



## 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	3
2.	Minikomputer Raspberry Pi	4
3.	Projekt urządzenia sterującego ogrzewaniem z wykorzystaniem pr	otokołu MQTT 5
3.1.	Dobór elementów systemu	6
3.2.	Schemat ideowy	7
3.3.	Raspberry Pi OS	
3.4.	Node-RED	14
3.5.	Opis programu	
3.6.	Testy systemu	
4.	Projekt systemu automatyki domowej Domoticz	
4.1.	Dobór elementów systemu	
4.2.	Schemat ideowy	
4.3.	Konfiguracja środowiska automatyki budynkowej Domoticz	
4.3.1	I. Konfiguracja serwera	
4.3.2	2. Konfiguracja klienta	
4.4.	Opis programu	
5.	Wykaz literatury	





## 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*).

	21/2 nower		5	EV nower
	3v3 power o			5 v power
	GPIO 2 (SDA) o	30	•	5V power
	GPIO 3 (SCL) o	<b>()</b>	<b>3</b>	Ground
	GPIO 4 (GPCLK0) o	0	<b>3</b>	GPIO 14 (TXD)
	Ground o	<b>0</b>	0	GPIO 15 (RXD)
	GPIO 17 o	00	3 0	GPIO 18 (PCM_CLK)
	GPIO 27 o	(B (	<u> </u>	Ground
	GPIO 22 o	• • • • •	0 0	GPIO 23
	3V3 power o	<b>()</b>	3	GPIO 24
	GPIO 10 (MOSI) o	(E) (	0 0	Ground
	GPIO 9 (MISO) o	200	<b>0</b>	GPIO 25
	GPIO 11 (SCLK) o		0 0	GPIO 8 (CE0)
	Ground •		0	GPIO 7 (CE1)
	GPIO 0 (ID_SD) o	(2) (	0	GPIO 1 (ID_SC)
	GPIO 5 o		•	Ground
	GPIO 6 o		0	GPIO 12 (PWM0)
	GPIO 13 (PWM1) 💿		Ø •	Ground
ATAL MIXE	GPIO 19 (PCM_FS) •		0	GPIO 16
	GPIO 26 o		B0	GPIO 20 (PCM_DIN)
	Ground o		00	GPIO 21 (PCM_DOUT)

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 – kotlownia*) 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:	-55°C : 125 °C
•	Dokładność pomiaru:	±0.5 °C
•	Rozdzielczość pomiaru:	9 – 12 bitów
•	Zasilanie:	3.3 – 5V DC
•	Obudowa:	ТО-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)



Europejski Fundusz Społeczny



#### Dioda LED z rezystancyjnym ograniczeniem prądowym

•	Kolor:	zielony
•	Prąd:	11 mA
•	Rezystancja szeregowa:	100 <b>Ω</b>







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*.

<b>Europejski</b> Wiedza Edukacja	<b>e</b> Rozwój	Europejski Fundusz Społe	eczny *	* * * * * *
🔓 Konfiguracja systemu	- 🗆 X	Konfiguracja systemu		- 🗆 X
OGÓLNE USŁUGI	OPCJE	OGÓLNE	USŁUGI	OPCJE
✓ ustaw hostname:       ispberrypi-client2 jocal         ✓ Ustaw login i hasło       Login:         Login:       pi         Hasło:	_	Włącz SSH Używaj uwierzyte Pozwól tylko na u Ustaw authorized_ke URUCHOM SSH	Iniania hasłem wierzytelnianie klucze s <sup>o</sup> dla 'pi': -KEYGEN	em publiczym

Ilnia Europaiska

Г

Rysunek 10 Konfiguracja systemu - zakładka Ogólne

Fundusze

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.



Unia Europejska Europejski Fundusz Społeczny



≥ pi@raspberrypi-client2: ~ × + ×			×
Windows PowerShell Copyright (C) Microsoft Corporation. All rights reserved.			
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows			
PS C:\Users\sers> <b>ssh</b> pi@raspberrypi-client2 pi@raspberrypi-client2's password: Linux raspberrypi-client2 6.6.31+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.6.31-1+rpt1 (2024-05-29) aar	ch64		
The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright.			
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law. Last login: Tue Aug 20 12:32:05 2024 from fe80::814c:f09:53c6:5fc4%wlan0			
SSH is enabled and the default password for the 'pi' user has not been changed. This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.			
pi@raspberrypi-client2:~ \$			

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: Opening ipv6 listen socket on port 1883.

#### 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 komenda:

#### 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/linuxinstallers/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*.



Europejski Fundusz Społeczny



pi@raspberrypi:~ \$ sudo node-red-start
Start Node-RED
Once Node-RED has started, point a browser at http://192.165.31.161:1880 On Pi Node-RED works better with the Firefox or Chrome browser
Use sudo systemctl enable nodered.service to autostart Node-RED at every boot Use sudo systemctl disable nodered.service to disable autostart on boot
To find more nodes and example flows - go to http://flows.nodered.org
Starting as root systemd service. 19 Aug 13:09:43 - [info] Welcome to Node-RED
<pre>19 Aug 13:09:43 - [info] Node-RED version: v4.0.2 19 Aug 13:09:43 - [info] Node.js version: v18.19.0 19 Aug 13:09:43 - [info] Linux 6.6.31+rpt-rpi-v8 arm64 LE 19 Aug 13:09:46 - [info] Loading palette nodes 19 Aug 13:09:49 - [info] Dashboard version 3.6.5 started at /ui 19 Aug 13:09:50 - [info] Settings file : /home/pi/.node-red/settings.js 19 Aug 13:09:50 - [info] Context store : 'default' [module=memory] 19 Aug 13:09:50 - [info] User directory : /home/pi/.node-red 19 Aug 13:09:50 - [info] User directory : /home/pi/.node-red 19 Aug 13:09:50 - [info] Flows file : /home/pi/.node-red/flows.json 19 Aug 13:09:50 - [info] Server now running at http://127.0.0.1:1880/ 19 Aug 13:09:50 - [warn]</pre>
Your flow credentials file is encrypted using a system-generated key. If the system-generated key is lost for any reason, your credentials file will not be recoverable, you will have to delete it and re-enter your credentials. You should set your own key using the 'credentialSecret' option in your settings file. Node-RED will then re-encrypt your credentials file using your chosen key the next time you deploy a change.
19 Aug 13:09:50 - [info] Starting flows 19 Aug 13:09:50 - [info] Started flows 19 Aug 13:09:50 - [info] [mqtt-broker:RPi] Connected to broker: mqtt://192.165.31.161:1883

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.



Europejski Fundusz Społeczny



User Settings		🗘 config 🚺 i 🗐 🔅 🔻 🔻
	Close	all unused
View	Nodes Install	✓ On all flows
Palette	Node-RED Community catalogue	mqtt-broker
Falette	Q dashboard 102 / 5005 ×	RPi 10
Keyboard	🗑 dashboard-evi 🗷	ui_base
Environment	A set of dashboard nodes for Node-RED • 1.0.2	Node-RED Dashboard
		ui_group
	## Install	[Home] Pomiary p 9
	O.0.2 III O Yours, O montus ago     Instan	[Home] Wykresy 3
	node-red-dashboard C*     A set of dashboard nodes for Node-RED     365	[Home] Pyły 4
		[Home] Pyly - wyk 4
	@cgjgh/node-red-dashboard-2-authentik-auth C Dashboard Auth with Authentik	[Home 2] Stan mie 1
	1.2.3 # 2 weeks ago install	[Home 2] Wykresy 2
		[Home 3] Default
	🔖 0.8.1 🏥 3 years, 11 months ago install	[Home 3] Default 2 1
		ui_tab
	s 1.0.21 💼 6 days ago install	Home 4
	<ul> <li>@5minds/node-red-dashboard-2-processcube-dynamic-table</li> <li>A ui component for showing dynamic Data with actions in a table</li> </ul>	Home 2 2
	🗞 1.1.10 🛗 6 days ago install	Home 3 2
	@5minds/node-red-dashboard-2-processcube-usertask-table A ui component for showing UserTasks in a table	Flow 1

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*).



Europejski Fundusz Społeczny



Node-RED			
9, filter nodes	Flow 1 Flow 2	Edit mqtt in nod	•
comment		Delete	Cancel Done
~ function		O Properties	• 21
function of switch of		Server	none v 🖉 🔸
change D		Action	Subscribe to single topic v
dij range o	and in the	III Topic	Topic
template	and a second sec	⊛ QoS	2 *
do delay d		(+ Output	auto-detect (parsed JSON object, string or buffi $\backsim$
o trigger o		Name	Name
exec b			
d Ster d			
¢ random ¢			
e smooth e			
~ network			
() matt in 👌			
of matt out			
http in			
chttp response 📀			
http request			
websocket in D		B O Enviro	
A ¥		W U Ehabiyo	

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).

Edit mqtt in node > Edit mqtt-broker node					
Delete			Cancel	Update	
Properties				\$	
Name	RPi				
Connection		Security	Messages		
Server	192.165.31.	161	Port 1883		
	🔽 Connect a	automatically			
	Use TLS				
Protocol	MQTT V3.1.	1	~		
Sclient ID	Leave blank for auto generated				
♥ Keep Alive 60					
i Session Vise clean session					

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*.

Edit mqtt in node					
Delete	Cancel Done				
Properties	to International				
Server	RPi ~ /				
Action	Subscribe to single topic v				
📰 Topic	rpi/temperature				
🛞 QoS	2 ~				
🕩 Output	auto-detect (parsed JSON object, string or buffi $\checkmark$				
Name	Name				

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().* 

Edit function node				
Delete			Cancel	Done
Properties				•
Name Pars	e temperature			-
Setup	On Start	On Message	On Stop	
1 const msg 2 3 4 var out = 5 6 7 return ou	_array = msg.pa {payload: pars t;	ayload.split(":"); seFloat(msg_array[1])	};	Su anno ann an ann

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.

Edit funct	ion nod	e							
Delete							Cancel		Done
Prope	rties							•	
Name Name		Parse	state						
Set	up		On	Start	On Me	ssage	On Stop		
1 2 3 4 5 6 7	const var s const const var o	msg_a tate = on_st off_s ut = {	mray msg ring trin payl	<pre>v = msg.payloa g_array[1].tri g = "ON"; ng = "OFF"; load: 0};</pre>	d.split( m();	":");			
8 9 10 11 12 13 14 15	if(on   o } els   o } retur	_strin ut.pay e if ( ut.pay n out;	ig.lo /load /load /load	ocaleCompare(s i = 1; _string.locale i = 0;	tate) == Compare(	0){ state) == (	9){		

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 oczekającego 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 °C:

```
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 oczekającego 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:	-55°C : 125 °C	
•	Dokładność pomiaru:	±0.5 °C	
•	Rozdzielczość pomiaru:	9 – 12 bitów	
•	Zasilanie:	3.3 – 5V DC	
•	Obudowa:	ТО-92	
	XX7 · / · 1 1		11

• 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 <b>Ω</b>



Rysunek 25 Czujnik temperatury DS18B20



Rysunek 27 Raspberry Pi 4B



Rysunek 26 Moduł dwuprzekaźnikowy



Rysunek 28 Moduł ESP32-DevKitC V4



Unia Europejska Europejski Fundusz Społeczny



## 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.



Europejski Fundusz Społeczny





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*  $\rightarrow$  *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.

Domot	CZ		🗾 Dashboa	ard 📍 Switche	s 🍯 Scenes	👃 Temperature	🌧 Weather	😵 Utility 🗙 Setup 🕶
Show 25 🗸 ent	ies						Sear	🗯 Hardware
ldx 🔻 Name		~ Enabled ~	Туре	^ A	ddress		^ Port	🔅 Devices
5 dumm	ý	Yes	Dummy (Does nothing, use for virtu only) Create Virtual Sensors	ual switches				O Energy Dashboard
4 ESP_	Client	Yes	MQTT Client Gateway with LAN int	erface Setup	192.165.31.161		1883	
3 Client	1	Yes	Domoticz - Remote Server		192.165.31.7		6144	📥 Users
2 Client		Yes	Domoticz - Remote Server		192.165.31.186		6144	V Cattings
Showing 1 to 4 or	4 entries							ス Settings
	_							Check for Update
Update	Delete							More Options
Enabl	d:							Log
Nan	e: Client1							
Туј	e: Domoticz - F	Remote Server						📥 My Profile
Log Lev	el: 🗾 Info	Status	Error					
Data Timeo	ut: Disabled							About
	Specifying a Do not ena	a Data Timeout will ble this option fo	restart the hardware device if no data r devices that do not receive data!	is received for the s	pecified time.			9 Logout
Remote Addres	s: 192.165.31	.186						
P	rt: 6144							
Usernan	ie: main							
Passwo	rd: •••••							
	Add							

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.



Europejski Fundusz Społeczny



Enabled:		
Name:	ESP Client	
Type:	<ul> <li>MQTT Client Gateway with LAN interface</li> </ul>	
Log Level:	Info Status Error	
Data Timeout:	Disabled v Specifying a Data Timeout will restart the hardware devi Do not enable this option for devices that do not re	ce if no data is received for the specified time. ceive data!
Remote Address:	192.165.31.161	
Port:	1883	
Lisemame <sup>.</sup>		
Password		
T assired.		
Prevent Loop:	True If False' this will cause the MQTT message to be echoe In most cases this is a bad scenario and should be avoi	d back. ded.
Publish Topic:	Flat Select the Topic('s) Domoticz will use to publish outgoin Flat - publish outgoing messages on topic (domoticz/ou Hierarchical - publish outgoing messages on topic (do Combined - Use both Flat and Hierarchical topic sch Index - publish outgoing messages on topic (domoticz/ Name - publish outgoing messages on topic (domoticz/ None - disable outgoing messages.	g messages. #j: moticz/out]/[\$floorplan name]/(\$plan name]. emes. fout]/(\$idx]. (with or without Retain bit) fout]/[\$name]. (with or without Retain bit)
	Note that Hierarchical only reports sensor updates for	sensors that are placed on a floorplan/plan.
Topic In Prefix:	domoticz/esp32/in ←	Leave empty for Disabled
Topic Out Prefix:	domoticz/esp32/out ←	Leave empty for Disabled
CA Filename:		
TLS Version:	tlsv1.2 v	
	Add	

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*).



Europejski Fundusz Społeczny



tdx       Name       ^ Enabled       Type       ^ Address       ^ Port       Data Timeout         5       dummy       Yes       Dummy (Does nothing, use for virtual switches only)       Disabled       Disabled         4       ESP_Client       Yes       MQTT Client Gateway with LAN interface       Setup       192.165.31.161       1883       Disabled         3       Client2       Yes       Domoticz - Remote Server       192.165.31.77       6144       Disabled         2       Client1       Yes       Domoticz - Remote Server       192.165.31.186       6144       Disabled         Showing 1 to 4 of 4 entries       First       Previous 1       Next       Intervious 1       Next         Update       Delete       Enabled:	Show 2	5 y entries							Sec	arch ·	
1       1		Name	∧ Enabl	od 🔨	Туре		م Add	7055	A Port	∧ Data	Timeout
4       ESP_Client       Yes       MQTT Client Gateway with LAN interface Setup       192.165.31.161       1883       Disabled         3       Client2       Yes       Domoticz - Remote Server       192.165.31.7       6144       Disabled         2       Client1       Yes       Domoticz - Remote Server       192.165.31.186       6144       Disabled         Showing 1 to 4 of 4 entries       First       Previous 1       Next         Update       Delete       First       Previous 1       Next         Vigdate       Delete       First       First       Previous 1       Next         Update       Delete       First       First       Previous 1       Next         Update       Delete       First       First       Previous 1       Next	5	dummy	Yes	u	Dummy (Does only) Create Vi	nothing, use for virtual s irtual Sensors	witches	1000	- T OIL	Disable	d
3       Client2       Yes       Domoticz - Remote Server       192.165.31.7       6144       Disabled         2       Client1       Yes       Domoticz - Remote Server       192.165.31.186       6144       Disabled         Showing 1 to 4 of 4 entries       First       Previous 1       Next         Update       Delete       First       Previous 1       Next         Log Level:       info       Status       Error         Disabled       Specified       Specified       Specified       Specified         Data Timeout:       Disabled       Specified       Specified time.       Specified time.	4	ESP_Clien	t Yes		MQTT Client G	ateway with LAN interfa	ce Setup 19	2.165.31.161	1883	Disable	ł
2       Olient1       Yes       Domoticz - Remote Server       192.165.31.186       6144       Disabled         Showing 1 to 4 of 4 entries       First       Previous 1       Next         Update       Delete       First       Previous 1       Next         Enabled:       Image: Service Servi	3	Client2	Yes		Domoticz - Rer	note Server	19	2.165.31.7	6144	Disable	ł
Showing 1 to 4 of 4 entries       First       Previous       1       Next         Update       Delete       Enabled:       Image: Common Comm	2	Client1	Yes		Domoticz - Rer	note Server	19:	2.165.31.186	6144	Disable	đ
Update       Delete         Enabled:	Showing	g 1 to 4 of 4 er	ntries								
Type: Dummy (Does nothing, use for virtual switches only) Log Level: Info Status Foro Data Timeout: Disabled Specifying a Data Timeout will restart the hardware device if no data is received for the specified time. Bo not triabile this option for devices that do not receive data!	Updat	e Dele Enabled:	te								
Log Level: Info Status Status Constant Status Statu	Updat	e Dele Enabled: Name:	te								
Data Timeout: Disabled Specifying a Data Timeout will restart the hardware device if no data is received for the specified time. Do not enable this option for devices that do not receive data!	Updat	e Dele Enabled: Name: Type:	te dummy	a use for	virtual switchas o	uniu) v					
Specifying a Data Timeout will restart the hardware device if no data is received for the specified time. Boo not enable this option for devices that do not receive data!	Updat	e Dele Enabled: Name: Type: Log Level:	dummy Dummy (Does nothin	g, use for	virtual switches o	nly) v					
	Updat	e Dele Enabled: Name: Type: Log Level: ata Timeout:	te dummy Dummy (Does nothin Info	g, use for Status	virtual switches o	nly) v					

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.

Create Virtual Sen	Create Virtual Sensor						
Name:	Dimmer						
Sensor Type:	Switch						
		OK Cancel					

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.

Domoticz			🗾 Dashboard	? Switches	🎬 Scenes	🜡 Temperature	🌧 Weather	😵 Utility	🗙 Setup -
Q Name, Desc, Idx, Status 🔥 All		11:45:00	🌞 🛧 20:39 🤟 19:11				earn Light/Switch	n 📀 Manu	al Light/Switch
Dimmer           Last Seen: 2024-08-21 11:31:38           Type: Light/Switch, Switch, On/Off           Log         Edit           Timers         Notifications	34 %								

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.

Domoticz	💹 Dashboard 🦻 Switches 🎬 Scenes	🖁 Temperature 🛛 🌧 Weather 🛛 😵 Utility 🕺 Setup 👻
Q Name, Desc, Idx, Status 🔥 All 🗸	12:08:37 🌼 ↑ 20:39 ↓ 19:11	Learn Light/Switch 😔 Manual Light/Switch
Dimmer         34 %           Last Seen: 2024-08-21 11:31:38         Type: Light/Switch, Dimmer           Log         Edit         Timers           Notifications         Notifications	Output         Off           Last Seen: 2024-08-21 12:08:37         Type: Lighting 2, AC, On/Off           Image: Comparison of the second s	

Rysunek 38 Wirtualne przyciski Domoticz





## 4.3.2. Konfiguracja klienta

Po zainstalowaniu i uruchomieniu usługi Domoticz na urządzeniu identyfikującym się jako klient uruchamiamy *Setup*  $\rightarrow$  *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.



Europejski Fundusz Społeczny



pi@raspbei	rypi-client1:~ \$	cat /	sys/kernel/de	bug/gpio	
gpiochip0:	GPIOs 512-569,	parent	: platform/fe	200000.gpio,	pinctrl-bcm2711:
gpio-512	(ID_SDA	)			
gpio-513	(ID_SCL	)			
gpio-514	(GPI02	)			
gpio-515	(GPIO3	)			
gpio-516	(GPIO4		onewire@0	) out	lo
gpio-517	(GPI05	)			
gpio-518	(GPI06	)			
gpio-519	(GPI07	)			
gpio-520	(GPIO8	)			
gpio-521	(GPI09	)			
gpio-522	(GPI010	)			
gpio-523	(GPI011	)			
gpio-524	(GPI012	)			
gpio-525	(GPI013	)			
gpio-526	(GPI014	)			
gpio-527	(GPI015	)			
gpio-528	(GPI016	)			
gpio-529	(GPI017	)			
qpio-530	(GPI018	)			

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*  $\rightarrow$  *Hardware* należy dodać nowe urządzenie wybierając typ *Generic sysfs GPIO*.



Rysunek 41 Dodanie nowego urządzenia - klient





Po dodaniu urządzenia należy w zakładce Setup  $\rightarrow$  Devices wybrać Add Device (symbol zielonej strzałki).

Domoticz Sould J	E Dashboard	Y Switches	Scenes	L Temperature	🌧 Weather	🤪 Utility	* Setup •			
Used Al Devices Not Used						PS H	ardware		2 Refresh	
Show 25 🗸 entries						<b>\$</b> 1	Devices	Sea	rch:	1
Idx         Hardware         ID         Unit         Name           Image: Second state         1         GPIO         30E0E02         19         Output	Type      SubType      Lighting 2      AC	ff		Data	_	🕐 Energ	y Dashboard	Tul n 🔜 n 💿 🖉 🗔 🕅	Last Seen + 2024-08-21 11:51:47	1
Showing 1 to 1 of 1 entries						4	Users	Firs		
						* *	Settings			
						Chec	k for Update			
						More	Options			
							Log			
						å M	y Profile			
						•	About			
						0	Logout			
Show 25 🗸 entries								s	earch	
🔲 Idx ^ Hardware ^ ID ^ Unit ^ Name	<ul> <li>Type</li> <li>SubType</li> </ul>			Data				∧ Ťit ∧ 🗖 ∧	Last Seen	-
GPIO 30E0E02 19 Output	Lighting 2 AC O	f						· · (0)/=	2024-08-21 11:51:47	
Showing 1 to 1 of Lentries								Add Devic	e <sup>pt</sup> Previous 1 Next La	

Rysunek 42 Definiowanie nowego urządzenia

Następnie należy wpisać nazwę urządzenia i zatwierdzić przyciskiem Add Device.

Add Device	
Name: As:	Output O Main Device O Sub/Slave Device
	Add Device Cancel

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*  $\rightarrow$  *Hardware* należy skonfigurować nowe ustawienie wybierając typ *1-Wire (System)*.

Show 25 🗸	entries								Se	arch :		
ldx 🔻 Nar	ne	^	Enabled ^	Туре		^ Ad	ldress	^	Port	^	Data Timeout	^
2 GF	90		Yes	Generic sys	ifs GPIO						Disabled	
Showing 1 to	1 of 1 ent	tries										
Update	Delet	e										
	_	_										
Er	abled:											
	Name:	DS18B20										
	Туре:	1-Wire (Syste	em)									
Log	Level:	Info	Status	Error								
Data Tir	meout:	Disabled Specifying a [ Do not enab	Data Timeout will le this option for	✓ restart the har devices that	rdware device if no data t do not receive data!	is received for the sp	ecified time.					
OWES	Path <sup>.</sup>											
Sens	or Poll			(milling of	ando)							
	Period:	5000		(minisecc	Jilds)							
Switch Poll F	Period:			(milliseco	onds)							
		Add										

Rysunek 45 Konfiguracja czujnika temperatury

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

Used	Al Device	s Not	Used											S Refresh
Show	25 ∨ en	tries											Searc	h.
	ld	к ^ Н	lardware 🔨	ID -	∧ Unit ∧	Name	^	Туре	<ul> <li>SubType</li> </ul>	^	Data	^ Ĭii ^ ➡ ^		Last Seen 👻
>	1 2	DS	S18B20	7922	34	Temperature	Ţ	emp	LaCrosse TX3	25.6 C			🖸 🖉 🗔 🕅	2024-08-21 12:06:32
	<u> </u>	GF	PIO	30E0E02	19	Output	L	ighting 2	AC	Off			Add Device	2024-08-21 11:53:46
Show	uing 1 to 2	of 2 entrie	15											Previous 1 Next Last
	Rysunek 46 Lista urządzeń klienta													
						Add Device	•							
								Name	Tempe	rature				
											Add Device	Cancel		

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).

Domoticz		📰 Dashboard	<b>?</b> Switches	🕤 Scenes	l Temperature	🌧 Weather 🛛 🔗 Utility	v ★ Setup -
Q Name, Desc, Idx, Status 🔥 All	✓ <sup>©</sup> ↑ ↓					🕒 Custom Grapt	Sort Forecast
Temperature           Image: Control of the sector	25.4° C						
Log Edit Notifications							

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

- 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

Rysunek 1 Pinout Raspberry Pi - listwa GPIO [5]	4
Rysunek 2 Idea systemu sterowania ogrzewaniem	5
Rysunek 3 Czujnik temperatury DS18B20	7
Rysunek 4 Moduł dwuprzekaźnikowy	7
Rysunek 5 Raspberry Pi 4B	7
Rysunek 6 Schemat ideowy klient - pomieszczenie	7
Rysunek 7 Schemat ideowy klient - kotłownia	8
Rysunek 8 Raspberry Pi Imager	8
Rysunek 9 Raspberry Pi Imager - konfiguracja	9
Rysunek 10 Konfiguracja systemu - zakładka Ogólne	10
Rysunek 11 Konfiguracja systemu - zakładka Usługi	10
Rysunek 12 Połączenie Raspberry Pi z wykorzystaniem SSH	11
Rysunek 13 Weryfikacja instalacji mosquitto	12
Rysunek 14 Plik mosquitto.conf po edycji	13
Rysunek 15 Uruchomienie Node-RED z terminala	15
Rysunek 16 Instalacja pakietu Node-RED-dashboard	16





Rysunek 17 Dodanie węzła MQTT	. 17
Rysunek 18 Konfiguracja brokera MQTT	.17
Rysunek 19 Ustawienia węzła Node-RED	. 18
Rysunek 20 Diagram Node-RED	. 18
Rysunek 21 Funkcja Parse temperature	. 19
Rysunek 22 Funkcja Parse state	. 20
Rysunek 23 Prezentacja graficzna przesyłanych danych	. 25
Rysunek 24 Idea systemu automatyki domowej	. 26
Rysunek 25 Czujnik temperatury DS18B20	. 28
Rysunek 26 Moduł dwuprzekaźnikowy	. 28
Rysunek 27 Raspberry Pi 4B	. 28
Rysunek 28 Moduł ESP32-DevKitC V4	. 28
Rysunek 29 Schemat ideowy klienta	. 29
Rysunek 30 Schemat ideowy klienta ESP	. 30
Rysunek 31 Domoticz - konfiguracja klienta Raspberry Pi	. 31
Rysunek 32 Domoticz - konfiguracja ESP jako MQTT Client	. 32
Rysunek 33 Tworzenie wirtualnego przycisku	. 33
Rysunek 34 Konfiguracja wirtualnego przycisku	. 33
Rysunek 35 Wirtualny switch w Domoticz	. 33
Rysunek 36 Okno edycji wirtualnego przycisku	. 34
Rysunek 37 Wirtualny przycisk ściemniacza	. 34
Rysunek 38 Wirtualne przyciski Domoticz	. 34
Rysunek 39 Tworzenie nowego użytkownika	. 35
Rysunek 40 Lista pinów GPIO	. 36
Rysunek 41 Dodanie nowego urządzenia - klient	. 36
Rysunek 42 Definiowanie nowego urządzenia	. 37
Rysunek 43 Dodaj urządzenie - konfigurator	. 37
Rysunek 44 Nowe urządzenie	. 37
Rysunek 45 Konfiguracja czujnika temperatury	. 38
Rysunek 46 Lista urządzeń klienta	. 38
Rysunek 47 Dodanie czujnika temperatury jako urządzenie	. 38





Rysunek	48 Czu	inik tem	neratury	na liście	urządzeń		39
Rysuner	TO CZU	jink tem	peratury	na nocie	urząuzen	 •••••••••••••••	