



UMCS

UNIwersytet Marii Curie-Skłodowskiej
w Lublinie

Wydział Matematyki, Fizyki i Informatyki

Kierunek: **informatyka**

Jan Bylina

nr albumu: 303827

Projekt oraz implementacja systemu gromadzenia rozproszonych danych z wykorzystaniem technologii LoRa

Design and implementation of the distributed data collection system
using LoRa technology

Praca licencjacka

napisana w Katedrze Oprogramowania Systemów Informatycznych

Instytutu Informatyki UMCS

pod kierunkiem **dr hab. Przemysława Stpiczyńskiego**

Lublin 2023

Spis treści

Wstęp	5
1 Wykorzystane narzędzia, technologie i protokoły	7
1.1 Urządzenia wykorzystywane w projekcie	7
1.1.1 ESP32	7
1.1.2 Raspberry Pi Pico	8
1.1.3 STM32	9
1.2 Języki programowania i technologie	9
1.2.1 C++ for Arduino	9
1.2.2 C for STM32	9
1.2.3 MicroPython for Raspberry Pi Pico	9
1.3 Bazy danych i pozostałe technologie	9
1.3.1 InfluxDB 2	9
1.3.2 Python for MQTT	10
1.4 Protokoły komunikacyjne	10
1.4.1 MQTT	10
1.4.2 LoRa	10
1.4.3 HTTP	10
1.5 Bazy danych i pozostałe technologie	10
1.5.1 InfluxDB 2	10
1.5.2 Docker	10
1.5.3 PlatformIO	10
2 Istniejące rozwiązania	11
2.1 LoRaWAN	11
2.1.1 The Things Network ?	11
2.1.2 ChirpStack ?	11
2.1.3 Lorient ?	11
2.2 Artykuły	11
2.3 Wpisy w sieci i blogach	11

3	Założenie i Implementacja	13
3.1	Podstawowe cele sieci	13
3.2	Części sieci	13
3.2.1	Węzły sieci	14
3.2.2	Stacja przekaźnikowa	14
3.2.3	Broker wiadomości	14
3.2.4	Baza danych	14
3.3	Wiadomości	15
3.4	Obsługa protokołu	15
3.4.1	Działanie węzłów	15
3.4.2	Działanie przekaźnika	15
3.4.3	Działanie bazy danych i programu zapisującego dane	15
3.5	Implementacja	16
3.5.1	Implementacja węzłów sieci	16
3.5.2	Implementacja stacji przekaźnikowej	17
3.5.3	Implementacja brokera wiadomości	17
3.5.4	Baza danych	18
3.5.5	Program zapisujący dane	18
4	Wdrożenie i testy	21
5	Wnioski i perspektywy rozwoju	23
	Spis listingów	25
	Spis tabel	27
	Spis rysunków	29
	Bibliografia	32

Wstęp

Tu treść wstępu WSTĘP WSTEP —

Rozdział 1

Wykorzystane narzędzia, technologie i protokoły

1.1 Urządzenia wykorzystywane w projekcie

1.1.1 ESP32

ESP32 to jednoukładowy mikrokontroler, zaprojektowany i produkowany przez firmę Espressif Systems. Jego najważniejsze cechy to:

- energooszczędny procesor RISC o częstotliwości do 240 MHz
- 520 kB pamięci SRAM
- WiFi 802.11 b/g/n
- Bluetooth
- liczne interfejsy cyfrowe i analogowe, w tym:
 - UART
 - I2C
 - SPI
 - I2S
 - CAN
 - ADC
 - DAC
 - PWM
 - Ethernet MAC

– USB 2.0

- ...

[21]

Powstało wiele wersji tego układu, różniące się m.in. szybkością procesora, ilością pamięci flash, ilością pinów, ilością interfejsów cyfrowych i analogowych, a także możliwością pracy w trybie bezprzewodowym (WiFi) lub przewodowym (Ethernet)[22]. Najczęściej układ te wykorzystywane różnych projektach IoT, zarówno jako czujniki, jak i serwery.[21]

W projekcie ESP32 zostało wykorzystane w dwóch płytkach TTGO T3 V1.6.1. Jedna z płytek została wykorzystana jako przekaźnik danych pomiędzy siecią LoRa a siecią WiFi, a druga jako część systemu zbierania danych z wykorzystaniem LoRa

1.1.2 Raspberry Pi Pico

Rasperry Pi Pico to płytka z mikrokontrolerem RP2040, zaprojektowana i produkowana przez firmę Raspberry Pi Foundation. Charakteryzuje się ona dwurdzeniowym procesorem ARM Cortex-M0+ o częstotliwości 133 MHz, 264 kB pamięci SRAM oraz 2 MB pamięci flash. Płytkę posiada również wiele interfejsów cyfrowych i analogowych, w tym:

- UART
- I2C
- SPI
- I2S
- ADC
- DAC
- PWM
- USB 1.1

[12, 11] Płytkę tę jest często wykorzystywana przez hobbystów do różnych projektów IoT, a także jako sterownik silników, czy kontroler robotów.[11]

W projekcie dwie płytki zostały wykorzystane jako część systemu zbierania danych.

1.1.3 STM32

STM32 to rodzina 32 bitowych mikrokontrolerów produkowanych przez firmę STMicroelectronics. Bazują one na architekturze ARM Cortex-M, oferują wysoką wydajność i energooszczędność. Cztery główne rodzaje mikrokontrolerów STM32 to:

- Rodzina płytek z częścią kodu F(4/5) i H — oferują największą wydajność
- Rodzina płytek z częścią kodu L — oferują największą energooszczędność
- Rodzina płytek z częścią kodu G/C/F(1/3) — do zastosowań ogólnych
- Rodzina płytek z częścią kodu W(L/B/BA) — do zastosowań bezprzewodowych.
[SRPAWDZIĆ]

[20] Głównym zastosowaniem STM32 są urządzenia wbudowane, w tym urządzenia medyczne, roboty, samochody, a także urządzenia IoT[NEED CITE]

W projekcie wykorzystano dwie płytki STM32WL55, które zostały wykorzystane jako część systemu zbierania danych.

1.2 Języki programowania i technologie

1.2.1 C++ for Arduino

—

1.2.2 C for STM32

—

1.2.3 MicroPython for Raspberry Pi Pico

MicroPython jest językiem

1.3 Bazy danych i pozostałe technologie

1.3.1 InfluxDB 2

—

1.3.2 Python for MQTT

1.4 Protokoły komunikacyjne

—

1.4.1 MQTT

—

1.4.2 LoRa

—

1.4.3 HTTP

—

1.5 Bazy danych i pozostałe technologie

1.5.1 InfluxDB 2

—

1.5.2 Docker

—

1.5.3 PlatformIO

—

Rozdział 2

Istniejące rozwiązania

—

2.1 LoRaWAN

—

2.1.1 The Things Network ?

—

2.1.2 ChirpStack ?

—

2.1.3 Lorient ?

—

2.2 Artykuły

—

2.3 Wpisy w sieci i blogach

Rozdział 3

Założenie i Implementacja

W poniższym rozdziale zaprocentowano założenia, potrzeby i projekt projektu Sieci Mesh na bazie LoRa

3.1 Podstawowe cele sieci

System ma kilka podstawowych założeń:

- Jeden centralny punkt gromadzenia danych
- Zapisywanie danych do bazy danych szeregów czasowych, w celu ich dalszego przetwarzania
- Zbieranie danych z dużego obszaru
- Niezależność od istniejących metod przesyłu danych (WiFi, Sieci komórkowe, Łączność satelitarna)
- Niezależność od platformy sprzętowej
- Zapewniać możliwie dużą dostarczalność pakietów
- Dane czasowe nie muszą być super dokładne (dopuszczalne są drobne opóźnienia)
- Węzły sieci tylko wysyłają dane, same nie konsumują przychodzących wiadomości

3.2 Części sieci

W celu uzyskania wyżej wspomnianych założeń, zaproponowano system złożonych z kilku części:

- Węzły sieci
- Stacje przekaźnikowe
- Broker wiadomości
- Baza danych

Na rysunku 3.1 przedstawiono schemat ideowy systemu.

3.2.1 Węzły sieci

Węzły sieci to urządzenia wyposażone w moduł LoRa i odpowiednie oprogramowanie pozwalające na pełną obsługę sieci. Gdy węzeł odbierze wiadomość, sprawdza jej poprawność i rozsyła ją dalej w celu zapewnienia jak największego zasięgu i dostarczalności

Urządzenie to może być również wyposażone w różnego rodzaju czujniki, które dostarczają danych bazy danych.

3.2.2 Stacja przekaźnikowa

Stacja przekaźnikowa to urządzenie wyposażone zarówno w moduł LoRa jak i moduł umożliwiający komunikację z siecią Internet (np. moduł WiFi lub moduł bazy Ethernet).

Urządzenie to odbiera przychodzące wiadomości LoRa i przesyła je do brokera wiadomości

3.2.3 Broker wiadomości

Broker wiadomości to program, działający na komputerze mającym dostęp do sieci, umożliwia on wydajną komunikację pomiędzy Stacją Przekaźnikową a Bazą Danych

3.2.4 Baza danych

Baza Danych umożliwiającą zapisywanie sporej ilości danych, uwzględniając również ich czas (baza danych szeregów czasowych). O zapisy danych z brokera wiadomości do bazy danych dba osobny program, który powinien sprawdzać również poprawność tych wiadomości, jak i dbać o to by nie zapisywać powtórzonych wiadomości

3.3 Wiadomości

Każda z wiadomości przesyłanych za pomocą tych sieci powinna mieć formę jak zaprezentowano na listingu 3.1

Zawiera ona pola:

- `ttl` — [Ang. time to live — czas życia] Wartość określająca maksymalną liczbę skoków pomiędzy węzłami sieci. Domyślnie wynosi 10, może zostać wydłużona w zależności od wielkości planowanej sieci
- `m_id` — UUID [1] wiadomości, gwarantujący niepowtarzalności tej wiadomości. Ułatwia również jej dalsze przetwarzanie
- `d_id` — numer identyfikacyjny urządzenia z którego pochodzi wiadomość
- `values` — słownik zawierający dane z urządzenia, do zapisania w bazie

3.4 Obsługa protokołu

3.4.1 Działanie węzłów

Wiadomości generowane są przez węzły sieci, zawierając wszystkie niezbędne pola (wymienione wyżej) i odczyty z czujników zamieszczonych na węźle. Następnie zostaje ona rozesłana do wszystkich węzłów w zasięgu (broadcasting).

Węzeł odbierając wiadomość, sprawdza jej poprawność (czy jest odpowiednio sformatowana, czy zawiera wszystkie potrzebne pola), i jeżeli wiadomość jest poprawna, a pole `ttl` jest większe od 0 rozsyła wiadomość dalej. Sprawdzanie wiadomości odbywa się by wyeliminować wiadomości niepoprawne z sieci.

3.4.2 Działanie przekaźnika

Przekaźnik odbiera wiadomości, i przesyła je do brokera wiadomości. Nie sprawdza poprawności wiadomości by zapewnić maksymalną wydajność i niezawodność.

3.4.3 Działanie bazy danych i programu zapisującego dane

Program pobiera kolejne wiadomości od brokera i przetwarza je w kolejności:

1. Sprawdzenie poprawności wiadomości
2. Sprawdzenie czy wiadomość nie została już sprawdzona (na podstawie pola `m_id`)

3. Zapisanie danych ze słownika `values` do bazy danych i przyporządkowanie ich do `d_id`, oznaczenie ich znacznikiem czasowym.
4. Zapisanie w pamięci operacyjnej `m_id`, potem do wykorzystania w kroku 2

Baza danych powinna przechowywać dane, takie jak:

- `d_id` — identyfikator urządzenia
- znacznik czasowy
- wartość pomiaru
- nazwa pomiaru

3.5 Implementacja

3.5.1 Implementacja węzłów sieci

W wyniku pracy nad systemem zbierania danych przygotowano implementację węzłów sieci w wykonrzystaniem 2 platform sprzętowych:

- Raspberry Pi Pico
- TTGO LoRa32

Raspberry Pi Pico

W wyniku pracy nad systemem zostały przygotowane dwa, identyczne urządzenia oparte o Raspberry Pi Pico. Urządzenie zostało wyposażone w moduł LoRa SX1262 [2](z wykorzystaniem płytki rozwojowej *Waveshare SX1262 LoRa Node Module* [3]) oraz czujnik temperatury i wilgotności DHT11. Urządzenie zostało zaprogramowane w języku MicroPython z wykorzystaniem biblioteki `micropySX126X` [9]. Urządzenie wysyła wiadomości zawierające odczyty z czujnika co 5 minut. Wysyłanie wiadomości odbywa się w sposób asynchroniczny, dzięki czemu urządzenie może wykonywać inne operacje w tym czasie.

TTGO LoRa32

W wyniku pracy nad systemem zostało przygotowane jedno urządzenie oparte o TTGO LoRa32. Urządzenie zostało wyposażone w moduł LoRa SX1276 [5]. Urządzenie zostało zaprogramowane w języku C++ z użyciem PlatformIO, ekosystemu do programowania urządzeń IoT. [17]. W programie została wykorzystane biblioteki:

- `arduino-LoRa` — biblioteka do obsługi modułu LoRa [19]
- `ArduinoJson` — biblioteka do obsługi formatu JSON [7]
- `Adafruit-SSD1306` — biblioteka do obsługi wyświetlacza [6] OLED
- `ESPRandom` — biblioteka do obsługi sprzętowego generatora liczb pseudolosowych [15]

Urządzenie wysyła wiadomości zawierające odczyty z czujnika co 5 minut. Wiadomości zawierają wartości losowe, wygenerowane za pomocą sprzętowego generatora liczb pseudolosowych zintegrowanego z mikrokontrolerem ESP32. Wysyłanie wiadomości odbywa się w sposób asynchroniczny, dzięki czemu urządzenie może wykonywać inne operacje w tym czasie.

3.5.2 Implementacja stacji przekaźnikowej

W wyniku pracy nad systemem zostało przygotowane jedno urządzenie oparte o płytkę TTGO LoRa32. Urządzenie zostało wyposażone w moduł LoRa SX1276 [5] oraz moduł WiFi ESP32 [21]. Urządzenie zostało zaprogramowane w języku C++ z wykorzystaniem PlatformIO [17]. W programie została wykorzystane biblioteki:

- `arduino-LoRa` — biblioteka do obsługi modułu LoRa [19]
- `Adafruit-SSD1306` — biblioteka do obsługi wyświetlacza OLED [6]
- `WiFi` — biblioteka do obsługi modułu WiFi ESP32, zawarta w pakiecie `arduino-esp32` [10]
- `PubSubClient` — biblioteka do obsługi protokołu MQTT [4]

Urządzenie odbiera wiadomości LoRa i przesyła je do brokera wiadomości za pomocą protokołu MQTT. Wysyłanie wiadomości odbywa się w sposób asynchroniczny, dzięki czemu urządzenie może wykonywać inne operacje w tym czasie. Urządzenie wyświetla na ekranie OLED informacje o stanie sieci LoRa, oraz otrzymanych wiadomościach. Urządzenie nie sprawdza poprawności wiadomości, by zapewnić jak największą wydajność i niezawodność.

3.5.3 Implementacja brokera wiadomości

Aby zapewnić niezawodność i wysoką wydajność komunikacji zdecydowano użyć otwartoźródłowego brokera wiadomości Mosquitto [16]. Jest to sprawdzony, wydajny i niezawodny program, który jest szeroko stosowany w systemach IoT, również w przemyśle. [16]

3.5.4 Baza danych

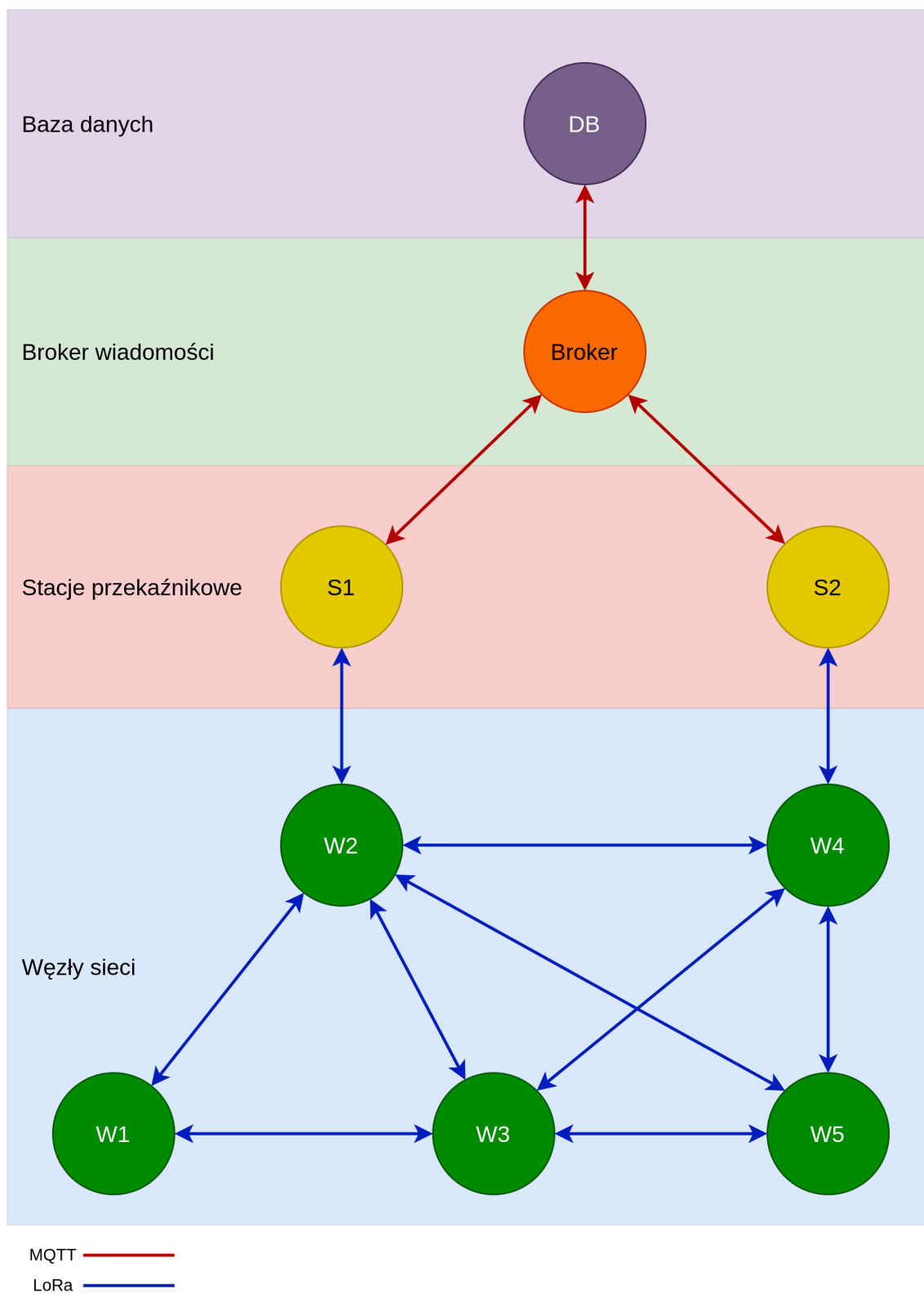
W celu zapisywania danych zdecydowano użyć bazy danych InfluxDB 2.0 [13]. Jest to baza danych szeregów czasowych, która jest wydajna i niezawodna. [13]

3.5.5 Program zapisujący dane

W celu zapisywania danych zdecydowano użyć programu napisanego w języku Python. Program został napisany w języku Python, z wykorzystaniem bibliotek:

- `paho-mqtt` — biblioteka do obsługi protokołu MQTT [8]
- `influxdb-client` — biblioteka do obsługi bazy danych InfluxDB [14]
- `redis-py` — biblioteka do obsługi bazy danych Redis [18]

Ostania, wymieniona biblioteka została użyta do połączenia się z bazą danych w pamięci w celu przechowywania identyfikatorów wiadomości, które zostały już sprawdzone. Dzięki temu program nie zapisuje powtórzonych wiadomości do bazy danych.



Rysunek 3.1: Diagram prezentujący schemat ideowy systemu

```
1 {  
2   "d_id": "id_233",  
3   "values": {  
4     "temp": "21",  
5     "hum": "50",  
6     "press": "1000",  
7     "light": "100",  
8     "co2": "1000",  
9     "pm25": "10",  
10    "pm10": "20"  
11  },  
12  "ttl": 10,  
13  "m_id": "eaa17a7b-9388-43b6-9310-731c942fc6b9"  
14 }
```

Listing 3.1: Przykładowa wiadomość przesyłana przez system

Rozdział 4

Wdrożenie i testy

Rozdział 5

Wnioski i perspektywy rozwoju

Spis listingów

3.1	Przykładowa wiadomość przesyłana przez system	20
-----	---	----

Spis tabel

Spis rysunków

3.1	Diagram prezentujący schemat ideowy systemu	19
-----	---	----

Bibliography

- [1] ietf. *A Universally Unique Identifier (UUID) URN Namespace*. 2005. (Visited on 04/23/2023).
- [2] Semtech. *SX1261/2 Low Power Long Range Transceiver*. 2019. URL: https://www.waveshare.com/w/upload/e/e1/DS_SX1261-2_V1.2.pdf (visited on 04/16/2023).
- [3] Waveshare. *Waveshare SX1262 LoRa Node Module 868MHz*. 2019. URL: <https://www.waveshare.com/wiki/Pico-LoRa-SX1262> (visited on 04/16/2023).
- [4] knolleary. *pubsubclient*. 2020. URL: <https://github.com/knolleary/pubsubclient> (visited on 04/16/2023).
- [5] Semtech. *SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver*. 2020. URL: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE (visited on 04/16/2023).
- [6] adafruit. *Adafruit_SSD1306*. 2023. URL: https://github.com/adafruit/Adafruit_SSD1306 (visited on 04/16/2023).
- [7] bblanchon. *ArduinoJson*. 2023. URL: <https://github.com/bblanchon/ArduinoJson> (visited on 04/16/2023).
- [8] Eclipse. *Paho MQTT Python Client*. 2023. URL: <https://www.eclipse.org/paho/> (visited on 04/23/2023).
- [9] ehong-tl. *micropySX126X*. 2023. URL: <https://github.com/ehong-tl/micropySX126X> (visited on 04/16/2023).
- [10] espressif. *arduino-esp32*. 2023. URL: <https://github.com/espressif/arduino-esp32> (visited on 04/16/2023).
- [11] Raspberry Pi Foundation. *Raspberry Pi Documentation*. 2023. URL: <https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html> (visited on 04/16/2023).

- [12] Raspberry Pi Foundation. *Raspberry Pi Pico Datasheet*. 2023. URL: <https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf> (visited on 04/16/2023).
- [13] InfluxData. *InfluxDB Documentation*. 2023. URL: <https://docs.influxdata.com/influxdb/v2.7/> (visited on 04/23/2023).
- [14] influxdata. *InfluxDB Python client library*. 2023. URL: <https://docs.influxdata.com/influxdb/v2.7/api-guide/client-libraries/python/> (visited on 04/23/2023).
- [15] moritz89. *ESPRandom*. 2023. URL: <https://github.com/moritz89/ESPRandom> (visited on 04/16/2023).
- [16] Eclipse Mosquitto. *Eclipse Mosquitto*. 2023. URL: <https://mosquitto.org/> (visited on 04/23/2023).
- [17] PlatformIO. *PlatformIO Documentation*. 2023. URL: <https://docs.platformio.org/en/latest/> (visited on 04/23/2023).
- [18] Redis. *redis-py*. 2023. URL: <https://redis.io/docs/clients/python/> (visited on 04/23/2023).
- [19] sandeepmistry. *arduino-LoRa*. 2023. URL: <https://github.com/sandeepmistry/arduino-LoRa> (visited on 04/16/2023).
- [20] STMicroelectronics. *STM32 Overview*. 2023. (Visited on 04/23/2023).
- [21] Espressif Systems. *ESP32 Datasheet*. 2023. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (visited on 04/16/2023).
- [22] Espressif Systems. *ESP32 SoCs*. 2023. URL: <https://www.espressif.com/en/products/socs> (visited on 04/16/2023).