

look up!

Raport z III etapu projektu

Zespół 5:

Michał Ciesielski
Karol Duszczyk
Kamil Kmiec

Projekt realizowany w ramach przedmiotu PBL-3
w semestrze 21Z

Opiekun: mgr inż. Marcin Kołakowski
2 lutego 2022

Spis treści

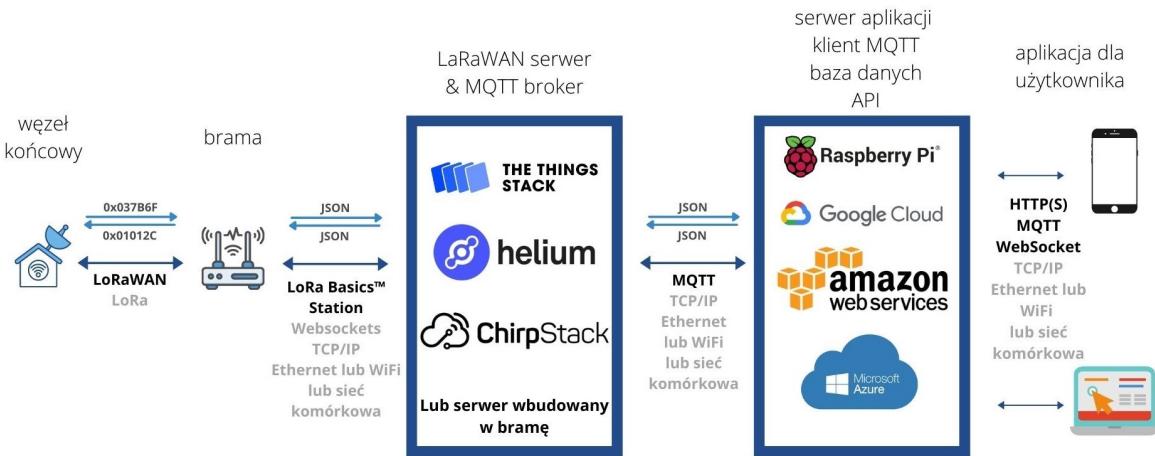
1 Przedstawienie tematyki projektu	3
2 Architektura sieci	3
3 Węzeł końcowy	3
3.1 Zbieranie i przygotowanie danych	3
3.1.1 Wykorzystane czujniki	3
3.1.2 Przygotowanie danych	3
3.2 Komunikacja w węźle końcowym	4
3.2.1 Omówienie	4
3.2.2 Realizacja	4
3.3 Prototyp	4
3.3.1 Budowa urządzenia	4
3.3.2 Pobór prądu	5
4 Komunikacja LoRaWAN	5
4.1 Format wiadomości	5
4.1.1 Wybór formatu danych	5
4.1.2 Wiadomości Uplink	6
4.1.3 Wiadomości Downlink	7
4.2 Wymagania formalne i prawne	7
4.3 Obliczenia teoretyczne	7
4.3.1 Parametry połączenia	7
4.3.2 Ilość wysyłanych danych	7
4.3.3 Czas wysyłania	8
5 Węzeł pośredniczący i serwer LoRaWAN	8
5.1 Omówienie	8
5.2 LoRa Basics™ Station	9
5.3 The Things Stack Community Edition	9
5.4 Własna brama w 432	9
5.5 Helium	9
6 Serwer, baza danych i prezentacja danych użytkownikowi	9
6.1 AWS	9
6.2 Lokalnie	11
6.2.1 Omówienie	11
6.2.2 Paho	11
6.2.3 InfluxDB	12
6.2.4 Grafana	12
7 Testy funkcjonalności	12
7.1 Testy komunikacji między węzłem końcowym a serwerem LoRaWAN	12
7.1.1 Plan testów	12
7.1.2 The Things Stack Community Edition	13
7.1.3 Własna brama w 432	13
7.1.4 Helium	13
7.1.5 Wnioski	15
7.2 Testy czujników	15
7.2.1 Test nr 1	15
7.2.2 Test nr 2	16
7.2.3 Testy z raportu II	17
7.2.4 Wnioski	18

8 Badania w labolatorium	18
8.1 Badania w labolatorium PTB	18
8.2 Obserwacja wiadomości uplink MQTT w Wireshark	21
9 Sposób implementacji sieci	21
10 Podsumowanie	22

1 Przedstawienie tematyki projektu

Celem projektu jest stworzenie rozproszonego systemu IoT umożliwiającego ocenę stopnia zachmurzenia nieba na danym obszarze. Obecnie taka ocena pokrywy chmur jest dokonywana poprzez analizę zdjęć satelitarnych lub wykorzystanie zaawansowanych czujników laserowych. Zdjęcia satelitarne wydają się wystarczającym źródłem informacji dla działań związanych z prognozowaniem pogody [8], jednak są niewystarczająco szczegółowe, aby spełnić wymagania osób zajmujących się zawodowo lub hobbystycznie obserwacjami astronomicznymi. Duża dokładność pomiarów dokonanych na obszarze prowadzenia takich obserwacji jest kwestią istotną, ponieważ nawet niewielkie zachmurzenie może skutecznie uniemożliwić prawidłowe ich wykonanie. Problem ten mogłoby rozwiązać opracowanie sieci czujników zbierających dane o aktualnie panujących warunkach pogodowych na danym obszarze połączonej z aplikacją prezentującą użytkownikowi interesujące go informacje w przystępnej formie.

2 Architektura sieci



Rysunek 1: Architektura sieci

Architektura sieci składa się z 5 części (jak pokazano na rys.1). Każdą z nich można modyfikować bez znaczącego wpływu na pozostałe (np. wymiana bramy publicznej na prywatną, zamiana serwera lokalnego na chmurowego, itp.). Pozwala to na łatwą skalalność rozwiązania i dopasowanie go do potrzeb użytkownika.

3 Wózek końcowy

3.1 Zbieranie i przygotowanie danych

3.1.1 Wykorzystane czujniki

W I etapie rozważaliśmy wykorzystanie różnych czujników - termometrów, czujnika zanieczyszczenia powietrza oraz kamery nocnej. Testy wykazały, że do poprawnego sprawdzania stanu zachmurzenia w zupełności wystarczy para czujników - IR MLX90614DCI (SEN0263) do mierzenia temperatury nieba oraz DS18B20 do mierzenia temperatury otoczenia.

3.1.2 Przygotowanie danych

W poprzednim etapie wysyialiśmy zmierzone wartości temperatury z rozdzielcością $0,1^{\circ}\text{C}$. Przerowadzone testy wykazały, że różnica temperatur z obu czujników dla bezchmurnego nieba jest o kilkakrotnie większa niż dla zachmurzonego. Z tego powodu uznaliśmy, że wystarczy przesyłać temperaturę z rozdzielcością 1°C . Czujnik SEN0263 pracuje w zakresie od $-70,01^{\circ}\text{C}$ do 270°C , a DS18B20 od -55°C do 125°C . Tak jak poprzednio, postanowiliśmy do odczytanej wartości dodać

$100^{\circ}C$, aby pozbyć się wartości ujemnych. Mimo dużego zakresu pracy czujników, w rzeczywistych warunkach temperatura otoczenia oraz temperatura nieba nie przekrocza $155^{\circ}C$, więc każdą z tych wartości możemy zapisać na 1 bajcie. Przed wysłaniem zebrane dane są przetwarzane do formatu, który zostanie omówiony w dalszej części raportu.

3.2 Komunikacja w węźle końcowym

3.2.1 Omówienie

Do komunikacji z bramą wykorzystujemy moduł LoRa-E5 mini, który jest podłączony do RPi zero W za pomocą przewodu USB. Łączy się on z dowolną bramą w jego zasięgu. Brama, która znajduje się w używanej sieci (np. The Things Stack, Helium lub własna sieć) rozpoznaje węzeł po APP_KEY i DEV_EUI i przekazuje jego wiadomość dalej. Urządzenie jest klasy A (najbardziej energoszczędne), więc wiadomości downlink są odbierane pojedynczo i tylko chwilę po wysłaniu wiadomości uplink.

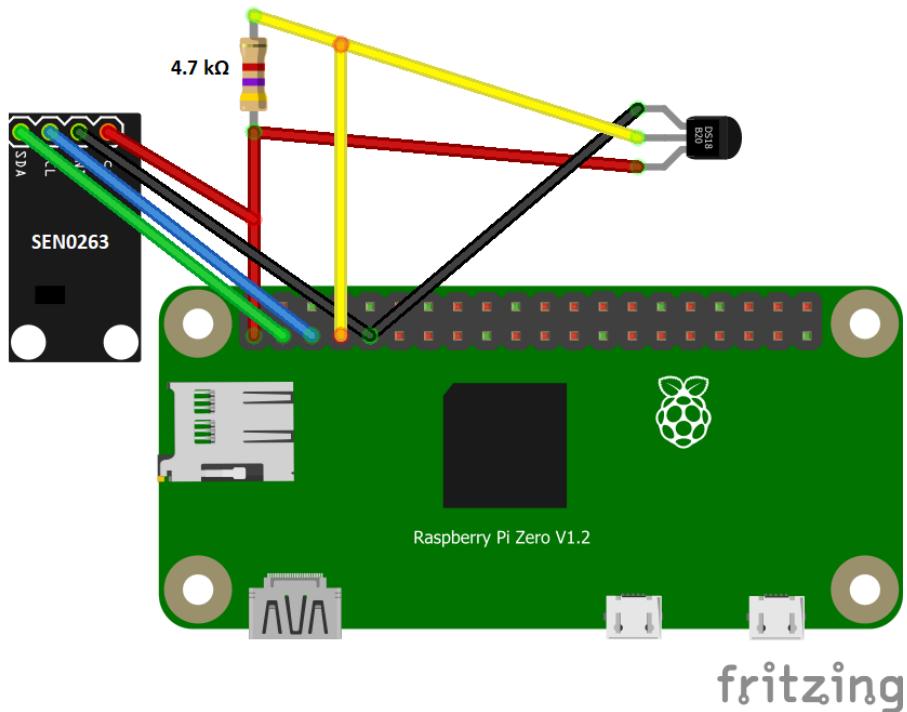
3.2.2 Realizacja

Kod odpowiedzialny za tę komunikację znajduje się w pliku *lora.py*. Najpierw inicjowane jest połączenie z urządzeniem USB. Następnie moduł dołącza się do bramy (próba połączenia jest powtarzana do skutku). Po nawiązaniu połączenia można wysyłać i odbierać wiadomości.

Odebrane wiadomości downlink są analizowane i na ich podstawie wykonywana jest konkretna akcja. W trakcie działania możliwa jest zdalna zmiana sieci/aplikacji LoRaWAN poprzez wysłanie nowego klucza APP_KEY i polecenia zrestartowania połączenia z siecią. Wszystkie możliwości obsługi urządzenia przez sieć LoRaWAN za pomocą wiadomości downlink zostaną omówione w dalszej części raportu.

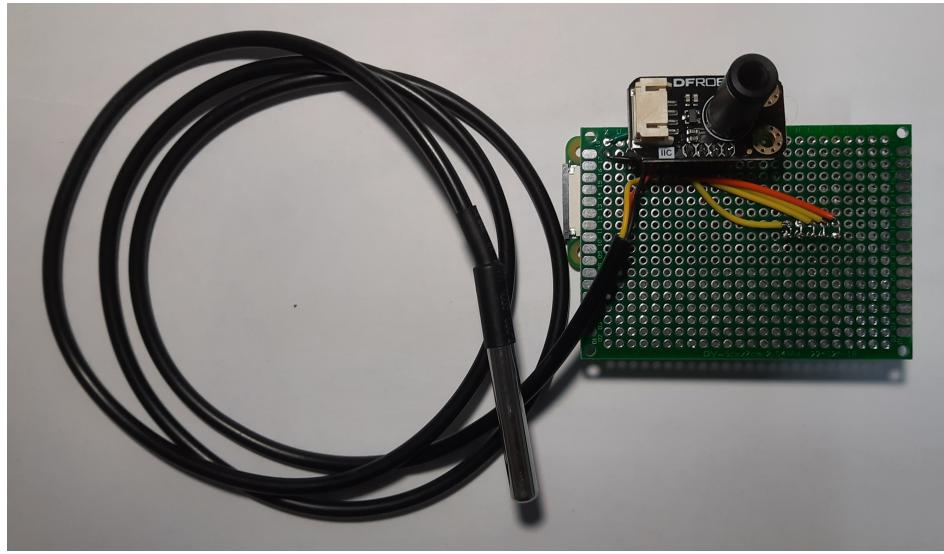
3.3 Prototyp

3.3.1 Budowa urządzenia



Rysunek 2: Schemat podłączenia czujników do RPi zero

Stworzyliśmy prototyp urządzenia końcowego, którego schemat został przedstawiony na rys. 2. Moduł LoRa-E5 mini podłączony jest do Raspberry Pi za pomocą przewodu USB. Aby połączenia między elementami były stabilne, przygotowaliśmy nakładkę na RPi wykorzystując płytę uniwersalną.



Rysunek 3: Nakładka z czujnikami

3.3.2 Pobór prądu

Raspberry Pi Zero

Poprzez wyłączenie HDMI, diod LED i Wi-Fi można zminimalizować pobór prądu przez RPi nawet do 80 mA (to nadal dużo jak na urządzenie IoT) [[9]], jednak przetwarzanie danych i obsługa programu znaczco zwiększa pobór prądu, dlatego dobrym rozwiązaniem byłoby zastąpienie RPi innym modułem, np. firmy STM.

STM32WLE5JC z modułem LoRa e5 mini

Zużywa ok. $75 \mu\text{A}$ w trybie czuwania, w czasie nadawania transmisji ok. 125 mA, ale transmisja jednej wiadomości zajmuje jedynie ok 50 ms, co stanowi niewielką część czasu pracy urządzenia.

DS18B20 i SEN0263 zużywają po ok. 1 mA.

4 Komunikacja LoRaWAN

4.1 Format wiadomości

4.1.1 Wybór formatu danych

Mimo wykorzystania tylko dwóch czujników chcemy zapewnić elastyczność naszego rozwiązania poprzez możliwość łatwej rozbudowy o dodatkowe czujniki. Z tego powodu nie możemy wysyłać samych mierzonych wartości (tak jak to robiliśmy w etapie II), ponieważ czujniki mogą się między sobą różnić i wysypane dane nie byłyby jednoznaczne (trzeba by było stosować inny format wiadomości dla każdej konfiguracji urządzenia pomiarowego). Jednocześnie chcemy zapewnić jak najmniejszy rozmiar wysyłanej wiadomości, ponieważ LoRaWAN oferuje małe przepływności i krótkie dozwolone czasy wysyłania, więc narzut dodatkowych informacji (poza zmierzonymi wartościami) w wiadomości musi być jak najmniejszy. Z tego powodu od razu odrzuciliśmy formaty takie jak JSON i zwykły tekst.

W komunikacji LoRa popularny jest format Cayenne Low Power Payload [7]. Dane z każdego czujnika są przesyłane w formacie:

1. Data Channel (1 bajt) - unikalny numer czujnika w danym urządzeniu.

2. Data Type (1 bajt) - typ wysyłanych danych np. temperatura, wilgotność itd. Typy są zdefiniowane w standardzie, dzięki czemu wystarczy wysłać 1 bajt oznaczający typ zamiast całej nazwy.
3. Dane (N bajtów) - zmierzona wartość, ilość bajtów zależy od typu danych i jest zdefiniowana w standardzie. Dla czujników temperatury są to 2 bajty z zapewnioną rozdzielczością $0,1^{\circ}C$.

Przesyłana ramka jest tworzona z połączenia ramek dla wszystkich czujników, z których dane są wysyłane.

Standard ten zapewnia dużą uniwersalność i bardzo łatwą możliwość rozbudowy urządzenia o wiele różnych czujników. Jednak nasze urządzenia są proste i nie potrzebujemy obsługi wielu różnych czujników (urządzenia będą raczej rozbudowywane o pojedyncze czujniki). Postanowiliśmy stworzyć własny format, który będzie lepiej dostosowany do naszych potrzeb i będzie zapewniał mniejszy rozmiar wiadomości, jednocześnie umożliwiając łatwą rozbudowę systemu.

Nasz format pozwala na przesyłanie odczytów z 8 różnych czujników, mając tylko 1 bajt narzutu. Pierwszy bajt w ramce (czytany od tyłu) informuje o tym, z których czujników będą wysyłane dane - każdy bit odpowiada jednemu czujnikowi, wartość 1 oznacza, że dane z tego czujnika będą wysyłane, a 0, że nie będą lub czujnik nie istnieje. Następnie wysyłany jest odpowiedni ciąg bajtów z danymi - ilość bajtów przypadająca na każdy czujnik jest z góry zdefiniowana w wykorzystywanym standardzie.



Rysunek 4: Przykładowy format wiadomości

4.1.2 Wiadomości Uplink

Aktualnie używany format wiadomości

Biorąc pod uwagę aktualnie używane czujniki, wysyłane wiadomości uplink mają następujący format:

Nr bitu (od najmniej znaczącego)	Typ danych	Ilość bajtów
1	temperatura otoczenia	1 (rozdz. $1^{\circ}C$, dodane $100^{\circ}C$)
2	temperatura nieba	1 (rozdz. $1^{\circ}C$, dodane $100^{\circ}C$)
3-6	nieużywane	-
7	stan baterii (niezaimplementowane)	1
8	komunikaty techniczne	2

Przykład:

Wiadomość '037b6f' oznacza:

- '03' = 00000011 - w tej wiadomości znajdują się wartości temperatury otoczenia i temperatury nieba
- '7b' = 123 - wartość temperatury otoczenia wynosi $123 - 100 = 23^{\circ}C$
- '6f' = 111 - wartość temperatury nieba wynosi $111 - 100 = 11^{\circ}C$

4.1.3 Wiadomości Downlink

Aktualnie używany format wiadomości

Do wiadomości typu downlink postanowiliśmy użyć formatu analogicznego do formatu wiadomości typu uplink:

Nr bitu (od najmniej znaczącego)	Typ danych	Ilość bajtów
1	czas między pomiarami [s]	2 (min. 20 s, max 65535 s)
2	obsługiwane czujniki	1 (układ taki, jak przy uplink)
3	nowy klucz APPKEY	16
4-7	nieużywane	-
8	komunikaty binarne	1

Komunikaty binarne zajmują 1 bajt. Każdy bit oznacza inny komunikat:

Nr bitu (od najmniej znaczącego)	Typ danych
1	restart urządzenia
2	żądanie wysłania stanu baterii (niezaimplementowane)
3	restart połączenia z siecią
4-8	nieużywane

4.2 Wymagania formalne i prawne

LoRa działa na częstotliwościach nielicencjonowanych, więc używając LoRaWAN musimy przestrzgać pewnych wymagań i ograniczeń.

- Korzystamy z europejskiej częstotliwości 868 MHz. Jeśli nasz system miałby działać w innych miejscach świata, musielibyśmy dostosować używaną częstotliwość do lokalnych wymagań.
- Maksymalna moc nadawania jest ograniczona do 25mW (14 dBm) dla wiadomości uplink (od urządzenia końcowego do serwera) oraz do 0,5W (27 dBm) dla wiadomości downlink (od serwera do urządzenia końcowego)
- Dodatkowym ograniczeniem jest maksymalny czas nadawania (duty cycle). Najczęściej wynosi on 1%, co oznacza, że możemy nadawać wiadomości uplink przez 36 sekund w ciągu godziny (864 sekundy w ciągu dnia). [6]

4.3 Obliczenia teoretyczne

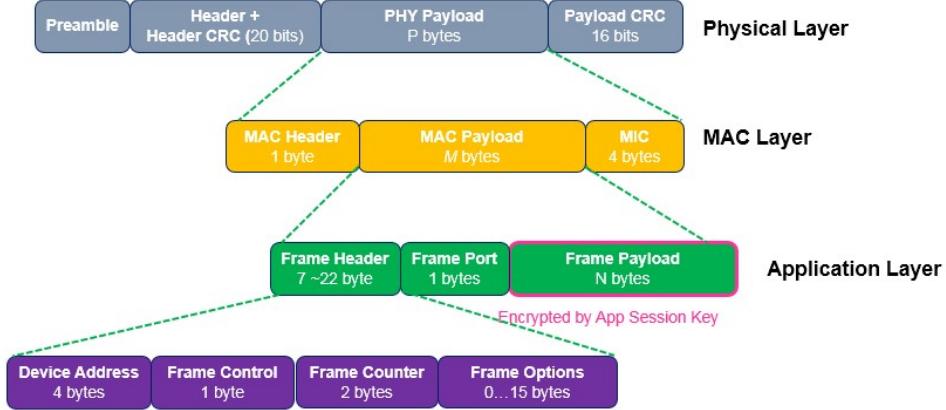
4.3.1 Parametry połączenia

Aktualnie używamy następujących ustawień:

- częstotliwość EU868
- pasmo 125 kHz
- szybkość kodowania (Code rate - CR) = 1
- współczynnik rozpraszania (Spreading factor - SF) = 7

4.3.2 Ilość wysyłanych danych

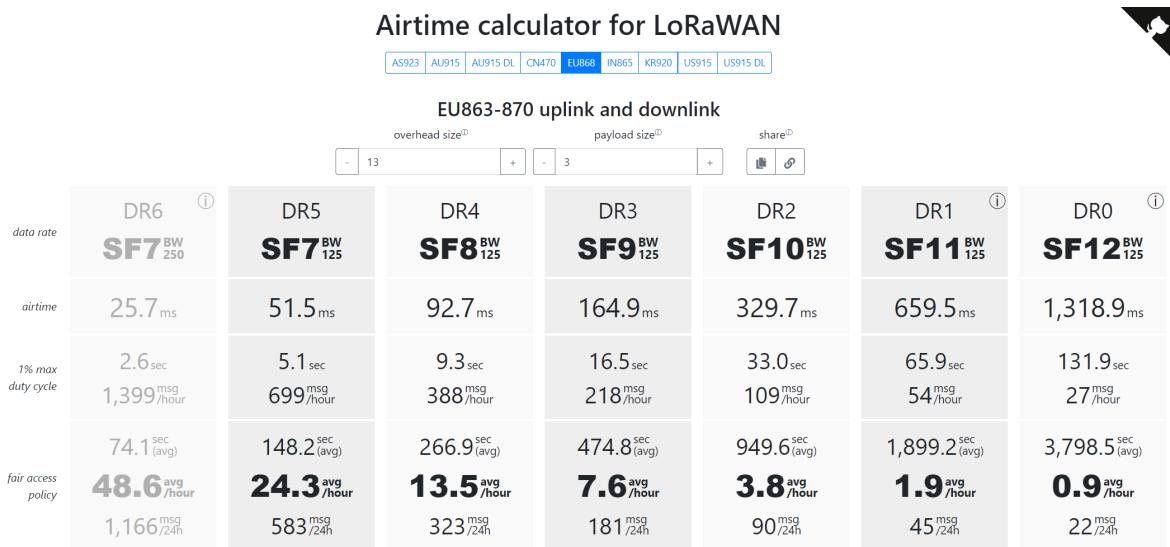
Obecnie zaimplementowane są tylko czujnik temperatury otoczenia i czujnik temperatury nieba. Zgodnie ze zdefiniowanym wyżej standardem, dane wysyłane w jednej wiadomości uplink zajmują 3 bajty. Poniżej została przedstawiona budowa ramki LoRa.



Rysunek 5: Budowa ramki LoRa [2]

W wiadomościach uplink (najczęściej przesyłane w naszej sieci) nagłówki warstwy aplikacji zajmują 7 bajtów, port 1 bajt, frame payload 3 bajty - razem ramka w warstwie aplikacji zajmuje 11 bajtów. W warstwie MAC dołożone zostają nagłówek MAC (1 bajt) oraz Message Integrity Code (4 bajty) - razem ramka w warstwie MAC zajmuje 16 bajtów.

4.3.3 Czas wysyłania



Rysunek 6: Wartości czasu nadawania obliczone w kalkulatorze [5] w zależności od ustawień

Przy naszych ustawieniach czas przesyłu jednej wiadomości uplink wynosi 51,5 ms. Przy zachowaniu wymaganego 1% duty cycle możemy wysyłać 699 wiadomości na godzinę. Publiczne sieci mogą nakładać na nas dodatkowe ograniczenia, np. korzystając z The Things Stack Community Edition możemy wysyłać tylko 24,3 wiadomości na godzinę.

5 Węzeł pośredniczący i serwer LoRaWAN

5.1 Omówienie

W tym projekcie węzłem pośredniczącym jest brama LoRaWAN. Z jednej strony komunikuje się ona z węzłem końcowym przez LoRa (LoRaWAN). Z drugiej strony komunikuje się z serwerem LoRaWAN (np. The Things Stack, Helium, ChirpStack) za pomocą łączna internetowego (LoRa Basics™ Station)

lub sama ma wbudowany serwer LoRaWAN. Serwer LoRaWAN pozwala zarządzać wszystkimi ustawieniami sieci (pozwala zarządzać wieloma bramami i projektami) i tworzyć klucze dostępu dla węzłów końcowych. Serwer przetwarza odebrane dane do formatu JSON i dodaje nagłówki (identyfikujące urządzenie, parametry transmisji, informacje o urządzeniu, znaczniki czasowe), a następnie udostępnia te dane za pomocą MQTT, HTTP lub innych protokołów. Najczęściej nie zapisuje danych bezpośrednio do bazy danych, więc musi się tym zająć główny serwer (serwer aplikacji), który komunikuje się z serwerem LoRaWAN.

5.2 LoRa Basics™ Station

LoRa Basics™ Station [1] to protokół służący do komunikacji między bramą a serwerem LoRaWAN. Zaimplementowany jest on w bramie i przekazuje pakiety (uplink i downlink) do serwera LoRaWAN (LoRaWAN Network Server - LNS) przez zabezpieczoną sieć IP. Pozwala także na synchronizację czasową bramki i serwera oraz aktualizowanie oprogramowania bramy. Protokół ten do komunikacji wykorzystuje WebSockets, a dane są przesyłane w formacie JSON.

W przeszłości był wykorzystywany protokół Semtech UDP Packet Forwarder (wykorzystujący UDP), ale obecnie jest on wycofywany z użycia.

5.3 The Things Stack Community Edition

The Things Stack to społecznościowa sieć bramek LoRaWAN rozmieszczonej po całym świecie (w tym w Warszawie). Wersja Community Edition pozwala na darmowe korzystanie z nich z pewnymi ograniczeniami. TTS zapewnia też za darmo chmurowy serwer LoRaWAN, z którego można sterować swoimi urządzeniami włączonymi do tej sieci. Rozwiązanie to zostało szerzej omówione w raportach z poprzednich etapów.

5.4 Własna brama w 432

W laboratorium 432 znajduje się bramka LoRaWAN RAK7268 WisGate Edge Lite 2 [3]. Została ona skonfigurowana i dodawana do sieci lokalnej. Posiada ona zintegrowany serwer LoRaWAN, więc nie trzeba używać zewnętrznego. Dostęp do bramki nie jest publiczny.

5.5 Helium

Helium [4] to społecznościowa sieć bramek LoRaWAN rozmieszczonej po całym świecie (w tym w Warszawie). Bramek w tej sieci jest więcej niż w TTS, ale korzystanie z nich wiąże się z opłatami. Opłaty zależą od ilości przesyłanych danych. Helium (tak samo jak TTS) zapewnia chmurowy serwer LoRaWAN, z którego można sterować swoimi urządzeniami włączonymi do tej sieci.

6 Serwer, baza danych i prezentacja danych użytkownikowi

6.1 AWS

Jednym z naszych pierwszych pomysłów na zapis zbieranych danych było wykorzystanie jednej z usług bazodanowych działających i zbudowanych w chmurze. Rozwiązanie takie ma kilka zalet, przede wszystkim rozpatrując pracę w zespole:

- dostęp do chmurowej bazy danych za pomocą interfejsu API od dostawcy lub interfejsu WWW;
- dynamiczna pojemność chmurowej bazy danych;
- kopie zapasowe danych umieszczone są na zewnętrznych serwerach, co zapewnia bezpieczeństwo danych w przypadku niespodziewanej awarii.

Pod uwagę braliśmy trzy najbardziej popularne chmurowe usługi bazodanowe na rynku, oferujące darmowe rozwiązania w zakresie przechowywania i analizy danych:

- Amazon Web Services (AWS);
- Google Cloud;
- Microsoft Azure.

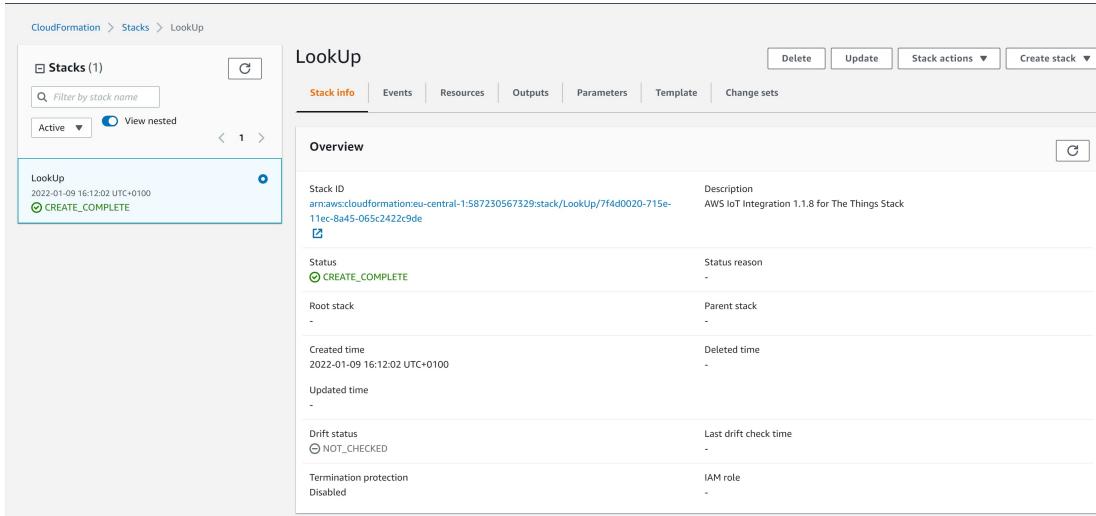
Każda z nich oferuje inny zakres usług oraz odpowiednie limity dotyczące m.in. łącznego rozmiaru danych w swojej darmowej opcji. Przeprowadziliśmy ich porównanie i zestawiliśmy je w poniższej tabeli.

Platforma	Protokoły komunikacji	Zalety	Top 3 obszary zastosowania	Plan płatności	Opcja darmowa	Darmowa opcja dla usług IoT
Amazon Web Services	<ul style="list-style-type: none"> • HTTP • MQTT (QoS 0,1) • WebSockets 	<ul style="list-style-type: none"> • łatwa integracja • autoryzacja • przyjazne programistom oprogramowanie 	<ul style="list-style-type: none"> • transport • fabryki • Smart City 	<ul style="list-style-type: none"> • 0,08\$ za milion urządzeń • 0,7-1\$ za milion wiadomości • 0,15\$ za milion żądań 	Na 12 miesięcy: <ul style="list-style-type: none"> • 2250000 minut łączności • 500 tys. wiadomości • 225 tys. połączonych urządzeń 	<ul style="list-style-type: none"> • dostępna na 12 miesięcy
Google Cloud	<ul style="list-style-type: none"> • HTTP • MQTT (QoS 0,1) 	<ul style="list-style-type: none"> • łatwa integracja • zarządzanie urządzeniami 	<ul style="list-style-type: none"> • energia • parkingi • transport i logistyka 	<ul style="list-style-type: none"> • 0,00045-0,0045\$ za MB danych 	<ul style="list-style-type: none"> • do pierwszych 250 MB 	<ul style="list-style-type: none"> • dostępna na 12 miesięcy z kredytem na 300\$ • duzo darmowych aplikacji
Microsoft Azure	<ul style="list-style-type: none"> • HTTP • MQTT (QoS 0,1) • AMQP • WebSockets 	<ul style="list-style-type: none"> • łatwa integracja • autoryzacja • zarządzanie urządzeniami • monitorowanie urządzeń • IoT Edge 	<ul style="list-style-type: none"> • służba zdrowia • sprzedaż • fabryki 	<ul style="list-style-type: none"> • podstawowa opcja: 10\$/miesiąc za urządzenie • standardowa opcja: 25\$/miesiąc za urządzenie 	<ul style="list-style-type: none"> • do 8 tys. wiadomości na dzień • do 500 urządzeń 	<ul style="list-style-type: none"> • dostępna na 12 miesięcy z kredyttem na 200\$ • 25+ zawsze darmowych usług

Rysunek 7: Zestawienie najważniejszych informacji dotyczących darmowych opcji porównywanych chmurowych usług bazodanowych.

Po zebraniu wiedzy oraz dokonaniu analizy postanowiliśmy wybrać AWS jako najlepsze rozwiązanie dla naszego projektu na tym etapie.

Przeprowadziliśmy integrację AWS z naszą aplikacją w The Things Stack, korzystając z wygenerowanego w konsoli klucza API. W usłudze AWS CloudFormation stworzyliśmy nowy stos dedykowany naszej aplikacji na TTS.



Rysunek 8: Stos w sułudze AWS CloudFormation dedykowany naszej aplikacji TTS.

Pomiary, które udało nam się zapisać w naszej bazie danych możemy zaobserować na poniższym zdjęciu. Tablica danych została zaprojektowana w ten sposób, aby zapisy można było sortować po urządzeniach oraz czasie zapisu wiadomości.

Items returned (50)			
	time_stam...	device_id	lorawan_uplink
□	164266670...	2CF7F1203...	{}
□	164277859...	2CF7F1203...	{}
□	164277741...	2CF7F1203...	{}
□	164266667...	2CF7F1203...	{}
□	164266723...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "22"}}
□	164266701...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "22"}}
□	164266721...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "22"}}
□	164266676...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "22"}}
□	164266811...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "23"}}
□	164266953...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "23"}}
□	164266872...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "23"}}
□	164267534...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "23"}}
□	164267503...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "23"}}
□	164266715...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "23"}}
□	164266693...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "23"}}
□	164266709...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "23"}}
□	164266886...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "23"}}
□	164266875...	2CF7F1203...	{"ambient_temp": {"N": "23"}, "sky_temp": {"N": "23"}}

Rysunek 9: Stos w usłudze AWS CloudFormation dedykowany naszej aplikacji TTS.

W usłudze IoT Core możemy również zaobserwować urządzenia podłączone do naszej aplikacji. Urządzenia są synchronizowane automatycznie po podłączeniu do naszej aplikacji na TTS.

Things (7) Info							
An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.							
		G	Advanced search	Run aggregations	Edit	Delete	Create things
□	Name		Thing type				
□	7083D57ED0048FD6		lorawan				
□	7083D57ED0048FD5		lorawan				
□	7083D57ED0048FD4		lorawan				
□	7083D57ED0048FD3		lorawan				
□	7083D57ED0048FC6		lorawan				
□	2CF7F12032304959		lorawan				
□	2CF7F12032304B54		lorawan				

Rysunek 10: Stos w usłudze AWS CloudFormation dedykowany naszej aplikacji TTS.

Z dalszego rozwoju tego rozwiązania postanowiliśmy zrezygnować w momencie rozpoczęcia zajęć w trybie zdalnym. W dłuższej perspektywie należałoby na pewno popracować nad lepszą formą zapisu danych do bazy za pomocą usługi funkcji Lambda.

6.2 Lokalnie

6.2.1 Omówienie

Postanowiliśmy też zrobić lokalny serwer na RPi. Do tego celu wykorzystaliśmy klienta MQTT Paho, który nasłuchuje dane od The Things Stack i zapisuje je do bazy danych InfluxDB. Do prezentacji danych użytkownikowi wykorzystaliśmy program Grafana.

6.2.2 Paho

Paho to klient MQTT, który został omówiony w poprzednim raporcie. Napisaliśmy program w Pythonie, który wykorzystuje bibliotekę Paho do subskrybowania tematów MQTT z TTS. Po otrzymaniu wiadomości typu uplink, przetwarza ją, odczytuje wartości temperatury i na ich podstawie szacuje poziom zachmurzenia (wg. skali opisanej we wnioskach z testów czujników w dalszej części tego raportu). Następnie odczytuje lokalizację danego czujnika z bazy danych InfluxDB i zapisuje wartość razem z lokalizacją do InfluxDB.

6.2.3 InfluxDB

InfluxDB to baza danych szeregow czasowych. Jest ona uruchomiona lokalnie na RPi. Zawiera 2 tabele: uplink3 (zawierającą dane z wiadomości uplink) i locations (zawierającą lokalizację czujników). Do obsługi zapytań w bazie danych wykorzystywana jest dedykowana biblioteka do języka Python.

name: locations				name: uplink3			
time	dev_id	latitude	longitude	time	ambient	delta	dev_id
1643544994506773508	fake-1	52.20323	20.962188	164355370713915437	19	32	fake-5
1643544994525608370	eui-70b3d57ed00486b4	52.218688	21.018944	1643553689386263208	10	2	fake-4
164354499453457187	eui-2cf7f12032304959	52.230688	21.01197	1643553673653705339	19	66	fake-3
164355293629231579	fake-2	52.22023	20.942288	1643553661442538714	19	34	fake-2
164355293655036643	fake-3	52.18363	20.902168	1643553517244352845	17	19	fake-1
164355293663709596	fake-4	52.18353	21.152188	164355356186438521	-31	51	fake-1
164355293672323901	fake-5	52.27393	21.097188	164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705339	19	66	fake-3
				1643553661442538714	19	34	fake-2
				1643553517244352845	17	19	fake-1
				164355356186438521	-31	51	fake-1
				1643553689386263208	10	2	fake-4
				164355370713915437	19	32	fake-5
				1643553673653705			

Środowisko testowe składało się z modułu LoRa-e5 mini podłączonego przewodem USB do laptopa. Moduł był sterowany przez aplikację Putty.

7.1.2 The Things Stack Community Edition

Udało się połączyć z bramką oraz zrealizować połączenia uplink i downlink w sieci The Things Stack.

Rysunek 13: Widok konsoli TTS po udanym przesłaniu wiadomości uplink

Testy były przeprowadzane wewnętrz budynku EiT. Za każdym razem moduł łączył się z bramką eui-0000024b0803180f zlokalizowaną na dachu budynku wydziału.

7.1.3 Własna brama w 432

Udało się (zanim uczelnia przeszła w tryb zdalny) połączyć oraz zrealizować połączenia uplink i downlink z bramką w laboratorium 432. Bramka ta posiada zintegrowany serwer LoRaWAN i to z niego korzystaliśmy.

7.1.4 Helium

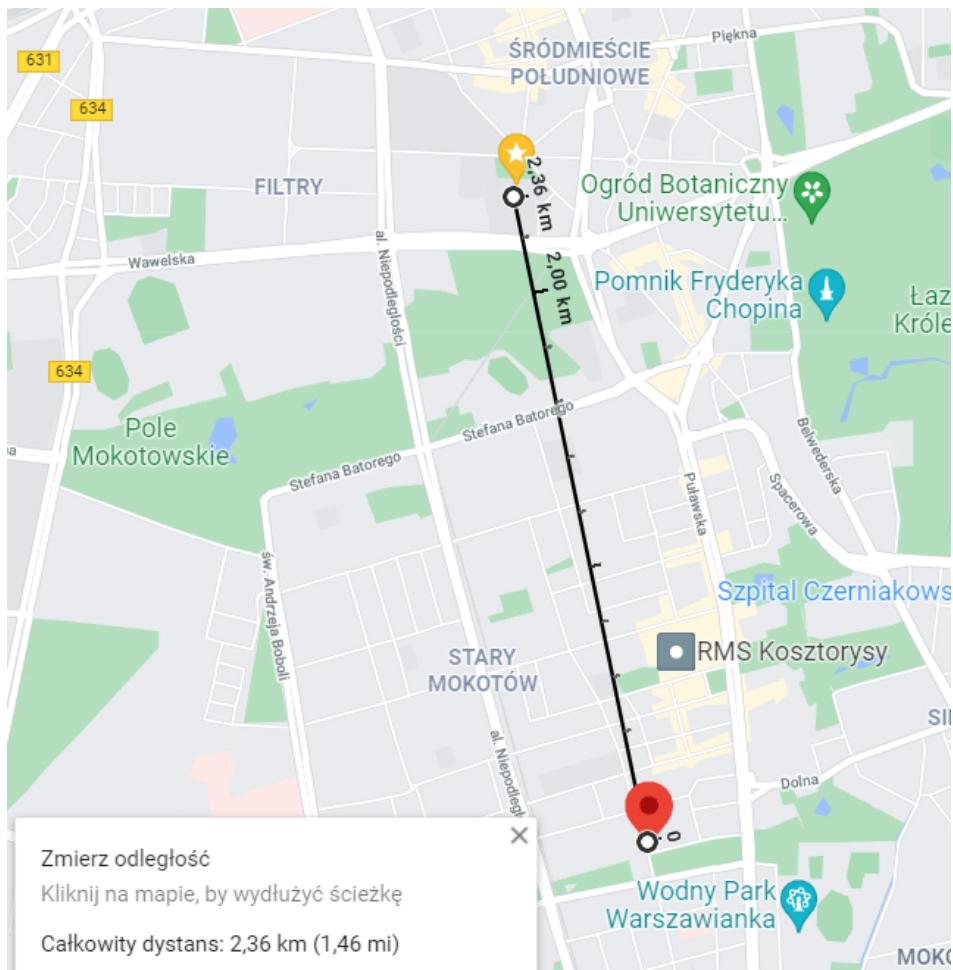
Udało się połączyć oraz zrealizować połączenia uplink i downlink w sieci Helium. Sieć ta ma wiele bramek na terenie Warszawy i komunikacja następowała z różnymi z nich. Przykładowo, udało nam się wysłać wiadomość uplink przez bramę amateur-orchid-seal z parametrami:

- odebrane RSSI -119 dB
- SNR -11 dB
- SF12BW125 (wiąże się to z dłuższym czasem wysyłania niż w przypadku zastosowania SF7)

Hotspot Name	RSSI	SNR	Frequency	Spreading	Time
amateur-orchid-seal	-119	-11.00	868.10	SF12BW125	Jan 20, 2022 10:34:04.437 AM

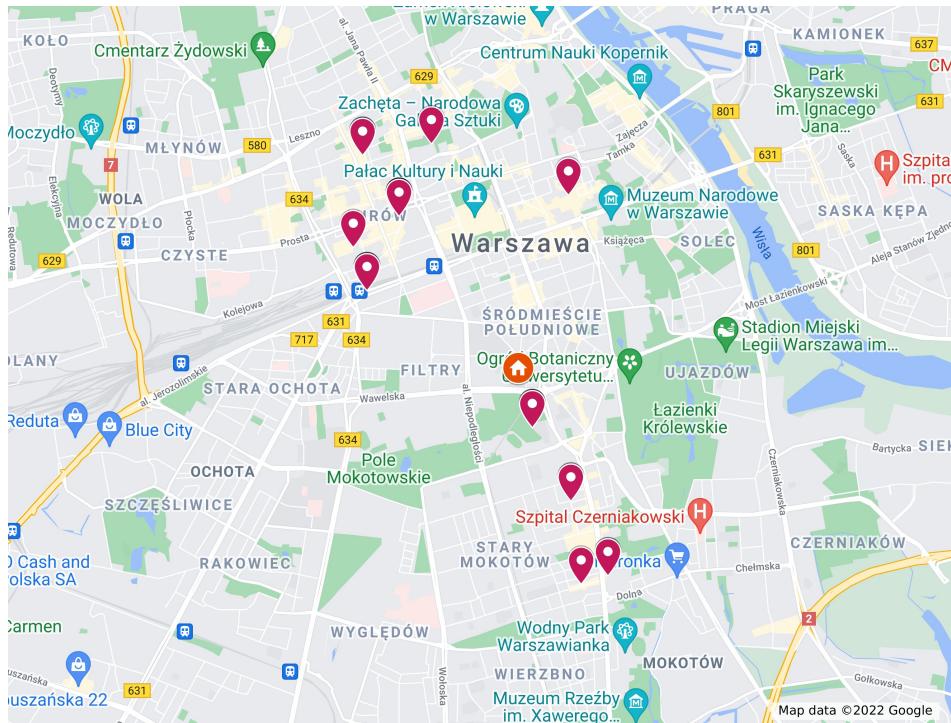
Rysunek 14: Widok konsoli Helium po udanym przesłaniu wiadomości uplink

Brama ta znajduje się 2,36 km od miejsca pomiaru (to dobry wynik w środowisku miejskim). Wykorzystuje ona antenę o zysku 6,5 dBi, która znajduje się na wysokości 30 m.



Rysunek 15: Pomiar odległości do bramki na mapie

Koszt wysłania 1 wiadomości uplink z payloadem wielkości 3B (takie wiadomości będą najczęściej wysyłane w tym projekcie) to 3DC, co odpowiada \$0.00003 USD.



Rysunek 16: Bramki Helium, z którymi udało nam się nawiązać komunikację

7.1.5 Wnioski

Przelaczanie się między różnymi sieciami LoRaWAN jest bardzo proste. Wymaga jedynie utworzenia aplikacji oraz dodania i skonfigurowania urządzenia. Moduł LoRaWAN wysyła wiadomości do wielu bramek LoRaWAN w danej sieci, więc nie trzeba definiować, której ma używać. Nawet w mieście da się osiągnąć kilka kilometrów zasięgu.

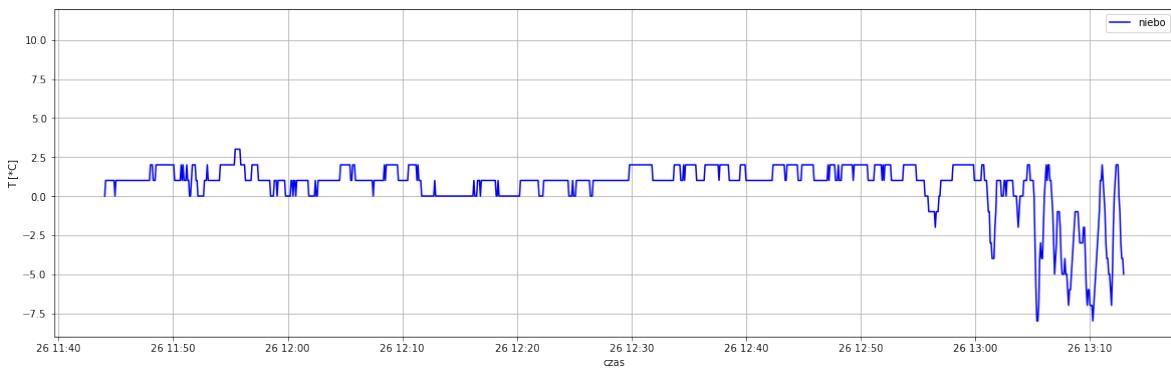
7.2 Testy czujników

7.2.1 Test nr 1

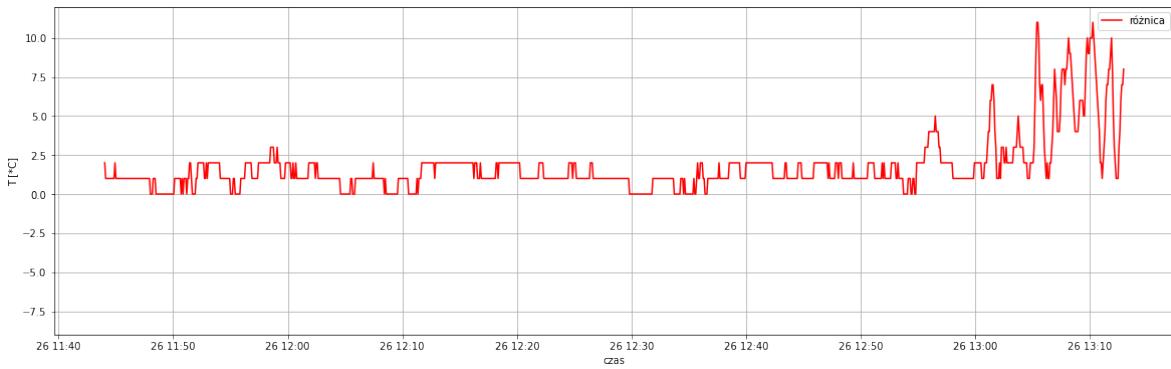
Test wykonany 26.01.2022 w godz. 11:44-13:13. Test przeprowadzony w warunkach dużego zachmurzenia. Pod koniec testu niebo zaczęło się przejaśniać. Dane zbierane z rozdzielcością 1°C (taką jaką jest przesyłana w projekcie) co 4s.



Rysunek 17: Zmiany temperatury otoczenia w czasie



Rysunek 18: Zmiany temperatury nieba w czasie



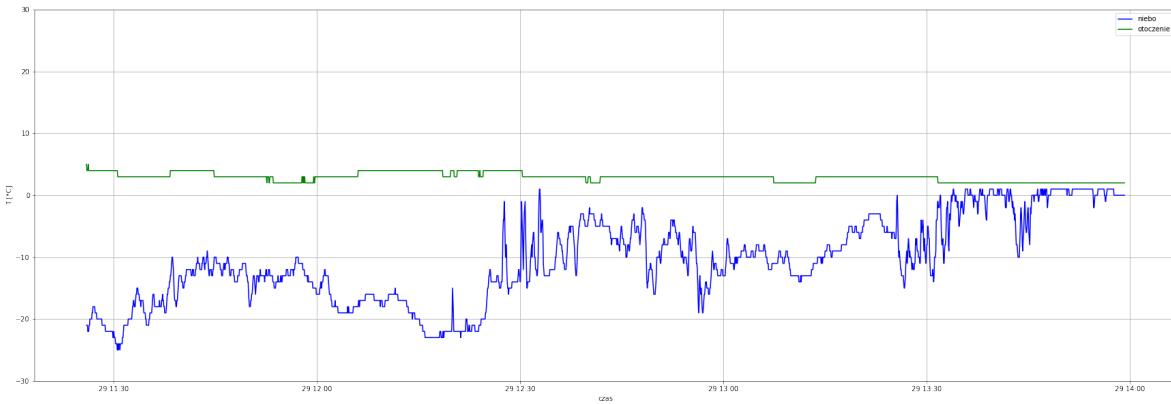
Rysunek 19: Zmiany różnicy temperatur w czasie

Możemy zauważyc, że temperatura otoczenia jest prawie stała (niewielkie wahania mogą wynikać z niedokładności czujnika i zaokrągleń) - zmienia się ona bardzo wolno w czasie.

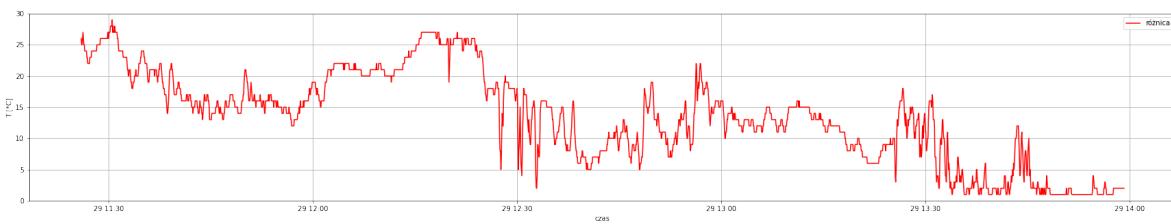
Gdy niebo jest jednolicie zachmurzone, temperatura nieba waha się tylko delikatnie ($0\text{--}3^{\circ}\text{C}$), więc wahania różnicy temperatur też są niewielkie i jest ona zbliżona do zera. Gdy na niebie pojawiają się przejaśnienia, ale nie jest ono całkowicie czyste, dochodzi do większych wahań temperatury nieba i różnic temperatur (chwilowe wahania dochodzą nawet do 10°C).

7.2.2 Test nr 2

Test wykonany 29.01.2022 w godz. 11:26-13:59. Dane zbierane z rozdzielcością 1°C (taką jaką jest przesyłana) co 3s.

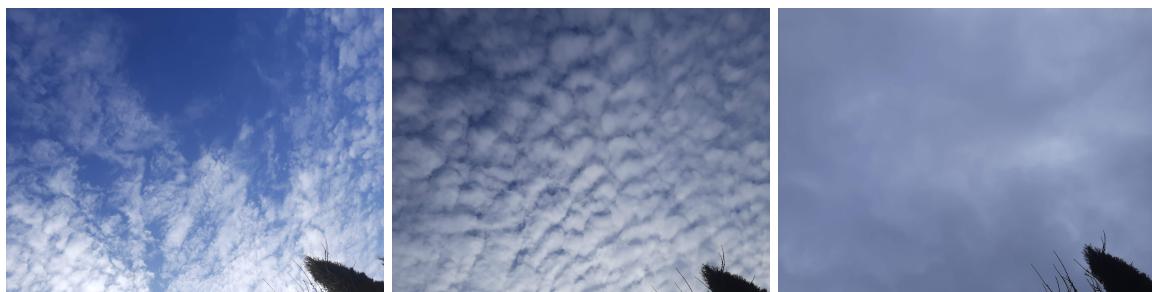


Rysunek 20: Zmiany temperatury nieba i otoczenia w czasie



Rysunek 21: Zmiany różnicy temperatur w czasie

Podczas testu temperatura otoczenia była prawie stała (ok. 3°C). Na początku na niebie były widoczne tylko pojedyncze chmury (rys. 22a). Temperatura nieba wynosiła ok. -20°C , a różnica temperatur ponad 20°C . Niebo nie było całkowicie czyste, więc temperatura się wała o kilka stopni. Koło godz. 11:40 zachmurzenie się zwiększyło, ale nadal było widoczne dużo przebłysków (rys. 22b). Temperatura nieba wała się od -10 do -20°C , a różnica temperatur wynosiła ok. 15°C (do 20°C). Koło godziny 12:20 mogliśmy znowu zaobserwować prawie bezchmurne niebo, a temperatury były zbliżone do początkowych (różnica temperatur powyżej 25°C). Od godziny 12:30 niebo było ciągle zachmurzone, ale nie w pełni. Występowały duże wahania temperatury (różnica temperatur wała się od 5 do 20°C). Po godzinie 13:30 niebo było już całkowicie zachmurzone (rys. 22c), a różnica temperatur zazwyczaj nie przekraczała 10°C . Mogliśmy zaobserwować krótkie przejaśnienie, podczas którego różnica temperatur wzrosła do ok. 10°C .



Rysunek 22: a) prawie bezchmurne b) większe, ale niepełne zachmurzenie c) całkowite zachmurzenie

7.2.3 Testy z raportu II

Warunki pogodowe nie pozwoliły nam w tym etapie na przeprowadzenie testów przy całkowicie bezchmurym niebie. Testy takie przeprowadziliśmy i opisaliśmy w poprzednim etapie (choć wtedy też niebo nie było całkowicie czyste). Różnica temperatur nieba i otoczenia wynosiła wtedy ponad 30°C .

7.2.4 Wnioski

Na podstawie przeprowadzonych przez nas obserwacji oraz pomiarów autora artykułu, na którym bazowaliśmy w poprzednich etapach [11] wyróżniliśmy 4 stany zachmurzenia nieba w zależności od różnicy temperatur:

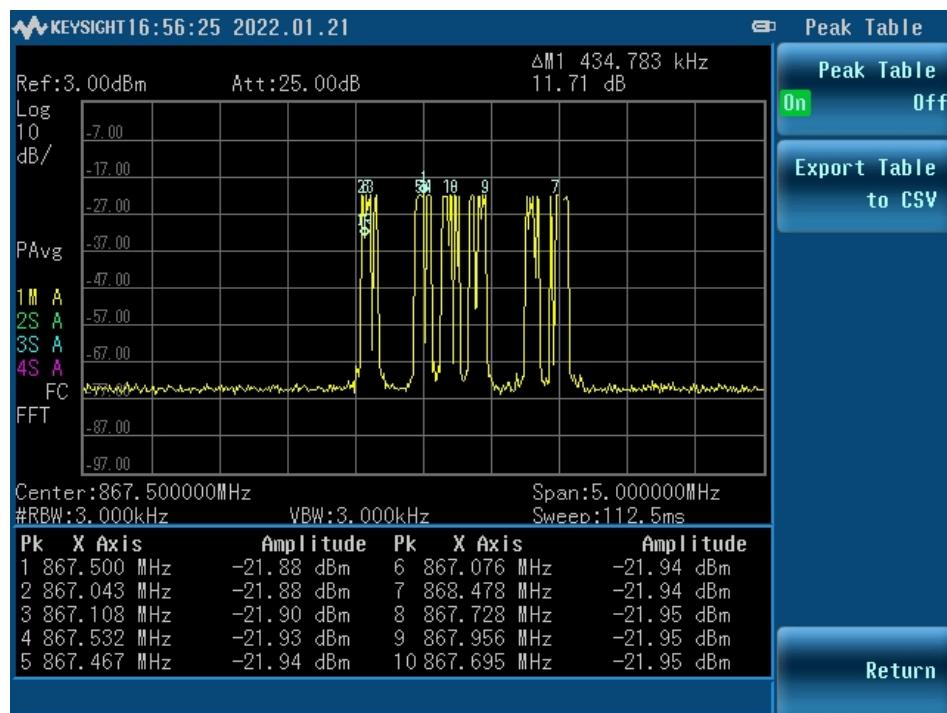
- 0 - brak zachmurzenia (różnica temperatur $\geq 40^{\circ}C$)
- 1 - małe zachmurzenie ($25^{\circ}C \leq$ różnica temperatur $< 40^{\circ}C$)
- 2 - średnie zachmurzenie ($10^{\circ}C \leq$ różnica temperatur $< 25^{\circ}C$)
- 3 - całkowite zachmurzenie (różnica temperatur $< 10^{\circ}C$)

W docelowym projekcie rozróżnianie stanu chmur mogłoby być wykonywane przy użyciu uczenia maszynowego, a ilość stanów zależna byłaby od potrzeb użytkownika.

8 Badania w labolatorium

8.1 Badania w labolatorium PTB

Przeprowadziliśmy testy komunikacji. Na rysunku 23 możemy zaobserwować transmisję w wybranych kanałach pasma 863-870 MHz (868 MHz) wykorzystywanych przez nasze urządzenie.



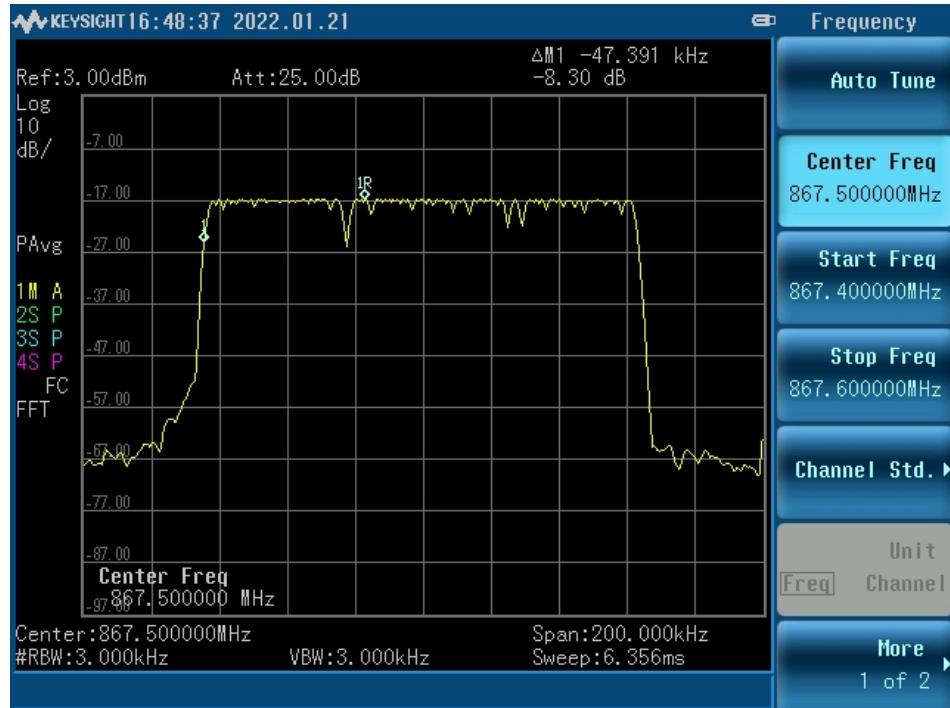
Rysunek 23: Transmisja w wybranych kanałach w paśmie 863-870 MHz

Urządzenie końcowe przesyła dane w losowym kanale z przedziału 0-7. W tym przypadku jesteśmy w stanie dostrzec transmisję w góre łączą w 6 kanałach [10]

- 867,1 MHz - kanał 3
- 867,5 MHz - kanał 5
- 867,7 MHz - kanał 6
- 867,9 MHz - kanał 7

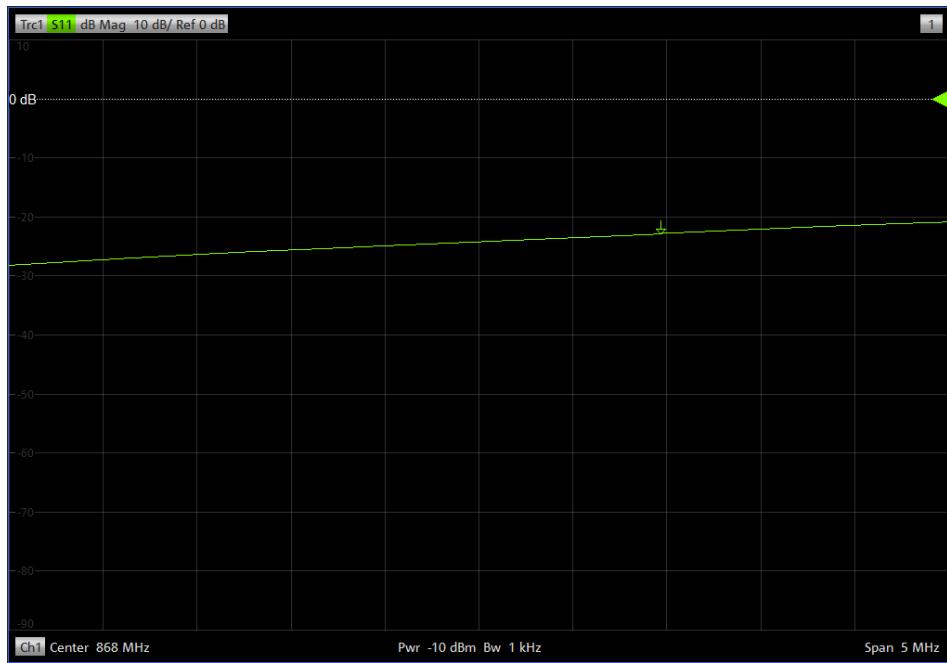
- 868,3 MHz - kanał 1
- 868,5 MHz - kanał 2

Dokonaliśmy dokładniejszą obserwację transmisji nadawanej w kanale 5 (częstotliwość środkowa 867,5 MHz) (rys. 24). Szerokość widma transmisji w tym kanale wyniosła ok. 125 kHz, co jest zgodne z wartością teoretyczną [10]. Moc odbierana przez analizator widma wyniosła ok. -17 dBm, co przy zastosowaniu między urządzeniem końcowym a analizatorem tłumika 20 dB i przewodami o niewielkiej rezystancji wskazuje na moc nadawania równą ok. 3 dBm.



Rysunek 24: Transmisja w kanale 5 (CF = 867,5 MHz)

Zbadaliśmy także dopasowanie wykorzystywanej przez urządzenie końcowe anteny przy wykorzystaniu analizatora obwodów. Co ciekawe, antena wykazuje bardzo dobre dopasowanie, gdy jest wyprostowana (straty odbicia ok. -25 dB)(rys. 25), natomiast po jej zgęściu dopasowanie gwałtownie się pogarsza - straty odbicia utrzymują się na poziomie ok. - 10 dB (rys. 26), czyli w zasadzie na granicy dobrego dopasowania, a nawet trochę powyżej tej granicy, co może powodować stratę mocy odbitej od wejścia anteny podczas prowadzenia transmisji (nawet ok. 20%).



Rysunek 25: Dopasowanie wyprostowanej anteny



Rysunek 26: Dopasowanie zgiętej anteny

8.2 Obserwacja wiadomości uplink MQTT w Wireshark

```
> Frame 68: 869 bytes on wire (6952 bits), 869 bytes captured (6952 bits) on interface \Device\NPF_{C749F463-70E1-4879-8B7E-C63C8B620CDA}, id 0
> Ethernet II, Src: TendaTec_da:65:18 (04:95:e6:da:65:18), Dst: Chongqin_b8:7f:7d (28:cd:c4:b8:7f:7d)
> Internet Protocol Version 4, Src: 52.212.223.226, Dst: 192.168.0.106
> Transmission Control Protocol, Src Port: 1883, Dst Port: 53828, Seq: 2, Ack: 1, Len: 815
    Source Port: 1883
    Destination Port: 53828
    [Stream index: 11]
    [TCP Segment Len: 815]
    Sequence Number: 2      (relative sequence number)
    Sequence Number (raw): 2271902160
    [Next Sequence Number: 817      (relative sequence number)]
    Acknowledgment Number: 1      (relative ack number)
    Acknowledgment number (raw): 826535650
    0101 .... = Header Length: 20 bytes (5)
    > Flags: 0x018 (PSH, ACK)
    Window: 219
    [Calculated window size: 219]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xbabd [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
    > [SEQ/ACK analysis]
    > [Timestamps]
    TCP payload (815 bytes)
    [PDU Size: 816]
    TCP segment data (815 bytes)
    > [2 Reassembled TCP Segments (816 bytes): #67(1), #68(815)]
        [Frame: 67, payload: 0-0 (1 byte)]
        [Frame: 68, payload: 1-815 (815 bytes)]
        [Segment count: 2]
        [Reassembled TCP length: 816]
        [Reassembled TCP Data: 30ad06003a76332f636c6f7564732d666c6f772d636f6e74726f6c4074746e2f64657669...]
    > MQ Telemetry Transport Protocol, Publish Message
        > [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet?)]
        > Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
            0011 .... = Message Type: Publish Message (3)
            .... 0... = DUP Flag: Not set
            .... .0. = QoS Level: At most once delivery (Fire and Forget) (0)
            .... .0 = Retain: Not set
        Msg Len: 813
        Topic Length: 58
        Topic: v3/clouds-flow-control@ttn/devices/eui-70b3d57ed00486b4/up
        Message: 7b22656e645f6465766963655f696473223a7b226465766963655f6964223a226575692...
```

Rysunek 27: Podgląd w Wireshark

Podejrzelismy w programie Wireshark wiadomość uplink (3 bajty danych z czujników) przesyłaną za pomocą MQTT od serwera TTS do klienta MQTT. Nie włączylismy szyfrowania wiadomości (dołączylismy do portu 1883, a nie 8883), więc mogliśmy podejrzeć temat i zawartość wiadomości. Wynika z tego, że w wersji produkcyjnej powinniśmy użyć połączenia szyfrowanego. Możemy zaobserwować, że wiadomości są przesyłane z poziomem QoS równym 0 (Fire and Forget), mimo że subskrybowaliśmy z QoS równym 2. Zgadza się to z dokumentacją TTS, gdzie zostało napisane, że maksymalny obsługiwany poziom QoS w TTS to 0. Mechanizm Retain Message nie został wykorzystany. Możemy zaobserwować warstwy komunikacji, które są wykorzystywane przez MQTT. W najniższej warstwie ramka ma 869 bajtów. Znaczącą część tej ramki stanowią temat oraz treść wiadomości (w formacie narzuconym przez TTS), narzuty innych warstw są niewielkie. Duży rozmiar wiadomości należy wziąć pod uwagę przy tworzeniu serwera odbierającego wiadomości z TTS.

9 Sposób implementacji sieci

Nasz system mógłby być implementowany na dwa sposoby:

- w wersji społecznościowej,
- w wersji przeznaczonej do zastosowań profesjonalnych.

W przypadku wersji społecznościowej, za rozwój całego systemu byłaby odpowiedzialna społeczność osób prowadzących obserwacje astronomiczne we własnym zakresie (najprawdopodobniej byliby to amatorzy oraz półprofesjonalni), chcących monitorować stan zachmurzenia nieba w okolicach, w których najczęściej obserwacje astronomiczne prowadzą. W tym wypadku implementacja takiej sieci musi uwzględniać:

- obecność bramki LoRaWAN w okolicy, w której dana osoba chciałaby zaimplementować nasz system; w przypadku gdy takiej bramki nie ma, dany użytkownik, lub grupa użytkowników musieliby zakupić bramkę umożliwiającą dostęp do wybranej sieci LoRaWAN;
- zakup urządzenia końcowego skonstruowanego przez nasz zespół, oraz jego samodzielnej implementację, wg. prowadzonych przez nas instrukcji.

Użytkownikowi udostępniane byłyby wtedy wszystkie dogodności związane z naszym systemem oraz mógłby korzystać z dostępnych informacji dotyczących stanu wszystkich urządzeń w danej sieci (przeprowadzanych przez nich pomiarów, stanu zachmurzenia oraz ich położenia).

W przypadku wersji przeznaczonej do zastosowań profesjonalnych, skierowanej przede wszystkim do obserwatoriów astronomicznych oraz profesjonalistów, umożliilibyśmy rozwój swojego systemu w ramach osobnej aplikacji, dzięki czemu jej użytkownicy mieliby dostęp do systemu na wyłączność. Implementacją takiej sieci zajmowalibyśmy się osobiście w ramach wykonania odpłatnej usługi. Wymagania dotyczące sieci, m.in. gęstości rozmieszczenia urządzeń końcowych, wyboru odpowiedniej bramki, charakterystyki komunikacji byłyby uzgadniane z użytkownikami wg. ich potrzeb. Dzięki zupełnie osobnej implementacji od wersji społecznościowej, w systemie tym unikaloby się przeciążeń sieci, a użytkownicy mieliby pełną kontrolę administracyjną nad swoim systemem.

10 Podsumowanie

Udało nam się zaplanować i stworzyć (przynajmniej w podstawowym zakresie) wszystkie węzły w naszej sieci. Stworzyliśmy prosty prototyp urządzenia końcowego, który zbiera dane z czujników oraz komunikuje się w obie strony z serwerem LoRaWAN. Przeprowadziliśmy pomiary różnicy temperatur nieba i otoczenia w różnych warunkach zachmurzenia, co pozwoliło na zweryfikowanie założeń teoretycznych z I etapu (możliwe jest analizowanie zachmurzenia na podstawie temperatury) oraz na podzielenie stopnia zachmurzenia na 4-stopniową skalę w zależności od wskazań czujników. Przetestowaliśmy 3 rozwiązania w zakresie bramek i serwerów LoRaWAN: The Things Stack Community Edition, Helium oraz bramkę w laboratorium 432. Zapoznaliśmy się z działaniem wszystkich tych rozwiązań i udało nam się użyć każdego z nich do przesłania wiadomości zarówno w dół, jak i w górę łącza. Używając sieci Helium udało nam się osiągnąć ponad 2 kilometry zasięgu w mieście, więc prawdopodobnie na terenach mniej zaludnionych (głównie tam znajdują się obserwatoria astronomiczne) zasięg ten byłby jeszcze większy. Zaobserwowałyśmy parametry połączenia LoRaWAN w warunkach laboratoryjnych na analizatorze widma oraz zbadaliśmy dopasowanie używanej anteny. Udało nam się odbierać dane z serwera LoRaWAN TTS i zapisywać je do bazy danych w serwisie AWS. Zrobiliśmy też lokalną implementację serwera, w której klient MQTT Paho odbierał (i wysyłał) wiadomości z TTS, przetwarzając je i zapisywał do bazy danych InfluxDB, a następnie zebrane dane były prezentowane na mapie w narzędziu Grafana. Niestety ze względu na wprowadzenie trybu zdalnego zajęć oraz brak czasu nie udało nam się zaimplementować lepszej aplikacji dla użytkownika, która pozwalałaby mu na zarządzanie urządzeniami końcowymi poprzez wysyłanie wiadomości downlink. Nie udało nam się też przeprowadzić wszystkich testów, które planowaliśmy. Mimo tego projekt możemy uznać za zakończony sukcesem. Udało nam się zaplanować i stworzyć działający prototyp sieci oraz wykonać na nim testy.

Literatura

- [1] Dokumentacja lora basics station. dostęp online [13-01-2022]. URL: <https://doc.sm.tc/station/>.
- [2] Lora network and protocol architecture frame structure. dostęp online [14-01-2022]. URL: <https://www.techplayon.com/lora-long-range-network-architecture-protocol-architecture-and-frame-formats/>.
- [3] Rak7268 wisgate edge lite 2 datasheet. dostęp online [24-01-2022]. URL: <https://docs.rakwireless.com/Product-Categories/WisGate/RAK7268/Datasheet/>.
- [4] Strona internetowa sieci helium. dostęp online [24-01-2022]. URL: <https://www.helium.com/lorawan>.
- [5] avbentem. Kalkulator czasu nadawania lorawan. dostęp online [22-12-2021]. URL: <https://avbentem.github.io/airtime-calculator/ttn/eu868/3>.
- [6] Eric B. Lora docs. dostęp online [18-01-2022]. URL: <https://lora.readthedocs.io/en/latest/>.
- [7] Cayenne. Cayenne low power docs. dostęp online [29-12-2021]. URL: <https://developers.mydevices.com/cayenne/docs/lora/#lora-cayenne-low-power-payload>.
- [8] Monika Dąbek. Zastosowanie informacji satelitarnej msg/seviri do oceny zachmurzenia ogólnego w Polsce, Warszawa 2011. Teledetekcja Środowiska, dostęp online [21-10-2021]. URL: http://geoinformatics.uw.edu.pl/wp-content/uploads/sites/26/2014/03/TS_v45_11-18.pdf.
- [9] Jeff Geerling. Raspberry pi zero - conserve power and reduce draw to 80ma. dostęp online [29-01-2022]. URL: <https://www.jeffgeerling.com/blogs/jeff-geerling/raspberry-pi-zero-conserve-energy>.
- [10] Troy Martin. Understanding lorawan frequencies. dostęp online [26-01-2022]. URL: <https://wifivitae.com/2021/06/29/lorawan-freq/>.
- [11] Chris Ramsay. Cloud detection - part one. dostęp online [25-10-2021]. URL: <https://chrisramsay.co.uk/posts/2014/03/cloud-detection-part-one/>.