Internet Rzeczy – przykład implementacji protokołu LoRaWAN

Andreas Kowol

Politechnika Opolska, Wydział Elektrotechniki, Automatyki i Informatyki, Instytut Elektroenergetyki i Energii Odnawialnej, ul. Prószkowska 76, 45-758 Opole

Streszczenie: W artykule przedstawiono proces konfiguracji mikroukładów LoPy do komunikacji radiowej, za pośrednictwem protokołu LoRaWAN. Mikroukłady umieszczono na podstawkach Pyscan firmy Pycom. Jeden zestaw zaprogramowano jako bramka komunikacyjna zwracająca otrzymane dane do portalu The Things Network przez wbudowany moduł Wi-Fi realizujący połączenie internetowe. Drugi zestaw pełni rolę zdalnego czujnika przesyłającego informacje o natężeniu oświetlenia. Dokonano rejestracji urządzeń w sieci TTN oraz zaimplementowano bazę danych. Baza danych pozwala na pobieranie wpisów przez REST API. Bramka została udostępniona na mapie dla innych użytkowników i stanowi bezpłatny punkt dostępu do sieci TTN. Programowania mikroukładów dokonano za pomocą języka MicroPython w środowisku Atom. Zrealizowany układ umożliwia przesyłanie danych z czujników na znaczne odległości, przy zachowaniu niskiego kosztu utrzymania systemu i niskiego zapotrzebowania na energię.

Słowa kluczowe: LoRaWAN, Pyscan, LoPy, komunikacja radiowa, The Things Network, Internet Rzeczy

1. Wprowadzenie

IoT czyli Internet of Things, tłumaczone dosłownie oznacza Internet Rzeczy. Kryje się za tym koncepcja mająca na celu umożliwienie urządzeniom wymianę informacji za pośrednictwem sieci. Przepływ informacji pozwala na poprawne funkcjonowanie zespołu urządzeń, gromadzenie danych lub jedynie informowanie użytkownika o obecnym stanie. Podłączenie niektórych urządzeń do sieci umożliwia zdalną kontrolę ich funkcjonalności na podstawie otrzymanych danych. Do łączenia pojedynczych urządzeń obecnie najczęściej wykorzystywana jest technologia bezprzewodowa (radiowa) a podłączone urządzenia pracują w topologii gwiazdy. Wiąże się to z podłączeniem urządzeń wykonawczych do bramki komunikacyjnej, której zadaniem jest przekazanie informacji do sieci. Zależnie od konfiguracji sieci, brama odbiera, wysyła, lub odbiera i wysyła pakiety danych do podłączonych do niej urządzeń umożliwiając im wzajemną komunikację. Brama sieciowa natomiast ma połączenie z innymi bramami lub z Internetem (rys. 1).

Rys. 1. Topologia sieci LoRaWAN wykorzystywana do przesyłania, gromadzenia oraz udostępniania danych

Fig. 1. LoRaWAN network topology used to transmit, storage and access data

Obecnie znanych jest kilka technologii pozwalających na przesyłanie danych drogą radiową – m.in. Wi-Fi, Bluetooth, LTE. Użytkownik, wybierając którąś z tych technologii jest zmuszony do kompromisu między zasięgiem, prędkością transmisji danych oraz opłatą za łącze (w przypadku LTE). Wi-Fi oferuje dużą przepustowość, jednak zasięg w normalnych warunkach jest mocno ograniczony. Dodatkowo pasmo 2,4 GHz jest wykorzystywane również przez inne technologie czy urządzenia użytku domowego pracujące na tym pasmie częstotliwości, co często wiąże się z wzajemnym zagłuszaniem sygnałów. Bluetooth jest technologią relatywnie prostą i tanią, jednak nie pozwala na połączenia większe niż 100 m, co mocno ogranicza użytkowników planujących łączyć urządzenia na znaczne odległości.

Autor korespondujący:

Andreas Kowol, a.kowol@doktorant.po.edu.pl

Artykuł recenzowany

nadesłany 30.05.2019 r., przyjęty do druku 28.06.2019 r.

Zezwala się na korzystanie z artykułu na warunkach licencji Creative Commons Uznanie autorstwa 3.0

Modulacja LoRa to warstwa fizyczna (PHY), natomiast protokół LoRaWAN pracuje w warstwie MAC (ang. *Medium Access Protocol*) czyli w warstwie łącza danych [1]. Pozwala na osiągnięcie dalekich zasięgów – do 5 km w terenach zurbanizowanych oraz do 15 km na terenach słabo zurbanizowanych [2]. Znaczną zaletą stosowania tego protokołu jest niskie zapotrzebowanie na energię. Węzły komunikacyjne mogą działać na baterii przez długi czas. Komunikacja o długim zasięgu oraz brak konieczności stałego dostępu do zasilania, pozwala na pracę czujników temperatury, wilgotci, nasłonecznienia itp. na farmach w odległych terenach wiejskich. Kompromisem tak dalekiego zasięgu jest niska szybkość transmisji danych, od 0,3 kb/s do 50 kb/s [3]. Zależy ona od długości komunikatu – wykorzystywana jest technika Adaptive Data Rate [2]. Ta technika pozwala na jeszcze mniejsze zużycie energii podczas transmisji danych.

Analizując powyższe informacje, można stwierdzić, że protokół LoRaWAN spełnia niemal wszystkie założenia komunikacji IoT. Dzięki wykorzystaniu nielicencjonowanego pasma, uruchomiony układ nie będzie wymagał kosztów utrzymania, co w przypadku uruchomienia większej liczby urządzeń końcowych generuje znaczne oszczędności. Dzięki dużemu zasięgowi nie jest wymagana duża liczba bramek komunikacyjnych do pokrycia zasięgiem dużego obszaru, co również przekłada się na obniżenie kosztów inwestycji.

2. Opis wykorzystanych urządzeń

Do stworzenia połączenia radiowego za pomocą protokołu LoRaWAN użyto dwóch zestawów urządzeń. Zestaw składa się z modułu umożliwiającego komunikację z siecią LoRa, który został zainstalowany na płytce rozszerzeń (rys. 4). Jeden zestaw został zaprogramowany jako bramka dostępowa, natomiast drugi jako węzeł końcowy.

Tabela 1. Specyfikacja modułu LoPy
Table 1. Module LoPy specification

CPU	Mikroprocesor Xtenasa dual-core 32-bit LX6 do 600 DMIPS Przyśpieszenie sprzętowe, Wielowątkowość Python Dodatkowy kompresor ULP, który monitoruje GPIO, kanały ADC i kontroluje większość urządzeń peryferyjnych w trybie uśpienia zużywając jedynie 25 µA
Pamięć	512 kB RAM 4 MB zewnętrznej pamięci flash
Wi-Fi	802.11b/g/n 16 Mbps
Bluetooth	Klasyczny z niskim zapotrzebowaniem na energię
LoRa	LoRaWAN 1.0.2 stack – Klasa urządzeń A i C Zasięg węzłów końcowych do 40 km Bramka komunikacyjna do 20 km (możliwość podłączenie do 100 węzłów końcowych)
Zegar czasu rzeczywistego	Tak
Zabezpieczenia	SSL/TLS WPA Enterprise
Hash/szyfrowanie	SHA MD5 DES AES

2.1. LoPy

LoPy to moduł firmy Pycom pozwalający na komunikację radiową przy użyciu protokołu LoRa dla częstotliwości 868 MHz. Zaimplementowano w nim również możliwość połączenia Bluetooth oraz Wi-Fi. LoPy oparty jest na systemie Linux, wspiera język programowania MicroPython. Ma dwurdzeniowy mikrokontroler oraz moduł ESP32 z 512 kB pamięci RAM (rys. 2, tab. 1).

Rys. 2. LoPy – moduł LoRa firmy Pycom [4]
Fig. 2. LoPy – LoRa module by Pycom company [4]

2.2. Pyscan

Moduł LoPy może zostać zainstalowany na różnych płytkach rozszerzeń oferowanych przez firmę Pycom. Każda z nich umożliwia implementację czujników oraz urządzeń zewnętrznych. W przykładzie została użyta płytka Pyscan. Znajduje się na niej akcelerometr, czujnik światła oraz przycisk definiowany przez użytkownika. Jako rozszerzenia Pyscan ma złącza do podłączenia czujników I/O, kamery, czytnika kodów kreskowych, czujnika podczerwieni lub ekranu LCD (rys. 3). Pyscan umożliwia również ładowanie baterii Li-Ion/Li-Po. Bateria

będzie ładowana, gdy płytka zostanie zasilona przez port micro USB, natomiast po odłączeniu portu bateria będzie źródłem zasilania. Dostępny jest również czytnik kart microSD. Firma Pycom udostępnia biblioteki do obsługi czytnika RFID/NFC oraz reszty podzespołów w serwisie Github [6]. Na płytce umieszczono komponenty:

- akcelerometr: ST LIS2HH12,
- czujnik natężenia światła: Lite-on LTR-329ALS-01,
- czytnik RFID/NFC: NXP MFRC63002HN, 151.

Rys. 3. Pyscan – płytka rozszerzeń dla modułu LoPy [5]
Fig. 3. Pyscan – extension board for LoPy module [5]

Rys. 4. Dwa wykorzystane zestawy zawierające LoPy oraz Pyscan
Fig. 4. Two used sets include LoPy and Pyscan

3. Konfiguracja urządzeń

3.1. Definicja bramki

Jeden z wyżej opisanych zestawów posłużył do stworzenia bramki komunikacyjnej. Zastosowano tu język MicroPython oraz darmowe środowisko programistyczne Atom [7], w którym zainstalowano rozszerzenie Pymakr, umożliwiające komunikację i wgrywanie programów oraz bibliotek do urządzeń firmy Pycom (rys. 5). Środowisko oraz rozszerzenie są w całości darmowe. Atom pobierany jest ze strony developera, a rozszerzenie można zainstalować z poziomu środowiska lub zewnętrznie.

Rys. 5. Interfejs środowiska Atom
Fig. 5. Interface of Atom environment

Podłączenie modułu LoPy za pośrednictwem płytki rozszerzeń i portu micro USB spowoduje, że środowisko Atom rozpozna urządzenie na jednym z portów COM. Poinformuje o tym stosownym komunikatem:

```
Connecting to COM27...
Pycom MicroPython 1.18.2.r3 [v1.8.6-849-a1641ca] on 2019-02-28; LoPy with ESP32
Type "help()" for more information.
```

Producent modułu umieścił w serwisie Github biblioteki z przykładami możliwości wykorzystania urządzeń [6], m.in. sposób zdefiniowania modułu jako bramka komunikacyjna (folder pycom-libraries-master\examples\lorawan-nano-gateway). Pliki potrzebne do uruchomienia bramki w sieci The Things Network to: config.py, main.py oraz nanogateway.py. W pierwszej kolejności należy zmodyfikować program config.py. Konieczne jest ustawienie odpowiedniej częstotliwości pracy, dla Europy będzie to wartość 868,1 MHz, podanie parametrów połączenia do sieci Wi-Fi, serwera oraz portu, do którego bramka ma się połączyć. Przykład konfiguracji podano poniżej.

```
import machine
import ubinascii
WIFI_MAC = ubinascii.hexlify(machine.unique_id()).upper()
GATEWAY_ID = WIFI_MAC[:6] + "FFFF" + WIFI_MAC[6:12]
SERVER = 'router.eu.thethings.network'
PORT = 1700
NTP = "pool.ntp.org"
NTP_PERIOD_S = 3600
WIFI_SSID = 'siecWiFi'
WIFI_PASS = 'haslo1234'
LORA_FREQUENCY = 868100000
LORA_GW_DR = "SF7BW125" # DR_5
LORA_NODE_DR = 5
```

Program nanogateway.py zawiera definicję klasy, która umożliwia komunikowanie się bramki z siecią TTN. Do poprawnego działania bramki nie jest wymagana modyfikacja. Main.py jest wczytywany jako pierwszy program po uruchomieniu urządzenia, algorytm sprawdza najpierw, czy w projekcie znajduje się plik boot.py, a następnie szuka programu main.py. Program ma

za zadanie wczytanie programu config.py oraz zaimportowanie klasy NanoGateway z programu nanogateway.py. Następnie do wywołanej klasy przypisane zostają wartości zawarte w config.py.

Rejestracja bramki do sieci wymaga wpisania EUI urządzenia. Jest to unikalny adres MAC modułu. Aby urządzenie zwróciło adres MAC w odpowiednim dla portalu formacie, wystarczy stworzyć krótki program, który za pomocą biblioteki network zwróci EUI. W analizowanym przypadku program został nazwany device_eui.py. Kod przedstawiono poniżej.

```
from network import WLAN
import binascii
wl = WLAN()
print("Gateway EUI: {}".format(binascii.hexlify(wl.mac())[6] + 'ffff'
+ binascii.hexlify(wl.mac())[6:]))
```

Tak przygotowany projekt można zaimportować do modułu LoPy. Ważne jest, aby zapisać projekt przed każdym załadowaniem do modułu. Konieczne jest nawiązanie połączenia na porcie COM, następnie należy zaznaczyć program main.py i załadować projekt przyciskiem *Upload*. Środowisko wgry wszystkie pliki znajdujące się w folderze na moduł a konsola zwróci następujące komunikaty:

```
Uploading project (main folder)...
[1/4] Writing file config.py (1kb)
[2/4] Writing file device_eui.py (0kb)
[3/4] Writing file main.py (1kb)
[4/4] Writing file nanogateway.py (15kb)

Upload done, resetting board...
OKets Jun 8 2016 00:22:57
rst:0x7 (TG0WDT_SYS_RESET),boot:0x17 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff8028,len:8
load:0x3fff8030,len:1688
load:0x4009fa00,len:0
load:0x4009fa00,len:14592
entry 0x400a059c
[ 86.502] Starting LoRaWAN nano gateway with id: b'240AC4FFFE026340'
[ 91.125] WiFi connected to: siecWiFi
[ 91.130] Syncing time with pool.ntp.org ...
[ 91.287] RTC NTP sync complete
[ 91.772] Opening UDP socket to router.eu.thethings.network (52.169.76.203) port 1700...
[ 91.786] Setting up the LoRa radio at 868.1 MHz using SF7BW125
[ 91.803] LoRaWAN nano gateway online
[ 91.808] You may now press ENTER to enter the REPL
```

W celu odczytania EUI modułu wystarczy zaznaczyć program device_eui.py, wcisnąć przycisk *Run*. Konsola zwróci EUI, np.:

```
Gateway EUI: b'240ac4fffe026340'
```

Uruchomioną bramkę należy zarejestrować na portalu TheThingsNetwork.org. W tym celu należy stworzyć darmowe konto pod adresem account.thethingsnetwork.org/register. Po założeniu konta i weryfikacji adresu e-mail pod adresem console.thethingsnetwork.org/gateways/register można zarejestrować bramkę. Konieczne do wypełnienia są następujące pola:

– Gateway EUI – pole do wpisania EUI modułu,

- I'm using the legacy packet forwarder – w przypadku konfiguracji urządzeń opartych na systemie Linux wymagane jest zaznaczenie tego pola,
- Description – opis bramki nadany przez użytkownika, który będzie widoczna przez innych użytkowników,
- Frequency Plan – miejsce działania bramki (w analizowanym przypadku Europe 868 MHz),
- Router – wybrać router stosowny do miejsca działania bramki (tutaj jest to ttn-router-eu),
- Podanie lokalizacji oraz umiejscowienia anteny nie jest konieczne.

Po zakończeniu procesu rejestracji, wszystkie bramki będą widoczne pod adresem console.thethingsnetwork.org/gateways. W zakładce *Overview* wyświetlone zostaną szczegółowe informacje – status, ostatnie logowanie do sieci, otrzymane paczki danych, wysłane paczki danych. Opis bramki można uzupełnić o dodatkowe informacje dotyczące modułu. Zakładka *Traffic* przedstawia szczegółowe informacje odnośnie otrzymywanych przez bramkę danych. W zakładce *Settings* znajdują się ustawienia bramki oraz możliwości wypełnienia dodatkowych informacji widocznych dla użytkowników chcących skorzystać z bramki. Po dodaniu lokalizacji bramki, będzie ona widoczna na mapie umieszczonej na stronie głównej po zaakceptowaniu przez administratora.

3.2. Definicja węzła końcowego (czujnika)

Drugi z zestawów został zdefiniowany jako węzeł końcowy. Jego zadaniem jest pomiar natężenia światła w 2 kanałach. Natężenie światła niebieskiego i czerwonego. Podobnie jak w przypadku bramki komunikacyjnej, do definicji urządzenia pomiarowego napisano program w oparciu o przykład udostępniony przez producenta. Komunikacja z urządzeniem w środowisku Atom będzie odbywać się na innym porcie. Jeśli autodetekcja portu została wyłączona w ustawieniach należy sprawdzić, do którego portu szeregowego system przypisał płytkę rozszerzeń Pyscan i z tym portem szeregowym nawiązać komunikację.

Do uruchomienia komunikacji czujnika z bramką potrzebny jest program config.py, użyty w projekcie bramki. W projekcie stworzono program main.py – po podaniu zasilania na węzeł końcowy, moduł automatycznie wczytuje program. Main.py bazuje na przykładzie abp_node.py w folderze z przykładami, ścieżka: `pycom-libraries-master/examples/lorawan-nano-gateway`. Program zmodyfikowano, aby wykonywał pomiar natężenia światła co 60 s, wyświetla wynik oraz wysyła go do portalu TTN przez bramkę komunikacyjną. Jeśli wartość pomiarowa światła na kanale mierzącym kolor czerwony przekroczy 1000 lx, to dioda LED będzie świeciła na czerwono, w przeciwnym wypadku będzie świeciła na zielono. Kod źródłowy programu przedstawiono poniżej:

```
from network import LoRa
from LTR329ALS01 import LTR329ALS01
from pyscan import Pyscan
import socket
import binascii
import struct
import time
import config
import _thread
import pycom

lora = LoRa(mode=LoRa.LORAWAN, region=LoRa.EU868)
dev_addr = struct.unpack(">I", binascii.unhexlify('22011257'))[0]
nwk_swkey = binascii.unhexlify(22C4E8C9BDEF-
92266C7D703F56302964)
app_swkey = binascii.unhexlify(315D7240E5D11269D9B5B916D-
36F0AF1)
```

```

for i in range(3, 16):
    lora.remove_channel(i)
lora.add_channel(0, frequency=config.LORA_FREQUENCY, dr_min=0, dr_max=5)
lora.add_channel(1, frequency=config.LORA_FREQUENCY, dr_min=0, dr_max=5)
lora.add_channel(2, frequency=config.LORA_FREQUENCY, dr_min=0, dr_max=5)
lora.join(activation=LoRa.ABP, auth=(dev_addr, nwk_swkey, app_swkey))
s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
s.setsockopt(socket.SOL_LORA, socket.SO_DR, config.LORA_NODE_DR)
s.setblocking(False)
py = Pyscan()
lt = LTR329ALS01(py)
RGB_BRIGHTNESS = 0x8
RGB_RED = (RGB_BRIGHTNESS << 16)
RGB_GREEN = (RGB_BRIGHTNESS << 8)
RGB_BLUE = (RGB_BRIGHTNESS)
pycom.heartbeat(False)
while True:
    pkt = b'PKT #' + str(lt.light())
    sendit = str(lt.light())
    print('Sending:', pkt)
    s.send(sendit)
    print("Light (channel Blue lux, channel Red lux): " + str(lt.light()))
    if lt.light()[1] > 1000:
        pycom.rgbled(RGB_RED)
    else:
        pycom.rgbled(RGB_GREEN)
    time.sleep(2)
    rx, port = s.recvfrom(256)
    if rx:
        print('Received: {}, on port: {}'.format(rx, port))
        time.sleep(58)

```

Do poprawnego działania programu w projekcie należy dodać folder `/lib` zawierający niezbędne biblioteki. Do folderu należy dodać bibliotekę do obsługi czujnika natężenia światła `LTR329ALS01.py` oraz dwie biblioteki niezbędne do obsługi płytki rozszerzeń `Pyscan`, wymagane ze względu na użycie czujnika natężenia światła: `pycoproc.py` i `pyscan.py`. Wszystkie biblioteki dostępne są w lokalizacji `pycom-libraries-master\pybytes\pyscan\lib`. Struktura projektu powinna wyglądać w następujący sposób:

Rys. 6. Struktura projektu definiującego urządzenie jako węzeł końcowy

Fig. 6. The structure of the project which is defining the device as the end node

Ostatnią czynnością poprzedzającą wgranie projektu do modułu jest podanie parametrów pozwalających na weryfikację czujnika w portalu The Things Network. W tym celu należy stworzyć nową aplikację, do której zostanie przypisane urządzenie. Będąc zalogowanym do portalu, należy przejść pod adres: `console.thethingsnetwork.org/applications` oraz wcisnąć przycisk *Add application*. Użytkownik zostanie popro-

szony o podanie unikalnej nazwy oraz opisu aplikacji. Pole *Application EUI* informuje, że portal sam przypisze numer, a router zostanie przypisany automatycznie zgodnie z tym podanym wcześniej w bramce. Możliwe jest teraz dodanie nowej aplikacji. Użytkownik zostanie przeniesiony do widoku aplikacji. Kolejnym krokiem jest przypisanie urządzenia do aplikacji. W sekcji *Devices* kliknąć przycisk *register device*. Należy podać niezmienną, unikalną nazwę urządzenia dla tej aplikacji. Konieczne jest tu podanie EUI urządzenia. Jak w przypadku bramki, adres MAC można wygenerować za pomocą skryptu `device_eui.py` opisanego wcześniej. Adres MAC wygenerowany przez skrypt należy podać w polu *Device EUI*. Po uzupełnieniu niezbędnych pól zatwierdzić wciskając *Add device*. Użytkownik zostanie przeniesiony do widoku urządzenia. Domyślnie aplikacja używa weryfikacji urządzenia opartej na rozpoczęciu sesji po odpowiedniej weryfikacji klucza aplikacji, nazywanej również *Over The Air Activation (OTAA)*. Przedstawiony powyżej kod został przygotowany do weryfikacji zwanej *Activation By Personalization (ABP)*. Pozwała ona na personalizację urządzenia. Należy zmienić sposób weryfikacji w zakładce *Settings*, zaznaczając *Activation Method* na *ABP* a następnie *Save*. Użytkownik zostanie przekierowany z powrotem na stronę urządzenia. Ostatnim krokiem przed załadowaniem projektu do modułu jest umieszczenie wygenerowanych przez program kluczy do kodu znajdującego się w programie `main.py`. Deklaracja zmiennej `dev_addr` definiuje wygenerowany *Device Address*, deklaracja zmiennej `nwk_swkey` wygenerowany *Network Session Key* a zmienna `app_swkey` deklarację *App Session Key*. Warto wspomnieć, że podczas pracy z protokołem radiowym, jakim jest LoRaWAN, każdy będzie mógł przechwycić i przechować wiadomości przysyłane między węzłem końcowym a bramką. *App Session Key* jest odpowiedzialny za szyfrowanie wiadomości. *Network Session Key* natomiast zapobiega manipulacji wiadomościami gdyż spowoduje to niepowodzenie sprawdzenia MIC. Po podaniu parametrów autoryzacji należy zapisać projekt i załadować do modułu. Konsola zwróci następujący komunikat:

Uploading project (main folder)...

Creating dir lib

[1/5] Writing file lib/LTR329ALS01.py (2kb)

[2/5] Writing file lib/pycoproc.py (10kb)

[3/5] Writing file lib/pyscan.py (0kb)

[4/5] Writing file config.py (1kb)

[5/5] Writing file main.py (2kb)

Upload done, resetting board...

OKets Jun 8 2016 00:22:57

rst:0x7 (TG0WDT_SYS_RESET),boot:0x17 (SPI_FAST_FLASH_BOOT)

configsip: 0, SPIWP:0xee

clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00

mode:DIO, clock div:1

load:0x3fff8028,len:8

load:0x3fff8030,len:1688

load:0x4009fa00,len:0

load:0x4009fa00,len:14592

entry 0x400a059c

Sending: b'PKT #(9, 13)'

Light (channel Blue lux, channel Red lux): (9, 13)

Komunikat informuje użytkownika, że do modułu zostały wgrane wszystkie elementy projektu, a sam moduł po restarcie płytki zaczął wykonywać zaprogramowany pomiar. Zwrócony został również tekst zawierający wynik pomiaru oraz informacja o wysłanej paczce danych. W celu ograniczenia ilości wysyłanych danych, paczka zawiera jedynie wartości bez informacji o tym, z którego kanału pochodzi.

4. Zestawienie połączeń IoT

4.1. Czynności wstępne

Na początku należy sprawdzić, czy czujnik jest widoczny przez portal TTN, tym samym czy została nawiązana komunikacja między bramką a węzłem. W tym celu w zakładce *Gateways* należy wybrać bramkę, do której przypisano czujnik i sprawdzić, czy liczba otrzymanych wiadomości rośnie (rys. 7). Zgodnie z algorytmem, wartość powinna co minutę rosnąć o jeden, zakładając, że żaden inny czujnik nie jest połączony z bramką.

Rys. 7. Podgląd bramki komunikacyjnej w serwisie TTN

Fig. 7. Gateway overview in the TTN service

Jeżeli bramka otrzymuje wiadomości, oznacza to, że nawiązano komunikację z węzłem końcowym. Następnie należy przejść do strony czujnika. W menu aplikacji odszukać przypisany czujnik, sprawdzić jego status oraz liczbę wysłanych ramek *Frames up* (rys. 8). Jeśli wartość jest równa zero, a status wyświetla komunikat *Never seen* lub sugeruje, że czujnik był widziany dawniej niż przewidziany w programie okres, należy zresetować licznik ramek. Jest to pewnego rodzaju zabezpieczenie przed atakiem polegającym na powtarzaniu i ponownym wysyłaniu paczek danych. Zabezpieczenie używane jest przy statycznej aktywacji urządzenia (ABP). Gdy urządzenie jest resetowane lub uruchamiane ponownie, licznik wysłanych i otrzymanych wartości jest resetowany. Jeżeli urządzenie wyśle wiadomość lub ją odbierze, odpowiedni licznik jest inkrementowany. Jeśli urządzenie lub sieć otrzyma komunikat z licznikiem ramek niższym niż ostatni, wiadomość zostanie zignorowana.

Dane otrzymane przez bramkę komunikacyjną wyświetlane są jednocześnie w dwóch miejscach. Pierwszym z nich jest widok czujnika i zakładka *Data*. Przedstawiona jest tam lista wiadomości otrzymanych przez bramkę. Każda linia zawiera datę otrzymania paczki danych, wartość licznika, port oraz wiadomość zakodowaną w systemie szesnastkowym. Drugim miejscem, gdzie można znaleźć przychodzące wiadomości jest widok aplikacji i zakładka *Data*. Lista przedstawia dane wszystkich czujników, które zostały przypisane do danej aplikacji. Wyświetlana jest też data otrzymania przez bramkę wiadomości, numer oraz port.

Rys. 8. Podgląd urządzenia w serwisie TTN

Fig. 8. Device overview in the TTN service

Dane są otrzymywane w systemie szesnastkowym, jednak w portalu TTN istnieje możliwość dodania skryptów w języku JavaScript do dekodowania danych. W tym celu należy z widoku aplikacji przejść do zakładki *Payload Formats*. Developer umieścił przykłady skryptów dla dekodera, konwertera i enkodera. Kod można napisać w edytorze na stronie lub użyć pola *Payload* do symulacji i przetestowania skryptu. Na potrzeby rozpatrywanego projektu stworzono dekodery, który zamieniał wiadomość z systemu szesnastkowego na format string. Dodatkowo dopisana zostaje wiadomość tekstowa oparta na tekście wyświetlanym w programie *Main.py* węzła końcowego. Poniżej przedstawiono kod źródłowy skryptu:

```
function Decoder(bytes, port) {
  var data = String.fromCharCode.apply(null, bytes);
  return {
    "Light (blue, red)": data ;
  }
}
```

Po dodaniu skryptu, zarówno w widoku aplikacji jak i w widoku czujnika, dane będą listowane z użyciem skryptu. Przetworzony format w postaci tekstu jest wyświetlany obok wiadomości zakodowanej w postaci heksadecymalnej (rys. 9).

Rys. 9. Otrzymywane dane przez bramkę komunikacyjną prezentowane na portalu TTN

Fig. 9. Data received through the communication gateway presented on the TTN portal

4.2. Archiwizacja danych

Portal The Things Network umożliwia integrację baz danych, aby przechowywać dane procesowe z czujników. Dostęp do danych jest możliwy dzięki aplikacji REST API lub z poziomu strony internetowej. Aby dodać bazę danych do aplikacji należy przejść do widoku aplikacji, a następnie wybrać zakładkę *Integrations*. Widok zakładki *Integrations* przedstawia wszystkie zaimplementowane wcześniej dodatki. Aby dodać nowy, należy kliknąć przycisk *Add integration*. Zostanie wyświetlona strona z dodatkami. Należy odszukać *Data Storage*. Po kliknięciu w ikonkę użytkownik zostanie przeniesiony do witryny przedstawiającej informacje o danym dodatku. Aby dodać, należy kliknąć ikonkę *Add integration*. Baza danych zostanie dodana do wybranej aplikacji.

W celu zalogowania się do bazy danych należy z poziomu widoku aplikacji przejść do zakładki *Integrations*, wybrać bazę danych, a następnie kliknąć *Go to platform*. Użytkownik zostanie przekierowany do strony zewnętrznej opartej na bibliotece swagger, aby ułatwić dostęp do danych. W celu autoryzacji należy kliknąć przycisk *Authorize*, a następnie podać *Access key*, który znajduje się w widoku aplikacji w portalu TTN. Należy go skopiować i wkleić na stronie bazy danych. Po uzyskaniu autoryzacji kliknąć link */api/v2/query*. Zostanie wyświetlona struktura, w jakiej zostają zapisywane dane. Po kliknięciu *Try it out!* użytkownik ma dostęp do informacji zapisanych przez urządzenie. Domyślnie wyświetlane są wpisy z ostatniej godziny, jednak możliwa jest zmiana parametru wyświetlania ostatnich wpisów w rubryce *last*. Poniżej przedstawiono przykład wpisu w formacie JSON generowanego w bazie danych po wysłaniu wiadomości z czujnika do bramki komunikacyjnej:


```
{
  "Light (blue, red)": "(25, 39)",
  "device_id": "lightsensor",
  „raw”: „KDI1LCAzOSk=”,
  „time”: „2019-05-04T16:10:27.650296383Z”
}
```

5. Podsumowanie

Protokół LoRaWAN kierowany jest do grona użytkowników mających na celu wymianę danych na większe odległości minimalizując przy tym koszty utrzymania połączenia. Przykładem może być uruchomiona w Amsterdamie sieć IoT oparta na protokole LoRaWAN, która wykorzystuje jedynie 10 bramek, aby zapewnić zasięg na terenie całego miasta [8]. Wdrożenie projektu trwało sześć tygodni. Obecnie mieszkańcy przejęli kontrolę nad projektem i sami tworzą strukturę inteligentnego miasta. Przy wsparciu portalu The Things Network rozszerzyli liczbę bramek do 59 [9]. Aktualnie rozwijana jest koncepcja monitoringu parkowania samochodów ciężarowych. Uruchomiono dedykowaną bramkę, do której będzie podpiętych 80 czujników [10]. Kolejnym wdrażanym projektem jest monitorowanie wolnych miejsc parkingowych w mieście. Znanymi przypadkami aplikacji w innych miastach to: systemy monitorowania jakości powietrza, inteligentne oświetlenie ulic, inteligentne wykrywanie zanieczyszczeń w wodzie, monitorowanie komunikacji miejskiej i informowanie pasażerów o czasie przyjazdu lub zarządzanie odpadami [11]. Przy założeniu, że bramka komunikacyjna zostanie połączona z Internetem, dostęp do informacji będzie możliwy z każdego miejsca na ziemi.

W artykule przedstawiono jeden ze sposobów konfiguracji dwóch urządzeń do wymiany danych przy pomocy protokołu LoRaWAN. Na rynku istnieje wiele innych urządzeń, które można skonfigurować w podobny sposób, przykładem jest popularne Raspberry Pi lub znacznie tańsze moduły radiowe, np. SX1276. Najtańszy zestaw urządzeń, który pozwoli na nawiązanie komunikacji między czujnikiem a bramką kosztuje około 12 \$, przy praktycznie zerowych kosztach utrzymania sieci. Dodatkowo portal The Things Network za darmo i w bardzo prosty sposób umożliwia użytkownikowi komunikację bramki z Internetem i dostęp do bazy danych. Umożliwia ona pozy-

skiwanie wpisów przez aplikację typu REST API. Za pomocą webowej aplikacji będzie można publikować informacje na stronie lub zapisywać w innej, prywatnej bazie danych. Następnie informacje mogą zostać przetworzone i analizowane w oparciu o stworzone przebiegi. Portal TTN umożliwia integrację wielu innych dodatków, o których nie wspomniano w artykule.

Bibliografia

1. A.I. Pop, U. Raza, P. Kulkarni, M. Sooriyabandara. *Does B directional Traffic Do More Harm Than Good in LoRaWAN Based LPWA Networks?*, GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Dec. 2017
2. Pinkas A. *LoRa – energooszczędna, długodystansowa sieć w natarciu*, <https://www.magazynprzemyslowy.pl/produkcja/LoRa-energooszczedna-dlugodystansowa-siec-w-natarciu,8726,1> [dostęp 19.05.2019]
3. Abid-Ali A. *Jak najlepiej podłączyć urządzenia do chmury – NB-IoT czy LoRaWAN?*, <https://mikrokontroler.pl/2018/02/09/najlepiej-podlaczyc-urzadzenia-chmury-nb-iot-lorawan/> [dostęp 19.05.2019]
4. Pycom.io, *Dokumentacja modułu LoPy*, <https://pycom.io/wp-content/uploads/2018/08/LoPy-specsheet.pdf> [dostęp 19.05.2019]
5. Pycom.io, *Dokumentacja płytki rozszerzeń pycan*, <https://docs.pycom.io/datasheets/boards/pycan.html> [dostęp 19.05.2019]
6. Github.com, *Przykłady producenta*, <https://github.com/pycom/pycom-libraries/archive/master.zip> [dostęp 19.05.2019]
7. Atom, Środowisko programistyczne, <https://atom.io/> [dostęp 19.05.2019]
8. Rs-online, *Czym jest LoRaWAN?*, <https://pl.rs-online.com/web/generalDisplay.html?id=i/lora> [dostęp 19.05.2019]
9. The Things Network, *Sieć LoRa w Amsterdamie*, <https://www.thethingsnetwork.org/community/amsterdam/> [dostęp 19.05.2019]
10. The Things Network, *Inteligentne monitorowanie parkowania ciężarówek*, <https://www.thethingsnetwork.org/community/amsterdam/post/smart-parking-application-at-the-port-of-amsterdam> [dostęp 19.05.2019]
11. Sematech.com, *Przykłady aplikacji z użyciem komunikacji radiowej opartej na protokole LoRa*, <https://www.semtech.com/lora/lora-applications/smart-cities> [dostęp 19.05.2019]

Internet of Things – an Example of Implementing the LoRaWAN Protocol

Abstract: The paper presents the process of LoPy chip configuration for radio communication using the LoRaWAN protocol. The microchips were placed on Pyscan extension board. One of the devices was programmed as a communication gate. The received data were transmitted to The Things Network portal via Internet connection. Second device was working as a remote sensor that transmits information of the light intensity to the gate. Devices were registered in the TTN network. Database was implemented in the project for easy access via REST API type of application.

Keywords: LoRaWAN, Pyscan, LoPy, radio communication, The Things Network, Internet of Things

mgr inż. Andreas Kowol

a.kowol@doktorant.po.edu.pl

Absolwent Wydziału Elektrotechniki, Automatyki i Informatyki, kierunku Elektrotechnika na Politechnice Opolskiej. W 2016 r. uzyskał tytuł inżyniera, a w 2017 r. – magistra inżyniera. Ukończył również studia na kierunku Technologie Energetyki Odnawialnej z tytułem inżyniera w 2018 r. Obecnie doktorant na Politechnice Opolskiej. Pracuje jako programista sterowników PLC w firmie zajmującej się automatyką przemysłową. Jego zainteresowania obejmują wykorzystanie protokołu LoRa w odniesieniu do Internetu Rzeczy oraz doskonalenie umiejętności programowania w języku Python.