VRIJE
UNIVERSITEIT
BRUSSEL

Engineering technology: Electronics-ICT

LORAWAN FOR SOUND POLLUTION

ICT design

Valentin Quevy

2021-2022

Promoter: Abdellah Touhafi
FACULTY OF ENGINEERING

LoRaWAN for Sound Pollution, ICT design

Valentin Quevy
Department of Industrial Engineering Electronics ICT
Vrije Universiteit Brussel

2021-2022

Acknowledgements

I thank the VUB for giving access to the fablab, professor Abdellah for providing wise advice and my family, who helped me keeping head above water and pursue this thesis.

Valentin

Abstract

The goal of this study was to design a Low Power Wide Area Network (LPWAN) capable of mapping noise pollution and contributing to the digitalization of future smart cities.

The research findings proved theoretical evidence for determining a suitable networking protocol, architecture and additional tools for our use-case. LoRa and LoRaWAN were discovered to be Wide Area networking protocols, allowing the use of low-power and low-cost microcontroller chips such as the ESP32 at the end-device level. Pycom's development boards use a set of open tools to enable rapid prototyping with microPython on a wide range of networks. The Things Network (TTN) runs the entire LoRaWAN stack and Telegraf, InfluxDB, and Grafana (TIG) are great tools for data visualization.

We worked on the design, architecture and implementation of an LPWAN. We have set up a single-hop network where the environment sound-level periodically is send by our acoustic nodes and forwarded by our gateway to our cloud-based network server. The final visualization is performed on our local Linux system, which hosts the application in Dockerized containers.

Our results showed that LoRa proved to be an adequate modulation technique by dynamically regulating the transmission range and data rate through a Spreading Factor (SF). On top of the LoRa radio, LoRaWAN provides an adequate protocol for setting up the upper infrastructure of our IoT application by defining the Medium Access Control layer (MAC). The ESP32 is a well-fitting microcontroller for the nodes, offering sufficient computation power while consuming little power.

This thesis was meant to be in participation with Louka Grignard; for the hardware design and sound analysis I refer to his future work.

Contents

Contents iv

List of Figures vi

List of Tables vi

Nomenclature vii

1 Introduction 1

2 Background and context 2
- 2.1 Internet of things . . . 2
  - 2.1.1 Network & Architecture . . . 3
- 2.2 Networking and data transmission . . . 3
  - 2.2.1 Units and definitions . . . 4
  - 2.2.2 What is LoRa? . . . 5
  - 2.2.3 Operated and private networks . . . 6
  - 2.2.4 LoRaWAN as networking layer . . . 6
    - 2.2.4.1 Device activation . . . 7

3 Methodology 8
- 3.1 Network architecture . . . 9
- 3.2 Node . . . 10
  - 3.2.1 Code description . . . 10
    - 3.2.1.1 Configuration . . . 10
    - 3.2.1.2 NoiseNode . . . 10
    - 3.2.1.3 Main . . . 11
- 3.3 Gateway . . . 11
- 3.4 Network and application server . . . 12
- 3.5 Local application . . . 13

4 Results 14
- 4.1 Signal analysis with Software Defined Radio . . . 14
- 4.2 Range . . . 15
- 4.3 Node power consumption . . . 16

5 Discussion 17

6 Conclusion 18
- 6.1 Further work . . . 18

A Range Test 19

B Code 20

Bibliography 21

List of Figures

2.1 Global share of IoT projects, IoT Analytics 2018 . . . 2
2.2 IoT main topics . . . 3
2.3 Star of stars, typical LoRaWAN topology . . . 3
2.4 IoT communication technologies comparison [8]. . . . 4
2.5 Uplink LoRaWAN packet visualized on TTN . . . 5
2.6 Spectogram of LoRa chirps [3] with annotations . . . 6
2.7 LoRaWAN packet structure [1] . . . 7
2.8 Overview of the LoRaWAN 1.0.2 OTAA join-procedure . . . 7

3.1 Bachelor Proof time-line . . . 8
3.2 Network architecture and data-flow of our setup . . . 9
3.3 Main components of our network . . . 9
3.4 NoiseNode class overview . . . 10
3.5 NoiseNode life-cycle . . . 11
3.6 Gateway life-cycle . . . 11
3.7 TTN network architecture . . . 12
3.8 Live activity of our node on TTN . . . 12
3.9 Live activity of our gateway on TTN . . . 13
3.10 TIG stack data-flow . . . 13
3.11 Sound Level Monitoring of our node in Grafana . . . 13

4.1 LoRa Signal Analysis with Software Defined Radio , GNU-radio . . . 14
4.2 Confirmed uplink message tracing with GNU Radio, REPL- and TTN console . . . 15
4.3 Testing our node's range at Plan Incliné de Ronquières . . . 15
4.4 NoiseNode Power consumption . . . 16

A.1 Map illustrating 7 measurement points testing out SF7-12 on different distances . . . 19

List of Tables

2.1 Communication technologies comparison . . . 3
2.2 Comparison of a private and operated network, Chirpstack and TTN . . . 6

3.1 Default LoRa configuration of our nodes . . . 10

A.1 Measurements reported on TTN . . . 19

Abreviations and Acronyms

ABP	Activation By Personalization
ADC	Analog to Digital Converter
ALOHA	Advocates of Linux Open-source Hawaii Association
AppEUI	64-bit globally unique application identifier in IEEE EUI64 address space
BW	Band Width
CRC	Cyclic Redundancy Check
CR	Coding Rate
CSS	Chirp Spread Spectrum
DevEUI	64-bit globally unique device identifier in IEEE EUI64 address space
DevNonce	Unique, random, 2-byte value generated by the end device
DR	Data Rate
ETSI	European Telecommunications Standards Institute
fft	Fast Fourier Transform
FSK	Frequency Shift-Keying
gRPC	Google Remote Procedure Call
HTTP	Hyper Text Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
LoRaWAN	Long Range Wide Area Network networking protocol
LoRa	Long Range spread spectrum modulation technique
LPWAN	Low Power Wide Area Network
MAC	Medium Access Control
MEMS	MicroElectroMechanical Systems
MIC	Message Integrity Code
MQTT	Message Queuing Telemetry Transport
OTAA	Over The Air Activation
PHY	Physical layer
RF	Radio Frequency
RSSI	Received signal strength indication
SF	Spreading Factor
SNR	Sound Noise Ratio
SQL	Structured Query Language
TIG	Telegraf, InfluxDB and Grafana
TTN	The Things Network
UDP	User Datagram Protocol

Chapter 1

Introduction

In this bachelor thesis, we will define IoT as small battery-powered 'things' that are wirelessly connected to the internet. Sensors and processing power are built into it

IoT environments are becoming more popular, and the technologies that support them are rapidly evolving, enabling new application possibilities such as smart cities, industry, transportation, remote healthcare, and others. [13]. On an ever-growing planet the data quantity produced is in an increasing trend. IoT environments provide numerous solutions for data collection and utilization. Among these solutions, LoRa and LoRaWAN are two of the most promising networking solutions. IoT's main challenges are scalability, security, self-organization, and energy efficiency, not to mention societal and business-related issues.

According to the European Health Concern, noise pollution affects millions of people and is something we must consider. It is a serious threat to human health, increasing the risk of heart disease, stroke, diabetes, sleep disturbance, stress, and cognitive impairment in children. But apparently, also a great trouble for the nearby wildlife [9]. The average background noise level in cities is 60 decibels (dB), which is harmful and considered a silent killer. These environmental parameters could be sensed and monitored...

It is necessary to communicate the urban-data. The advantage of LoRa communication is its long-range capability with low power consumption. LoRa employs a Chirp Spread Spectrum modulation, which provides remarkable immunity to interference while retaining the low-power characteristics of traditional Frequency-Shift Keying (FSK) modulation. The implementation is low-cost because it makes use of international radio spectrum reserved for industrial, scientific, and medical (ISM) purposes, which requires no license fee. The band is open for use, which means there will be a lot of chatter and noise, limiting the possible data rate. You must follow regional standards, such as ETSI in Europe, when operating in those bands. The data throughput that such bands can achieve is very low. However, LoRa can be useful for periodic transmission of small packets. Above this is LoRaWAN, the network stack rooted on the LoRa physical layer that enables the upper networking layers.

This bachelor's thesis examines the requirements and limitations of our IoT application, 'Urban sound monitoring.' We talk about the network architecture for our acoustic meters. Other communication technologies and protocols such as LoRa, Zigbee, NB-IoT, and so on will be considered. Later, we'll look at how we can visualize the data we've gathered and provide some analytics and insights. In the final chapters, I evaluate and conclude on the proposed system.

The goal of this thesis is to deploy our smart acoustic meters in a LoRaWAN network for smart cities of the future. There are numerous questions that arise when designing such a system. How do we design and deploy such a network? Which technologies and protocols can help the project realize its full potential? What can be done with the collected data? ... these are the questions we will attempt to answer.

The main topics and core concepts will be explained in following sections involving a large technical background. Setting up an IoT-environment can be a challenging work. First, we provide a broad overview of how such a network could function and investigate the major components that make it up. Next, we'll go over the various technologies and protocols we'll be employing, such as LoRa and LoRaWAN.

2.1 Internet of things

The Internet of Things is an expanding network of interconnected objects. Devices that can interact with their surroundings and detect changes and events. Figure 2.1 helps situating the IoT markets, key-sectors and current trend. [7].

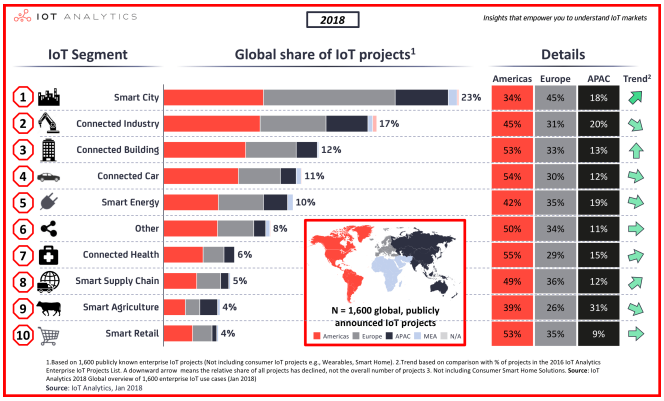

Figure 2.1: Global share of IoT projects, IoT Analytics 2018

Smart cities, connected buildings, connected healthcare, and smart supply chain are seeing widespread adoption in North America and Europe.

All IoT segments are made up of a mix of hardware, software, and other system architectures. Comprising a variety of processes such as computation, sensing, networking, and data sharing.

Sensors are used for sensing. Utilizing low-power protocols and technologies requires computation to maximize band usage and energy efficiency.

Data management is crucial. To avoid privacy and security issues, the data collected by the devices must be carefully managed. The network architecture defines the data flow and must be carefully chosen to avoid traffic congestion and collisions. [11].

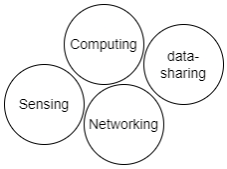

Figure 2.2: IoT main topics

2.1.1 Network & Architecture

An IoT networking system can be made up of a variety of components such as sensors, actuators, protocols, cloud services, and so on. The data path will be determined by the network architecture. A network's topology is crucial to its performance. There exists numerous topologies: point-to-point, bus, ring, star, tree, mesh, and hybrid. Each is appropriate for a specific set of circumstances.

A star topology can be useful and easier to implement in the case of a small application. Managing the entire network from a central hub, with devices that can be added and removed without disrupting network operations. Bandwidth will be defined by the central hub and performance by the network density.

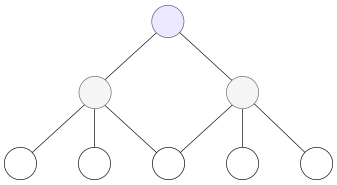

Figure 2.3: Star of stars, typical LoRaWAN topology

2.2 Networking and data transmission

Data often ends in the cloud. Before, this data must transit via a low power communication technology from the nodes to a gateway and must be chosen according to the IoT application requirements.

Following table gives a brief overview of the characteristics of interesting wireless network protocols.

Technology	Range	Power usage	cost	security	Bandwith	Data rate
LoRa	<5km (urban) <20 km (rural)	very low	very low	medium	125 & 250 kHz	0.3 - 5.5 kbps
SigFox	<50 km	very low	very low	low	100 Hz	100 bps
NB-IoT	<10 km	low	medium	high	200 kHz	200 kbps
BLE	<100 m	very low	very low	medium	1 MHz	125 kbps - 2 Mbps
Wi-Fi	<100 m	medium	high	low	2.4 GHz / 5 GHz	54 Mbps
ZigBee	<30 m	medium	very low	medium	2 MHz	250 kbps

Table 2.1: Communication technologies comparison

Mapping sound in an area require sufficient coverage. WiFi, BLE and Zigbee are used for short range applications. NB-IoT uses an licensed band, meaning you have to pay for the deployed infrastructure. Figure 2.4 gives a deeper comparison of the most suitable IoT technologies. Sigfox and LoRa are good candidates for our use-case with high coverage, very low power usage at low cost. However with LoRa we can reach a higher data rate and transmission duty-cycle. LoRaWAN integrates the AES128 as encryption standard making the communication relatively secure.

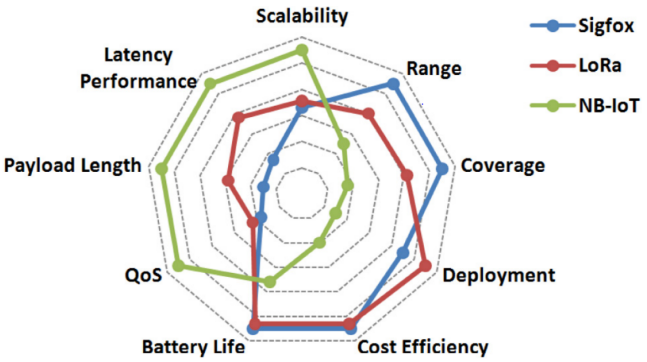

Figure 2.4: IoT communication technologies comparison [8].

2.2.1 Units and definitions

In this work we handle wireless communication through radio frequencies. It is important to know the base notions, definitions and units to use Radio Frequencies. RF is the oscillation rate of an alternating electromagnetic field in the frequency range from around 20 kHz to around 300 GHz [12].

The following lines define some of the most important definitions of radio transmission and contextualize them with an example on fig. 2.5.

- **RSSI** stands for the Received Signal Strength Indication. It is used as a measurement of how well a receiver can "hear" a signal from a sender. (dBm)
- **Sensitivity** is the minimal RSSI a receiver can accept in order to decode a signal. (dBm)
- **SNR** stands for Sound Over Noise Ratio, that is the ratio between the received signal- and the noise floor power level.

$$SNR(dB) = 10log(P_{\text{signal}}/P_{\text{noise}}) \tag{2.1}$$

- **Link Budget** is the sum of the power of transmission (P_{tx}), gains (G) and losses (L) of a signal from a sender to the final receiver. We define the minimum link budget as the difference between the power of transmission of the sender and the sensibility of the receiver. (dB)

$$P_{\text{rx}}(dBm) = P_{\text{tx}} + G - L \tag{2.2}$$

The ratio of two values of power is expressed in **dB**, **dBm** is with reference to one milliwatt. These are frequently used units in radio transmissions.

Figure 2.5 shows an uplink LoRaWAN packet received by our FiPy single-channel gateway using the SX1276 LoRa transceiver from Semtech. We can observe some of the recorded parameters of the transmission. We got an RSSI of -91 dB and SNR of 6 dB, this is indicating a rellatively weak but less corrupted signal. Typical LoRa SNR values are between -20 dB and 10 dB, for RSSI this ranges from 0 to -120 dBm.

```
"rx_metadata": [
  {
    "gateway_ids": {
      "gateway_id": "fcf5c4fffe0df9e4",
      "eui": "FCF5C4FFFE0DF9E4"
    },
    "time": "2022-04-29T15:58:11.504165Z",
    "timestamp": 108874522,
    "rssi": -91,
    "channel_rssi": -91,
    "snr": 6,
    "uplink_token": "Ch4KHAoQZmNmNWM0ZmZmZ
  }
],
```


Figure 2.5: Uplink LoRaWAN packet visualized on TTN

2.2.2 What is LoRa?

LoRa is a chirped spread spectrum (CSS)-based physical layer technology that operates in the unlicensed sub-GHz ISM band. CSS is a type of wideband linear frequency modulation in which the chirp pulse is a sweep of frequency over time. The modulation technique advantages are its affordability, low-power and long-range capability. LoRa is based on ALOHA principles. A node broadcasts data whenever data is available to send on a channel with a pre-configured data rate (DR) and transmission power (Ptx). LoRa works with a variety of ISM frequencies, including 868 MHz in Europe, 915 MHz in North America, and 433 MHz in the United States. In Europe the 868 MHz channel is divided in 8 sub-channels. LoRa is a single-hop technique that uses gateways to forward messages from a node to a central server as illustrated on fig. 2.3.

The communication performance is defined by the Spreading Factor (SF), bandwidth (BW), coding rate (CR) and Ptx. SF defines number of bits per symbol and vary from 7 to 12. A larger SF increases the time on air, which increases energy consumption, reduces the data rate, and improves communication range. The CR denotes the number of useful bits (4) which are going to be encoded by 5, 6, 7 or 8 transmission bits. Multiple LoRa broadcasts with distinct SFs are quasiorthogonal, allowing for multiple transmissions with various SFs at the same time. [10]. The chirp or symbol duration (T_S) is defined by eq. (2.3).

$$T_S = \frac{2^{SF}}{BW} \tag{2.3}$$

Depending on previous parameters LoRa enables data throughput ranging from 300 bps to 50 kbps. The useful bitrate can be calculated with eq. (2.4).

$$R_b = SF \times \frac{BW}{2^{SF}} \times CR \tag{2.4}$$

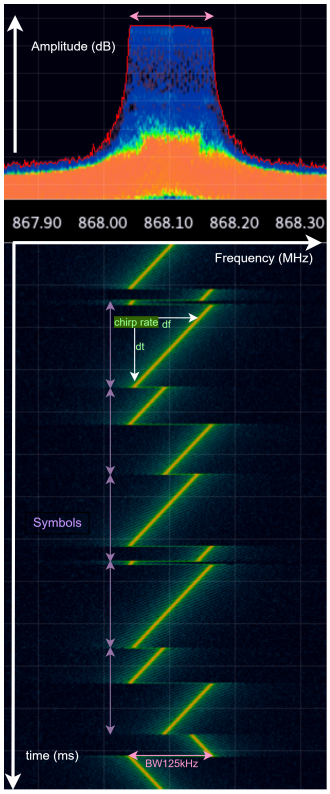

Figure 2.6: Spectogram of LoRa chirps [3] with annotations

2.2.3 Operated and private networks

The LoRaWAN protocol is maintained by the LoRa-Alliance. In Europe the ETSI-standards defines the rules but additional restrictions may apply if a network operator is used. They restrict the network occupancy, depending on the chosen channel. This may vary between 0.1 % and 1%.

There are two possibilities of network type, namely a private or operated network. Each has their own pros and cons. table 2.2 compares Chirpstack and The Thing Network (TTN), a private and an operated network.

	Chirpstack	TTN
Cost subscription	free	base fee and €1.5/device
Infrastructure cost	high (Gateways and servers)	low (included in subscription)
Required skills	high	low
Duty cycle (uplink)	36*sec*/*hour* (ETSI-standards)	according to subscription ($\leq$ ETSI)

Table 2.2: Comparison of a private and operated network, Chirpstack and TTN

2.2.4 LoRaWAN as networking layer

LoRaWAN act on top of the physical LoRa layer as the MAC and application layer. It encapsulates the frames so that they are suitable for transmission. The main features enable public and private deployment options, firmware updates over the air, geolocation, end-to-end security and long-range communication. According to the type of network, different kind of rules exists which are explained in section 2.2.3.

Figure 2.7 determines the packet structure. At the top we have the physical layer followed by the MAC- and the application layer. The physical layer is composed of a preamble, PHY header, CRC headers and the PHY payload. The preamble chirps are meant for signal detection, frame and frequency synching at the highest coding rate (4/8). The PHY-header gives the frame information. Finally the Cyclic Redundancy Check checks the integrity of it. In implicit mode, the PHY-header and CRC could be removed in order to decrease the overall duty-cycle.

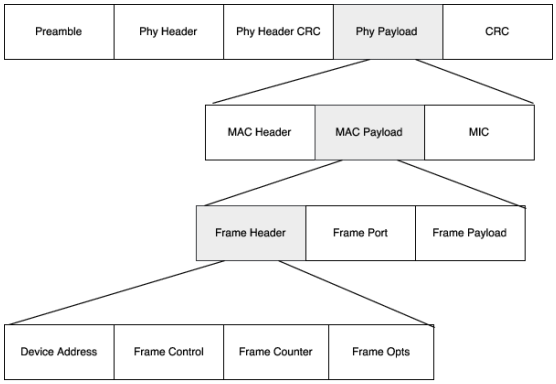

Figure 2.7: LoRaWAN packet structure [1]

At the MAC-layer, the MAC header defines the protocol version and message type in use. The Message Integrity Code will authenticate the frame at the network server. The frame header specifies the device adress, the frame control, the frame counter and the frame options at the application server. Finally the frame port bounds the frame payload to the correct application.

2.2.4.1 Device activation

Activation of the devices can happen through OTAA or ABP. These activation modes define the keys for decoding the MIC and the frame payload. With OTAA, new keys are generated on each session. OTAA is more reliable and less vulnerable against attacks but requires downlink data flow for the join-procedure.

OTAA does not use a fixed security session like ABP. This means that the keys (DevAddr, NwkSKey and AppSKey) are not hard-coded on the device, in contrary with ABP, but derived by the network server with the DevEUI, AppEUI and the AppKey. The join-procedure will end with a downlink join-accept sent by the network server which confirms the authentification and resets the frame counter. All this makes the switching to another network effortless. Figure 2.8 illustrates the join-procedure.

OTAA provides a basic security and enables a flexibel design.

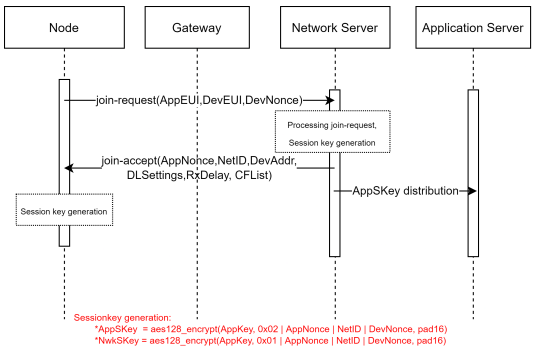

Figure 2.8: Overview of the LoRaWAN 1.0.2 OTAA join-procedure

Methodology

During this Bachelor Proof, we went through several stages, as illustrated on fig. 3.1. Developing our IoT solution did not always went that smoothly. We sometimes had to go several steps back in order to fix unexpected problems showing up before diving in the further stages.

We first had to learn a lot of new concepts and tools before developping our network. Starting at **stage 0** we first did a lot of research about the LoRa & LoRaWAN protocols and learned how to use the Linux operating system. In **stage 1** we first tried to set up the Network Server locally but we finally turned to a cloud-based solution with TTN for ease of deployment, it took quite a long time before having our first gateway deployed in the network (**stage 2**). Rapidly after this we had our first node deployed, initially activated with ABP, last with OTAA (**stage 3**).

Moving into **stage 4** we had set up our multi-channel gateway and discovered the "LoRa: Orthogonality" with our second node deployed in the network (**stage 5**). We were able to send on the same channel at different SF's at the same time! We used Pycom's libraries, which are well documented but hide a lot of surprises...

In the last stages we optimized the node software and visualized it locally (**stage 6&7**). Finally we did some experminents testing our network out (**stage 8**).

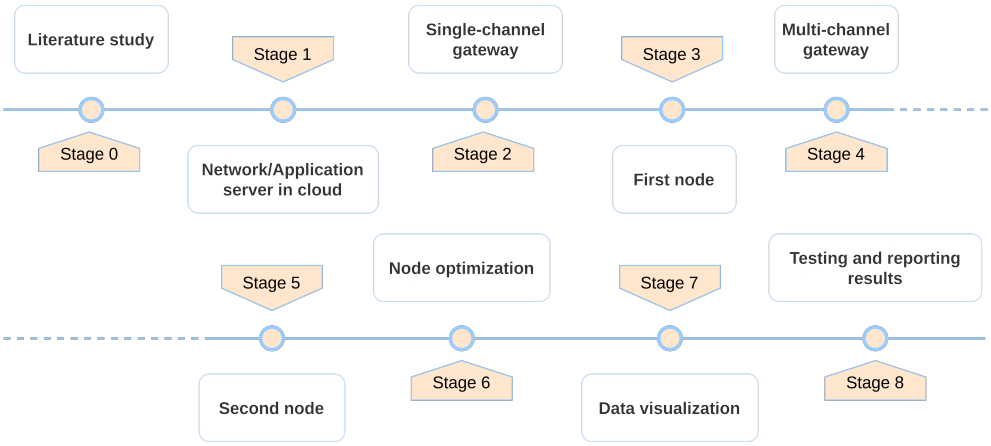

Figure 3.1: Bachelor Proof time-line

3.1 Network architecture

The installation of a monitoring system includes both hardware and software components. In this chapter we will define each of the components which are constituting our network. We ran several experiments to find the best practical setup for improving the quality of the LoRa signal and increasing the packet delivery ratio.

Our nodes and gateways are powered by a FiPy and GPy board. We will go into greater detail about the hardware in Louka Grignard's bachelor thesis. The nodes use LoRaWAN packets to send data from the sensor at regular intervals to our multi-channel gateway in a single hop. By enabling Adaptive Data Rate, we allow the network server to optimize the network's data rates, airtime, and energy consumption by adjusting the SF, BW, and transmission power. The LoRaWAN network is a star-of-stars network, with the gateway serving as the nodes' central hub for packet forwarding to the cloud. These packets are forwarded to the TTN cloud server via UDP. TTN's network and application server will then identify and decode those. Finally, using docker containers hosted on a central computer, useful data will be gathered, processed, and delivered in a user interface.

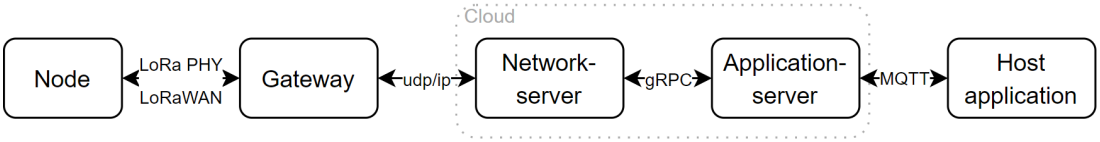

Figure 3.2: Network architecture and data-flow of our setup

Figure 3.3: Main components of our network

The following sections will deconstruct the operation of each component.

3.2 Node

We have our nodes at the start of our IoT-chain. The nodes are in charge of acoustically sensing the environment, computing the data, and transmitting it via LoRa.

Our microphone module will sense sound. With code loaded on the flash memory, the ESP32 microcontroller-based board will compute the frequency spectrum and sound level out of the sensor readings and send the processed data over LoRa to our gateway.

3.2.1 Code description

At the node level, the primary task was to write software. The code is written in microPython and makes use of the Pycom libraries to access the protocols available on this board, primarily the network module. Our code is divided into three scripts: config, NoiseNode, and main.

3.2.1.1 Configuration

In the configuration file, we define the parameters and keys that our nodes will use. Our node's default configuration is to be activated with OTAA in a LoRaWAN network with ADR enabled, optimizing the datarate to achieve more efficient communication.

mode	region	class	activation	channel	adr
LoRaWAN	Europe	A	OTAA	0 (868.1 MHz)	active

Table 3.1: Default LoRa configuration of our nodes

3.2.1.2 NoiseNode

The NoiseNode script will be our main library. On fig. 3.4 we define our node as a class with useful properties and methods which we will be using in our main script.

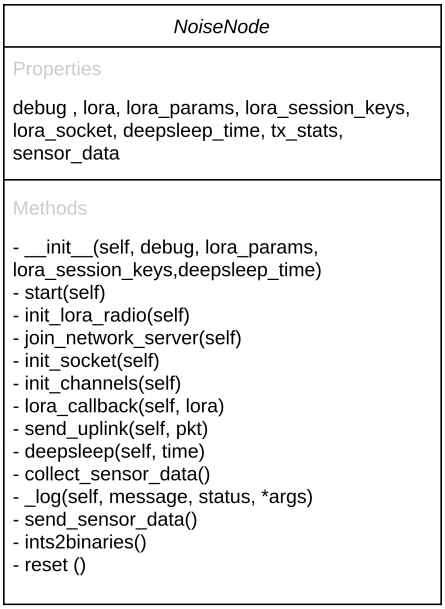

Figure 3.4: NoiseNode class overview

3.2.1.3 Main

The main script will define our nodes' life-cycle, as shown in fig. 3.5, where each block corresponds to a method.

First, a "Noise Node" object will be created and configured with the pre-defined settings. LoRa parameters include mode, region, class, activation-mode, frequency, channels, and data rate. Deep sleep time and packet confirmation will be set as well. Once the node has been configured, the LoRa socket is **initialized**, and a LoRa-callback function handles LoRa transmission events such as downlink reception, uplink transmission, and transmission failure.

Following that, our node will attempt to **join** the network server via join-requests using the pre-defined activation mode and authentication keys. This process can be time-consuming; saving the connection in the nvram will be useful after waking up from deep sleep mode.

The analog-micro will output a value that will be **collected** from an ADC-pin, and this value will be used to compute the sound-level and spectrum using a fft-algorithm. Finally, the LoRa-socket will attempt to **send** this data packed in a LoRaWAN packet via the connected antenna. It will send the packet a predetermined number of times until it receives a downlink confirmation. The node will then save the join-session to the NVRAM and **enter deep sleep**. The entire procedure can be repeated.

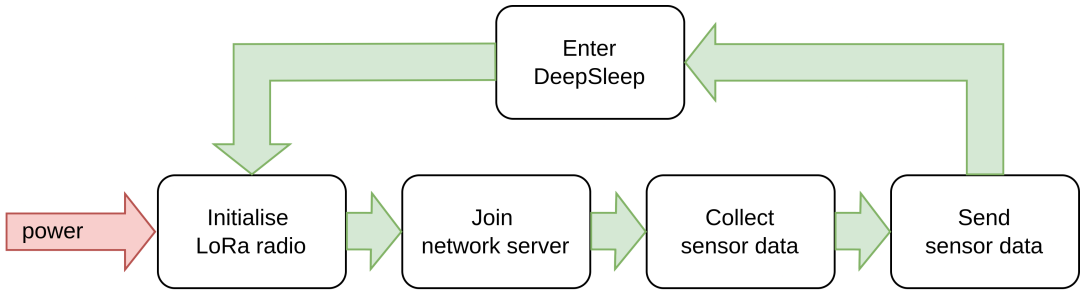

Figure 3.5: NoiseNode life-cycle

3.3 Gateway

The gateway is the next link in our chain. Its job is to route LoRaWAN packets to the network server (TTN). The gateway and our node share many similarities, such as the ability to send and receive packets through LoRa. It, on the other hand, has a WiFi connection and remains active to listen for incoming LoRaWAN packets and UDP threads.

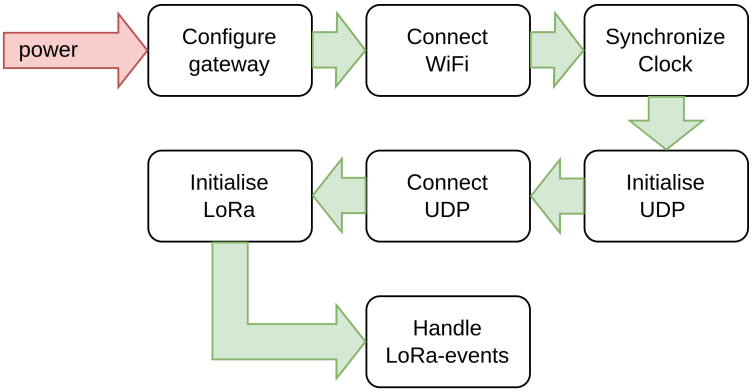

Figure 3.6: Gateway life-cycle

2 types of gateways were developed, one single- and one multi channel gateway. The multi-channel is the gateway in use for this project.

The software and configuration for our multi-channel gateway are provided by Pycom and TTN. The main task is to forward LoRaWAN packets and to send the gateway status to the network server on a regular basis via UDP. Forwarding occurs by listening to LoRa communication on all pre-configured channels and spreading factors. In addition, if the device was activated via OTAA, the gateway could indeed send packets for join-accepts and confirmed-uplinks.

Figure 3.6 shows the life-cycle of our single-channel gateway; the multi-channel gateway code is based on this but listens on all configured channels and SF's. This type of gateway could run on the same board as our node, which would require less expensive hardware but would limit network density and data rate optimization.

3.4 Network and application server

The Things Network is positioned between our gateway and local application. It handles the routing and processing of the LoRaWAN packets before sending it to our final application. Figure 3.7 we can see how the TTN cloud server is composed.

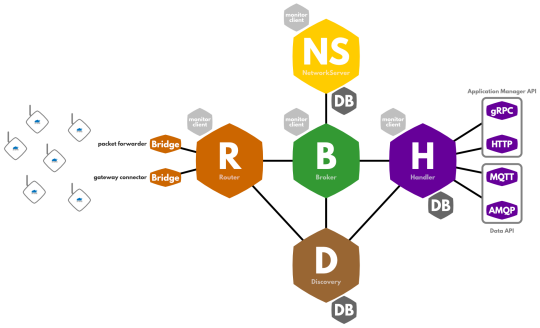

Figure 3.7: TTN network architecture

For ease of use we decided to opt for the TTN Community edition instead of an entire private network like Chirpstack or The Things Stack. This supports the required LoRaWAN version (1.0.2) defined by the LoPy board and allows us 30 seconds uplink time per day. Data routing can easily be monitored after configuration of the app and devices. Figure 3.8 shows all the different types of messages our node handles: join-request, join-accept, uplink and downlink. On fig. 3.9 we observe the live activity of our gateway.

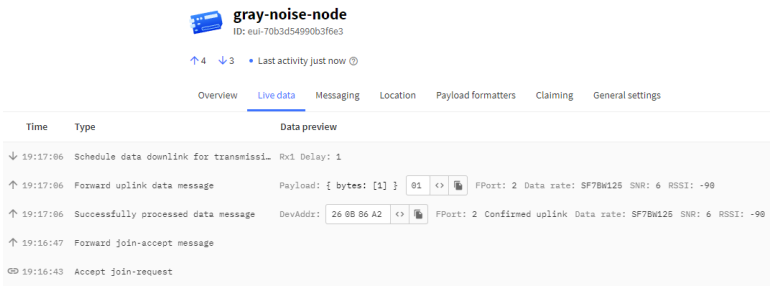

Figure 3.8: Live activity of our node on TTN

Figure 3.9: Live activity of our gateway on TTN

3.5 Local application

Representation of the data will be done on our local Linux system. We use the open source tool TIG-stack containerized with docker for rapid and easy deployment. TIG stands for Telegraf, InfluxDB and Grafana and will be used for collection, storage and graphing of our time stamped metrics.

Telegraf will collect the metrics from the cloud-server through the TTN's MQTT broker and push the metrics via HTTP in our InfluxDB database. Both tools provide a quick integration for using MQTT.

InfluxDB provides an SQL-like language handling the HTTP requests firing up from grafana. Finally our Grafana reports the metrics analytics pulled from InfluxDB.

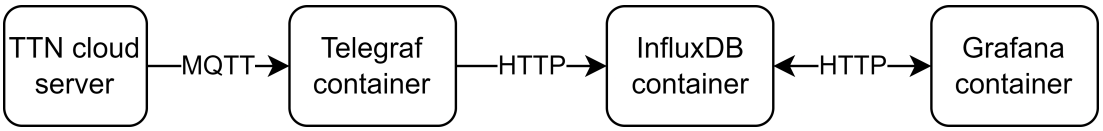

Figure 3.10: TIG stack data-flow

Figure 3.11 shows our Grafana-dashboard. This dashboard is composed by two main blocks: a level-meter and a point graph. In this way we can monitor the node's environment sound-level and visualize relationships in the data.

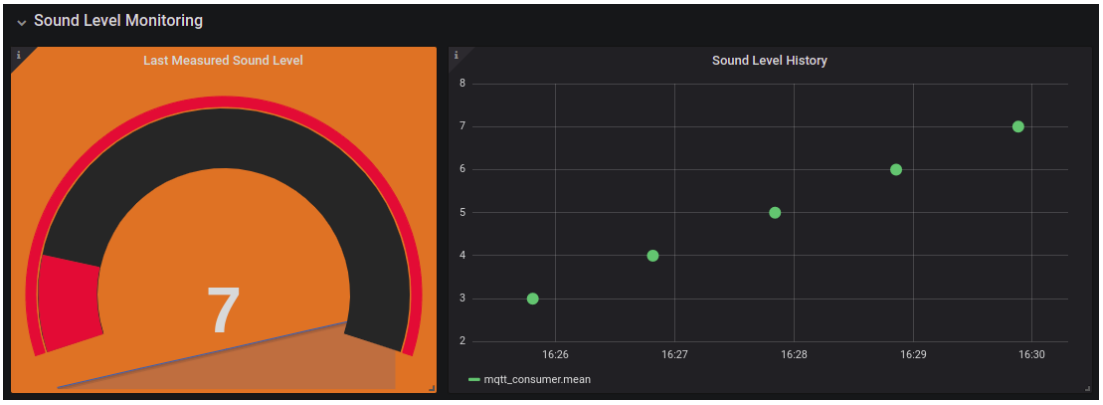

Figure 3.11: Sound Level Monitoring of our node in Grafana

In the following sections, we will analyze and test the performance of our system. The LoRa signal, network latency, communication range, and power consumption will all be evaluated.

4.1 Signal analysis with Software Defined Radio

Analyzing the LoRa RF can be done with the aid of GNU-radio and a low-cost RTL-SDR dongle. Its usefull to use this kind of software to check the signal frequency-spectrum live and see when our devices really emits.

On fig. 4.1 we can observe a RF spectrum analysis of the LoRa modulation accompanied by the noise our antenna received. With this we can deduce the channel and bandwith on which we were emitting. In this case we were emitting on channel 0 (868.1 MHz) with a bandwidth of 125 kHz. The observations are coherent to our node configuration who was set at this particular time on the same channel with SF of 7 and a bandwidth of 125 kHz.

On fig. 4.2 we traced a confirmed uplink message. We can notice three distinct lines. The first two traces represent the transmission attempts, the third one is the downlink confirmation which confirms our three seconds rx delay.

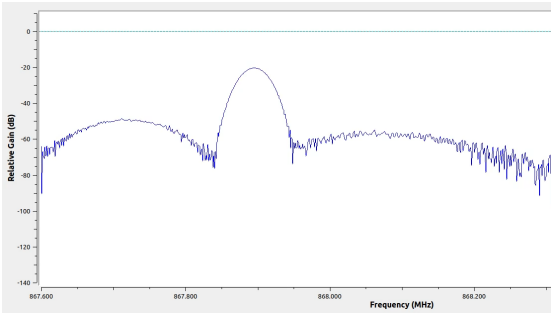

Figure 4.1: LoRa Signal Analysis with Software Defined Radio , GNU-radio

Following code-snippet defines the function we used to analyze the signal at varying SF's.

```
def SpreadFactorTest(self):
    for i in range(0,6):
        self.datarate = i
        self.send_uplink(self.array2packet(list(range(20))))
        time.sleep(10)
```

The signal and network latency are showed in our video, url: https://youtu.be/cZKYumqe4U8

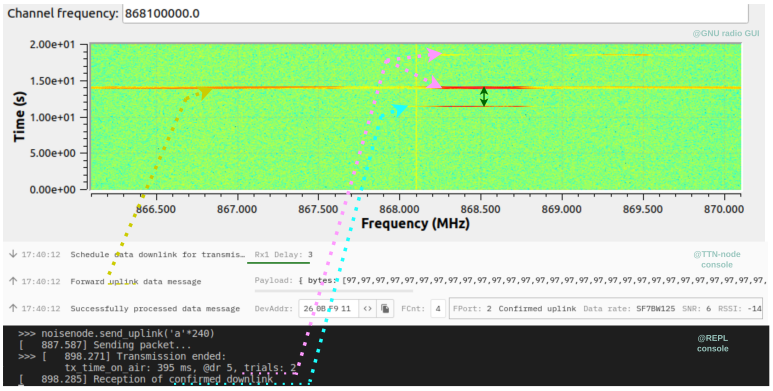

Figure 4.2: Confirmed uplink message tracing with GNU Radio, REPL- and TTN console

4.2 Range

We ran a range test in Ronquières, testing all the spreading factors at various points along the Brussels-Charleroi Canal, a large area with a long line of sight that reduces the possibility of interference. The map and measurments are shown in appendix A.

When sending with a higher SF, we observed a higher packet delivery ratio in locations further away from the gateway. We were able to send packets up to 3100 meters away at SF10-12.

Figure 4.3: Testing our node's range at Plan Incliné de Ronquières

4.3 Node power consumption

We conducted a power consumption analysis, mapping the different processes running on our node with timestamps on fig. 4.4. This was measured with the Nordic Semiconductor's Power Profiler Kit II connected to the FiPy board at 3,7V.

In section 3.2.1, fig. 3.5 describes the set of action our node is executing: *boot, initialization, joining, collecting, sending and deepsleep.*

First, it is powered up, and our node enters **cycle 0** beginning with a boot flagged as *POWER_ON_RESET* lasting ±2 seconds drawing 25.7mA on average. Shortly after this the LoRa radio is initialized and the join-request is sent with SF12BW125 (1.5 seconds with peaks at 213mA), which open the receiving slot (rx slot) after 5 seconds delay. The node takes sensor readings in a very short period of time and sends them at SF12BW125 (138mA, 2.4s). This is followed by a pre-defined receiving delay (rx delay) of one second before the next rx slot is opened (43.3mA) for transmission confirmation. Once the confirmed-downlink is received, it immediately enters deepsleep at an average of 320µA.

The following cycles are significantly shorter as a result of skipping the join-procedure by restoring the settings and join-session from the NVRAM and adjusting the data rate with ADR to SF7BW12. Cycle 1 takes 3.46s at 37.9mA and can vary depending on the SF specified in the ADR acknowledgement.

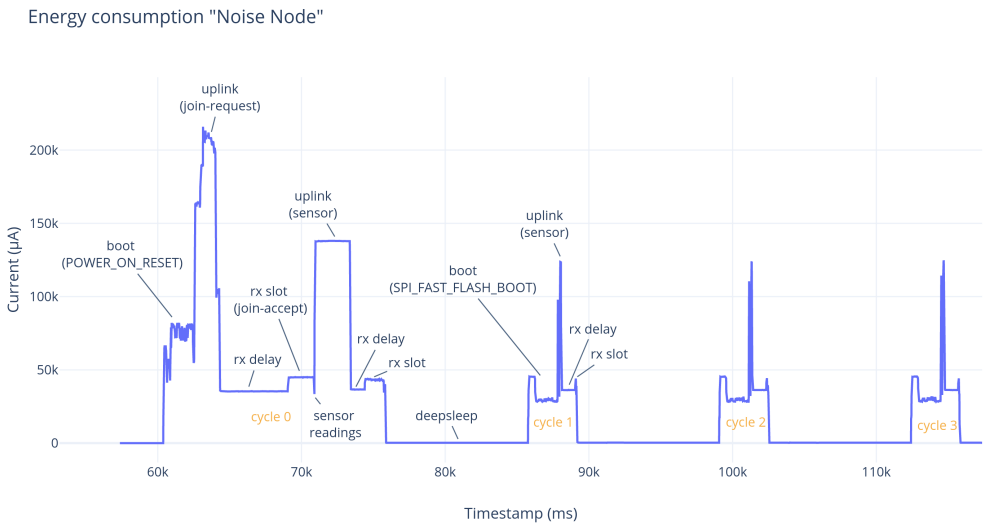

Figure 4.4: NoiseNode Power consumption

LoRaWAN. While the LoRa network provides good coverage while consuming little power, the achievable data rates in LoRa can be limited for smart city applications that require continuous streaming, frequent data collection, or large model updates.

Pycom boards enabled for rapid prototyping but are limited in their application; employing a top-level language like microPython hides a lot of underlying code... When using their libraries, various difficulties were discovered, such as the inability to define the power of transmission and the coding rate in LoRaWAN mode.

TTN allows users to use their cloud LoRaWAN stack, which has a number of benefits, including not needing to operate the server on a local system. But we are limited in our ability to employ full LoRaWAN features, duty cycle, and disposable channels. We'd have to pay for a membership to access all of LoRaWAN's features.

GNU radio has been quite helpful in debugging the LoRa transmission. This allowed us to see what was really going on and estimate the duration of the communications. We discovered that our device was not logging all of the transmissions that were taking place. With this, we were able to trace the join-procedure and transmission-retries.

Range test. We noticed that increasing the SF significantly increased the reception rate, especially at points further away from the gateway. We expected a greater range, but we could only receive packets from 3.1km away. This could undoubtedly be improved if we could set the transmission power to its maximum (14dBm) and use larger antennas.

Power-profile. The power-profile of our nodes gave us a lot of insight into what was going on and allowed us to map the processes in time precisely. It also revealed that the board's power consumption deviates significantly from the values specified in the datasheet. This adds another dimension to the device's comprehension, giving us more perspectives than the console logs do.

Chapter 6

Conclusion

This bachelor's thesis looked into the deployment of a LoRaWAN network in response to the need for wireless sensor systems that monitor environmental noise pollution. We developed a solution towards the design of an LPWAN for acoustic monitoring. By developing an architecture that makes use of LoRa and LoRaWAN for end-device networking. We simplified the design by using TTN services for the network and Pycom boards for the hardware. We developed the node software, making the most of our nodes, and configured the cloud-server and our local application to monitor the metrics in real time. With this research we contributed the development of future smart cities.

In the future, Louka Grignard will develop the micro-module that will allow us to finally test our network with real data.

6.1 Further work

Recommendations for future work on the topic:

- Deployment of LoRaWAN Network Server stack (Chirpstack) on a private server.
- Edge computing on the gateways/nodes to improve response times and saving bandwidth.
  - On-edge Environmental Sound Classification to reduce power-consumption and privacy issues.
  - Big Data Analytics, to provide smart decisions based on collected data for enhancing smart city services.

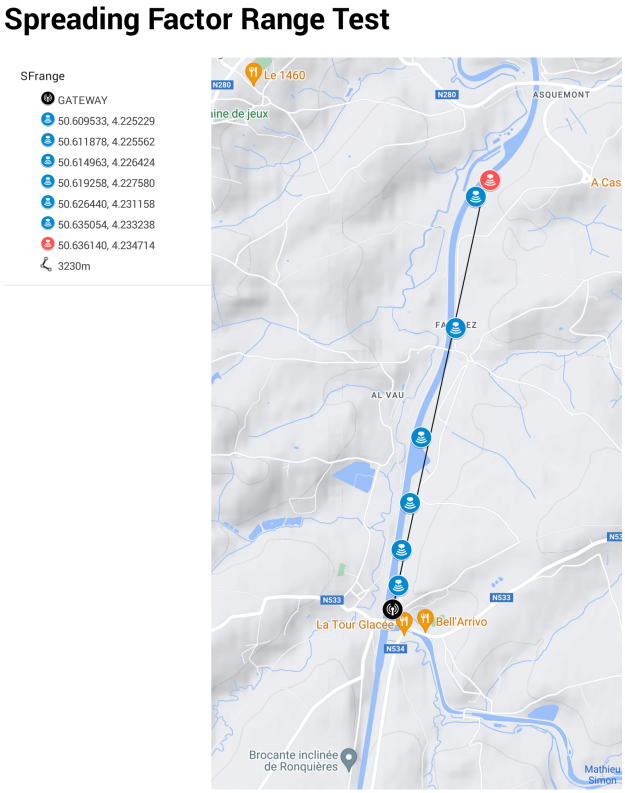

Figure A.1: Map illustrating 7 measurement points testing out SF7-12 on different distances

distance (m)	SF	RSSI (dB)	SNR (dB)
180	7-12	-80	10.5
440	7-12	-85	11.5
790	7-12	-95	9.1
1280	7-12	-108	-8.5
2100	7-12	-109	-7
3100	10-12	-109	-15.5
3200	/	/	/

Table A.1: Measurements reported on TTN

Code

Our github repository contains the complete code as well as more detailed documentation for the node, gateway and local application.

https://github.com/valigatotuS/IoT-For-Sound-Pollution

Below a short code-snippet, describing our smart accoustic-meters routine.

```
""" Main code for a lora node tested on TTN with LoRaWAN, OTAA & ADR """

import config, time
from NoiseNode import NoiseNode

if __name__ == '__main__':
    noisenode = NoiseNode(
        debug             = config.DEBUG,
        lora_params       = config.LORA_PARAMETERS,
        lora_session_keys = config.LORA_SESSION_KEYS,
        deepsleep_time    = config.DEEPSLEEP_TIME,
        )

    # starting the LoRaWAN Noise Node
    noisenode._log("Starting Noise Node with id: %s" % noisenode
        .lora_session_keys['dev_eui'])
    noisenode.init_lora_radio()
    noisenode.join_network_server()

    if noisenode.debug:
        # Entering RPL (interactive MicroPython prompt)
        input('Press ENTER to enter the REPL')

    else:
        # Sending noise level on regular interval via LoRa
        noisenode.collect_sensor_data() # analog reads
        noisenode.send_sensor_data()    # sending then deepsleep
```

.

Listing B.1: main.py code, describing the accoustic-meters routine

Bibliography

[1] E. Aras et al. 'Exploring the Security Vulnerabilities of LoRa'. In: June 2017, pp. 1–6.

[2] F. Y. Aznaveh and M. Mansub Bassiri. 'Evaluation of using LoRaWAN to implement AMI in big city of Tehran'. In: *2019 3rd International Conference on Internet of Things and Applications (IoT)*. 2019, pp. 1–4.

[3] *Decodinglora*. URL: https://revspace.nl/DecodingLora.

[4] A. Fox. *What Is A Good Signal-To-Noise Ratio For A Microphone?* URL: https://mynewmicrophone.com/what-is-a-good-signal-to-noise-ratio-for-a-microphone/ (visited on 04/26/2022).

[5] A. Fox. *What Is Microphone Sensitivity? An In-Depth Description*. URL: https://mynewmicrophone.com/microphone-sensitivity/ (visited on 04/26/2022).

[6] C. Ge. *HOW TO BIAS AN OP-AMP*. 2016. URL: http://fab.cba.mit.edu/classes/863.16/section.Architecture/people/Ge/bias_opamp.pdf (visited on 04/30/2022).

[7] K. L. Lueth. *The top 10 IOT segments in 2018 – based on 1,600 real IOT projects*. Apr. 2022. URL: https://iot-analytics.com/top-10-iot-segments-2018-real-iot-projects/.

[8] K. Mekki et al. 'A comparative study of LPWAN technologies for large-scale IoT deployment'. In: 5 (Mar. 2019), pp. 1–7.

[9] J. M. B. Morillas et al. 'Noise pollution and urban planning'. In: *Current Pollution Reports* 4.3 (2018), pp. 208–219.

[10] J. Petajajarvi et al. 'Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage'. In: *International Journal of Distributed Sensor Networks* Vol. 13 (Mar. 2017), pp. 1–16.

[11] K. Routh and T. Pal. 'A survey on technological, business and societal aspects of Internet of Things by Q3, 2017'. In: *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*. 2018, pp. 1–4.

[12] J. Scarpati. *Radio Frequency*. Apr. 2022. URL: https://en.wikipedia.org/wiki/Radio_frequency.

[13] E. Sisinni et al. 'LoRaWAN Range Extender for Industrial IoT'. In: *IEEE Transactions on Industrial Informatics* 16.8 (2020), pp. 5607–5616.

[14] Unknown. *IoT Tutorials*. 2018. URL: https://data-flair.training/blogs/iot-hardware/ (visited on 04/30/2022).

[15] P. Wegner. *Global IOT market size grew 22% in 2021 - these 16 factors affect the growth trajectory to 2027*. Mar. 2022. URL: https://iot-analytics.com/iot-market-size/#:~:text=At%5C%20this%5C%20point%5C%2C%5C%20IoT%5C%20Analytics,lowered%5C%20from%5C%20the%5C%20previous%5C%20year..

[16] E. P. Yadav, E. A. Mittal, and H. Yadav. 'IoT: Challenges and Issues in Indian Perspective'. In: *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*. 2018, pp. 1–5.