


SPRAWOZDANIE NR 2			
Nazwa ćwiczenia	System sterowania ogrzewaniem w pomieszczeniu		 <b>POLITECHNIKA BYDGOSKA</b> Wydział Telekomunikacji, Informatyki i Elektrotechniki
Przedmiot	Internet rzeczy i systemy wbudowane		
Student grupa	Adrian Gwiazdowski grupa I		
Data ćwiczeń	21.10.22-2.12.22	13.12.22	Data oddania sprawozdania
Ocena, uwagi			

## 1. Cel Ćwiczenia

Celem laboratorium jest opracowanie oprogramowania i zbudowanie prototypu systemu do sterowania ogrzewaniem w pomieszczeniu. System musi monitorować temperaturę w pomieszczeniu, wykrywać otwarcie okien i poprzez aplikację pozwalać na ustawienie odpowiedniej temperatury.

## 2. Przebieg

### Zadanie 1:

Zbudowanie prototypu urządzenia z zachowaniem następujących zaleceń:

- Czujnik otwarcia drzwi i okien musi być połączony z mikrokontrolerem ESP32
- Czujnik środowiskowy musi być połączony z mikrokomputerem Raspberry Pi 4B
- Dioda RGB ma symulować głowicę termostatyczną i może być połączona z dowolnym urządzeniem.



### 2.1. Zadanie 2:

Zaimplementowanie oprogramowania na mikrokontroler ESP32 w jednym z obsługiwanych języków. Oprogramowaniu musi spełniać następujące funkcje:

1. Odczyt stanu czujnika otwarcia drzwi i okien;
2. Komunikacja z raspberry pi zawierająca informację o otwarciu i zamknięciu okna w wybranym protokole komunikacyjnym;
3. Obsługa diody RGB zapewniając płynną zmianę koloru w zależności od mocy grzania w skali od 1 do 5 oraz świecąc niebieskim światłem w momencie wyłączenia grzania.

```
1  import time
2  import ds18x20
3  import esp
4  import machine
5  import micropython
6  import network
7  import onewire
8  import ubinascii
9  from machine import Pin
10 from umqttsimple import MQTTClient
11 import gc
12
13 esp.osdebug(None)
14 gc.collect()
15
16 state = machine.Pin(25, machine.Pin.IN)
17 windowState = "0" # Z - zamknięte, 0 - Otwarte
18
19 ssid = 'POCO X3 NFC'
20 password = '12345677'
21 mqtt_server = '192.168.234.3'
22
23 client_id = ubinascii.hexlify(machine.unique_id())
24
25 topic_pub_temp = b'sensor/temp'
26
27 last_message = 0
28 message_interval = 5
29
30 station = network.WLAN(network.STA_IF)
31 station.active(True)
32 station.connect(ssid, password)
33 print('Connection successful')
```

```

38 def connect_mqtt():
39     global client_id, mqtt_server
40     client = MQTTClient(client_id, mqtt_server)
41     client.connect()
42     print('Connected to %s MQTT broker' % (mqtt_server))
43     return client
44
45
46 def restart_and_reconnect():
47     print('Failed to connect to MQTT broker. Reconnecting...')
48     time.sleep(10)
49     machine.reset()
50
51
52     try:
53         client = connect_mqtt()
54     except OSError as e:
55         restart_and_reconnect()
56
57 while True:
58     try:
59         if (time.time() - last_message) > message_interval:
60             if state.value() == 0:
61                 windowState = "0"
62             else:
63                 windowState = "1"
64             client.publish(topic_pub_temp, windowState)
65             last_message = time.time()
66     except OSError as e:
67         restart_and_reconnect()

```

1. Za odczyt stanu czujnika otwarcia drzwi i okien odpowiadają linie kodu 17. Łączy on zmienną state z pinem nr 25 na płytce ESP32 oraz ustawia go jako input. Dzięki temu możemy monitorować stan czujnika otwarcia.
2. Komunikacja z raspberry pi odbywa się przez Wi-fi dzięki narzędziu mosquitto. Linie kodu odpowiedzialne za komunikację:
  - 19-21** Zmienne zawierające dane sieci Wi-fi oraz adres IP raspberry pi.
  - 23** Zmienna zawierająca identyfikator ESP32.
  - 25** Zmienna zawierająca nazwę tematu pod którym będzie publikowany stan czujnika.
  - 30-33** Odpowiada za połączenie z Wi-fi.

38-43 Jest to funkcja, która ustanawia połączenie z brokerem na raspberry pi.  
46-49 Jest to funkcja odpowiadająca za reset ESP32 w razie błędu połączenia klienta MQTT.

52-55 Wykorzystanie powyższych funkcji w celu nawiązania połączenia.

57-67 Pętla wysyłająca stan czujnika na wcześniej ustawiony temat w interwale 5s.

Kod na raspberry pi odpowiedzialny za subskrypcje tematu oraz odbieranie wiadomości:

```
20 Broker = "192.168.234.3"
21 sub_topic = "sensor/temp"
```

```
46 def on_connect(client, userdata, flags, rc):
47     print("Connected with result code " + str(rc))
48     client.subscribe(sub_topic)
49
50
51 def on_message(client, userdata, msg):
52     global windowState
53     message = str(msg.payload)
54     message = message.replace("b", "")
55     message = message.replace("'", "")
56     if int(message) == 0:
57         windowState = "Zamkniete"
58         lbl44.config(text="Zamkniete")
59     else:
60         windowState = "Otwarte"
61         lbl44.config(text="Otwarte")
62
63
64 def connection():
65     client = mqtt.Client()
66     client.on_connect = on_connect
67     client.on_message = on_message
68     client.connect(Broker, 1883, 60)
69     client.loop_start()
```

3. Obsługa diody odbywa się przez raspberry pi. Kod odpowiedzialny za obsługę:

```
13 pixels = neopixel.NeoPixel(board.D10, 7)
```

13 Zmienna inicjalizująca diodę RGB, potrzebuje numeru pinu oraz rodzaju diody.

```
85 def temperatureChange(temp):
86     global heatLevel
87     if temp <= 0:
88         pixels.fill((0, 0, 255))
89         heatLevel = 0
90         lbl22.config(text=0)
91     elif temp > 0 and temp <= 6:
92         pixels.fill((0, 255, 0))
93         heatLevel = 1
94         lbl22.config(text=1)
95     elif temp > 6 and temp <= 12:
96         pixels.fill((180, 255, 0))
97         heatLevel = 2
98         lbl22.config(text=2)
99     elif temp > 12 and temp <= 18:
100         pixels.fill((255, 255, 0))
101         heatLevel = 3
102         lbl22.config(text=3)
103     elif temp > 18 and temp <= 24:
104         pixels.fill((255, 144, 0))
105         heatLevel = 4
106         lbl22.config(text=4)
107     elif temp > 24:
108         pixels.fill((255, 0, 0))
109         heatLevel = 5
110         lbl22.config(text=5)
111     pixels.write()
```

**85-111** Funkcja odpowiedzialna za zapalenie diody na kolor zależny od przedziału temperatury.

### 3.3. Zadanie 3:

Zaimplementowanie oprogramowania na mikrokomputer Raspberry Pi w jednym z obsługiwanych języków programowania. Oprogramowaniu musi spełniać następujące funkcje:

1. Obsługa czujnika środowiskowego BME680 – odczyt temperatury.
2. Ponadto na mikrokomputerze musi powstać oprogramowanie dla użytkownika pozwalające monitorować temperaturę w pomieszczeniu, wyświetlać informację o poziomie ogrzewania, ustawiać wybraną temperaturę w pomieszczeniu oraz wyświetlać archiwalne temperatury wraz z poziomem otwarcia grzejników i informacją o otwarciu i

zamknięciu okna. Oprogramowanie musi również umożliwiać podłączenie więcej niż jednego modułu ESP32 do mikrokomputera.

## 1. Obsługa czujnika środowiskowego BME680 – odczyt temperatury.

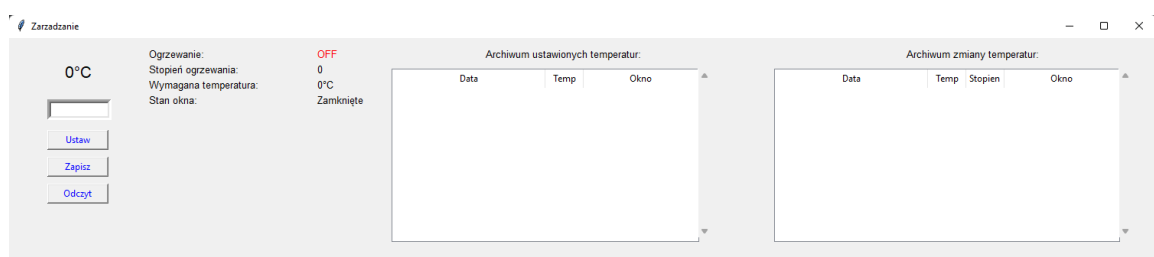
```
23 sensor = bme680.BME680()
24 sensor.set_temperature_oversample(bme680.OS_8X)
25 sensor.set_filter(bme680.FILTER_SIZE_3)
26
27
28 def sens():
29     while True:
30         if sensor.get_sensor_data():
31             temp_sensor_1 = (sensor.data.temperature)
32             global temp_sensor
33             temp_sensor = temp_sensor_1
34             if float(temp_desire) <= temp_sensor and state == True:
35                 temperatureLed(0)
36             elif float(temp_desire) > temp_sensor and state == False:
37                 temperatureLed(int(temp_desire))
38             lbl0.config(text=str(temp_sensor) + '°C')
39             date = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
40             tree1.insert('', '0', values=(date, str(temp_sensor) + '°C', heatLevel, windowState))
41             list1.append((str(date), str(temp_sensor) + '°C', heatLevel, windowState))
42             time.sleep(5)
```

**23-25** Opowiadająca za inicjalizacją czujnika.

**28-42** Jest to funkcja uruchamiająca pętlę, która pobiera temperaturę z czujnika w interwale 5s oraz wykonuje także niezbędny kod związany z interfejsem graficznym programu.

## 2. Graficzny interfejs użytkownika został zbudowany za pomocą biblioteki Tkinter. Elementy użyte:

- a. Label
- b. Entry
- c. Button
- d. Treeview



0°C

Ustaw

Zapisz

Odczyt

Ogrzewanie:

Stopień ogrzewania:

Wymagana temperatura:

Stan okna:

OFF

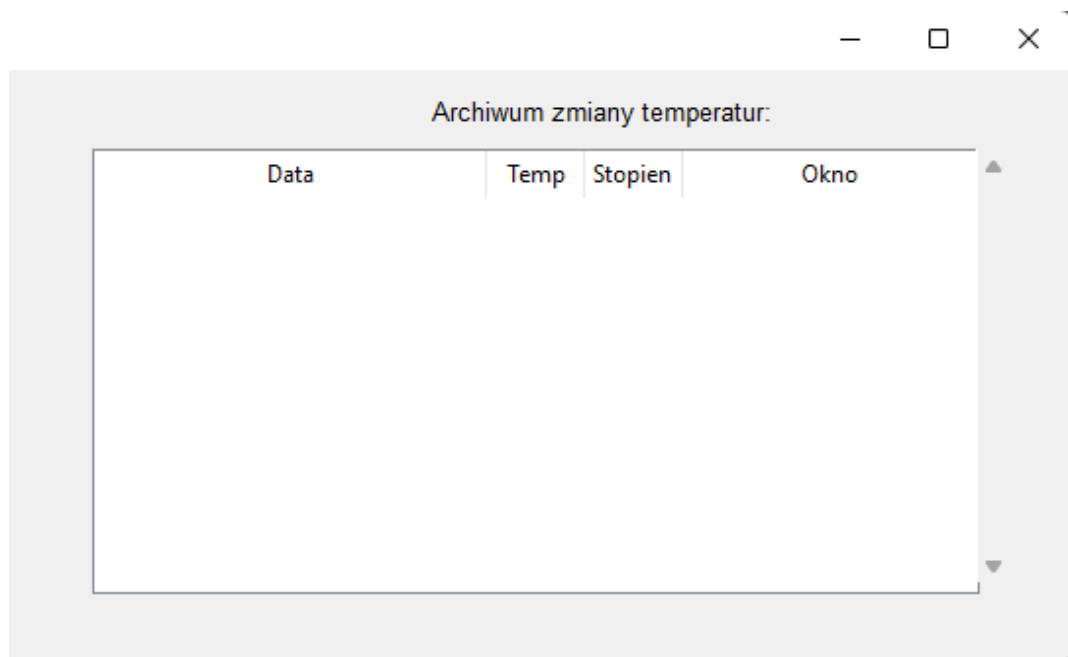
0

0°C

Zamknięte

[illegible]





### 3. Wnioski

Obsługa czujników oraz diod odbyła się bez większych problemów dzięki bibliotekom dołączonym do nich. Interfejs także nie sprawił większych problemów. Jedyną przeszkodą było wymyślenie oraz implementacja algorytmu sterowania systemem, a także połączenie MQTT.