

Politechnika Warszawska

W Y D Z I A Ł E L E K T R Y C Z N Y



Instytut Elektrotechniki Teoretycznej i Systemów Informacyjno-Pomiarowych

Praca dyplomowa magisterska

na kierunku Elektrotechnika
w specjalności Systemy Wbudowane

Prywatna sieć czujnikowa wykorzystująca standard LoRa

inż. Mikołaj Rosiński

numer albumu 290988

promotor
dr inż. Łukasz Makowski

WARSZAWA 2023

Prywatna sieć czujnikowa wykorzystująca standard LoRa
Streszczenie

Streszczenie po polsku

Słowa kluczowe:

Private sensor network using the LoRa standard
Abstract

Abstract in English

Keywords:

Spis treści

1	Wstęp	9
2	Sieci w standardzie LoRa	11
2.1	LoRaWAN	11
3	Przygotowanie środowiska programistycznego	13
3.1	Rozpoczęcie projektu z PlatformIO Core	13
3.2	Praca z PlatformIO	14
3.2.1	Uruchamianie projektu	14
3.2.2	Zarządzanie bibliotekami	15
4	Implementacja oprogramowania	17
4.1	Framework oraz biblioteki	17
4.1.1	Wykorzystane biblioteki	18
4.1.2	Ograniczenia związane z wykorzystaniem Arduino oraz STM32duino	18
4.2	Implementacja oprogramowania elementów sieci	18
4.2.1	Oprogramowanie modułu MASTER	21
4.2.2	Oprogramowanie modułów SLAVE	21
4.3	Implementacja oprogramowania modułu serwera sieciowego	21
5	Podstawowe testy implementacji	23
6	Badania działającej sieci	25
7	Podsumowanie	27
	Bibliografia	29
	Spis rysunków	31

Rozdział 1

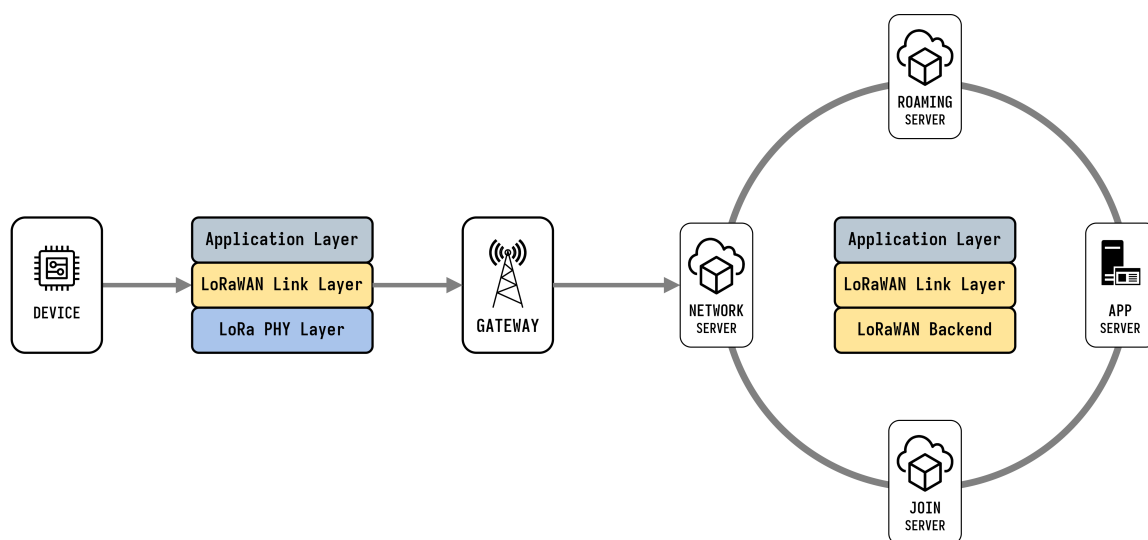
Wstęp

Intro

Rozdział 2

Sieci w standardzie LoRa

2.1 LoRaWAN



Rysunek 1. Schemat architektury sieci LoRaWAN

Rozdział 3

Przygotowanie środowiska programistycznego

Oprogramowanie wszystkich elementów zostało napisane z wykorzystaniem PlatformIO. Narzędzie pozwala na budowanie pod systemy wbudowane na wiele platform [1], w tym wykorzystane do zbudowania sieci STMicroelectronics STM32 Nucleo. Do kompilacji kodu źródłowego możliwe jest użycie wtyczki do edytora Visual Studio Code „PlatformIO IDE” lub samodzielnego narzędzia CLI (ang. *Command Line Interface*) „PlatformIO Core”.

Rozwiązanie to zostało wybrane jako główne narzędzie do kompilacji oraz wgrywania kodu źródłowego, z uwagi na to, że działa na wielu platformach. Dzięki temu nie jest wymagane instalowanie oraz ustawianie osobnych, dedykowanych środowisk dla każdej z wykorzystywanych platform. Jedynym wymogiem, aby móc zacząć pracę jest zainicjowanie projektu oraz ustawienie podstawowej konfiguracji. Zadanie to jest bardzo proste, ponieważ dokumentacja narzędzia jest rozbudowana i bardzo szczegółowa.

3.1 Rozpoczęcie projektu z PlatformIO Core

Całość sieci składa się z dwóch oddzielnych projektów – pierwszy z nich to projekt uniwersalny dla modułów MASTER oraz SLAVE sieci LoRa, drugi natomiast wykorzystywany jest do mikrokontrolera Adafruit Feather M0. Aby rozpocząć nowy projekt, należy wykorzystać komendę, gdzie argumentem jest docelowy mikrokontroler:

```
pio project init --board <board>
```

W przypadku projektu dla sieci LoRa wykorzystane zostały płytki Nucleo L152RE, stąd argumentem było `nucleo_l152`, natomiast dla projektu serwera sieci lokalnej – `adafruit_feather_m0`. Użycie komendy rozpoczyna proces tworzenia nowego projektu. Na podstawie podanego argumentu tworzony jest plik konfiguracyjny. Zdefiniowane zostają platforma projektu oraz wykorzystywany framework. W przypadku obu projektów wybrany został ten wykorzystywany przez Arduino z uwagi

na dużą dostępność bibliotek, które działają bez potrzeby modyfikowania ich kodu źródłowego. Dodatkowo zdefiniowana została tutaj prędkość transmisji portu szeregowego.

W projekcie dla modułów sieci wykonana została modyfikacja pliku konfiguracyjnego – elementy wygenerowane przez narzędzie CLI PlatformIO przeniesione zostały do osobnej sekcji [base_config], natomiast konfiguracje dla poszczególnych modułów znajdują się w dedykowanych „środowiskach”. Wprowadzone zmiany zostały dokładniej opisane w sekcjach o implementacji oprogramowania na poszczególne moduły (4.2, 4.3).

Poza plikiem konfiguracyjnym, narzędzie generuje też podstawową strukturę plików całego. Powstaje folder src, który dedykowany jest dla plików źródłowych, include dla plików nagłówkowych, lib dla bibliotek lokalnych oraz tests do testów jednostkowych, jeżeli planowane jest użycie ich.

3.2 Praca z PlatformIO

Po stworzeniu projektu możliwe jest przystąpienie do pisania kodu źródłowego na wybraną platformę. PlatformIO udostępnia możliwość kompilowania kodu oraz wgrywania go na docelowe urządzenie poprzez jedną komendę lub jeden przycisk w edytorze tekstu. Jest to bardzo dobre rozwiązanie, ponieważ dzięki temu możliwe jest skupienie się na rozwoju kodu źródłowego, zamiast czekania aż projekt będzie możliwy do uruchomienia i sprawdzenia.

3.2.1 Uruchamianie projektu

Uruchomienie projektu jest w przypadku PlatformIO rozumiane poprzez wykonanie kompilacji (build), wgranie skompilowanego kodu na urządzenie docelowe (upload) lub wykonanie zdefiniowanego zestawu testów jednostkowych (test). Aby uruchomić projekt należy wykorzystać komendę:

```
pio run [OPTIONS]
```

Argumentami dodatkowymi mogą być:

- environment: element konfiguracji projektu, który określa zależności w kwestiach kompilacji (np. flagi budowania projektu), programowania (wgrywania kodu) docelowych urządzeń, testów jednostkowych lub wykorzystanych bibliotek,
- target: cel uruchomienia (np. kompilacja albo kombinacja kilku celów jednocześnie),
- upload-port: port, do którego podłączone jest urządzenie i na które ma zostać wgrany kod. Szczególnie użyteczne w przypadku, gdy pracuje się na wielu urządzeniach jednocześnie,
- monitor-port: port, na którym po zakończeniu procesu ma zostać otwarty monitor portu szeregowego.

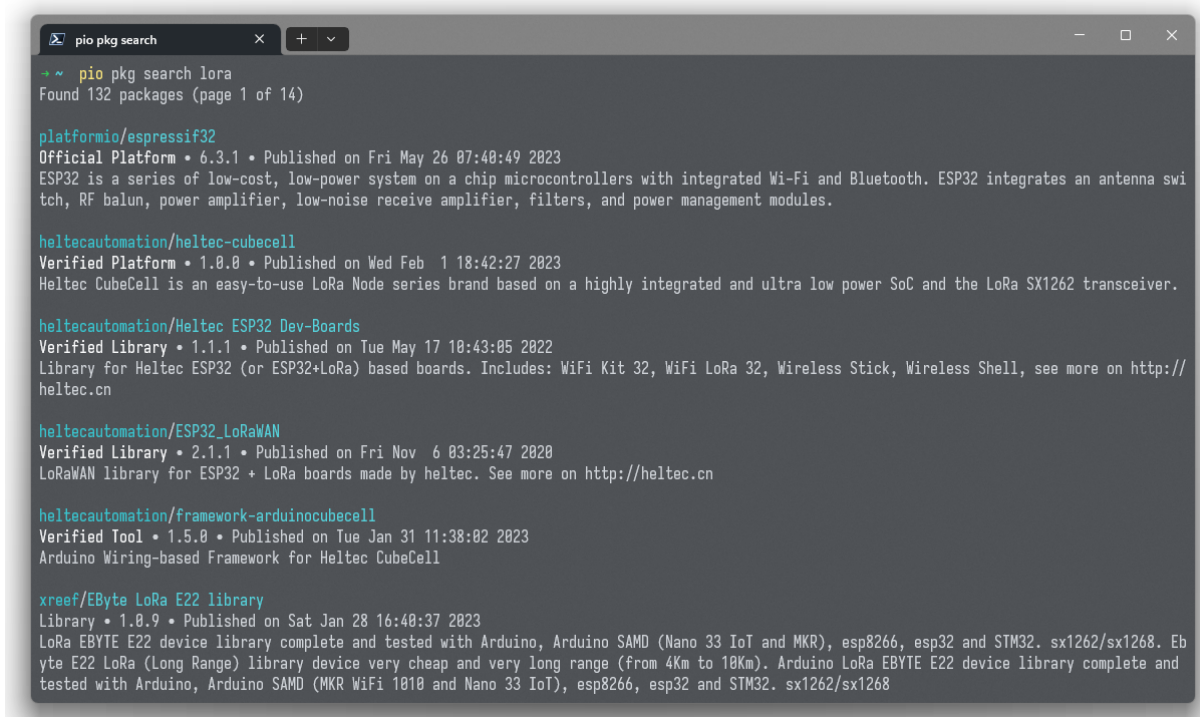
W przypadku opcji związanych z portem, jeżeli nie zostaną sprecyzowane (podane jako argument do komendy), PlatformIO będzie próbował wykryć je automatycznie. Dostępne jest jeszcze kilka innych opcji, jednakże są one znacznie rzadziej wykorzystywane, ponieważ ich domyślne opcje są tymi, które są najczęściej ustawiane.

3.2.2 Zarządzanie bibliotekami

PlatformIO posiada wbudowany moduł dedykowany do zarządzania bibliotekami oraz innymi zasobami dołączanymi do projektu. Dzięki wykorzystaniu odpowiedniej podkomendy z zestawu:

```
pio pkg [COMMAND]
```

możliwe jest przeszukiwanie, instalowanie z, aktualizacja lub publikowanie do rejestru dostępnych bibliotek. Podczas wyszukiwania możliwe jest też zastosowanie filtrów, które w znacznym stopniu zmniejszają ilość wyników i przybliżają do znalezienia tego pasującego. Wykorzystując tę operację zainstalowane zostały potrzebne do projektów biblioteki (wbudowane dla frameworku Arduino, tak jak „Wire” czy te, które opublikowane zostały na platformie GitHub i dodane do rejestru PlatformIO). Na rys. 2 przedstawiony zostały przykładowy wynik wyszukiwania dostępnych bibliotek związanych z hasłem „LoRa”. Każdy wynik zawiera informację: nazwę, typ paczki, biblioteki, która została znaleziona, najnowszą wersję, datę publikacji oraz krótki opis tego czym dana paczka, biblioteka są. Komenda pokazuje także informacje o tym ile wyników zostało znalezionych.



```
pio pkg search
~ pio pkg search lora
Found 132 packages (page 1 of 14)

platformio/espressif32
Official Platform • 6.3.1 • Published on Fri May 26 07:40:49 2023
ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and Bluetooth. ESP32 integrates an antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules.

heltecautomation/heltec-cubecell
Verified Platform • 1.0.0 • Published on Wed Feb 1 18:42:27 2023
Heltec CubeCell is an easy-to-use LoRa Node series brand based on a highly integrated and ultra low power SoC and the LoRa SX1262 transceiver.

heltecautomation/Heltec ESP32 Dev-Boards
Verified Library • 1.1.1 • Published on Tue May 17 10:43:05 2022
Library for Heltec ESP32 (or ESP32+LoRa) based boards. Includes: WiFi Kit 32, WiFi LoRa 32, Wireless Stick, Wireless Shell, see more on http://heltec.cn

heltecautomation/ESP32_LoRaWAN
Verified Library • 2.1.1 • Published on Fri Nov 6 03:25:47 2020
LoRaWAN library for ESP32 + LoRa boards made by heltec. See more on http://heltec.cn

heltecautomation/framework-arduino-cubecell
Verified Tool • 1.5.0 • Published on Tue Jan 31 11:38:02 2023
Arduino Wiring-based Framework for Heltec CubeCell

xreef/EByte LoRa E22 library
Library • 1.0.9 • Published on Sat Jan 28 16:40:37 2023
LoRa EBYTE E22 device library complete and tested with Arduino, Arduino SAMD (Nano 33 IoT and MKR), esp8266, esp32 and STM32. sx1262/sx1268. Ebyte E22 LoRa (Long Range) library device very cheap and very long range (from 4Km to 10Km). Arduino LoRa EBYTE E22 device library complete and tested with Arduino, Arduino SAMD (MKR WiFi 1010 and Nano 33 IoT), esp8266, esp32 and STM32. sx1262/sx1268
```

Rysunek 2. Wyniki wyszukiwania bibliotek powiązanych z hasłem „LoRa”

Rozdział 4

Implementacja oprogramowania

Całość oprogramowania wykorzystuje język programowania C++. Projektowana oraz implementowana sieć składa się z dwóch typów modułów, stąd też pojawiła się potrzeba zainicjowania dwóch osobnych projektów – jednego pod elementy sieci LoRa oraz drugiego, dedykowanego dla modułu serwera sieciowego (ang. *webserver*), z uwagi na zupełnie inną platformę sprzętową. Firmware napisany został z wykorzystaniem kilku różnych podejść:

- modułowego: każdy plik źródłowy odpowiada za zbiór funkcji wykonujących określone zadania (np. praca z biblioteką do modułów LoRa zaimplementowana jest w pliku `lora.cpp`),
- obiektowego: większość elementów kodu źródłowego jest reprezentowana w postaci osobnego obiektu. Każdy z nich posiada swoje funkcje oraz pełni określone zadania (np. obiekt „bme” ma za zadanie umożliwić współpracę z sensorami dostępnymi na płytce czujników BME280, która podłączona jest do każdego modułu SLAVE).

Ponadto, z uwagi na wykorzystanie modułów posiadających duże ilości pamięci RAM dostępnej na oprogramowanie, wykorzystane zostały dostępne w nowszych wersjach języka C++ – funkcje szablonowe (ang. *template functions*) lub pętle typu `for-range`. Są to elementy, które znacznie ułatwiły implementację kodu oraz pozwoliły na minimalizację powtarzalności pewnych elementów.

4.1 Framework oraz biblioteki

Bazą do oprogramowania na wszystkich modułach jest framework Arduino oraz jego modyfikacja pod platformę STM32 – `stm32duino`, która pozwala na wykorzystanie pełnej funkcjonalności rdzenia Arduino [2]. Pomimo tego, że biblioteki HAL (ang. *Hardware Abstraction Layer*) oraz framework STM32 są narzędziami dedykowanymi, w przypadku tego projektu nie można było ich zastosować. Oryginalna biblioteka do obsługi modułów rozszerzeń LoRa została wycofana z użytku na rzecz nowszej implementacji, pod nowszą wersję płytek Nucleo z wbudowanym hardware.

4.1.1 Wykorzystane biblioteki

Do implementacji oprogramowania na wszystkie moduły wykorzystanych zostało kilka bibliotek, które pozwalały na dodanie pełnego zakresu funkcjonalności do każdego z projektów.

W przypadku bibliotek zewnętrznych (niebędących częścią rdzenia Arduino) były to:

- STM32duino I-NUCLEO-LRWAN1: biblioteka do uruchomienia oraz pracy z modułem rozszerzeń LoRa. Pozwala ona na pracę w dwóch trybach: LoRaRadio – implementacja wykorzystująca tylko standard dolnej warstwy sprzętowej LoRa oraz LoRaWAN – dodająca możliwość podłączenia modułów do istniejącej sieci LoRa oraz wysyłanie i odbieranie z niej wiadomości,
- Adafruit BME280 Library: biblioteka dedykowana do modułów BME280, pozwalająca na zbieranie danych z sensorów, wykorzystując do tego magistralę SPI albo I2C (w zależności od posiadanego modułu rozszerzeń),
- Adafruit BusIO: uniwersalna biblioteka dodająca pewien poziom abstrakcji do komunikacji po magistralach I2C oraz SPI,
- WiFi101: biblioteka, która daje możliwość wykorzystania modułu WiFi obecnego na płytce Adafruit Feather M0 (wykorzystanej do uruchomienia serwera w sieci lokalnej).

Ponadto, wykorzystane zostały biblioteki I2C oraz SPI, dostępne w rdzeniu Arduino. Potrzebne były one do uzyskania komunikacji pomiędzy mikrokontrolerem Adafruit Feather M0 a modułem WiFi, sensorami BM280 podłączonymi do modułów SLAVE oraz do stworzenia połączenia pomiędzy modułem MASTER a płytką z serwerem sieci lokalnej.

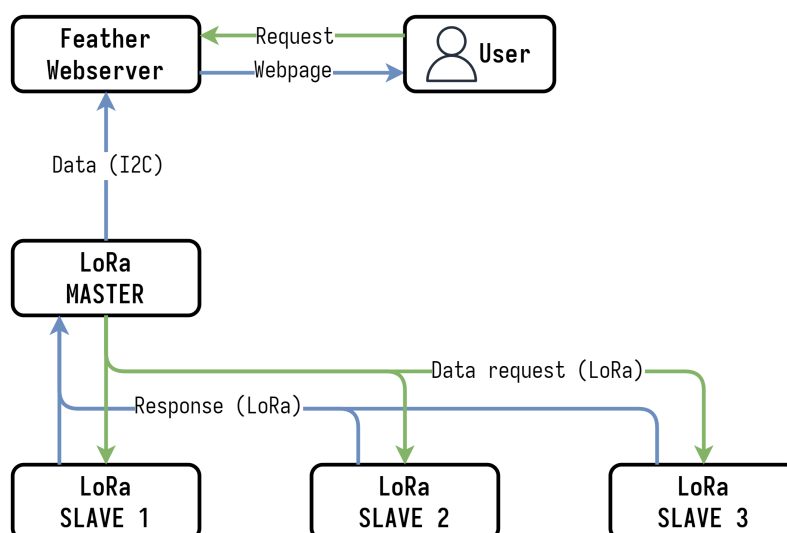
4.1.2 Ograniczenia związane z wykorzystaniem Arduino oraz STM32duino

STM32duino, pomimo tego, że ułatwił, bądź w ogóle pozwolił na pracowanie z wykorzystywanymi modułami, nie jest platformą idealną, pozbawioną ograniczeń. Jedynym z nich, które w dość znacznym stopniu utrudniło implementację oprogramowania dla modułów sieci, był brak przerw programowych oraz ograniczone możliwości zastosowania przerw sprzętowych. Stąd też pojawił się wymóg zastosowania pewnych obejść, jednocześnie tracąc na wydajności implementowanego rozwiązania. Ponadto, występowały też problemy związane z działaniem magistrali I2C, tutaj w przypadku modułów Feather oraz standardowego Arduino – niemożliwe było wykorzystanie wyświetlacza OLED pracującego na magistrali I2C oraz zarejestrowania samego mikrokontrolera jako części, z którą można komunikować się po tej magistrali.

4.2 Implementacja oprogramowania elementów sieci

Zaprojektowana sieć składała się w sumie z pięciu modułów – 4 z nich stanowiły elementy sieci LoRa, natomiast ostatni był wykorzystywany jako serwer w sieci lokalnej. W projekcie nie została wykorzystana pełna funkcjonalność LoRaWAN oraz typowa dla niej architektura (przedstawiona w sekcji 2.1, rys. 1), ponieważ implementacja takiego rozwiązania jest bardzo kosztowna i wymaga znacznie większej ilości elementów. Aby móc skorzystać ze specyfikacji wymagane jest posiadanie

bramy (ang. *gateway*) oraz serwerów odpowiedzialnych za przyłączanie urządzeń, zarządzanie siecią oraz serwera aplikacyjnego. Z uwagi na to zastosowana została dużo prostsza i mniej wymagająca metoda budowania sieci, opierająca się na wykorzystaniu modułów w formie nadajników radiowych, pracujących w standardzie LoRa. Schemat ideowy budowanej sieci przedstawiony został na rys. 3.



Rysunek 3. Schemat zbudowanej sieci, z oznaczonymi elementami komunikacji

Oprogramowanie dla modułów pracujących w sieci LoRa zostało zaimplementowane w formie uniwersalnej – jeden projekt zawiera elementy dla modułu MASTER oraz modułów SLAVE. Plik konfiguracyjny projektu zawiera flagę, która definiuje, na jaki typ modułu kod zostanie skompilowany. Co więcej, w przypadku modułów SLAVE dodana została też flaga informująca o tym, jakie ID przypisane zostaje danej płytce. Rozwiązanie to odgrywa znaczącą rolę w tym, jak wiadomości są przesyłane w sieci. Fragment pliku konfiguracyjnego, który odpowiedzialny jest za definiowanie tych elementów przedstawiony został na listingu 1.

```

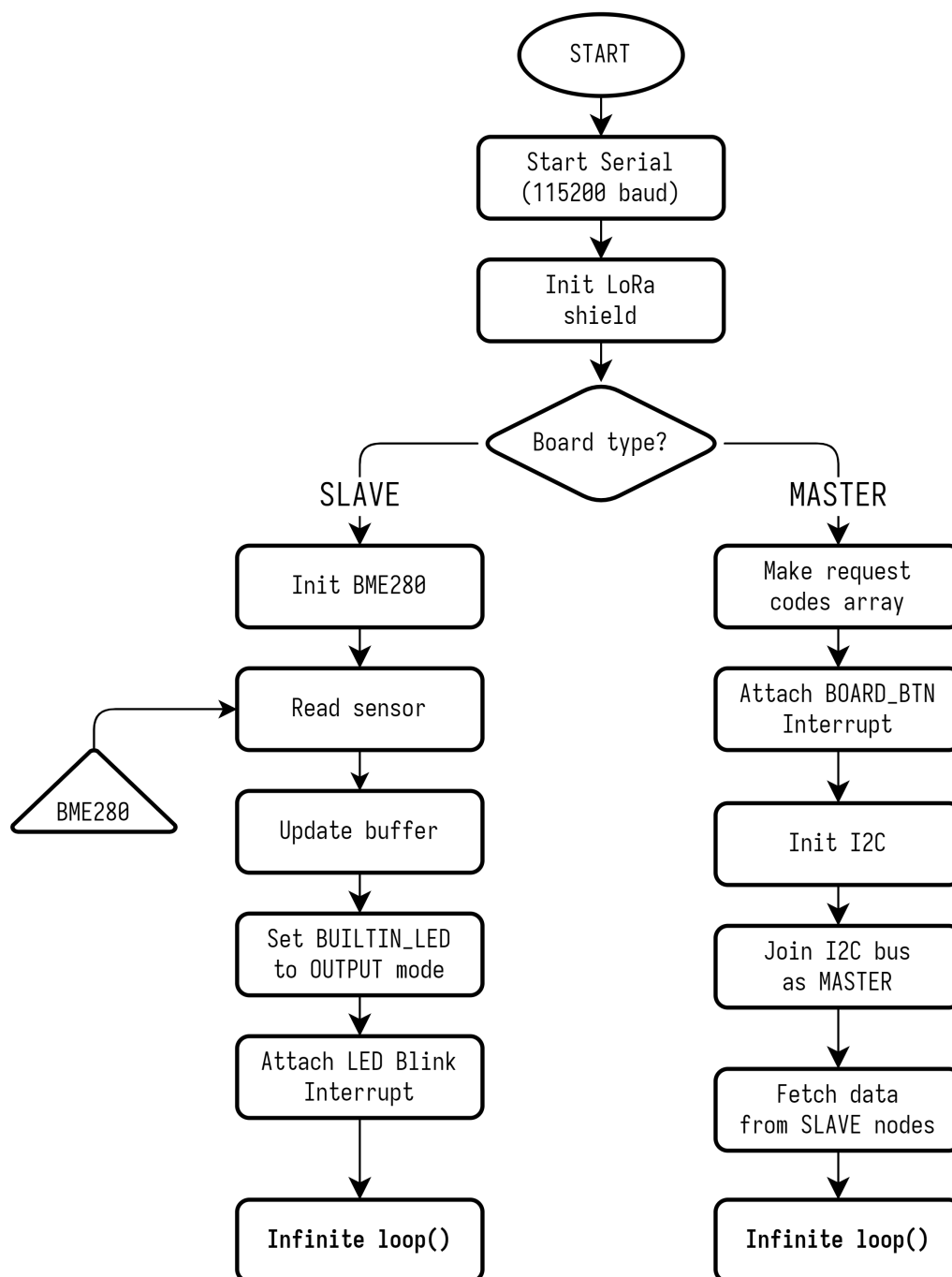
1 [env:SLAVE1]
2 extends = base_config
3 build_flags =
4     ; Set board type
5     -DBOARD_TYPE=lorar::SLAVE
6     ; Set board ID
7     -DBOARD_ID=0x01

```

Listing 1. Fragment pliku konfiguracyjnego (tutaj dla SLAVE1) odpowiedzialny za definicję typu oraz ID modułu

Wykorzystanie frameworku Arduino wymagało zastosowania pewnych schematów podczas implementacji. Dlatego też całość kodu podzielona jest na dwie sekcje `setup()` oraz `loop()`, wykonywane odpowiednio raz, podczas startu modułu oraz w nieskończonej pętli, dopóki płytka ma zasilanie. Na

rys. 4 przedstawiony został schemat blokowy zaimplementowanego oprogramowania – części zawartej w sekcji `setup()`.



Rysunek 4. Schemat blokowy części `setup()` oprogramowania modułów sieci LoRa, z podziałem na typ płytki

Oba typy oprogramowania zaczynają od ustawienia portu szeregowego na 115200 baud (szybkość transmisji), następnie inicjowane jest rozszerzenie LoRa. Logowana jest informacja o typie płytki,

a następnie kod oczekuje na informacje o starcie modułu rozszerzenia. W przypadku błędu oraz poprawnego startu na port szeregowy wystawiana jest odpowiednia informacja.

Następnie, w zależności od typu płytki, wykonywane jest kilka operacji. W przypadku modułów SLAVE są to:

1. przygotowanie sensora BME280 oraz pobranie z niego danych,
2. aktualizacja zawartości bufora (wykorzystywanego do przechowywania odczytanych wartości),
3. przygotowanie diody LED, która informuje o trwającej komunikacji w sieci,
4. przygotowanie przerwania, wykorzystywanego do obsługi nowych zapytań.

Natomiast dla modułów MASTER wykonywany jest inny zestaw operacji, z uwagi na to, że taki moduł pełni zupełnie inną funkcję w sieci:

1. przygotowanie tablicy z „kodami” zapytań (jedno bajtowe wartości do określenia czego żąda MASTER),
2. inicjacja magistrali I2C i podłączenie modułu jako MASTER,
3. wykonanie podprogramu wysyłającego zapytania oraz odbierającego odpowiedzi od SLAVE-ów, tak aby tuż po starcie można było odczytać dane z sieci.

Ostatnim krokiem w obu przypadkach jest przejście do nieskończonej pętli i wykonywanie instrukcji w niej zawartych, wykorzystując do tego określony okres zegara.

4.2.1 Oprogramowanie modułu MASTER

4.2.2 Oprogramowanie modułów SLAVE

4.3 Implementacja oprogramowania modułu serwera sieciowego

Rozdział 5

Podstawowe testy implementacji

Rozdział 6

Badania działającej sieci

Rozdział 7

Podsumowanie

Summary

Bibliografia

- [1] PlatformIO, *Documentation*, Development Platforms.
- [2] STM32duino, *Documentation*, Arduino core for STM32 MCUs.

Spis rysunków

1	Schemat architektury sieci LoRaWAN	11
2	Wyniki wyszukiwania bibliotek powiązanych z hasłem „LoRa”	15
3	Schemat zbudowanej sieci, z oznaczonymi elementami komunikacji	19
4	Schemat blokowy części <code>setup()</code> oprogramowania modułów sieci LoRa, z podziałem na typ płytki	20