Akademia Techniczno-Informatyczna w Naukach Stosowanych we Wrocławiu

Skrypt do zajęć Projekt specjalnościowy platformy IoT opartej o Rasberry Pi (Projekt 1)

Paweł Dobrowolski

Wrocław 2024

Skrypt przygotowany w ramach projektu „Kształcenie przyszłości: internet rzeczy w zrównoważonej automatyce i robotyce" współfinansowanego ze środków Unii Europejskiej w ramach Programu Fundusze Europejskie dla Rozwoju Społecznego 2021-2027, Priorytet 1 Umiejętności, Działanie 01.05 Umiejętności w szkolnictwie wyższym, Typ projektu Dostosowanie oferty podmiotów systemu szkolnictwa wyższego do potrzeb rozwoju gospodarki oraz zielonej i cyfrowej transformacji.

Spis treści

1. Wstęp

W 2012 roku pierwszy raz na rynku ukazał się minikomputer Raspberry Pi. Dzięki elastyczności w konfiguracji oraz otwartemu na modyfikację systemowi Raspbian (dystrybucja oparta na Linux) szybko stał się popularny wśród hobbystów. Kolejne wersje Raspberry Pi zyskiwały coraz więcej sympatyków ze względu na różnorodne zastosowania. Raspberry Pi stał się nie tylko modułem programowalnym z dostępem do sieci. Można było skonfigurować go jako serwer, stworzyć klaster z minikomputerów Raspberry Pi czy wykorzystać jako sterownik automatyki domowej. Pojawiły się też wersje oraz oprogramowanie dla automatyki przemysłowej.

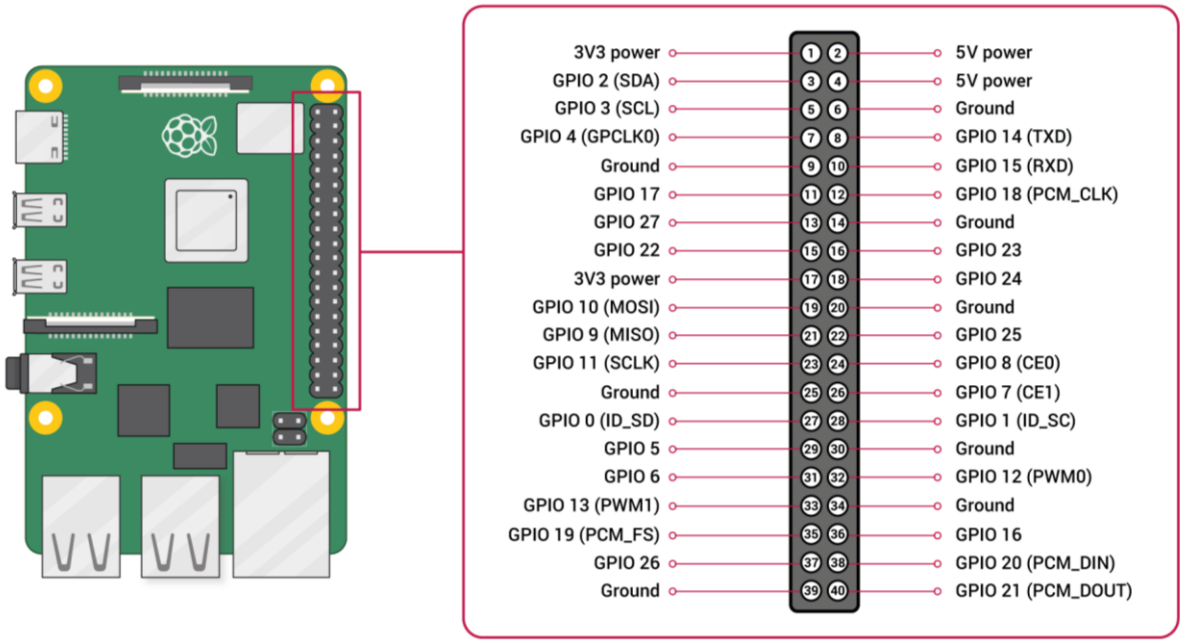
Mnogość zastosowań sprawiła, że chętnie zaczęto go wykorzystywać również w aplikacjach Internetu rzeczy. Wbudowane Wi-Fi oraz złącze Ethernet, swobodnie konfigurowalne piny GPIO, wiele magistrali komunikacyjnych (UART, I2C, SPI, 1-wire) oraz wysoka moc obliczeniowa spowodowała, że Raspberry Pi stał się jedną z najbardziej popularnych platform wykorzystywanych w IoT.

W niniejszej pracy omówiono dwa projekty wykorzystujące Raspberry Pi. Pierwszym z nich jest projekt sterowania ogrzewaniem z wykorzystaniem protokołu MQTT oraz platformy Node-RED. W jego skład wchodzi broker MQTT z usługą Node-RED zainstalowany na jednym Rapsberry Pi oraz dwa urządzenia – klienci, którzy wymieniają się danymi z brokerem (publikacja i subskrypcja tematów). Drugi projekt wykorzystuje środowisko automatyki domowej Domoticz. Składa się on z serwera (Raspberry Pi) oraz dwóch klientów. Jednym z klientów jest Raspberry Pi umożliwiające pomiar temperatury oraz sterowanie włącz / wyłącz np. oświetleniem czy opuść / podnieś w przypadku sterowania roletami. Drugim klientem jest urządzenie oparte o moduł ESP32-WROOM-32, którego zadaniem jest sterowanie jasnością oświetlenia.

2. Minikomputer Raspberry Pi

Raspberry Pi 4B posiada 40-pinowe złącze GPIO umożliwiające sterowanie różnego typu układami wykonawczymi. Dzięki obecności protokołów komunikacyjnych (UART, I2C, SPI, 1-wire) możliwy jest odczyt wartości z różnych czujników np. temperatury, ciśnienia, jakości powietrza, IMU, RTC. Zainstalowany system operacyjny Raspberry Pi OS pozwala na swobodną konfigurację według indywidulanych potrzeb użytkownika i aplikacji. Do sterowania automatyką domową można wykorzystać system Domoticz, do analizy obrazu z kamery pakiet OpenCV, a do zbierania danych z wielu czujników broker MQTT.

Dostęp do pinów GPIO można uzyskać za pomocą dostępnych bibliotek m.in. w języku python (biblioteka *RPi.GPIO*).



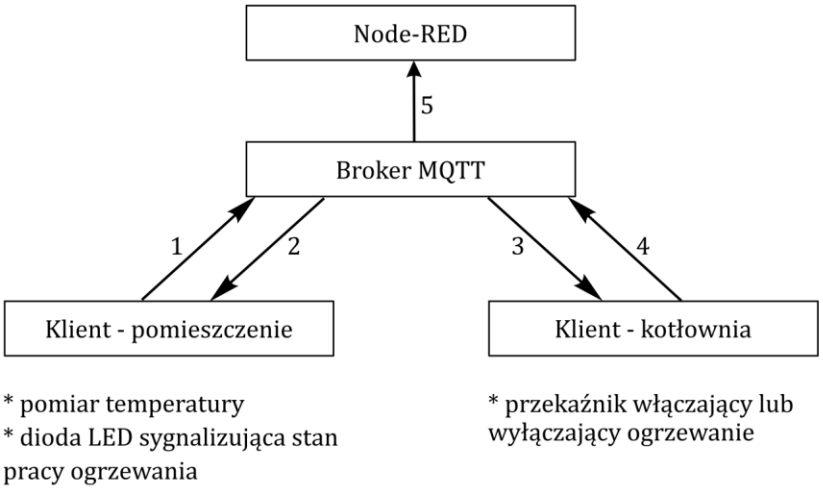
Rysunek 1 Pinout Raspberry Pi - listwa GPIO [5]

Raspberry Pi 4B posiada również:

- 2 złącza microHDMI,
- 4 złącza USB typu A,
- złącze RJ-45 (Ethernet),
- złącze audio,
- złącze kamery (CSI),
- złącze wyświetlacza.

3. Projekt urządzenia sterującego ogrzewaniem z wykorzystaniem protokołu MQTT

Projekt urządzenia sterującego ogrzewaniem z wykorzystaniem protokołu MQTT oparto na dwóch urządzeniach tzw. klientach oraz jednym serwerze (broker). W realizacji opisanego poniżej projektu założono, że sterowanie ogrzewaniem będzie polegało na włączaniu i wyłączaniu pieca elektrycznego lub elektrozaworu, który umożliwia zasilanie grzejników gorącą wodą. Przełączanie będzie wykonywane przez moduł przekaźnika wykonawczego, którego obciążalność styków wynosi do 10A dla prądu przemiennego przy obciążeniu rezystancyjnym. Będzie on wyzwalany za pomocą Raspberry Pi (*klient – kotłownia*) w sytuacji, gdy temperatura w pomieszczeniu będzie zbyt niska. Do odczytu temperatury w pomieszczeniu wykorzystano cyfrowy czujnik temperatury DS18B20, który komunikuje się z Raspberry Pi (*klient – pomieszczenie*) wykorzystując interfejs 1-wire. Do sygnalizacji stanu ogrzewania (włączone lub wyłączone) wykorzystano diodę LED.



Rysunek 2 Idea systemu sterowania ogrzewaniem

Oznaczenia wykorzystane na powyższym rysunku:

1. Publikowanie wartości temperatury do brokera
2. Subskrybowanie informacji o stanie pracy ogrzewania
3. Subskrybowanie wartości temperatury
4. Publikowanie stanu pracy ogrzewania do brokera
5. Subskrybowanie danych z brokera i ich graficzna prezentacja

3.1. Dobór elementów systemu

Właściwy dobór elementów systemu (modułów) oraz znajomość ich kluczowych parametrów znacznie ułatwia konstruowanie urządzeń prototypowych. Poniżej opisano kluczowe moduły elektroniczne wykorzystane do budowy systemu wraz z ich najważniejszymi parametrami. W poniższym zestawieniu nie uwzględniono przewodów połączeniowych czy płytek stykowych.

Minikomputer Raspberry Pi 4B – 3 szt.

- Komunikacja bezprzewodowa: Wi-Fi w paśmie 2.4 GHz / 5.0 GHz i standardzie 802.11b/g/n/ac; Bluetooth, BLE 5.0
- Ilość wyprowadzeń GPIO: 28
- Interfejsy komunikacyjne: UART, I2C, SPI, PWM
- Taktowanie: 4 x 1.5 GHz
- Pamięć RAM: 2 GB
- Napięcie zasilania: 5.2V / 3A

Czujnik temperatury DS18B20

- Interfejs komunikacyjny: 1-wire
- Zakres pomiaru temperatury: -55ºC : 125 ºC
- Dokładność pomiaru: ±0.5 ºC
- Rozdzielczość pomiaru: 9 – 12 bitów
- Zasilanie: 3.3 – 5V DC
- Obudowa: TO-92
- Wyjście danych: pin DQ typu open-collector

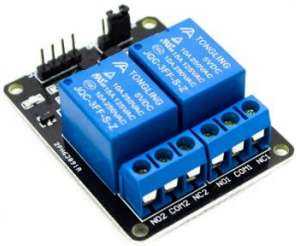
Moduł dwuprzekaźnikowy

- Napięcie zasilania cewki przekaźnika: 5V
- Aktywacja przekaźnika: stan niski (LOW)
- Maksymalne obciążenie styków: 10A, 250 VAC (obciążenie rezystancyjne)

Dioda LED z rezystancyjnym ograniczeniem prądowym

- Kolor: zielony
- Prąd: 11 mA
- Rezystancja szeregowa: 100 Ω

Rysunek 3 Czujnik temperatury DS18B20

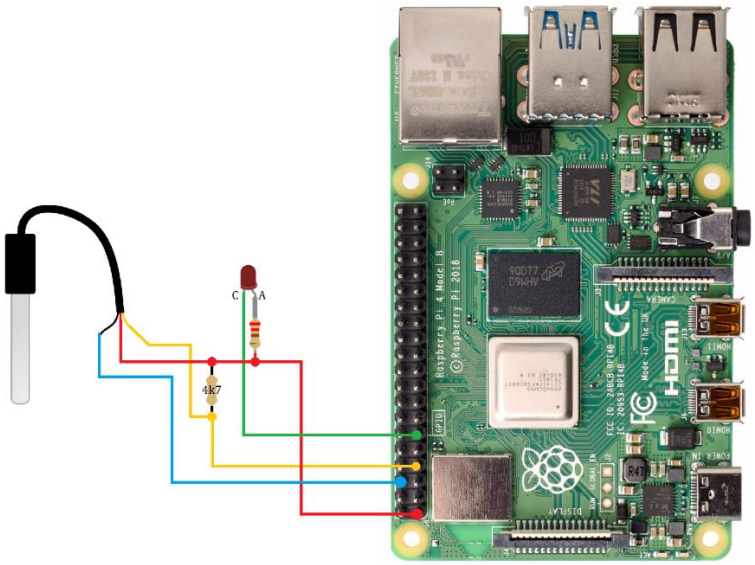


Rysunek 4 Moduł dwuprzekaźnikowy

Rysunek 5 Raspberry Pi 4B

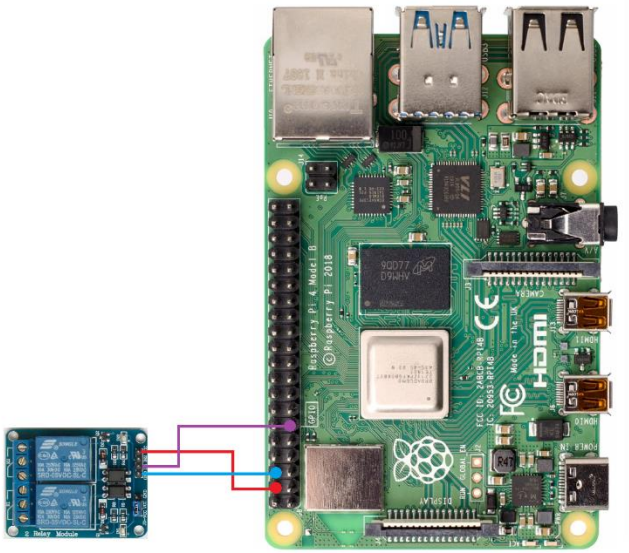
3.2.Schemat ideowy

Na szkicu poniżej zaprezentowano schemat połączeń modułów elektronicznych z minikomputerem Raspberry Pi 4. Poniższe połączenie pozwala na uruchomienie i poprawne działanie programu z punktu *6.5 Opis programu* dla układu *klient - pomieszczenie.* Moduły elektroniczne tj. czujnik temperatury oraz dioda LED zasilane są napięciem 3.3V. Czujnik DS18B20 podłączony jest do pinu 7 (GPIO4), a dioda LED do pinu 11 (GPIO17).



Rysunek 6 Schemat ideowy klient - pomieszczenie

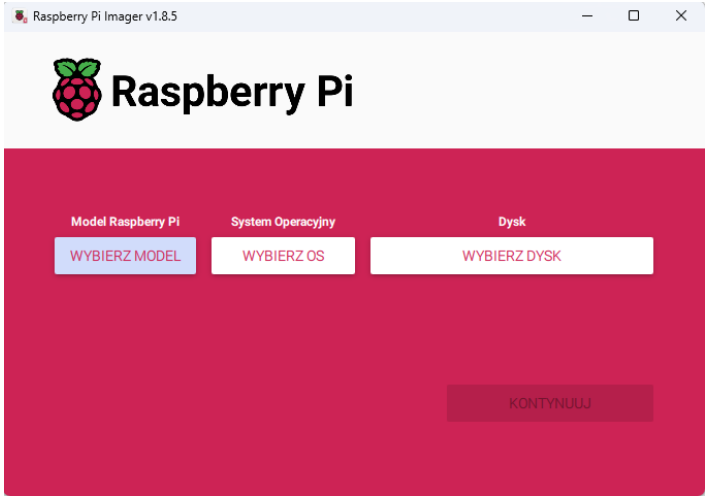
Na szkicu poniżej zaprezentowano schemat połączeń modułów elektronicznych z minikomputerem Raspberry Pi 4 dla układu *klient – kotłownia.* Moduł przekaźnikowy podłączony jest do pinu 11 (GPIO17).



Rysunek 7 Schemat ideowy klient - kotłownia

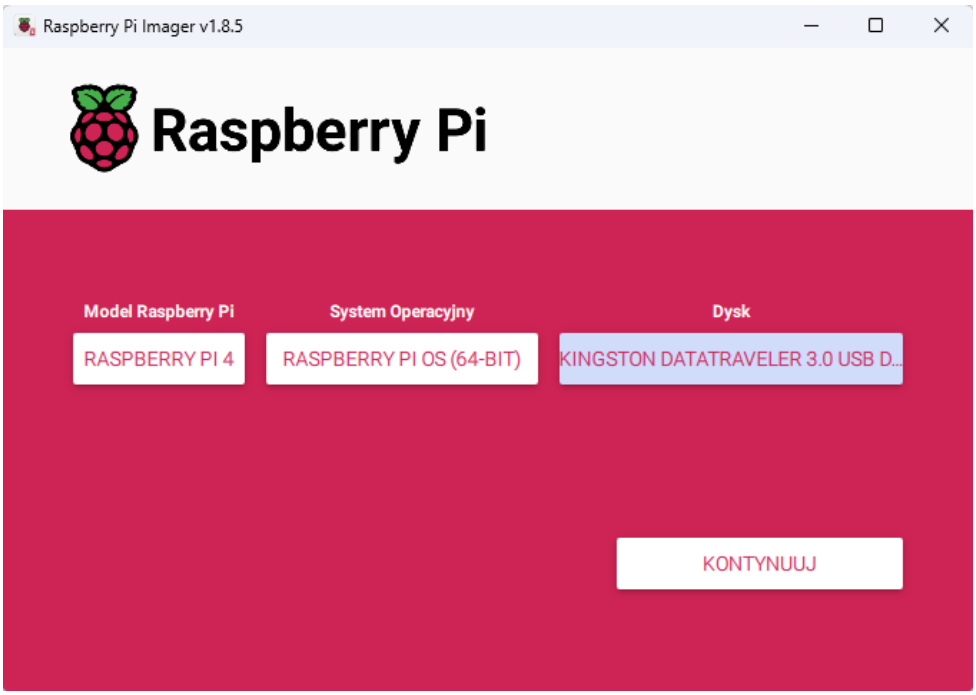
3.3.Raspberry Pi OS

Aby uruchomić i właściwie skonfigurować minikomputer Raspberry Pi 4 konieczne jest zainstalowanie systemu operacyjnego Raspberry Pi OS na karcie microSD, z której uruchamiany jest system. Instalacja odbywa się z wykorzystaniem dedykowanego narzędzia *Raspberry Pi Imager*.



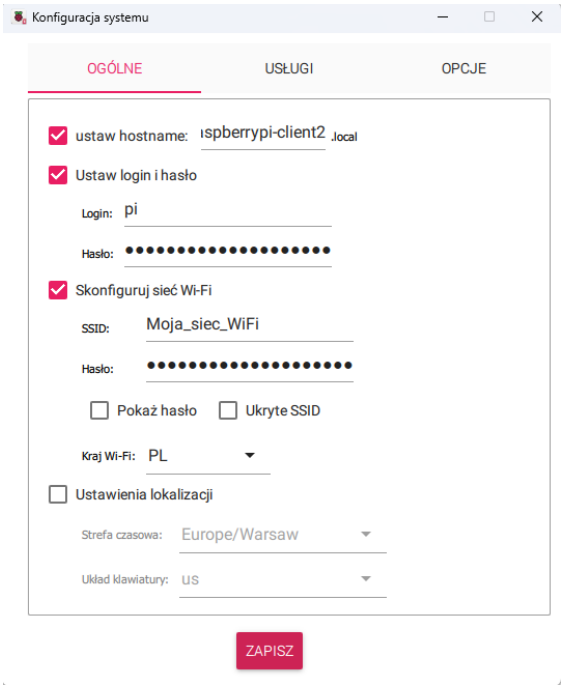
Rysunek 8 Raspberry Pi Imager

Po jego uruchomieniu należy uzupełnić trzy pola: wybrać model Raspberry Pi, do którego wgrywamy system; wybrać typ systemu operacyjnego oraz wskazać nośnik, na którym system zostanie zainstalowany. W przypadku niniejszego projektu wybrano Raspberry Pi 4B, a system to Raspberry Pi OS 64-bit (*A port of Debian Bookworm with the Raspberry Pi Desktop (Recommended)*). Po wskazaniu nośnika (karty microSD) należy kliknąć przycisk *Kontynuuj.*

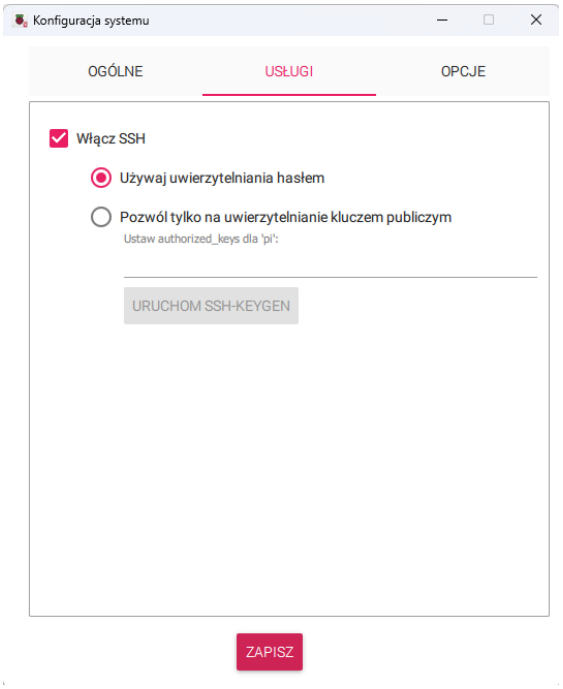


Rysunek 9 Raspberry Pi Imager - konfiguracja

W następnym kroku pojawia się okno dotyczące ustawień personalizacji systemu operacyjnego. Należy wybrać opcję *Edytuj ustawienia.* W zakładce *Ogólne* należy wpisać nazwę sieci Wi-Fi, z którą Raspberry Pi ma się połączyć. W tej zakładce ustawia się też login oraz hasło, które są konieczne do połączenia się po SSH. Domyślnie login to *pi,* a hasło to *raspberry*. W przypadku, gdy w sieci będzie kilka minikomputerów Raspberry należy ustawić *hostname* tak, aby każde z urządzeń miało swoją własną, niepowtarzalną nazwę. Sugeruje się, aby nazwa odpowiadała funkcji urządzenia, np. *raspberrypi-broker-mqtt.local*. W drugiej zakładce tj. *Usługi* należy uruchomić połączenie SSH zaznaczając checkbox przy opcji *Włącz SSH.* Należy również pozostawić aktywne *Używaj uwierzytelnienia hasłem.*



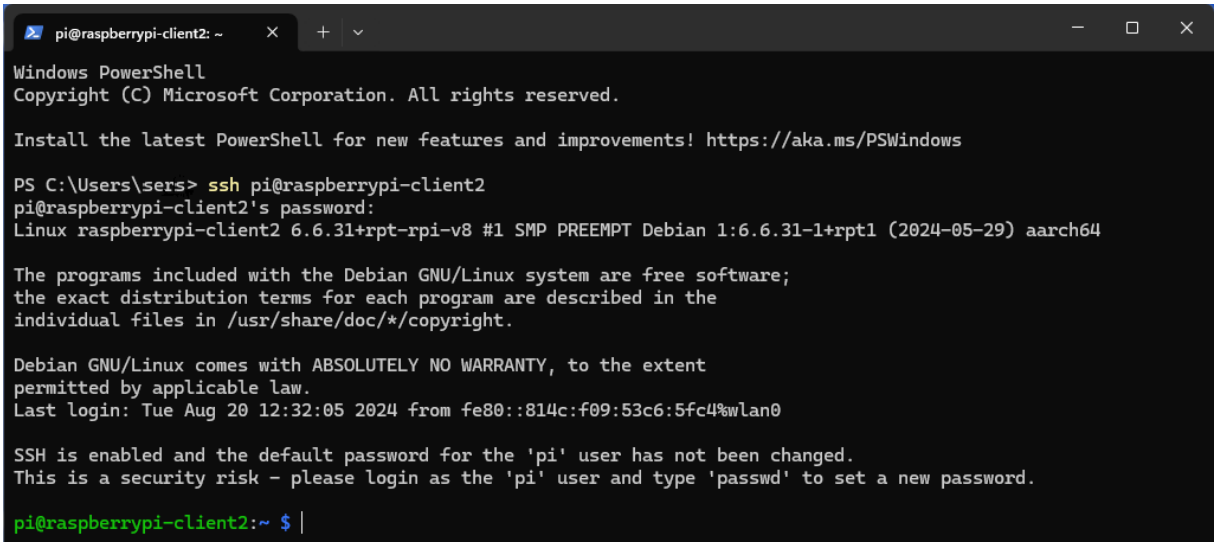
Rysunek 10 Konfiguracja systemu - zakładka Ogólne



Rysunek 11 Konfiguracja systemu - zakładka Usługi

Po wprowadzeniu wszystkich wyżej wymienionych danych należy zapisać ustawienia oraz rozpocząć proces instalacji systemu. Po jego zakończeniu należy przełożyć kartę microSD z komputera do Raspberry Pi, podłączyć zasilacz Raspberry Pi (zasilacz z wtykiem USB-C) oraz włączyć go do gniazda sieciowego. Po pojawieniu się zasilania minikomputer rozpocznie ładowanie systemu (sygnalizuje to migająca zielona dioda).

Aby połączyć się z Raspberry Pi najprościej jest wykorzystać do tego celu aktywowany na etapie instalacji protokół SSH. Jeżeli komputer oraz Raspberry Pi znajdują się w tej samej sieci Wi-Fi należy uruchomić aplikację *Windows Powershell* (dotyczy systemów Windows 10 i nowszych). W przypadku starszych systemów operacyjnych należy wykorzystać program Putty łącząc się po SSH z adresem portu 22. Polecenie *ssh pi@hostname* otwiera port i nawiązuje połączenie z urządzeniem o nazwie lokalnej *hostname.* Nazwa ta jest ustawiana podczas edycji ustawień przy instalacji systemu (zakładka *Ogólne*). W przypadku odnalezienia urządzenia i połączenia się z nim wymagane jest podanie loginu i hasła ustanowionego podczas instalacji systemu. Po poprawnym zalogowaniu połączenie z minikomputerem zostało nawiązane.



Rysunek 12 Połączenie Raspberry Pi z wykorzystaniem SSH

Po instalacji systemu należy przeprowadzić instalację pakietów wymaganych do uruchomienia protokołu MQTT:

- *mosquitto-clients* oraz przeprowadzić jego konfigurację,
- *python3-paho-mqtt*.

Instalacja *mosquitto-clients* polega na wpisaniu w terminal poniższej komendy:

sudo apt install -y mosquitto mosquitto-clients

Natomiast instalacja *python3-paho-mqtt* rozpoczyna się po wpisaniu w terminalu i zatwierdzeniu przyciskiem *Enter*:

sudo apt install python3-paho-mqtt

Powyższe kroki, które opisano (od momentu instalacji systemu operacyjnego do instalacji pakietów) należy wykonać dla każdego Raspberry Pi (w niniejszym projekcie 3 szt.).

Broker MQTT – konfiguracja

*Poniższe kroki dotyczą **wyłącznie** konfiguracji brokera MQTT*

Po instalacji pakietu *mosquitto-clients* należy włączyć automatyczne uruchamianie usługi podczas ładowania systemu operacyjnego. Pozwoli to na automatyczne uruchomienie brokera

MQTT i transmisję danych bez konieczności manualnego jej uruchamiana po restarcie systemu. Autostart usługi uruchamia się wpisując w terminal polecenie:

sudo systemctl enable mosquitto.service

W celu weryfikacji poprawności instalacji i uruchomienia usługi MQTT należy wpisać polecenie:

mosquitto -v

```
pi@raspberrypi:~ $ mosquitto -v
1724165390: mosquitto version 2.0.11 starting
1724165390: Using default config.
1724165390: Starting in local only mode. Connections will only be p
ossible from clients running on this machine.
1724165390: Create a configuration file which defines a listener to
 allow remote access.
1724165390: For more details see https://mosquitto.org/documentatio
n/authentication-methods/
1724165390: Opening ipv4 listen socket on port 1883.
1724165390: Error: Address already in use
1724165390: Opening ipv6 listen socket on port 1883.
1724165390: Error: Address already in use
```

Rysunek 13 Weryfikacja instalacji mosquitto

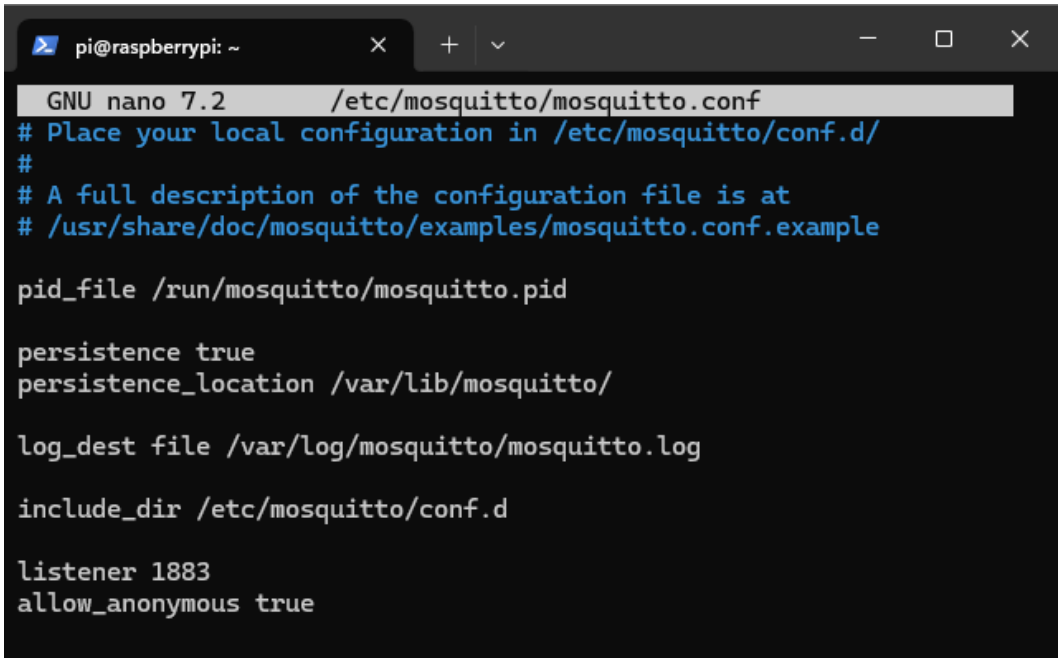
Broker MQTT może pracować w dwóch trybach: z autoryzacją oraz bez autoryzacji. Uruchomienie autoryzacji wymaga zdefiniowania użytkowników oraz haseł, za pomocą których łączą się oni z brokerem. W niniejszym przykładzie wykorzystano tryb bez autoryzacji przesyłanych danych. Oznacza to, że każde urządzenie może przesłać dane do brokera, a on je odbierze. Aby wyłączyć autoryzację należy edytować plik *mosquitto.conf.* W celu jego otwarcia edytorem *nano* wpisujemy poniższą komendę:

sudo nano /etc/mosquitto/mosquitto.conf

Na końcu pliku należy dopisać dwie linijki kodu:

listener 1883

allow_anonymous true



Rysunek 14 Plik mosquitto.conf po edycji

Po zapisaniu zmian i zamknięciu edytora (*Ctrl + X, Y, Enter*) należy zrestartować usługę komendą:

sudo systemctl restart mosquitto

Po wykonaniu powyższych czynności broker MQTT działa w tle i oczekuje na przesyłane wiadomości. Aby można było zweryfikować, czy dane docierają do brokera należy wykorzystać poniższe polecenie:

mosquitto_sub -h localhost -t rpi/test -q 1

gdzie:

rpi/test to nazwa przesyłanego tematu (topic),

-q 1 dopisujemy, gdy urządzenie wysyłające (publikujące) dane ma ustawione qos (*quietly of service*). Gdy ustawienia brak, należy je pominąć.

3.4.Node-RED

Mając zainstalowany pakiet *Mosquitto Broker* należy zainstalować pakiet Node-RED. W terminalu należy wpisać poniższą komendę:

bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)

Po zakończeniu instalacji program / usługa jest gotowa do uruchomienia. Dostępne są cztery tryby jej uruchomienia:

- *node-red-start* uruchamia usługę wraz z logami. Zamknięcie terminala nie wyłącza usługi.
- *node-red-stop* zatrzymuje działanie usługi,
- *node-red-restart* resetuje usługę,
- *node-red-log* wyświetla logi usługi.

Istnieje również możliwość uruchamiania usługi Node-RED automatycznie przy starcie systemu. Opcję tę można uaktywnić wpisując komendę:

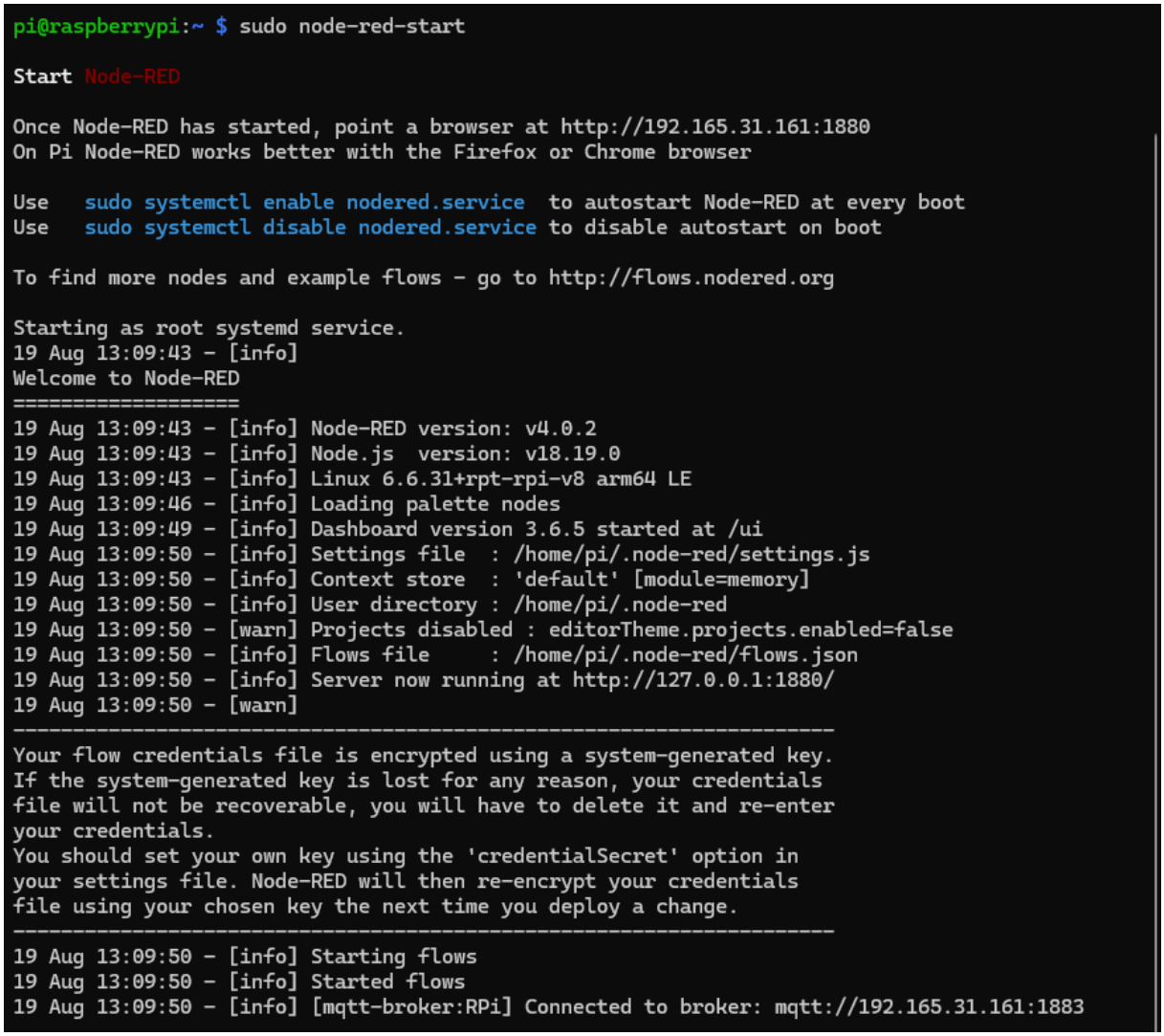
sudo systemctl enable nodered.service

Aby usunąć usługę autostartu należy parametr *enable* zmienić na *disable*.

Mając zainstalowaną usługę należy ją uruchomić wpisując na terminalu komendę:

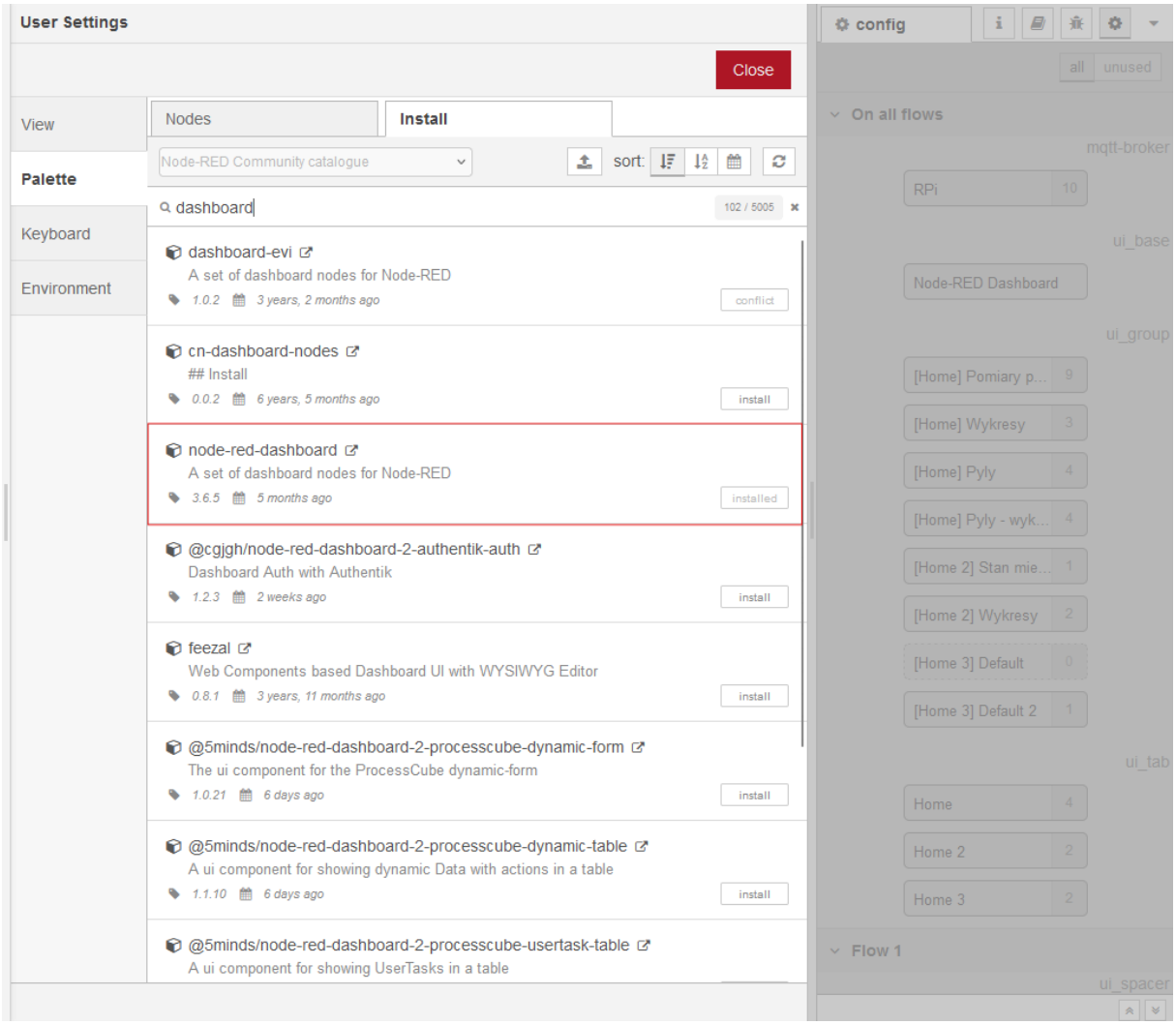
sudo node-red-start

Po wpisaniu komendy na terminalu pojawią się dane, za pomocą których można uruchomić wersję webową oprogramowania. W prezentowanym przykładzie będzie to *http://192.165.31.161:1880.*



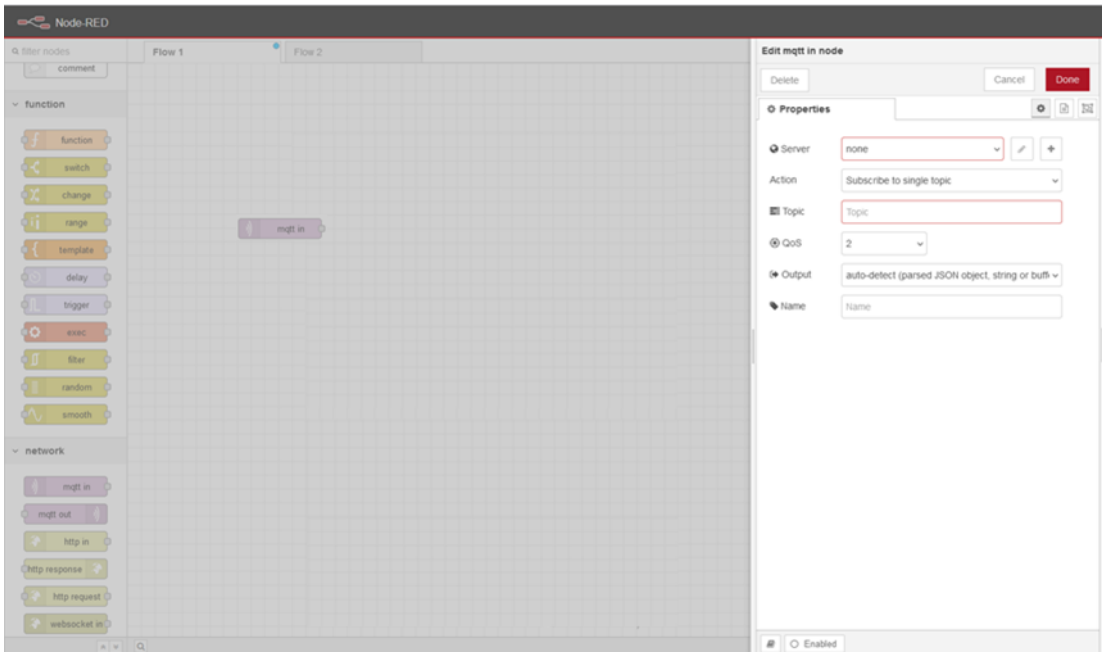
Rysunek 15 Uruchomienie Node-RED z terminala

Po wpisaniu adresu w przeglądarkę pojawia się okno konfiguracyjne, za pomocą którego można dodawać węzły oraz przekazywać dane pomiędzy publisher'em a subscriber'em. Aby móc prezentować dane w sposób liczbowy, a także graficzny (na wykresach) należy doinstalować pakiet *node-red-dashboard.* Instalację należy wykonać za pomocą narzędzia *Manage palette* (opcja dostępna po kliknięciu w trzy poziome kreski znajdujące się w prawym górnym rogu ekranu). Po uruchomieniu narzędzia należy wyszukać pakiet oraz go zainstalować.



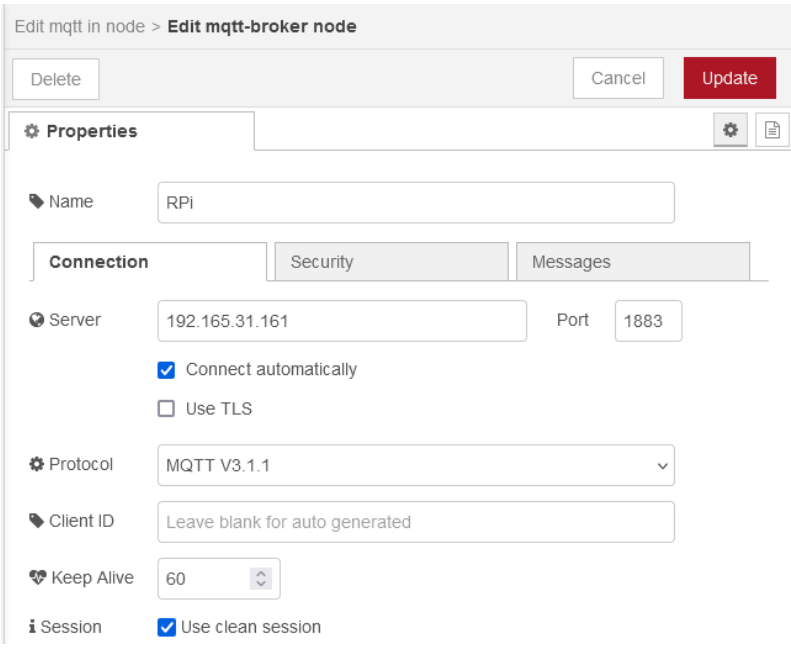
Rysunek 16 Instalacja pakietu Node-RED-dashboard

Po instalacji pakietu *node-red-dashboard* należy powrócić na stronę główną (*Flow 1*). Aby dodać węzeł MQTT z zakładki *network* należy wybrać opcję *mqtt in.* Pojawi się wówczas okno konfiguracyjne, w którym należy wybrać serwer (*Rapsberry Pi*) oraz wpisać temat (*Topic*).



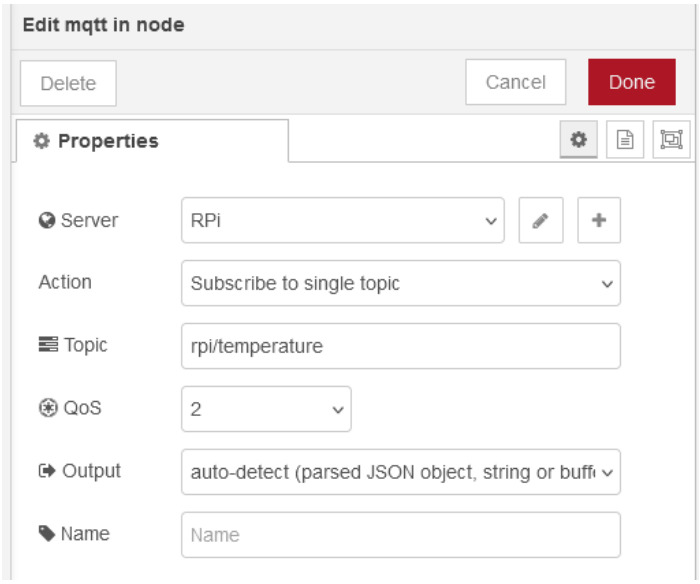
Rysunek 17 Dodanie węzła MQTT

W przypadku pierwszego uruchomienia lista serwerów będzie pusta. Należy wówczas skonfigurować nowy serwer klikając na symbol [+]. W nowo otwartym oknie *Właściwości* należy wpisać nazwę serwera, jego adres IP, wersję protokołu MQTT oraz ustawić parametr *Keep Alive,* który określa maksymalny czas (w sekundach) na odpowiedź, po upłynięciu którego urządzenie jest automatycznie rozłączane (w przypadku braku odpowiedzi).



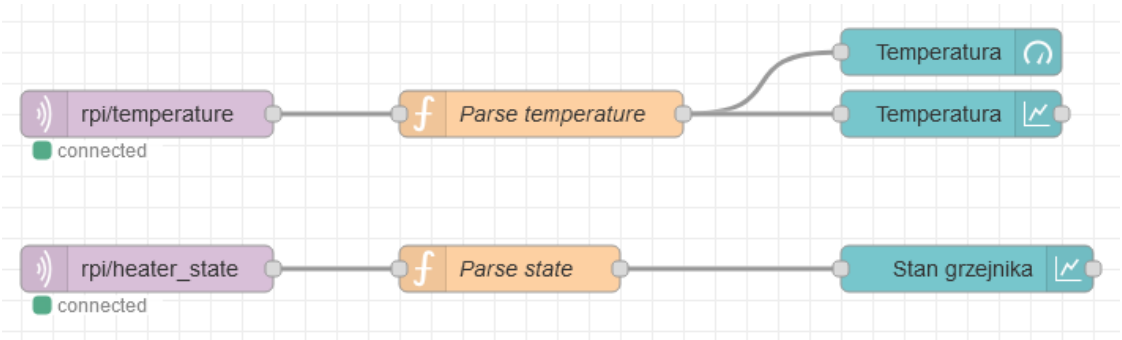
Rysunek 18 Konfiguracja brokera MQTT

Po dodaniu ustawień brokera należy skonfigurować węzły wejściowe. W przypadku omawianego projektu będą to dwie informacje przesyłane i wizualizowane w Node-RED: temperatura przesyłana z *klient-pomieszczenie* do *klient-kotłownia* (temat *rpi/temperature*) oraz stan pracy ogrzewania (temat *rpi/heater_state*) przesyłany z *klient-kotłownia* do *klient-pomieszczenie.*



Rysunek 19 Ustawienia węzła Node-RED

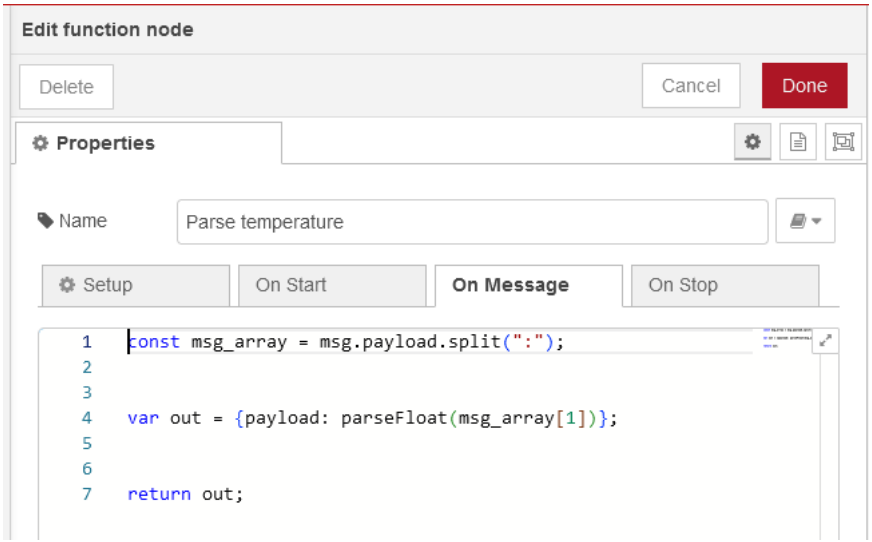
Finalny diagram w Node-RED przedstawiono na rysunku poniżej:



Rysunek 20 Diagram Node-RED

Bloki fioletowe to bloki z grupy *mqtt in*, które subskrybują dany temat. Bloki pomarańczowe to bloki z grupy *function*, w których można przekształcić otrzymaną wiadomość wykorzystując język JavaScript. Bloki w kolorze turkusowym odpowiadają za reprezentację graficzną przesłanych danych – wartości liczbowe oraz wykresy.

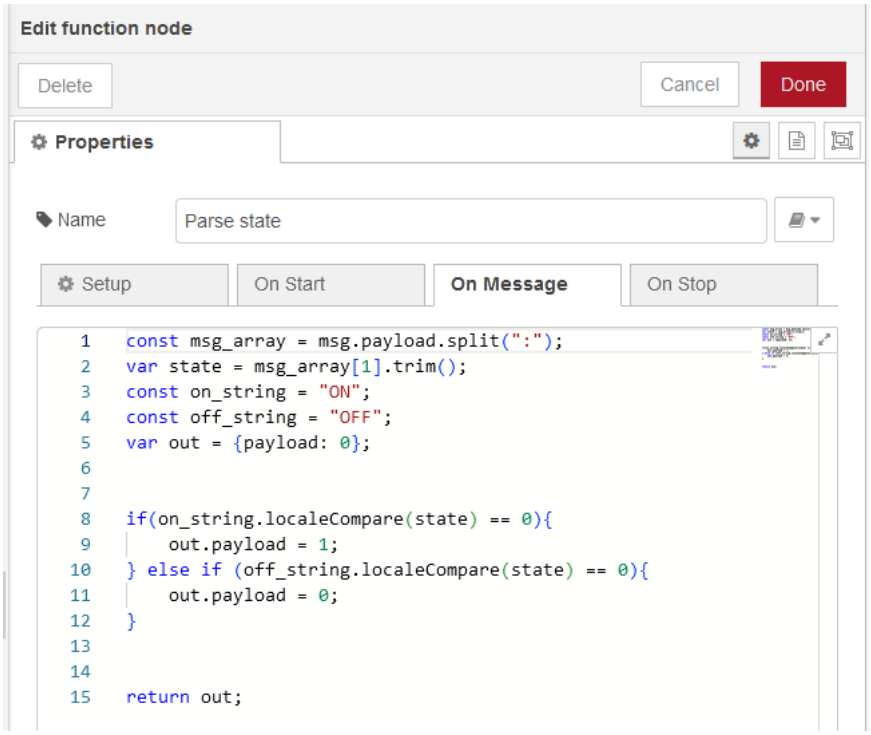
Blok funkcyjny *Parse temperature* odpowiada za wyodrębnienie z przesyłanej wiadomości wartości temperatury. Urządzenie *klient-pomieszczenie* wysyła wiadomość o temperaturze w następującym formacie: *"Temperature is: " + str(temperature)* np. *Temperature is: 28.2547*. Taki format uniemożliwia poprawny odczyt temperatury i wyświetlenie jej na wskaźniku cyfrowym oraz wykresie. Aby wyłuskać wartość temperatury wykorzystano funkcję *split()*, która dzieli frazę na osobne stringi i zapisuje je do macierzy stringów *msg_array* w przypadku napotkania znaku określonego w argumencie funkcji *split()*. Następnie wartość temperatury jest konwertowana z typu string na float funkcją *parseFloat()*.



Rysunek 21 Funkcja Parse temperature

Blok funkcyjny *Parse state* odpowiada za wyodrębnienie z przesyłanej wiadomości stanu ogrzewania (włączone lub wyłączone). Urządzenie *klient-kotłownia* wysyła wiadomość o stanie ogrzewania w następującym formacie: *"Heater status: ON"* lub *"Heater status: OFF"*. Taki format uniemożliwia poprawny odczyt stanu ogrzewania i zwizualizowaniu go na wykresie. Aby wyłuskać stan ON lub OFF (1 lub 0 w zapisie bitowym) wykorzystano funkcję *split()*, która dzieli frazę na osobne stringi i zapisuje je do macierzy stringów *msg_array* w przypadku napotkania znaku określonego w argumencie funkcji *split()*. Wywołanie funkcji *trim()* usuwa białe znaki z danego stringa. Następnie następuje porównanie wartości wyodrębnionej z wiadomości MQTT ze wzorcem (zmienne *on_string* i *off_string*) za pomocą wyrażenia *on_string.localeCompare(state)*. W przypadku pozytywnego wyniku porównania

(funkcja *on_string.localeCompare()* zwraca wartość 0) zwracana jest wartość 1 dla stanu ogrzewania ON oraz 0 dla stanu ogrzewania OFF.



Rysunek 22 Funkcja Parse state

3.5. Opis programu

Odbiór danych przez brokera MQTT jest możliwy wyłącznie, gdy klienci będą publikować dane. Poniżej przedstawiono kod umożliwiający publikację oraz subskrybowanie danych z brokera dla urządzeń *klient-pomieszczenie* oraz *klient-kotłownia*. Kod napisano w języku python, który umożliwia bezpośredni dostęp do interfejsów komunikacyjnych Raspberry Pi oraz pinów GPIO.

Program urządzenia klient – pomieszczenie

Import bibliotek obsługujących odczyt temperatury z czujnika DS18B20, protokół MQTT oraz sterowanie pinami GPIO:

```
import paho.mqtt.client as mqtt
from w1thermsensor import W1ThermSensor
import time
import RPi.GPIO as GPIO
```

Konfiguracja pinu określającego stan ogrzewania (dioda LED):

```
heater_pin = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(heater_pin, GPIO.OUT)
```

Konfiguracja brokera oraz tematów przesyłanych wiadomości do brokera MQTT:

```
hostname = "raspberrypi"
broker_port = 1883
topic_temperature = "rpi/temperature"
topic_heater = "rpi/heater_state"
heater_state = 0
```

Reakcja programu na rozłączenie z brokerem:

```
def on_disconnect(client, userdata, rc):
    logging.debug("Disconnected result code "+str(rc))
    client.loop_stop()
```

Funkcja definiująca połączenie z brokerem i subskrypcję określonego tematu:

```
def on_connect(client, userdata, flags, rc):
    print("Connected to MQTT broker with result code: " + str(rc))
    client.subscribe(topic_heater, qos=1)
```

Reakcja programu na przyjście wiadomości:

```
def on_message(client, userdata, msg):
    global heater_state
    m = msg.payload.decode()
    state = (m.split(':', 1)[1]).strip()
    if state == "ON":
        heater_state = 1
    elif state == "OFF":
```

```
heater_state = 0
```

Uruchomienie klienta MQTT:

```
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.on_disconnect = on_disconnect
```

Odczyt temperatury:

```
sensor = W1ThermSensor()
```

Połączenie z brokerem i utworzenie wątku oczekującego na potwierdzenia otrzymania wiadomości:

```
client.connect(hostname, broker_port, 60)
```

Uruchomienie wątku zbierającego potwierdzenia wiadomości. W przypadku braku komendy możliwe jest wysłanie maksymalnie 20 wiadomości. Konieczny restart usługi, by przywrócić funkcjonalność.

```
client.loop_start()

while True:
```

Tworzenie i publikowanie wiadomości o temperaturze:

```
    temperature = sensor.get_temperature()
    message = "Temperature is: " + str(temperature)
    client.publish(topic_temperature, message, qos=1)
```

Reakcja na odebrane dane o stanie ogrzewania (włącz / wyłącz LED):

```
    if heater_state == 1:
        GPIO.output(heater_pin, GPIO.HIGH)
    else:
        GPIO.output(heater_pin, GPIO.LOW)

    time.sleep(1)
```

Program urządzenia klient – kotłownia

Import bibliotek protokół MQTT oraz sterowanie pinami GPIO:

```
import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO
```

Konfiguracja pinu włączającego / wyłączającego ogrzewanie:

```
heater_pin = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(heater_pin, GPIO.OUT)
GPIO.output(heater_pin, GPIO.HIGH)
```

Konfiguracja brokera oraz tematów przesyłanych wiadomości do brokera MQTT:

```
hostname = "raspberrypi"
broker_port = 1883
topic_temperature = "rpi/temperature"
topic_heater = "rpi/heater_state"
```

Uruchomienie klienta MQTT:

```
client = mqtt.Client()
```

Deklaracja zmiennych oraz progu załączania ogrzewania:

```
current_temperature = 0.0
set_temperature = 27.0
heater_state = 0
```

Reakcja programu na rozłączenie z brokerem:

```
def on_disconnect(client, userdata, rc):
    logging.debug("Disconnected result code "+str(rc))
    client.loop_stop()
```

Funkcja definiująca połączenie z brokerem i subskrypcję określonego tematu:

```
def on_connect(client, userdata, flags, rc):
    print("Connected to MQTT broker with result code: " + str(rc))
    client.subscribe(topic_temperature, qos=1)
```

Reakcja programu na przyjście wiadomości:

```
def on_message(client, userdata, msg):
    global heater_state
    m = msg.payload.decode()
    t_str = m.split(':', 1)[1]
    current_temperature = float(t_str)
```

Regulacja temperatury z histerezą 1 ºC:

```
    if (current_temperature < set_temperature - 0.5) and (heater_state == 0):
        heater_state = 1
        GPIO.output(heater_pin, GPIO.LOW)

    elif (current_temperature > set_temperature + 0.5) and (heater_state == 1):
        heater_state = 0
        GPIO.output(heater_pin, GPIO.HIGH)
```

Przesyłanie wiadomości o stanie ogrzewania (włączone / wyłączone):

```
    if heater_state == 1:
        client.publish(topic_heater, "Heater status: ON", qos=1)
    else:
        client.publish(topic_heater, "Heater status: OFF", qos=1)

client.on_disconnect = on_disconnect
client.on_connect = on_connect
client.on_message = on_message
```

Połączenie z brokerem i utworzenie wątku oczekującego na potwierdzenia otrzymania wiadomości:

```
client.connect(hostname, broker_port, 60)
```

Oczekiwanie na nowe wiadomości

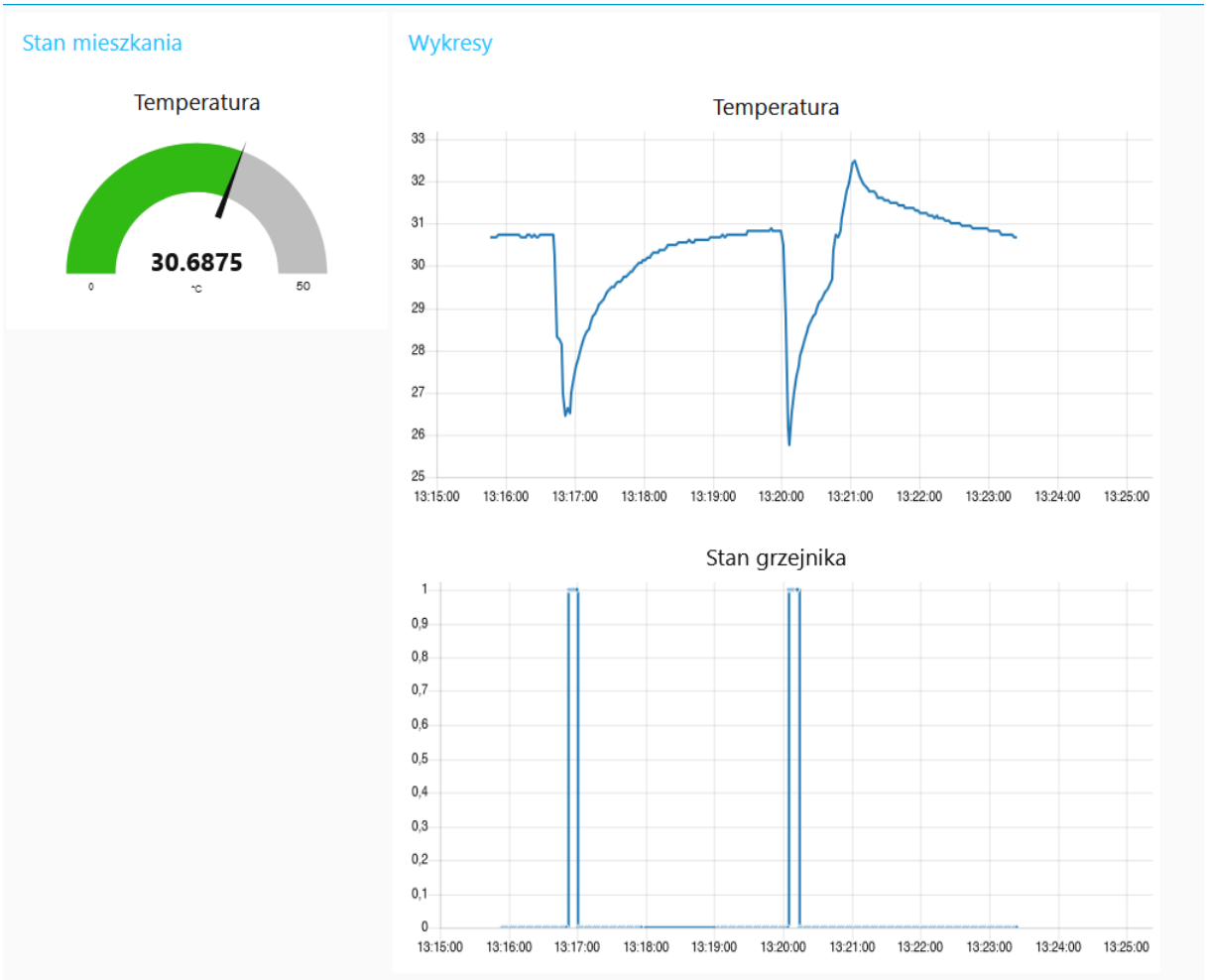
```
client.loop_forever()
```

3.6. Testy systemu

Po kompilacji i uruchomieniu skryptów python poleceniem *python nazwa_programu.py* należy uruchomić Node-RED *user interface*, który w sposób graficzny zobrazuje przesyłane dane. Aplikacja graficzna dostępna jest z poziomu przeglądarki pod adresem:

http://192.165.31.161:1880/ui

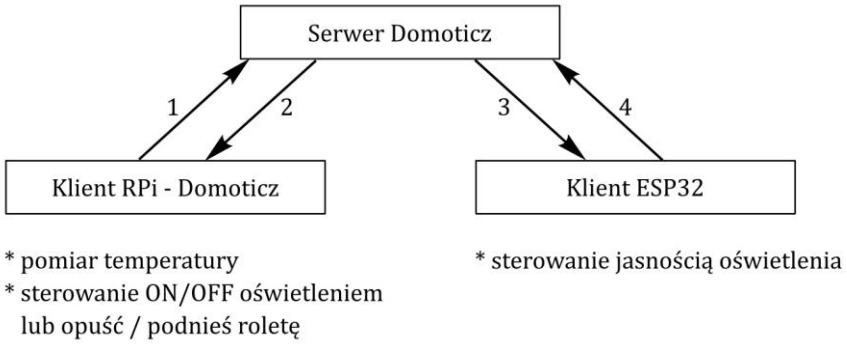
Po jej uruchomieniu otworzy się okno jak, na poniższym rysunku. Dane będą wyświetlane i aktualizowane w czasie rzeczywistym po każdej odebranej wiadomości.



Rysunek 23 Prezentacja graficzna przesyłanych danych

4. Projekt systemu automatyki domowej Domoticz

Przedstawiony poniżej projekt wykorzystuje środowisko automatyki domowej Domoticz, która do wymiany danych wykorzystuje protokół MQTT. Składa się on z serwera (Raspberry Pi) oraz dwóch klientów. Jednym z klientów jest Raspberry Pi umożliwiające pomiar temperatury oraz sterowanie włącz / wyłącz np. oświetleniem czy opuść / podnieś w przypadku sterowania roletami. Drugim klientem jest urządzenie oparte o moduł ESP32, którego zadaniem jest sterowanie jasnością oświetlenia. Umożliwia to sygnał PWM, który jest podłączony do diody LED symbolizującej oświetlenie w pokoju. W praktycznym zastosowaniu diodę LED należałoby zamienić na układ optotriaka z głównym triakiem załączającym (optotriak bez układu ZCD (zero-cross-detect)). Taki układ pozwoliłby na sterownie jasnością oświetlenia ze standardowych żarówek jak i energooszczędnych LEDowych.



Rysunek 24 Idea systemu automatyki domowej

Przedstawiony w niniejszym punkcie przykład systemu automatyki domowej umożliwia proste sterowanie jasnością oświetlenia, jego włączaniem lub wyłączaniem oraz odczyt temperatury. Sterowanie jasnością odbywa się za pomocą slidera (suwaka) z poziomu aplikacji serwera. W niej można również włączyć lub wyłączyć oświetlenie lub odczytać wartość temperatury. Zaprezentowany materiał może służyć jako wprowadzenie do systemu Domoticz tak, aby po jego pełnej konfiguracji i dodaniu scen stworzyć prosty system automatyki domowej.

4.1. Dobór elementów systemu

Właściwy dobór elementów systemu (modułów) oraz znajomość ich kluczowych parametrów znacznie ułatwia konstruowanie urządzeń prototypowych. Poniżej opisano kluczowe moduły elektroniczne wykorzystane do budowy systemu wraz z ich najważniejszymi parametrami. W poniższym zestawieniu nie uwzględniono przewodów połączeniowych czy płytek stykowych.

Minikomputer Raspberry Pi 4B – 2 szt.

- Komunikacja bezprzewodowa: Wi-Fi w paśmie 2.4 GHz / 5.0 GHz i standardzie 802.11b/g/n/ac; Bluetooth, BLE 5.0
- Ilość wyprowadzeń GPIO: 28
- Interfejsy komunikacyjne: UART, I2C, SPI, PWM
- Taktowanie: 4 x 1.5 GHz
- Pamięć RAM: 2 GB
- Napięcie zasilania: 5.2V / 3A

Czujnik temperatury DS18B20

- Interfejs komunikacyjny: 1-wire
- Zakres pomiaru temperatury: -55ºC : 125 ºC
- Dokładność pomiaru: ±0.5 ºC
- Rozdzielczość pomiaru: 9 – 12 bitów
- Zasilanie: 3.3 – 5V DC
- Obudowa: TO-92
- Wyjście danych: pin DQ typu open-collector

Moduł dwuprzekaźnikowy

- Napięcie zasilania cewki przekaźnika: 5V
- Aktywacja przekaźnika: stan niski (LOW)
- Maksymalne obciążenie styków: 10A, 250 VAC (obciążenie rezystancyjne)

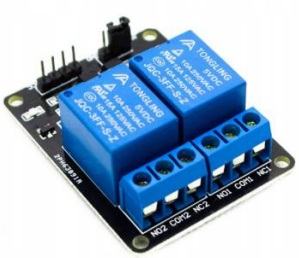
Moduł mikrokontrolera z Wi-Fi ESP32-DevKitC V4

- Komunikacja bezprzewodowa: Wi-Fi w paśmie 2,4 GHz i standardzie 802.11b/g/n; Bluetooth, BLE 4.2
- Ilość wyprowadzeń GPIO: 34
- Interfejsy komunikacyjne: UART, I2C, SPI, SDIO, PWM, I2S, ADC, DAC
- Taktowanie: do 240MHz
- Pamięć ROM: 448 KB
- Pamięć SRAM: 520 KB
- Pamięć Flash SPI: 4 MB
- Napięcie zasilania: od 3 V do 3,6 V

Dioda LED z rezystancyjnym ograniczeniem prądowym

- Kolor: zielony
- Prąd: 11 mA
- Rezystancja szeregowa: 100 Ω

Rysunek 25 Czujnik temperatury DS18B20



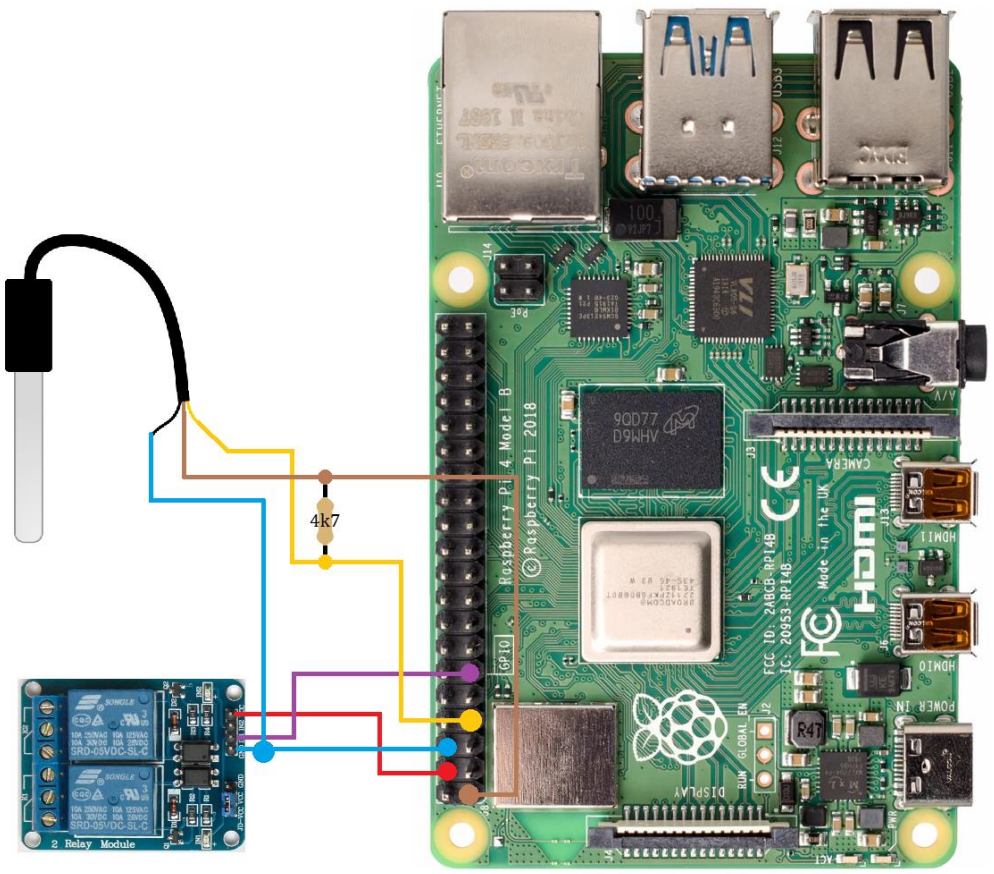
Rysunek 26 Moduł dwuprzekaźnikowy

Rysunek 27 Raspberry Pi 4B

Rysunek 28 Moduł ESP32-DevKitC V4

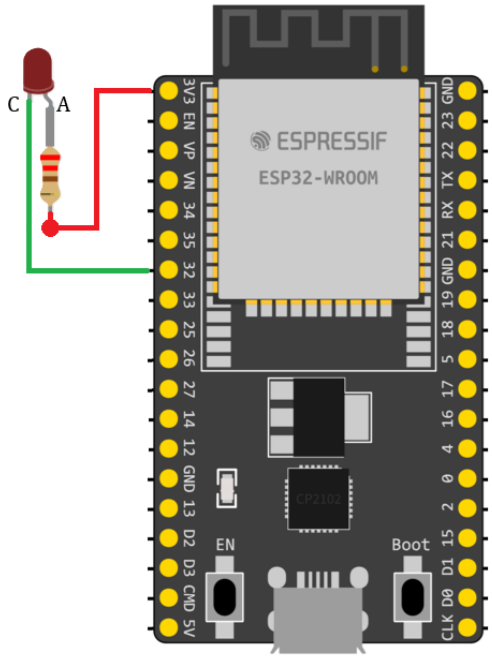
4.2.Schemat ideowy

Na szkicu poniżej zaprezentowano schemat połączeń modułów elektronicznych z minikomputerem Raspberry Pi. Poniższe połączenie pozwala na uruchomienie i poprawne działanie środowiska opisanego w punkcie *4.3.2 Konfiguracja klienta.* Moduły elektroniczne tj. czujnik temperatury zasilany jest napięciem 3.3V, a moduł przekaźnikowy napięciem 5V. Czujnik DS18B20 podłączony jest do pinu 7 (GPIO4), a moduł przekaźnikowy do pinu 11 (GPIO17).



Rysunek 29 Schemat ideowy klienta

Na szkicu poniżej zaprezentowano schemat połączeń modułów elektronicznych z układem ESP32-WROOM-32E. Dioda LED, której jasnością sterujemy z serwera Domoticz podłączona jest do pinu 32.



Rysunek 30 Schemat ideowy klienta ESP

4.3.Konfiguracja środowiska automatyki budynkowej Domoticz

Po uruchomieniu i zalogowaniu się przez protokół SSH do Raspberry Pi należy pobrać i zainstalować pakiet Domoticz wpisując w terminal poniższą komendę:

sudo bash -c "$(curl -sSfL https://install.domoticz.com)"

Po zakończeniu instalacji oprogramowania środowisko automatyki domowej Domoticz jest gotowe do uruchomienia i konfiguracji. Usługa odpowiedzialna za uruchomienie środowiska uruchamia się wraz ze startem systemu Raspberry Pi OS.

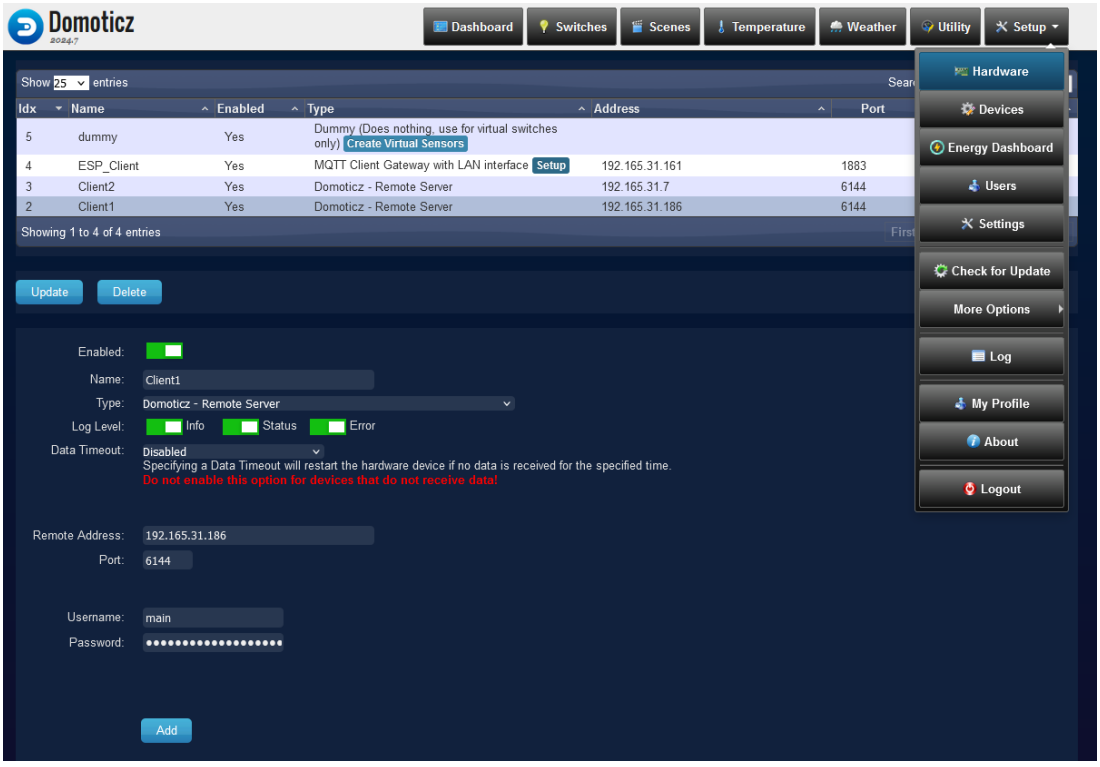
4.3.1. Konfiguracja serwera

Po instalacji środowiska automatyki domowej Domoticz należy uruchomić graficzne środowisko konfiguracyjne. Jest one dostępne z poziomu aplikacji webowej pod adresem IP:

http://192.165.31.161:8080

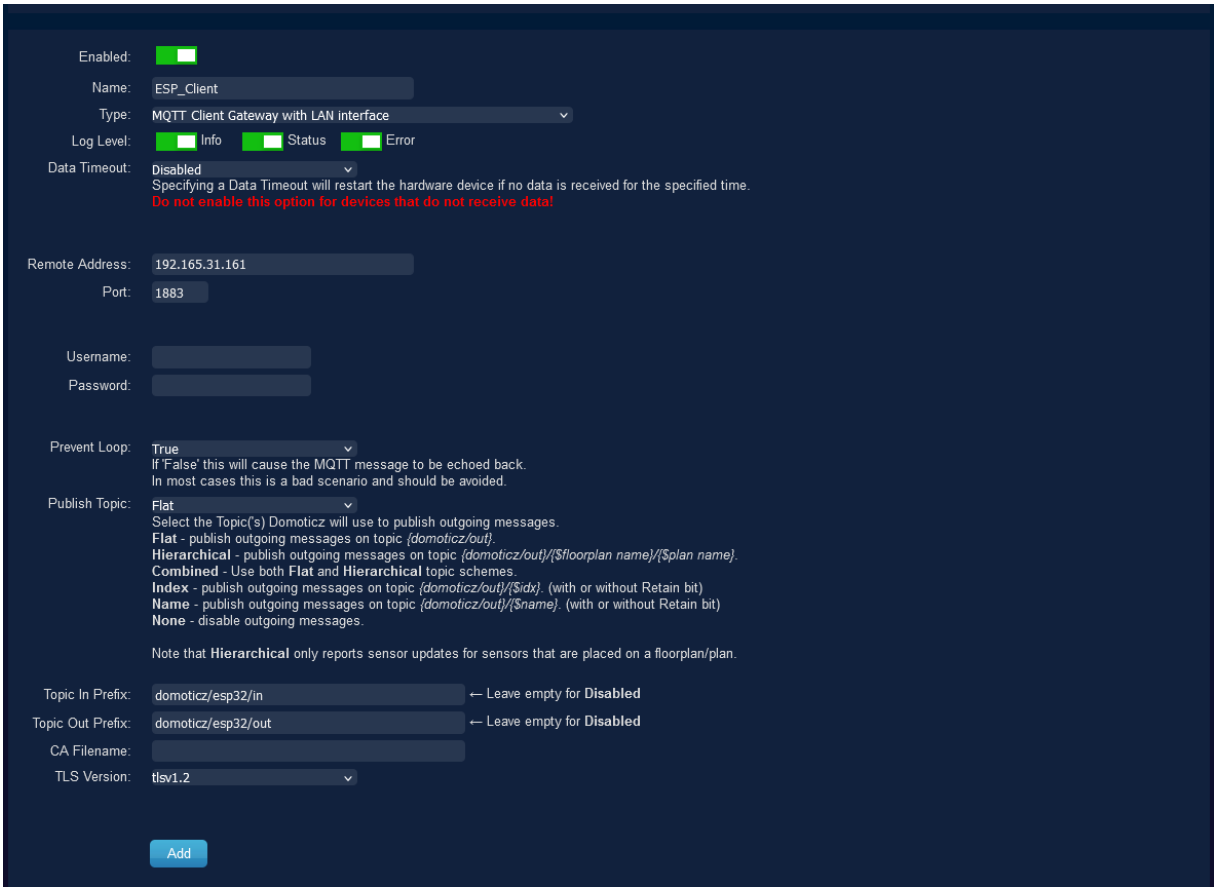
Po jego uruchomieniu ukazuje się panel konfiguracyjny Domoticz. Pierwszym krokiem jest dodanie urządzeń (klientów), którzy są w sieci lokalnej i będą realizowali funkcje pomiarowe

lub uruchamiające urządzenia wykonawcze. W zakładce *Setup* → *Hardware* należy dodać wszystkich klientów właściwie ich nazywając oraz identyfikując platformę (*Type*). Dla klientów zbudowanych na Raspberry Pi należy wybrać *Domoticz – Remote Server* oraz wpisać odpowiadający urządzeniu adres IP.



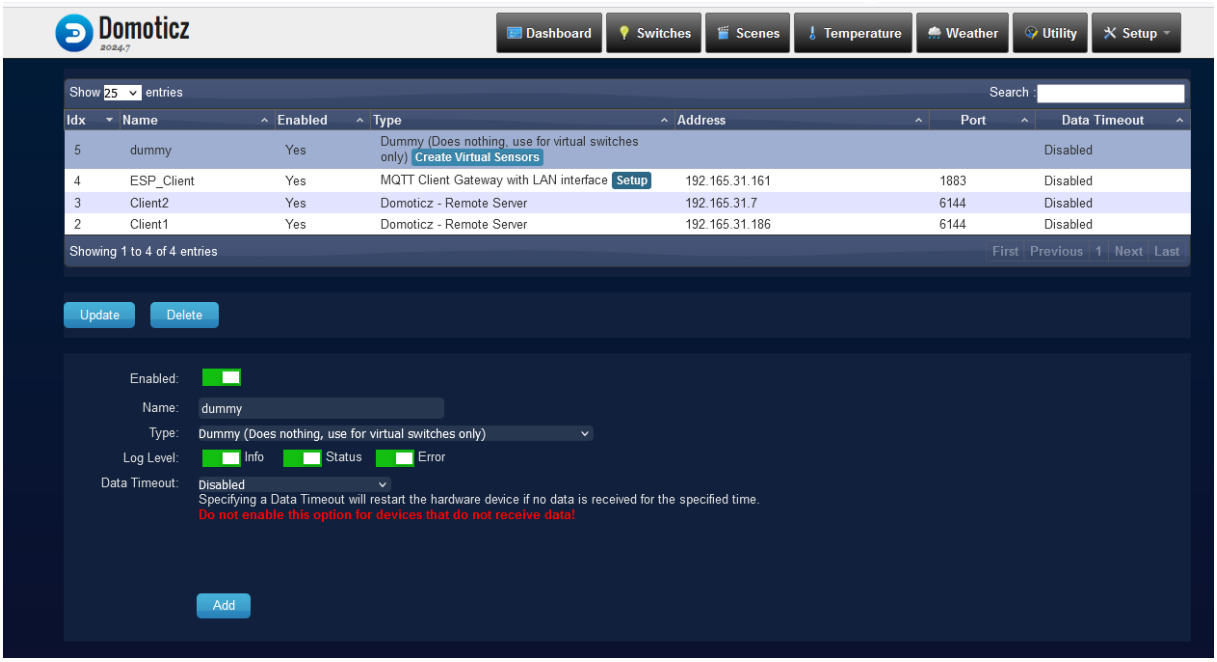
Rysunek 31 Domoticz - konfiguracja klienta Raspberry Pi

W przypadku klienta opartego o platformę ESP wybieramy typ *MQTT Client Gateway with LAN interface* oraz wpisujemy odpowiadający mu adres IP. Dodatkowo należy nadać temat (*topic*) wiadomości wychodzących i przychodzących do ESP (pola *Topic In Prefix, Topic Out Prefix*). Dzięki tym tematom możliwe jest przesyłanie i odbieranie wiadomości przez ESP w formacie JSON.



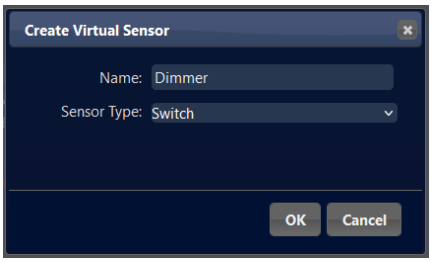
Rysunek 32 Domoticz - konfiguracja ESP jako MQTT Client

Aby móc sterować np. jasnością oświetlenia z poziomu aplikacji Domoticz należy zdefiniować wirtualny przycisk. Tworzy się go z wykorzystaniem tego samego kreatora, co w przypadku deklaracji klienta ESP czy Raspberry Pi. W pole określające typ należy wybrać opcję Dummy (*Does nothing, use for virtual switches only*).



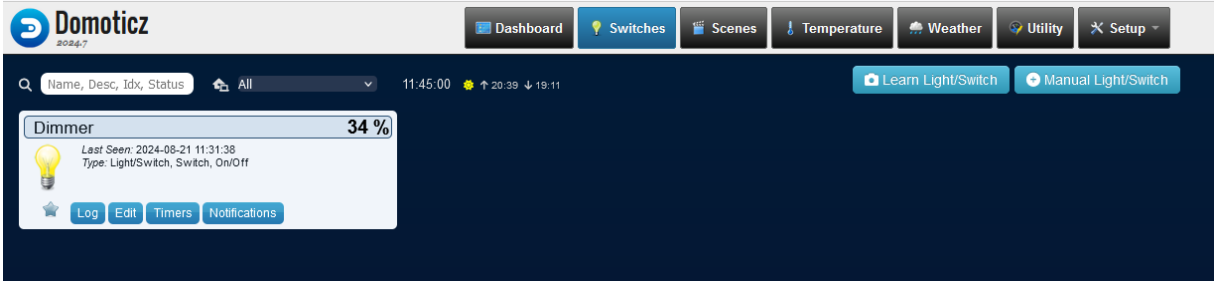
Rysunek 33 Tworzenie wirtualnego przycisku

Po dodaniu wirtualnego przycisku należy wybrać opcję *Create Virtual Sensors*. Pojawi się wówczas okno, w którym należy nadać nazwę oraz wybrać typ wirtualnego sensora / przycisku.



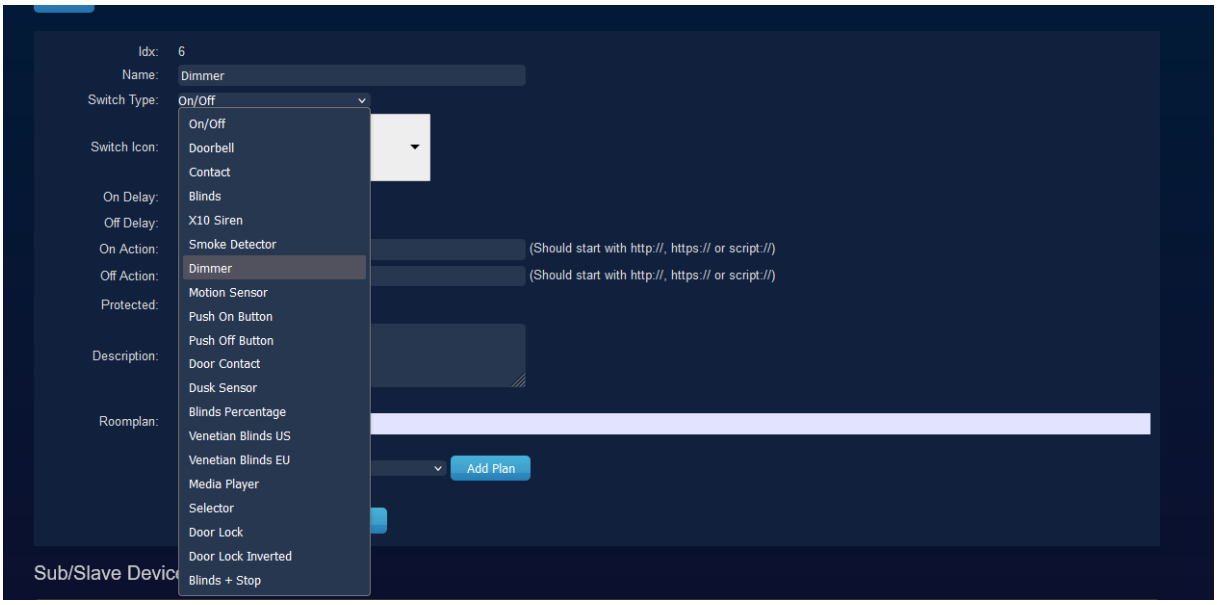
Rysunek 34 Konfiguracja wirtualnego przycisku

Po akceptacji ustawień w zakładce *Switches* pojawi się wirtualny switch. Aby zmienić jego ustawienia należy wybrać opcję *Edit*, które otworzy okno edycji przycisku.



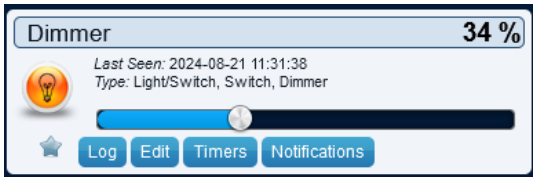
Rysunek 35 Wirtualny switch w Domoticz

W prezentowanym przykładzie *virtual switch* pełni funkcję sterowania jasnością oświetlenia. W związku z tym należy zmienić jego typ na *Dimmer.* Zmiana ta spowoduje, że na symbolu wirtualnego przycisku pojawi się suwak, za pomocą którego będzie można zmieniać jasność oświetlenia.



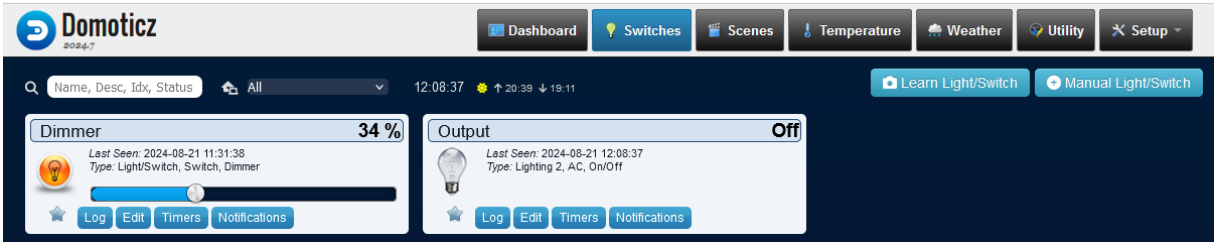
Rysunek 36 Okno edycji wirtualnego przycisku

Po akceptacji zmian wirtualny przycisk zmieni swój wygląd i pojawi się suwak oraz odczyt procentowy wartości ustawionej na nim.



Rysunek 37 Wirtualny przycisk ściemniacza

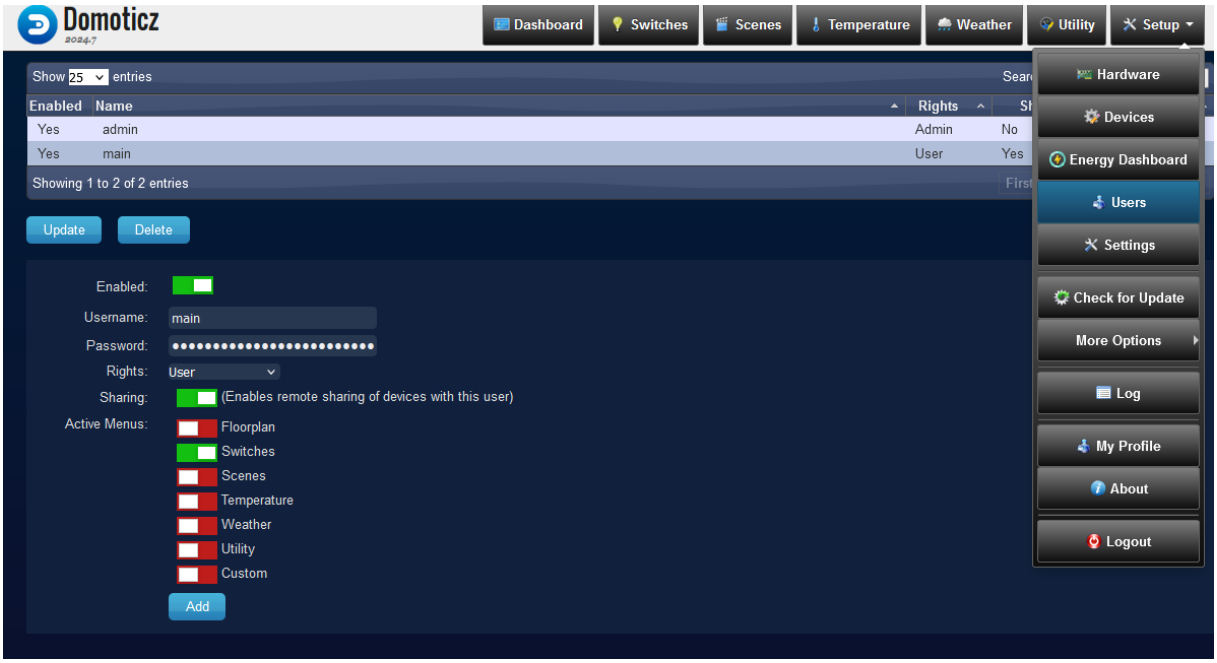
Do pełnej konfiguracji systemu należy jeszcze dodać wirtualny przycisk typu ON/OFF jak na rysunku poniżej.



Rysunek 38 Wirtualne przyciski Domoticz

4.3.2. Konfiguracja klienta

Po zainstalowaniu i uruchomieniu usługi Domoticz na urządzeniu identyfikującym się jako klient uruchamiamy *Setup* → *Users* i tworzymy nowego użytkownika wciskając przycisk *Add*.



Rysunek 39 Tworzenie nowego użytkownika

Aby móc sterować dowolnym zewnętrznym urządzeniem, np. modułem przekaźnikowym lub odczytywać stan przycisku czy czujnika konieczne jest zmapowanie pinów oraz ich konfiguracja. Wpisując w terminal poniższą komendę:

cat /sys/kernel/debug/gpio

wyświetli się lista pinów GPIO wraz z ich numerami identyfikacyjnymi.

Rysunek 40 Lista pinów GPIO

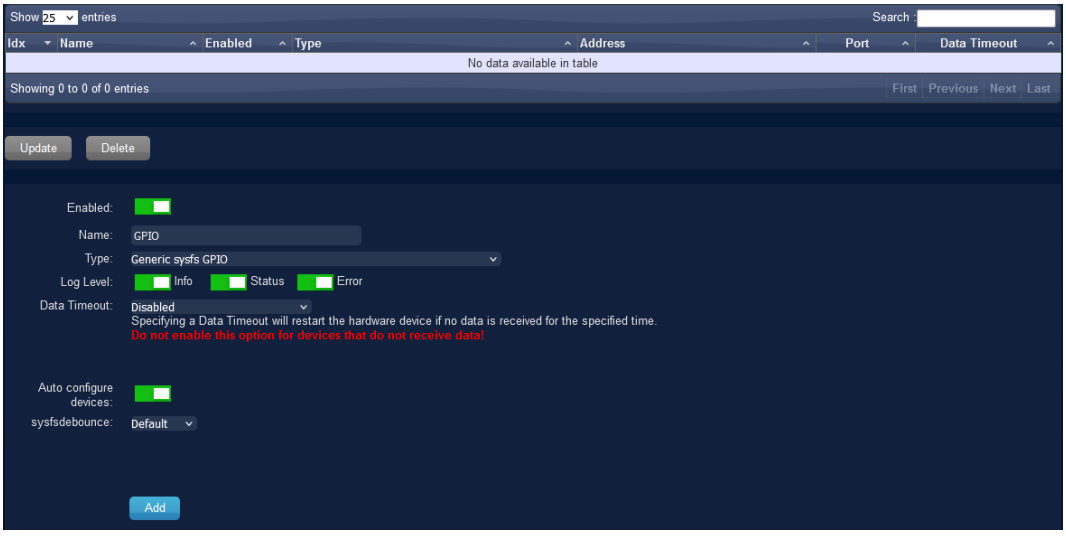
Aby wysterować przekaźnikiem podłączonym do GPIO17 należy odszukać jego numer na wyświetlonej liście. Następnie za pomocą komendy:

echo 529 >/sys/class/gpio/export

oraz

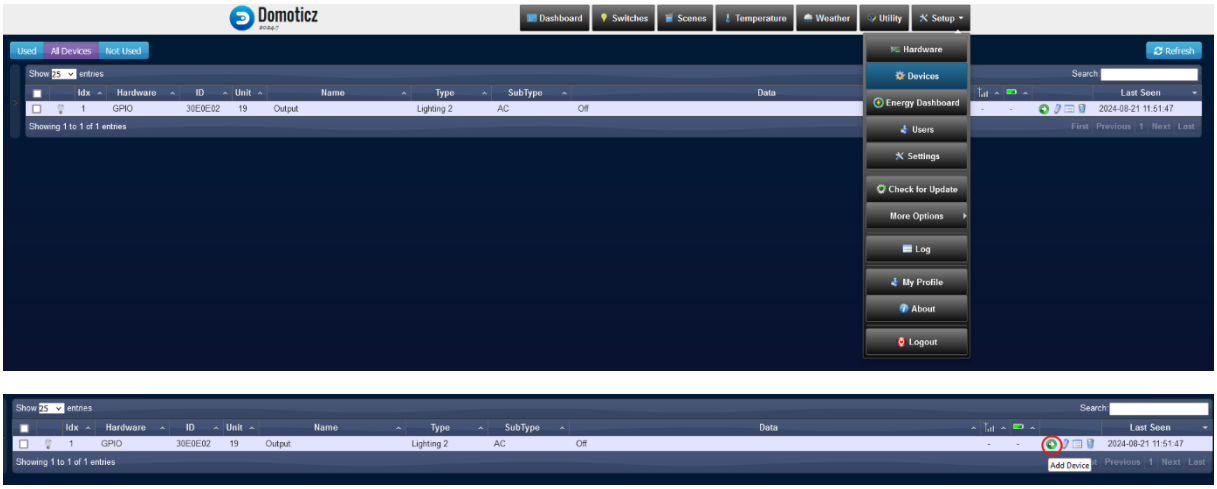
echo out > /sys/class/gpio/gpio529/direction

ustawia się kierunek pinu GPIO17. Mając poprawnie ustawiony kierunek pinu (wyjście) w zakładce *Setup* → *Hardware* należy dodać nowe urządzenie wybierając typ *Generic sysfs GPIO*.



Rysunek 41 Dodanie nowego urządzenia - klient

Po dodaniu urządzenia należy w zakładce *Setup* → *Devices* wybrać *Add Device* (symbol zielonej strzałki).

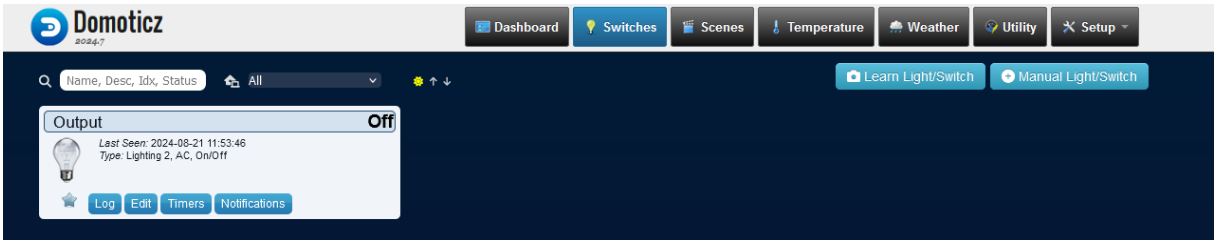


Rysunek 42 Definiowanie nowego urządzenia

Następnie należy wpisać nazwę urządzenia i zatwierdzić przyciskiem *Add Device.*

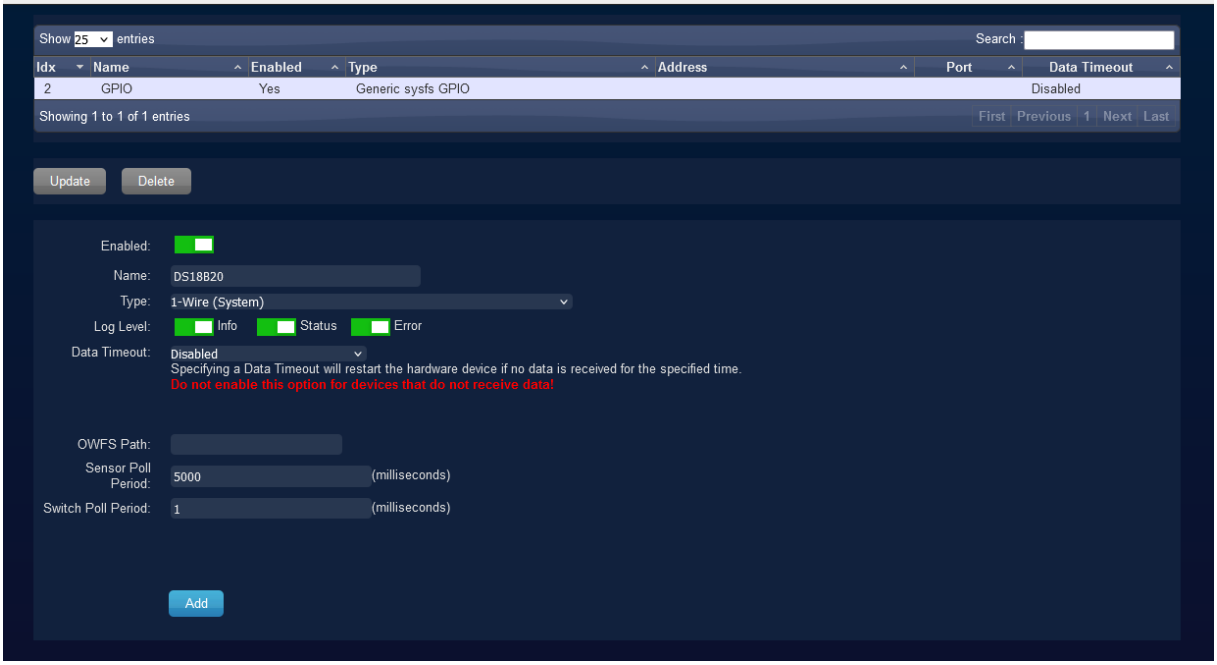
Rysunek 43 Dodaj urządzenie - konfigurator

Po dodaniu nowego urządzenia pojawia się ono w zakładce *Switches* i jest automatycznie podłączone do GPIO17. W przypadku ustawienia większej ilości pinów jako wyjście pojawiłoby się kilka urządzeń, a każde z nich byłoby przypisane do jednego pinu wyjściowego.



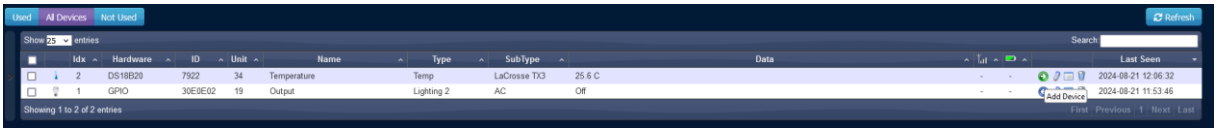
Rysunek 44 Nowe urządzenie

Dodanie czujnika temperatury DS18B20 wykorzystującego komunikację 1-wire odbywa się w sposób zbliżony do konfiguracji przycisków. W zakładce *Setup* → *Hardware* należy skonfigurować nowe ustawienie wybierając typ *1-Wire (System)*.

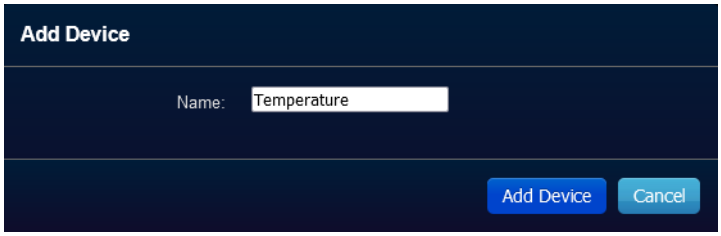


Rysunek 45 Konfiguracja czujnika temperatury

Po skonfigurowaniu urządzenia należy w zakładce *Setup* → *Devices* wybrać *Add Device* (symbol zielonej strzałki) oraz dodać urządzenie.



Rysunek 46 Lista urządzeń klienta



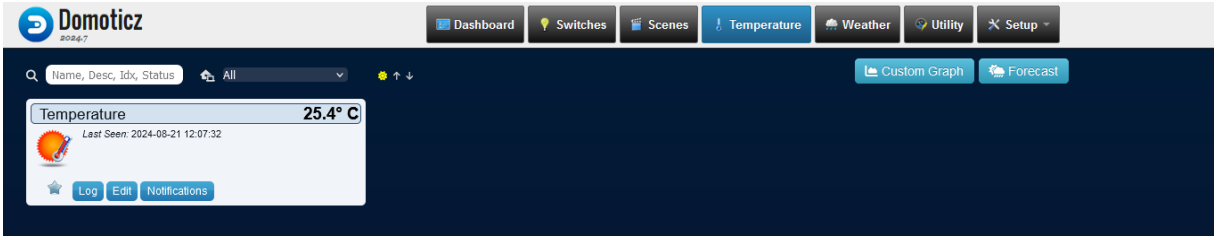
Rysunek 47 Dodanie czujnika temperatury jako urządzenie

Po dodaniu czujnika temperatury jako urządzenie w zakładce *Temperature* pojawi się pole z wartością temperatury. W tym miejscu można również dokonać edycji parametrów lub

konfiguracji czujnika. Aby podłączyć czujnik temperatury do właściwego pinu należy ponownie wykorzystać komendę:

cat /sys/kernel/debug/gpio

i zweryfikować, który z pinów został skonfigurowany jako one-wire. W prezentowanym przypadku jest to GPIO4 (Rysunek 40).



Rysunek 48 Czujnik temperatury na liście urządzeń

4.4.Opis programu

Wymagane biblioteki:

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <ArduinoMqttClient.h>
#include <ArduinoJson.h>

#define LAMP_PIN 32
```

Dane połączenia sieciowego po Wi-Fi:

```
char ssid[] = "Moja_siec_WiFi";
char pass[] = "Moje_WiFi123";
```

Dane brokera MQTT:

```
const char broker[] = "192.165.31.161";
int        port     = 1883;
```

Tematy subskrybowane u brokera MQTT:

```
const char topic[]  = "domoticz/esp32/out";
```

Definicja obiektów client (połączenia Wi-Fi) oraz mqttClient:

```
WiFiClient client;
MqttClient mqttClient(client);

unsigned long previousMillis = 0;
int lamp_brightness;

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

Inicjalizacja pinu sterowania oświetleniem:

```
    pinMode(LAMP_PIN, OUTPUT);
```

Inicjalizacja Wi-Fi:

```
    WiFi.begin(ssid, pass);
    int wifi_ctr = 0;
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }

    Serial.println("WiFi connected");
```

Inicjalizacja połączenia jako klient MQTT:

```
    if (!mqttClient.connect(broker, port))
    {
        Serial.print("MQTT connection failed! Error code = ");
        Serial.println(mqttClient.connectError());
        while (1);
    }

    Serial.println("You're connected to the MQTT broker!");
    Serial.println();
```

Subskrybowanie tematów i odbieranie wiadomości:

```
    mqttClient.onMessage(onMqttMessage);
    mqttClient.subscribe(topic);
}

void loop()
{
```

Utrzymywanie połączenia z brokerem MQTT:

```
    mqttClient.poll();
}
```

Funkcja odbierająca dane w postaci JSON i wyodrębniająca poziom jasności:

```
void onMqttMessage(int messageSize)
{
    String payload = "";
    JsonDocument json_msg;

    while (mqttClient.available())
    {
        payload += (char)mqttClient.read();
    }

    deserializeJson(json_msg, payload);
    lamp_brightness = json_msg["svalue1"];
    Serial.println(lamp_brightness);
```

Sterowanie jasnością oświetlenia:

```
    int pwm_duty = (255 * lamp_brightness) / 100;
    analogWrite(LAMP_PIN, pwm_duty);
}
```

5. Wykaz literatury

[1]. Alexandru Radovici, Cristian Rusu, Ioana Culic, „Komercyjne i przemysłowe aplikacje Internetu rzeczy na Raspberry Pi", Apress, 2020

[2]. Markus Edenhauser, „Node-RED: IoT Programmierung mit ESP32 & MQTT", pixeledi.eu, 2023

[3]. Dhairya Parikh, „Raspberry Pi and MQTT Essentials. A complete guide to helping you build innovative full-scale prototype projects using Raspberry Pi and MQTT protocol", Packt Publishing, 2022

[4]. Dokumentacje techniczne modułów elektronicznych użytych w projektach pobrane ze stron producentów lub dostawców

[5]. Źródło internetowe, dostęp 21.08.2024 r. https://www.raspberrypi.com/documentation /computers/raspberry-pi.html

Spis rysunków