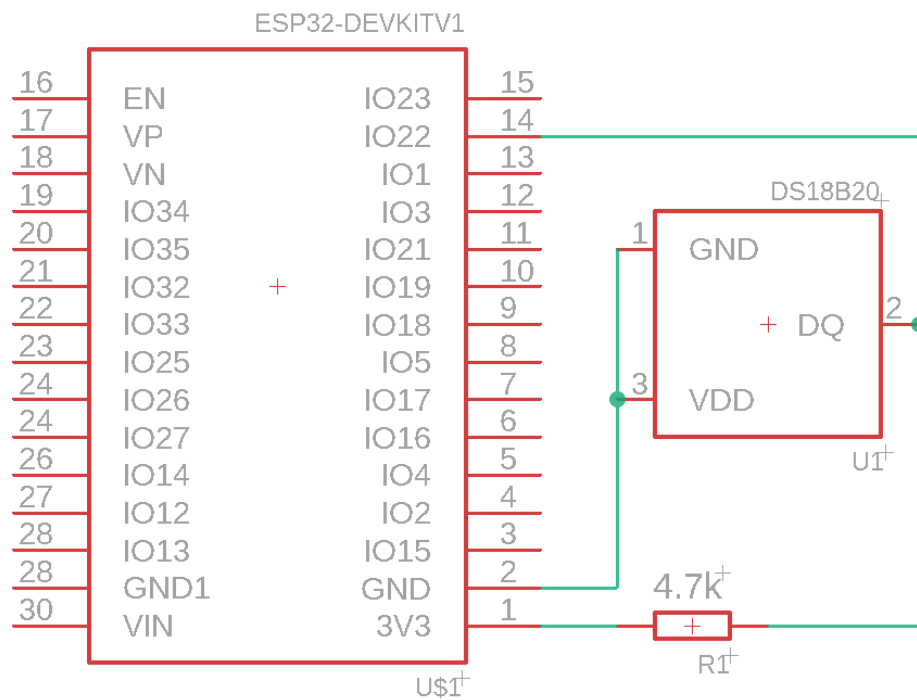
Głównym założeniem projektu było zbudowanie trzech płytek zawierających ESP-WROOM-32 z czego jedna jest skonfigurowana jako 'server' i ustawiona jako AccesPoint hostujący stronę internetową wyświetlającą wyniki pomiaru temperatury zebrane z pozostałych dwóch płytek skonfigurowanych jako 'sender' wysyłających je w postaci eventów. Płytką 'server' posiada czerwoną diodę LED sygnalizującą konfigurację serwera i AccesPointa po jej włączeniu. Płytką 'sender' posiada czujnik temperatury DS18B20.

2. Schematy i płytki

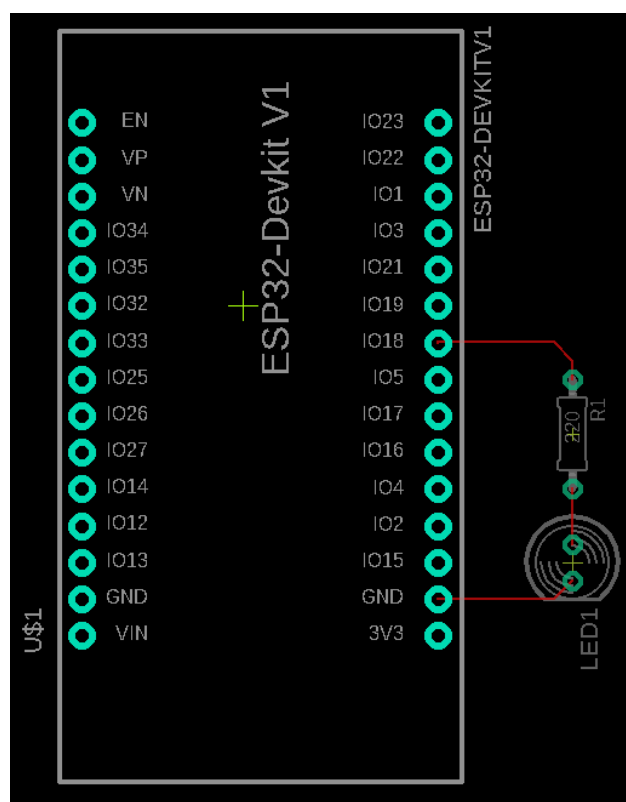
Schemat 'server'



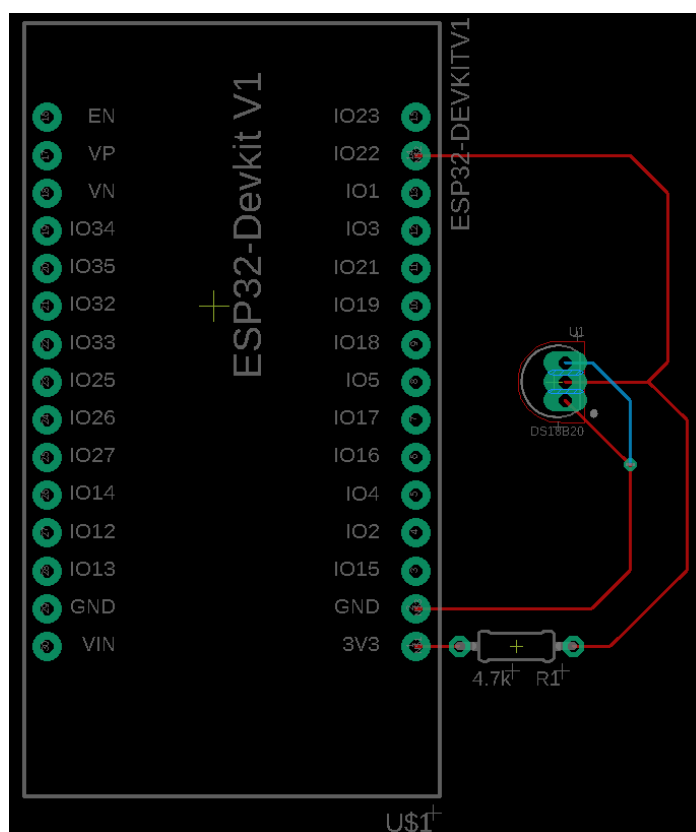
Schemat 'sender'



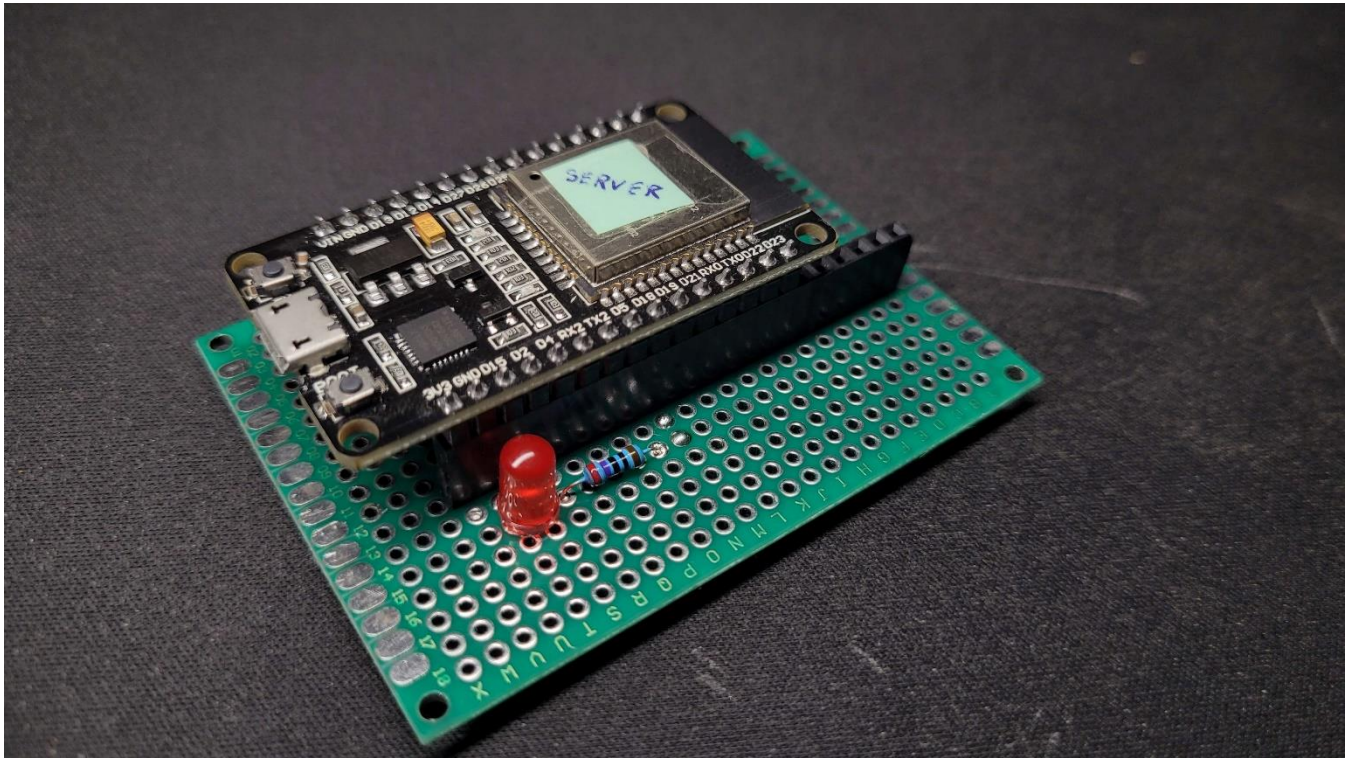
Płytki 'server'



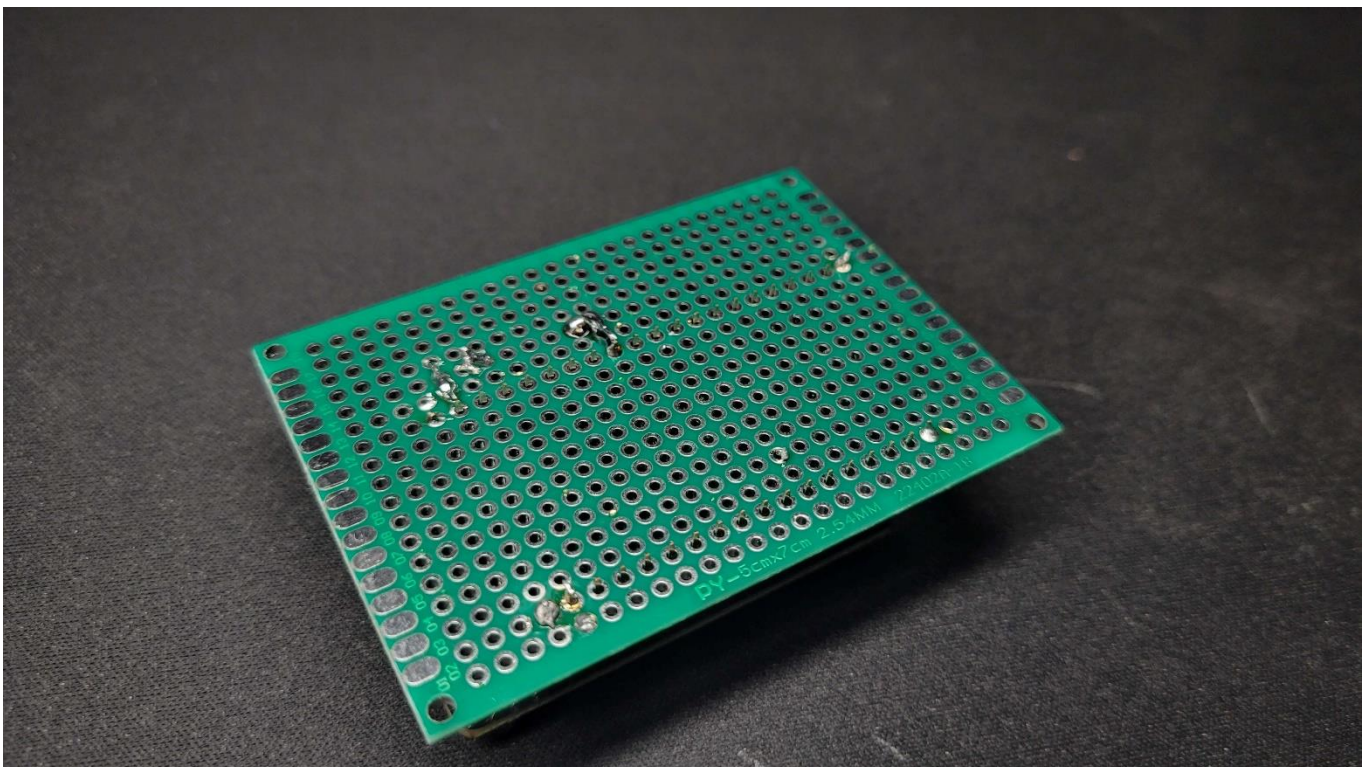
Płytki 'sender'



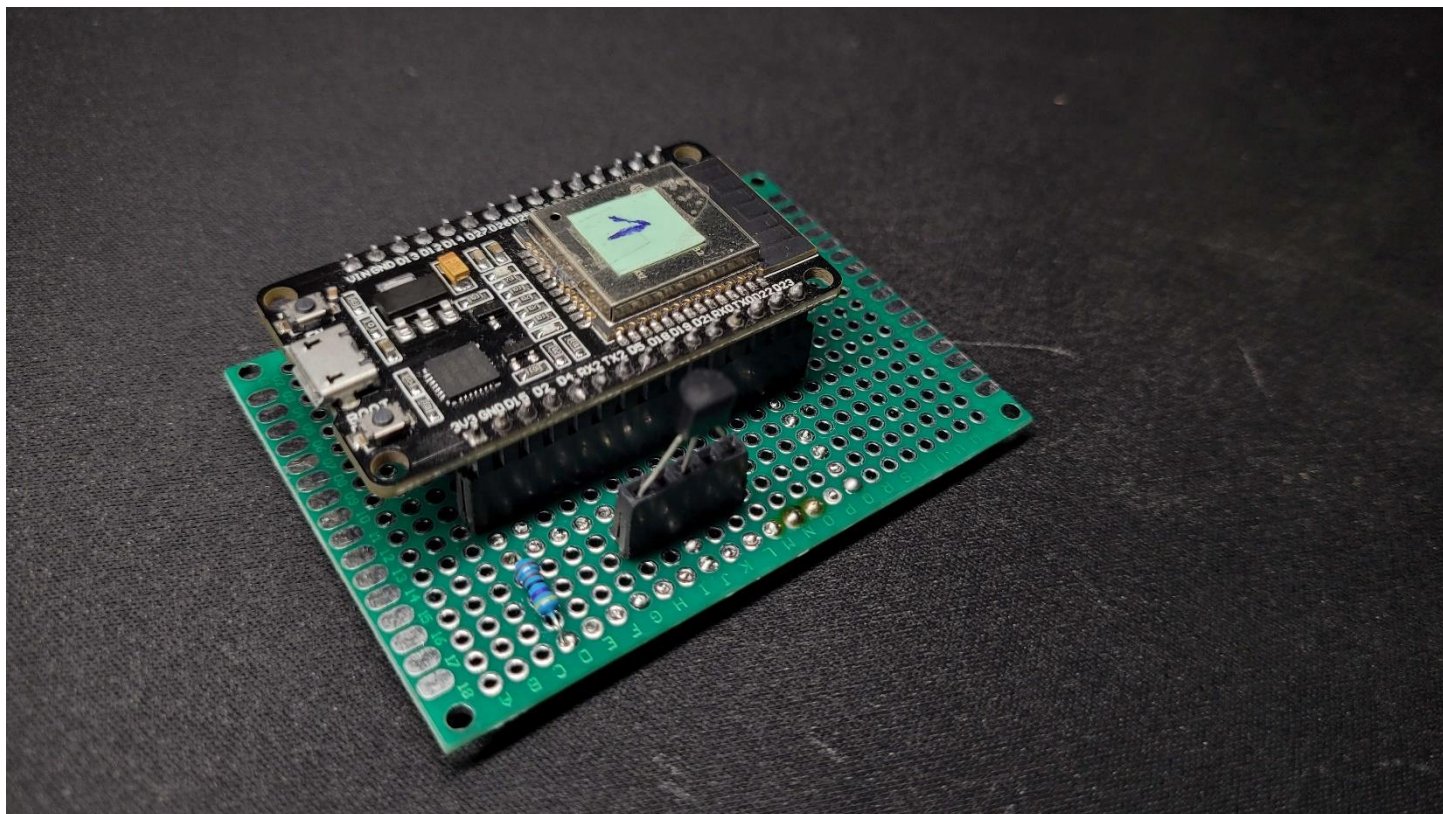
3. Zdjęcia płytek



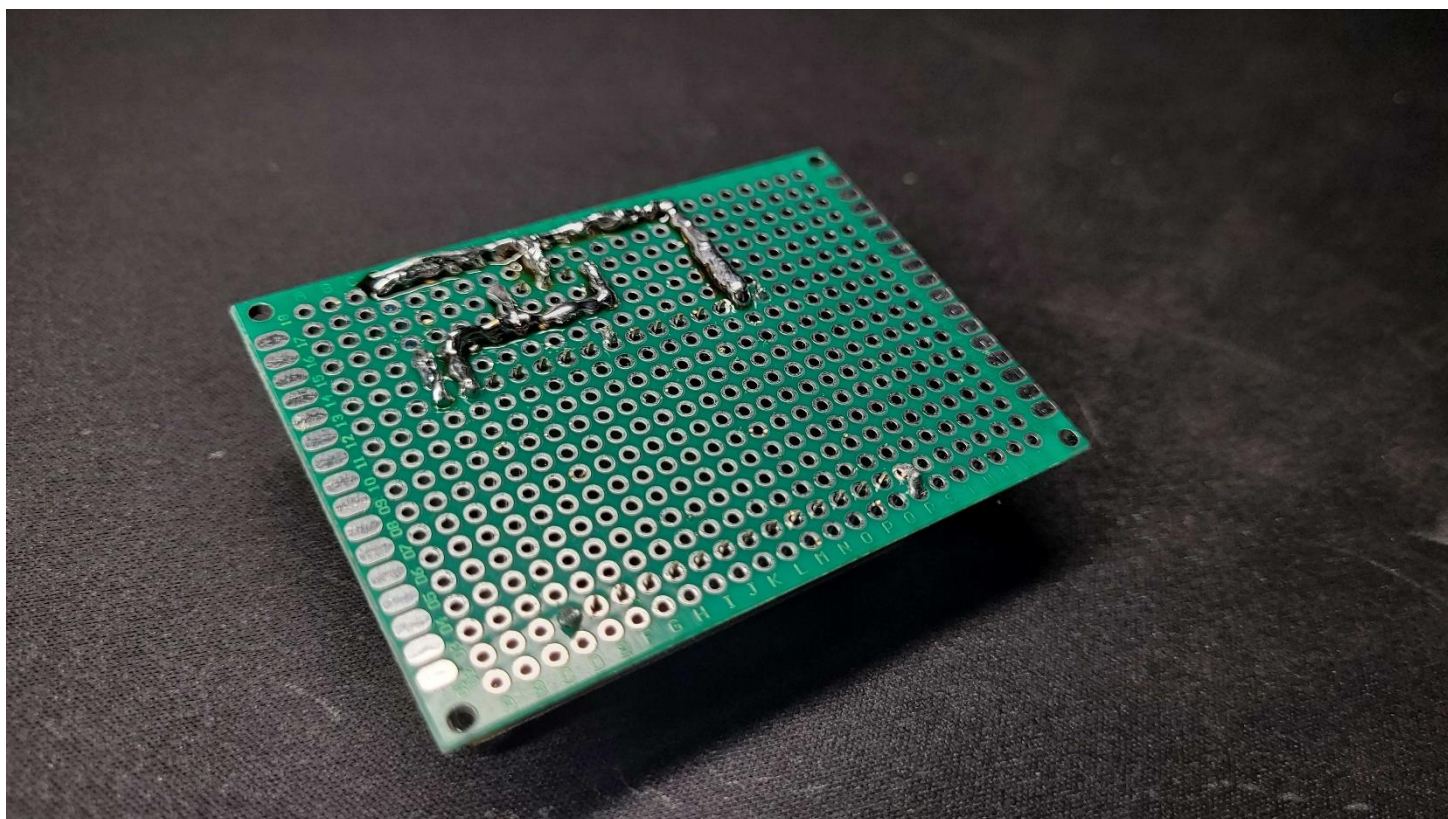
Przednia strona płytki 'sever'



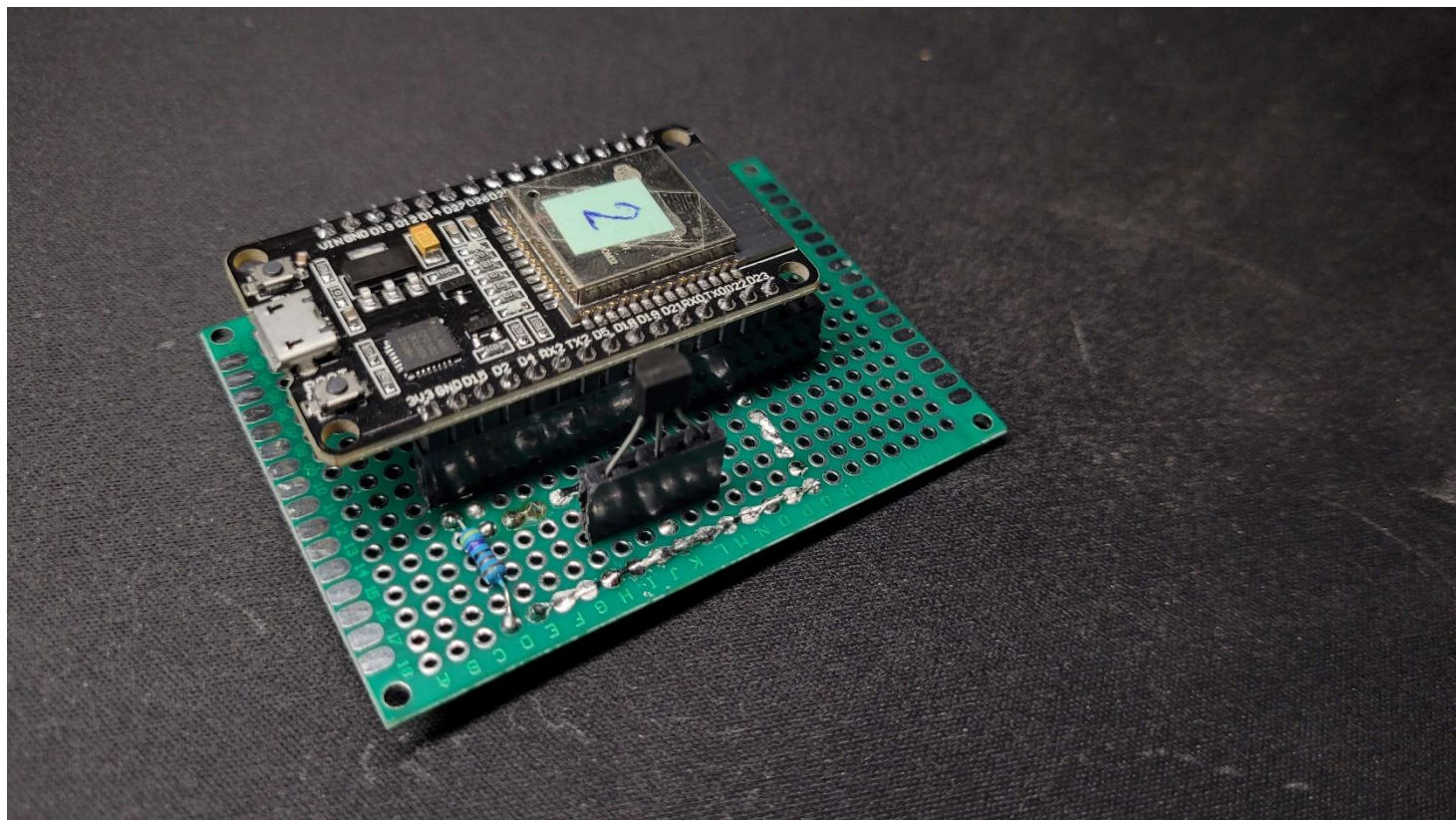
Tylna strona płytki 'sever'



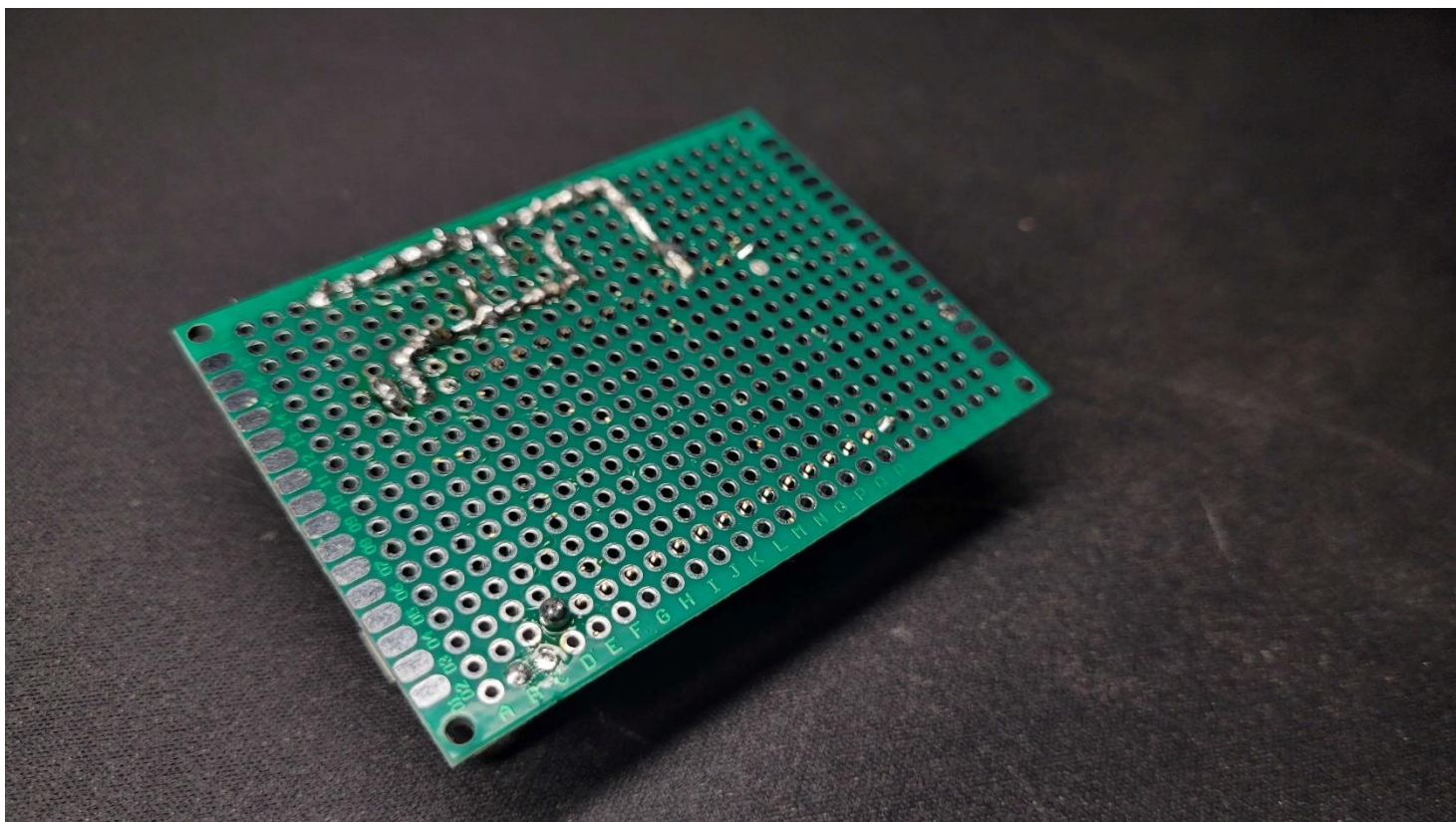
Przednia strona płytki 'sender1'



Tylna strona płytki 'sender1'



Przednia strona płytki 'sender2'



Tylna strona płytki 'sender2'

4.Kod

Kod płytki 'server'

```
#include <esp_now.h>
#include <WiFi.h>
#include "ESPAsyncWebServer.h"
#include "html.h"
#include <Arduino_JSON.h>

#define LED_PIN 18

typedef struct struct_message {
    int id;
    float temp;
    unsigned int readingId;
} struct_message;

const char* ssid = ""; // :)
const char* password = ""; // :)
struct_message incomingReadings;
JSONVar board;
AsyncWebServer server(80);
AsyncEventSource events("/events");

void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int len)
{
    memcpy(&incomingReadings, incomingData, sizeof(incomingReadings));
    board["id"] = incomingReadings.id;
    board["temperature"] = incomingReadings.temp;
    board["readingId"] = String(incomingReadings.readingId);
    String jsonString = JSON.stringify(board);
    events.send(jsonString.c_str(), "new_readings", millis());
}

void setup() {
    pinMode(LED_PIN, OUTPUT);
    digitalWrite(LED_PIN, HIGH);

    WiFi.mode(WIFI_AP_STA);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(1000);
        Serial.println("Setting as a Wi-Fi Station..");
    }
}
```



```

if (esp_now_init() != ESP_OK)
{
    Serial.println("Error initializing ESP-NOW");
    return;
}
esp_now_register_recv_cb(OnDataRecv);

server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html);
});
events.onConnect([](AsyncEventSourceClient *client){
    if(client->lastId())
    {
        Serial.printf("Client reconnected! Last message ID that it got is: %u\n",
client->lastId());
    }
    client->send("hello!", NULL, millis(), 10000);
});

server.addHandler(&events);
server.begin();
digitalWrite(LED_PIN, LOW);
}

void loop() {
    static unsigned long lastEventTime = millis();
    static const unsigned long EVENT_INTERVAL_MS = 5000;
    if ((millis() - lastEventTime) > EVENT_INTERVAL_MS)
    {
        events.send("ping",NULL,millis());
        lastEventTime = millis();
    }
}

```

Kod płytki 'sender'

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <esp_now.h>
#include <esp_wifi.h>
#include <WiFi.h>

#define BOARD_ID 2
const int oneWireBus = 22;

OneWire oneWire(oneWireBus);
DallasTemperature sensors(&oneWire);
uint8_t broadcastAddress[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00}; // ;)
constexpr char WIFI_SSID[] = ""; // :)

typedef struct struct_message {
    int id;
    float temp;
    int readingId;
} struct_message;

struct_message myData;

unsigned long previousMillis = 0;
const long interval = 3000;
unsigned int readingId = 0;

int32_t getWiFiChannel(const char *ssid) {
    if (int32_t n = WiFi.scanNetworks()) {
        for (uint8_t i=0; i<n; i++) {
            if (!strcmp(ssid, WiFi.SSID(i).c_str())) {
                return WiFi.channel(i);
            }
        }
    }
    return 0;
}

float readTemperature() {
    sensors.requestTemperatures();
    float temperatureC = sensors.getTempCByIndex(0);

    if (isnan(temperatureC) || temperatureC == -127) {
        Serial.println("Failed to read temperature!");
        return 0;
    }
}
```

```

    else {
        Serial.println(temperatureC);
        return temperatureC;
    }
}

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
    Serial.print("\r\nLast Packet Send Status:\t");
    Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" :
"Delivery Fail");
}

void setup() {
    Serial.begin(9600);
    sensors.begin();

    WiFi.mode(WIFI_STA);
    int32_t channel = getWiFiChannel(WIFI_SSID);
    esp_wifi_set_promiscuous(true);
    esp_wifi_set_channel(channel, WIFI_SECOND_CHAN_NONE);
    esp_wifi_set_promiscuous(false);

    if (esp_now_init() != ESP_OK) {
        Serial.println("Error initializing ESP-NOW");
        return;
    }
    esp_now_register_send_cb(OnDataSent);
    esp_now_peer_info_t peerInfo;
    memcpy(peerInfo.peer_addr, broadcastAddress, 6);
    peerInfo.encrypt = false;

    if (esp_now_add_peer(&peerInfo) != ESP_OK){
        Serial.println("Failed to add peer");
        return;
    }
}

void loop() {

    unsigned long currentMillis = millis();

    if (currentMillis - previousMillis >= interval) {
        previousMillis = currentMillis;
        myData.id = BOARD_ID;
        myData.temp = readTemperature();
        myData.readingId = readingId++;
    }
}

```



```
    esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));
    if (result == ESP_OK) {
        Serial.println("Sent with success");
    }
    else {
        Serial.println("Error sending the data");
    }
}
}
```

Kod strony internetowej

```
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html>
<head>
  <title>Temperature site</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href='http://fonts.googleapis.com/css?family=Roboto' rel='stylesheet'
type='text/css'>
  <style>
    html {
      font-family: 'Roboto', sans-serif;
      display: inline-block;
      text-align: center;
    }

    p {
      font-size: 1.2rem;
    }

    body {
      margin: 0;
    }

    .topnav {
      overflow: hidden;
      background-color: #1a3c22;
      color: rgb(255, 255, 255);
      font-size: 1.7rem;
    }

    .content {
      padding: 30px;
    }

    .card {
      background-color: white;
      box-shadow: 2px 2px 25px 1px rgba(81, 81, 81, 0.5);
      border-radius: 25px;
      border: 2px solid #c7c7c7;
      padding: 20px;
      width: 200px;
      height: 150px;
    }

    .cards {
      max-width: 700px;
```



```

        <div class="card temperature">
            <h4>Sensor 2</h4><p><span class="reading"><span id="t2">...</span>
&deg;C</span></p><p class="packet">Reading ID: <span id="rt2"></span></p>
        </div>

        <div class="card temperature">
            <h4>Sensor x</h4><p><span class="reading"><span id="t3"></span>---
</span></p><p class="packet">Preview additional board<span id="rt3"></span></p>
        </div>

    </div>
</div>

<script>
if (!!window.EventSource) {
    var source = new EventSource('/events');

    source.addEventListener('new_readings', function(e) {
        console.log("new_readings", e.data);
        var obj = JSON.parse(e.data);
        document.getElementById("t"+obj.id).innerHTML = obj.temperature.toFixed(2);
        document.getElementById("rt"+obj.id).innerHTML = obj.readingId;

    }, false);
}
</script>

</body>
</html>
)rawliteral";

```

5. Przykład działania



Temperature measurements at home

Sensor 1

21.56 °C

Reading ID: 21

Sensor 2

26.31 °C

Reading ID: 16

Sensor x

[Preview additional board](#)

6. Kosztorys

3x ESP-WROOM-32	88,95
2x DS18B20	39,8
3x Płytki uniwersalna 50x70mm	13,5
Podstawki pod ESP32	9
Czerwony LED	0
Dostawa	25
Suma	176,25zł

7. Wnioski

Nad projektem spędziłem sporo czasu, lecz uważam, że nie był to czas stracony! Poznałem dużo nowych rzeczy, które przydadzą mi się nie tylko przy systemach mikroprocesorowych. Poznałem w jaki sposób działa przesyłanie danych przy pomocy JSON w urządzeniach IOT a także nauczyłem się lutować jak i również zapoznałem się z IDE od Arduino.