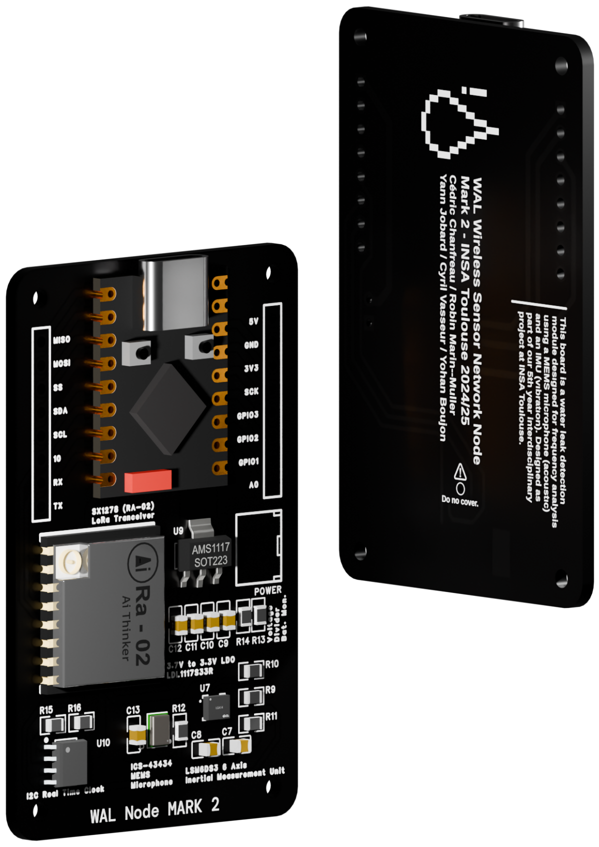
Smart Water Leak Detection in Water Networks

ISS Project Report

Yohan Boujon, Cédric Chanfreau, Yann Jobard, Robin Marin–Muller, Cyril Vasseur


WAL Node MARK 2
SX1278 (RA-02) LoRa Transceiver
AMS1117 SOT223
POWER
ICS-43434 MEMS Microphone
LSM6DS3 6 Axis Inertial Measurement Unit
I2C Real Time Clock
WAL Wireless Sensor Network Node
Mark 2 - INSA Toulouse 2024/25
Cédric Chanfreau / Robin Marin--Muller
Yann Jobard / Cyril Vasseur / Yohan Boujon
Do no cover.

Institut National des Sciences Appliquées de Toulouse

January 28, 2025

Contents

Introduction

1.1 Context

In France water leaks result in 1/5th (20%) of the water transported on the water distribution network being waster.[1] Water and especially drinking water is a precious ressource that needs to be handled as such. Current water network infrastrucutres include minimal to no way to monitor its strutural integrity, this problem have to be adressed to ensure optimal and leak free water distribution.

This issue is made worse by the fact that much of the water network in France is old and built with materials prone to leaks, such as cast iron and steel. Smaller towns and rural areas often face even greater challenges, as they have longer pipelines and fewer resources to fix or replace them. Without proper upgrades and monitoring systems, these problems will continue to grow, making it harder to ensure a reliable and efficient water supply.

1.2 Problem Statement

Existing methods for detecting and addressing water leaks such as manual inspections, pressure tests, and acoustic sensors, are often intrusive, very costly, and inefficient in certain environments. These limitation can and must be adressed by a modern, innovative, non-intrusive, and cost effective approach to a sustainable water network infrastructure.

1.3 Project Objectives

The goal of this project is to bring an innovative solution for detecting water leaks on public water network infrastructure. Our solution leverage cutting edge sensors, complex distributed wireless sensor network and advanced analysis algorithms to tackle this important challenge.

Our objective is not to provide a as-is working solution for direct industrial deployment but a Proof-of-Concept. While being mainly a research project, we kept the potential industrial future in mind during all the design steps.

1.4 Report Structure

This document will explain in a detailed and technical manner the solution we propose and the choice we made along the way. We will first begin by a state of the art on the different ways to address our water leak detection problem. This section will help us situate our approach in the water detection system landscape and propose an innovative solution. We will then continue by explaining in great details our solution while trying to be as technical and rigorous as possible.

1.5 Methodology

This work was done with the agile method. This means that the software development activities and the hardware follow an Agile process and are divided into work iterations.

- An iteration is called a sprint and lasts a minimum of 4 weeks.
- An epic is a body of work that can be carried out over several sprints.
- Each epic is divided into user stories.
- A user story is a set of tasks that can be carried out over the course of a sprint.
- User stories are planned for a sprint during sprint planning.
- User stories are managed in a backlog.

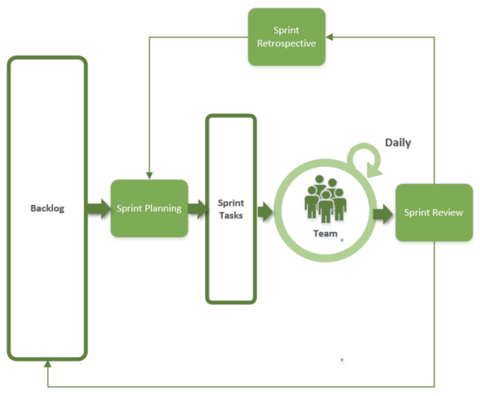


Figure 1.1: Agile Method

To use this method efficiently we used Plane an open website to schedule task with agile status.

State of the Art

2.1 Traditional Leak Detection Methods

Traditional methods for detecting leaks in water distribution systems have evolved over decades, focusing on manual and sensor-based approaches. These techniques, while useful, often face limitations in accuracy, cost, and practicality.

2.1.1 Acoustic Techniques

The oldest and most widely used methods for identifying leaks in water pipelines are probably acoustic techniques. Theses methods relies on detecting– most of the time by ear– the noise generated by water leaks which typically include a specific hissing "shhhh" sound caused by water escaping pipes under pressure.

The effectiveness of acoustic detection techniques greatly depends on three main factors: the pipe material, the size of the leak and the surrounding noise levels. For example, metal based pipes like cast iron transmit sound significantly more effectively than plastic pipes making it easier[2] to detect potential leaks.

Another significant limitation is the necessity to rely on skilled technician to manually collect and interpret acoustic signals. This narrow down greatly the scalability of the solution and point out the impossibility to apply aforesaid techniques for such a complex and wide issue.

2.1.2 Invasive Methods and Their Drawbacks

Invasive methods involve directly accessing the pipeline to inspect or monitor its condition.

2.2 Modern Technologies

Recent advancements in sensors technology (MEMS - Micro Electro-Mechanical Systems) machine learning and intelligent algorithms enable modern leak detection solutions. Theses solution offers significant improvements in accuracy, scalability and cost-effectiveness over traditional methods. These modern approaches are far more resilient to external disturbances and make them ideal for harsh environments.

2.2.1 Acoustic and Vibration Sensors

Modern solutions extensively uses MEMS acoustic and vibration sensors, they provide very high sensivity, adaptibility and can be easily integrated in modern Wireless Sensor Networks. "Acoustic techniques have proven to be effective and are widely used to locate leaks in water distribution pipes in many years".[3]

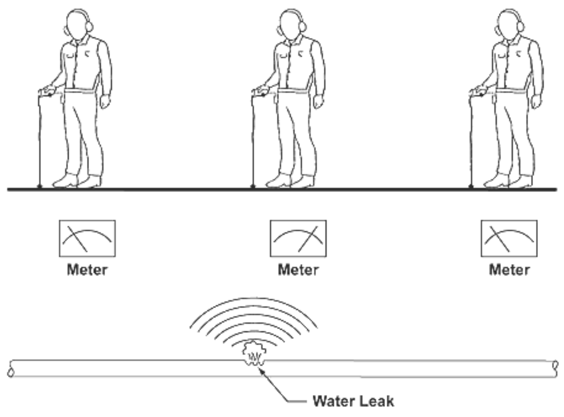


Figure 2.1: Pinpointing Acoustic Leark Detection Method[4]

High signal-to-noise ratio accelerometers can capture vibration frequencies caused by leaks, even in plastic pipes where attenuation rates are higher. Marmarokopos et al.[5] demonstrated that these sensors enable leak detection with greater precision when placed near potential leak points.

Wavelet transforms applied to acoustic signals improve leak signature detection in noisy environments. This technique excels in separating leak-related frequencies from background noise, as shown by Ahadi and Bakhtiar[6]

Vibration sensors integrated with wireless networks allow continuous monitoring of water pipelines with minimal energy consumption. Virk et al. (2020) highlighted the effectiveness of low-power sensors in detecting leaks and minimizing water wastage.[7]

Hydrophones for Low-Frequency Waves: In buried pipelines, hydrophones detect low-frequency acoustic waves more effectively than accelerometers, especially in low signal-to-noise ratio environments. Gao and Liu (2017) demonstrated that hydrophones outperform accelerometers in such conditions (Gao et Liu, 2017).[3]

2.2.2 Artificial Intelligence (AI) Applications

AI technologies have revolutionized leak detection by automating data analysis and improving classification accuracy.

Machine Learning Models: Tools like XGBoost and Support Vector Machines (SVM) are being used to analyze and classify vibration and acoustic data to detect leaks. For example Lee and Kim (2023)[8] achieved 99.79% accuracy using XGBoost on vibration data.

Deep Learning with CNNs: Convolutional Neural Networks (CNNs) are used to handle complex vibration data. Choi and Im (2023)[9] found that CNN-based models outperformed traditional SVMs.

Predictive Maintenance: AI systems don't just detect leaks they can predict them before they happen. Gong et al. (2020)[10] developed algorithms to spot cracks and leaks even in noisy urban environments.

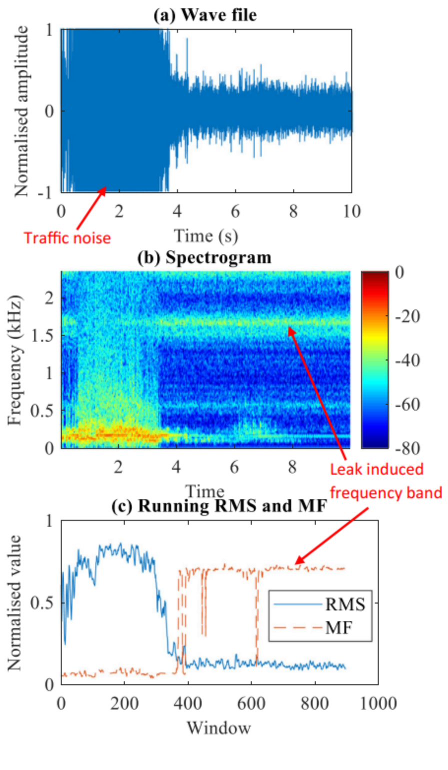


Figure 2.2: Spectral and RMS Analysis of Leak Scenario[10]

Edge Computing: With machine learning on the edge, data can be processed on the spot, this reduces latency and allows large scale monitoring without requiring extensive data transmission.[11]

2.3 Summary

2.3.1 Identified Gaps

Traditional leak detection methods like acoustic techniques or invasive inspections are limited and not scalable. They depend on skilled techninians, very high costs and limited accuracy espcially in complex and difficult to access environments. On the other hand, while modern approaches leverage advanced techniques like highly sensitive sensors and complex detection/classification algorithms they lack real world large scale implementations and associated issues like power consumption, WSNs implementation or communication. Moreover, the lack of efficient all-in-one cost-effective solution limits the large-scale adoption of these technologies. These points highlight the need for an innovative, low-power, non-intrusive, and scalable system that integrates advanced sensing, machine learning, and energy-efficient communication protocols.

2.3.2 Justification for Proposed Approaches

The proposed approach bridge the limitations of existing solutions by combining modern MEMS acoustic and vibration sensors, advanced spectral analysis and machine learning classification techniques into in all-in-one hands-on solution. Additionally, a user-friendly interface ensures better system monitoring and large-scale adoption.

System Design and Architecture

In this section we will describe our system design process from the overall system architecture to the mechanical design passing by the PCB design. We will try to explain our choices in a concise but technical manner.

3.1 Principles

Our idea is based on the fact that water leak emits specific and potentially differentiable spectral signatures. Theses signatures may take different shapes, whether they come from different vibrations sources (acoustic or mechanical) our goal is to detect and classify them.

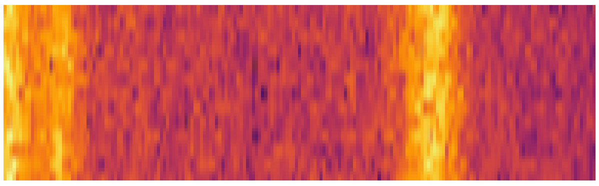
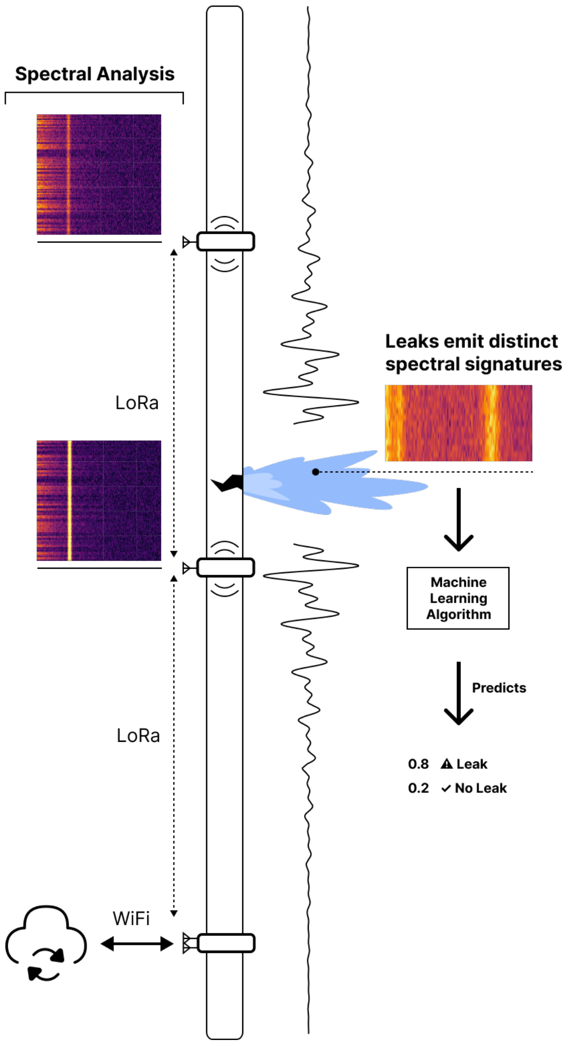


Figure 3.1: 8kHz Typical Leak Spectral Signature

3.1.1 Leak Detection Principle

Leak localization in a pipeline is achieved by analyzing the amplitude of the leak's acoustic or vibration signal detected at two sensor nodes, A and B, positioned on either side of the suspected leak. The method relies on the principle of signal attenuation, where the amplitude of the leak's spectral signature decreases as it travels through the pipe.


Spectral Analysis
LoRa
Leaks emit distinct spectral signatures
Machine Learning Algorithm
Predicts
0.8 Leak
0.2 No Leak
LoRa
WiFi
Figure 3.2: Leak Detection Principle

3.1.2 Leak Localization Principle

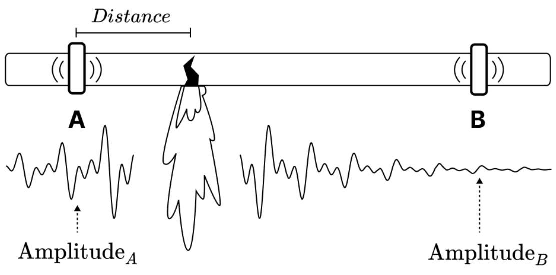


Figure 3.3: Leak Localization Principle

As explained earlier, when a leak occurs, it generates acoustic signals that propagate in both directions along the pipe, the amplitude of these signals, Amplitude$_A$ at Node A and Amplitude$_B$ at Node B, depends on the proximity of the leak to each node: the closer node detects a higher amplitude due to lower attenuation.

Using the attenuation model, the distance from the Node A to the leak is given by:

$$Distance = \frac{log(\text{Amplitude}_A/\text{Amplitude}_B)}{log(\alpha)}$$

Where:

- Amplitude_A and Amplitude_B are the measured amplitudes at Nodes A and B.
- α is the attenuation factor which depend on multiple factors like pipe mateial or leak ferquency.

As we know precisely the position of each node on the network (from the installation) we can deduce very precisely the location of the leak.

3.2 Overall System Architecture

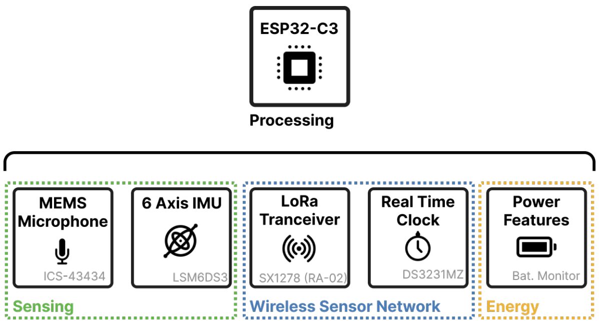


Figure 3.4: System Description Diagram

3.2.1 Architecture Description

Illustrates the system's overall design, from data collection to leak detection.

3.3 Prototyping Pipe Network

To test our model, we had to find a way of emulating leaks from a controlled model. Two functionalities had to be present on the model, the first being the leak simulator with different types of leak and the second a classic case simulator to emulate the daily use of water consumption.

3.3.1 Physical Model Overview

So we basically have this schema done :

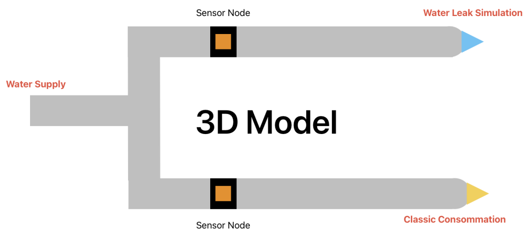


Figure 3.5: Representation of the Physical Model

3.3.2 Physical Model Prototype

The final version of our prototype is :

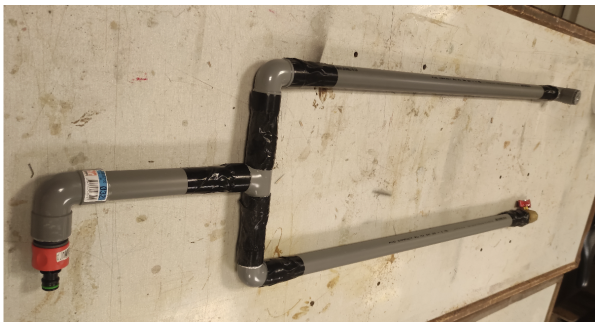


Figure 3.6: Prototype of Physical Model

3.3.3 Challenges

Producing the model was a real challenge. In fact, we ran into problems right from the start of the design because none of the team had ever done piping before. As a result, the final pipe had many uncontrolled leaks, and in the end it didn't work, so we had to find another solution.

3.4 PCB Design

3.4.1 Choice of EasyEDA

While being fairly experienced with classical CAD softwares like Altium Designer or KiCAD we made the thoughtful choice to go for EasyEDA for its extensive ready-to-use components library. Furthermore, we knew we would manufacture our PCBs using JCLPCB, EasyEDA is conveniently integrated to it. All theses choices arise from the tight time constraints and objectives to be achieved quickly.
All designs are accessible here.

3.4.2 Mark 1 Circuit Design

As we designed Mark 1 very early in the project, we didn't fully understand the project's scope. To accelerate the design process, we anticipated the needs and included untested components and multiple parts for the same function. For example, we integrated two different microphone technologies (analog and MEMS) to ensure at least one would work effectively.

Designing the boards was very challenging, as we were running short on time we had to make very quick decision about the component choices and decided to skip to the majority of prototyping phases to directly jump to the PCB production phase. This decision paid off, and was highly rewarding: due to careful design and technical considerations the soldered PCB came fully working. All the components (IMU, Microphone, Microcontroller, Analog Microphone, LoRa Module) were functionning as expected. And we could imediately start working on the implementation of the features.

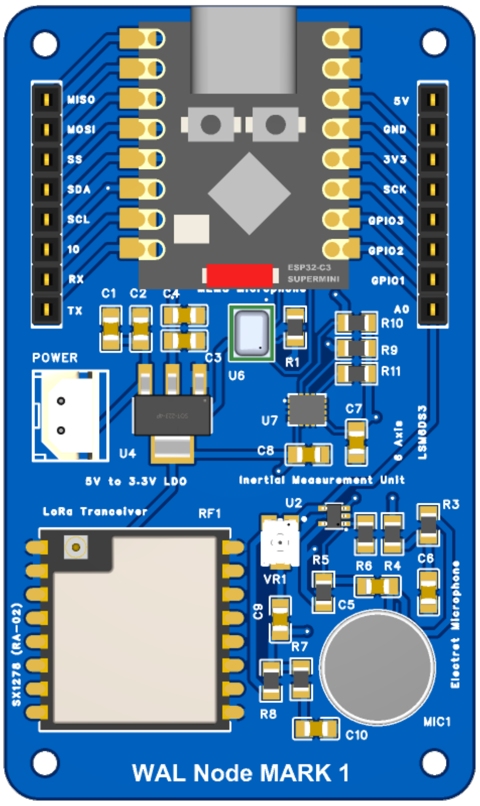


Figure 3.7: Mark 1 PCB Render

3.4.3 Mark 2 Circuit Design

Once we had worked with Mark 1 for a while, we noted some technical improvements to be made and quickly started the design process for the improved Mark 2.

When designing Mark 2, we corrected the errors of Mark 1 and added needed functionalities as we became more experienced with the project. The MEMS microphone used in Mark 1 (INMP441) was deprecated, so we switched to the up-to-date ICS-43434 equivalent. We implemented new features like the RTC module for synchronization (once the communication protocol was defined), a battery monitoring circuit, and a better low dropout LDO to improve energy efficiency and meet the high standards we imposed on ourselves.

Improvements compared to MARK1 design:

1. **Replaced LDO:** Upgraded to a very low dropout (VLD) LDO with minimal quiescent current to optimize the conversion from 3.7V to 3.3V, improving energy efficiency and battery life.

2. **Reset Pin Integration:** Connected the reset pin to the SX1278 module, enabling more reliable control over the LoRa chip.

3. **Microphone Upgrade:** Removed the analog electret microphone as the MEMS microphone is far superior, and switched from the deprecated INMP441 to the new ICS-43434.

4. **Battery Monitoring System:** Added a system for real-time monitoring of the battery state to improve energy management.

5. **Real-Time Clock (RTC):** Added an RTC to support node synchronization.

6. **Routing Improvements:** Optimized PCB routing with techniques dynamic track width and better topology.

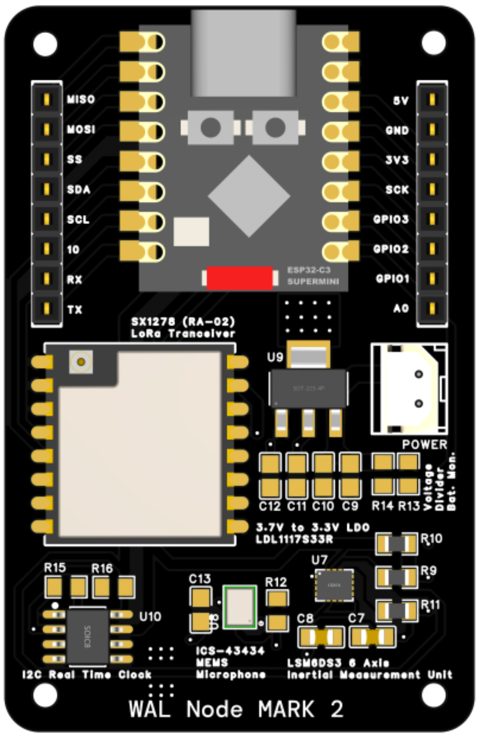


Figure 3.8: Mark 2 PCB Design

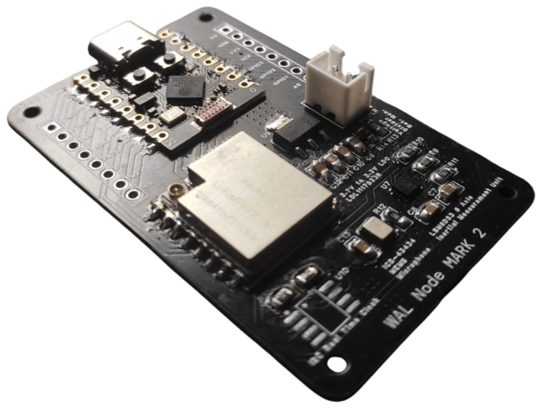


Figure 3.9: Mark 2 Assembled

3.5 Mechanical Model Design

The mechanical model design aims to create the physical housing that protects and integrates the electronic board into the water distribution network. This involves the consideration of the materials, dimensions, and assembly methods to ensure durability and adaptability.

3.5.1 Choice of SolidWorks

SolidWorks was chosen as the primary software for designing the mechanical model due to its robust features and user-friendly interface. SolidWorks provides advanced tools for 3D modeling, simulation, and analysis.

3.5.2 Board Housing Design

The sensor case was designed to house the electronic board that detects leaks in water pipelines. The design process involved creating a 3D model of the case, considering factors such as size, shape, and material. The case was designed to be durable, with appropriate openings for sensor connections and mounting points for integration into the water distribution network.

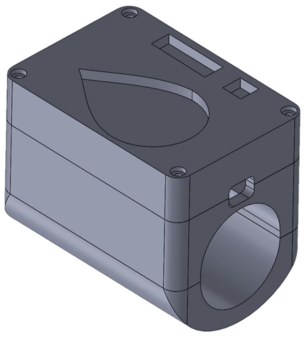


Figure 3.10: 3D Model Case

For the sound sensor, a dedicated space was created within the model to ensure direct contact with the pipeline. This direct contact is essential for accurately detecting leaks with good informations. Additionally, screw holes were added to the case to securely fix both the housing and the electronic board. This ensures that the components are firmly held in place, preventing any false vibration readings.

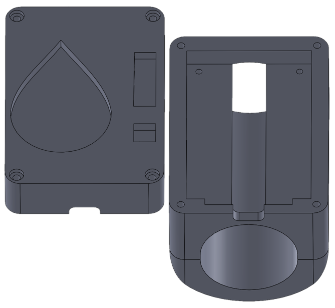


Figure 3.11: Interior 3D Model

3.5.3 Simulation and Testing

Simulation and testing were conducted to ensure the mechanical model meets the required specifications and performs reliably under various conditions. Different models of the sensor case were 3D printed and tested iteratively. Each one was evaluated to identify potential improvements, and the design was refined accordingly.

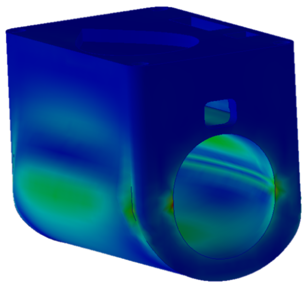


Figure 3.12: Stress Simulation

The forces exerted around the pipe are minimal, but this study is necessary to efficiently model the housing. The simulation was performed to ensure that the casing design could withstand external constraints without compromising structural integrity. The analysis focused on identifying stress concentration points and ensuring that the material could withstand these forces without excessive deformation.

3.5.4 Improvements and Future Work

Based on the simulation and testing results, several improvements were identified for the mechanical model. These include optimizing the design for better performance and enhancing the ease of assembly. Future work will focus on implementing new improvements and conducting further testing to ensure the model meets all requirements.

Additionally, we plan to develop multiple models to accommodate different pipeline sizes, ensuring versatility and adaptability in various water distribution networks. Currently, we are using PLA (Polylactic Acid) for 3D printing the sensor cases. In the future, we will explore and select an ideal plastic material that provides optimal waterproofing and durability, enhancing the overall design and protection of the electronic board. Exploring new materials and manufacturing techniques will be considered to further improve the design.

Communication Protocol Architecture

4.0.1 Motivations

The **What A Leak** project aims to optimize its performance for data sharing via multiple nodes. These nodes need to communicate within a certain range, and because they can be many and far away from each other, a specific protocol has to be designed. There are two layers in this protocol, the first one being highlighted in Sec.4.0.8 will be the data processing layer, with multiple types that can be transferred and fields specific to the project's needs. The second layer will be discussed in Sec. **??**, each nodes will be able to communicate with each other to transfer data over long range topology. These nodes need to identify the fastest path and will be helped by the main node. A cryptographic approach will be discussed in the Sub Sec. **??** to help each node secure its data to the main gateway. There are some important constraints that we first need to talk about before entering into the depth of the tailored protocol. In Sec. **??** every choices in term of timing, bandwidth, transmission power will be explained in details.

4.0.2 Node discovery and topology

Spanning tree algorithm, topology will be stored in the main gateway.

4.0.3 Addressing each Node

ESP-32 are equipped with a Wi-Fi chip, whereas it has a Wi-Fi antenna or not, its MAC Address is globally unique accross all devices. *Espressif* has the following identifier: ***18:8B:0E***, the other 6 bytes can be used as a unique identifier. This will be used in the first check to give each device a node ID that will be unique for the given topology.

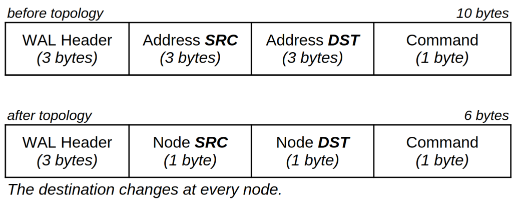


Figure 4.1: Start of the packet in two scenarios

During the topology payload each node will communicate to the main node their MAC address on 3 bytes. Then an address on 1 byte will be sent back. This is made so that each packets will be smaller afterwards. As we can see on the picture above this does not change anything on the payload, only the final size. For the device to actually know which type of address the device has to read we have to look into the *WAL header*.

4.0.4 Header

The start of each packet is strictly defined with 9 bits which are composed of "*wal*" in morse code. Then the version number on 7 bits each, for both major and minor version. If the version is not compatible or the 9 bits, which are equal to ***0xd4*** are not recognized, the packet is instantly **rejected**.

1 bit is entirely dedicated to the type of address, either on 3 bytes (MAC based address), or 1 byte (Node ID given by the gateway). It can be described as the following:

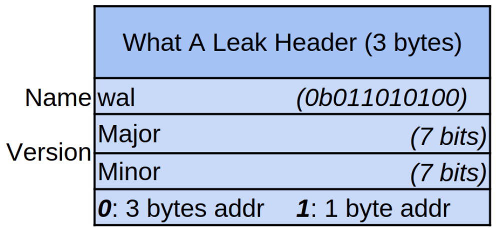


Figure 4.2: Bits of the header

4.0.5 Command payload

After checking if the packet has a correct header and if the data that is gathered is destinated to the correct device, a final check has to be done which is the command payload. This defines how the data should like like afterwards. It is entirely contained in 1 byte with the following bits:

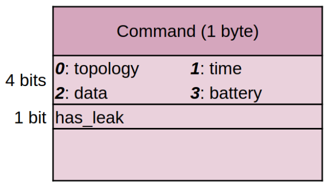


Figure 4.3: Bits of the command

As we can see there are 4 types of command that can be sent by each device for now. It is stored in 4 bits, which can go up to 15 different commands that are yet to be defined (user-based could be applied here). After these command bytes we can have various data. One bit is dedicated to the *has_leak* derivative which is a verdict sent by the node's internal AI and FFT[1] calculus.

4.0.6 Topology payload

Each node on the network need to gather a unique *Node ID* as well as its underlying RSSI[2]. There could be multiple nodes that are close to the source device: by this mean, it can send multiple node with multiple level of signal strengths. This is how the payload can be described:

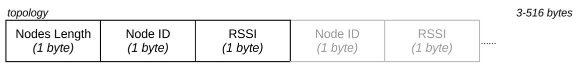


Figure 4.4: Topology Payload

4.0.7 Time payload

The internal RTC[3] of each node may not be synchronize properly. With this payload, a ping-pong can be established between the closest device possible, calculating the possible drift. After testing with 10 exchanges, the EPOCH can be given on 8 bytes. The device has to add its delay and this way synchronize completely with the main gate. It has not been tested yet and may be fully precise, however, the time difference could be so minimal that it does not interfer with the FFT.

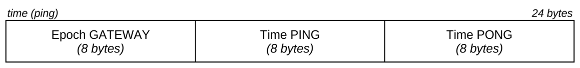


Figure 4.5: Time Payload

4.0.8 Data payload

Once everything is setup, it is now the time to send data. A data payload can have multiple types in itself, this is why there is 2 bytes header with the type (5-bits) and the size (13-bits). The 4 first bits are indicator of either 8,16,32 or 64 bits integers or a floating value (on 32-bits), a final 5th bit determine the signess of the underlying data.

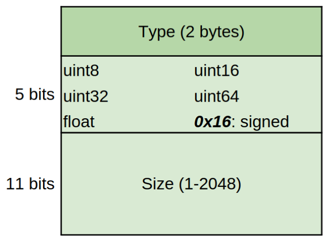


Figure 4.6: Type bits in Data Payload

The data can then be delivered on *2048 bytes maximum*.

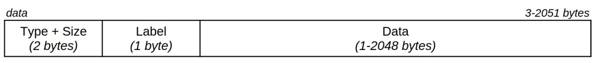


Figure 4.7: Data Payload

4.0.9 Battery payload

Sometimes, we need to check the battery level of each node, a simple value on 1 byte can be send with the same logic as the topology payload seen in Sec.4.0.6.

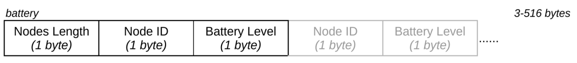


Figure 4.8: Battery Payload

[1]**FFT**: Fast Fourier Transform
[2]**RSSI**: Received Signal Strength Indicator
[3]**RTC**: Real-Time Clock

Implementation

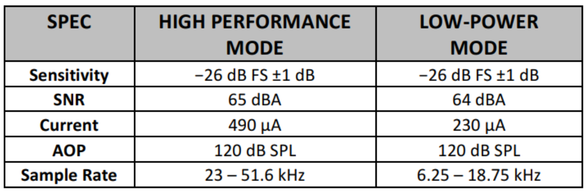
5.1 Data Collection

As detailed in the detailed architecture, each node consists of multiple sensors that need to be integrated with the microcontroller. This integration is essential for collecting physical data and enabling the various analytical processes needed.

5.1.1 Sensor Specifications

ICS-43434 MEMS Microphone

As detailed in the Principle section, leaks emit acoustic signals that travel through the pipe wall and surrounding medium, as the medium has a direct influence on the quality of the transmission and the system must be medium-agnostic, we had to choose a high performance acoustic sensor. The ICS-43434 is a MEMS microphone that perfecty fit our need. This sensor use the I2S communication protocol, it is in a way I2C for sound signals.



SPEC	HIGH PERFORMANCE MODE	LOW-POWER MODE
Sensitivity	−26 dB FS ±1 dB	−26 dB FS ±1 dB
SNR	65 dBA	64 dBA
Current	490 µA	230 µA
AOP	120 dB SPL	120 dB SPL
Sample Rate	23 – 51.6 kHz	6.25 – 18.75 kHz

Figure 5.1: ICS-43434 Features[12]

LSM6DS3TR MEMS IMU Accelerometer Gyroscope

In order to measure even the most sublte mecanical vibrations we needed an inertial measurement unit that provide high SNR and sensibility. Alternatively, we wanted to measure a wide range of frequencies hence a wide bandwidth. Most IMUs have a limited sample rate, the widely used MPU6050 for example only provide a 500Hz bandwith on the Accelerometer. To adress this issue we looked for a high performance IMU and found the LSM6DS3TR[13] that provide a 6.664kHz maximum sample rate resulting in a 3.3kHz maximum bandwidth. This sensor provide a standard I2C interface.

5.1.2 Data Collection Methods

We developed a hardware abstraction layer for both the ICS-43434 MEMS Microphone and the LSM6DS3TR MEMS IMU Accelerometer Gyroscope to facilitate the integration with other tasks running on freeRTOS. This layer provides a simple interface for data collection.

Microphone Data Collection

The ICS-43434 MEMS Microphone uses the I2S communication protocol to transmit audio data. The microcontroller is configured to operate in I2S master mode, receiving audio data from the microphone. The data is sampled at a specified rate and stored in a buffer for processing.

- Initialization: Configure the I2S interface with including sample rate, bit resolution and channel format.
- Data Acquisition: Continuously read audio data from the I2S interface and store it in a DMA buffer.
- Data Processing: Apply necessary signal processing techniques to extract meaningful information from the raw audio data.

IMU Data Collection

The LSM6DS3TR MEMS IMU uses I2C to transmit sensor data. The microcontroller communicates with the IMU to retrieve acceleration and gyroscope data at a constant interval.

- Initialization: Configure the I2C interface and initialize the IMU with the desired settings, including sample rate and measurement range.
- Data Acquisition: Periodically read acceleration and gyroscope data from the IMU registers.
- Processing: Apply calibration to ensure accurate sensor data.

5.1.3 Fast Fourier Transform

Detecting water leaks isn't an easy task. the challenge mainly lies in the fact that leaks emit acoustic signals that blend into their surrouding spectral environment. These signals are most of the time masked by noise from surround activities like traffic from the road above or machinery. Our solution had to extract meaningful features ;or leak spectral signatures; as we call them from this highly chaotic and unpredictable background.

To tackle this problem we used the Fast Fourier

Transform that converts raw time-domain signals into the frequency domain, where leak signatures become more distinguishable.

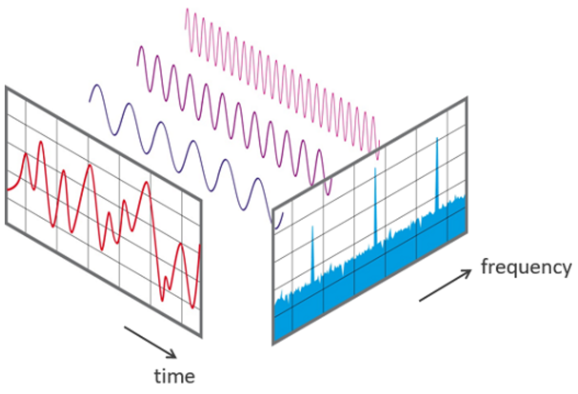


Figure 5.2: FFT Principle[14]

FFT Implementation

Initially, we implemented FFT from scratch in C. While this was a very good way to learn, the implementation was inefficient and too slow. To address this, we adopted the ESP-IDF DSP library which is optimized for the ESP32 architecture: this significantly improved performance.

Buffer Initialization

- Global buffers are allocated and aligned to cache lines for optimal performance on the ESP32.
- A Hann window is applied to reduce spectral leakage during the FFT process.

Signal Processing

- Raw 16-bit data from the sensors is converted to float and multiplied by the Hann window function.
- The complex input array is filled, where real values are taken from the processed data, and imaginary values are initialized to zero.

FFT Computation

- The FFT is computed using the `dsps_fft2r_fc32` function from the ESP-IDF DSP library.
- The complex data is transformed to real format.

Magnitude Calculation

- The magnitude (for the power spectrum) is calculated for the first half of the FFT result:

$$\text{Magnitude}[i] = 10 \cdot \log_{10}\left(\frac{\text{Re}[i]^2 + \text{Im}[i]^2}{\text{N}}\right)$$

First Results At first, we had problems with aliasing which happens when the sampling rate is too low to capture the signal's frequencies properly. This caused distortion and made it hard to analyse. To fix this, we increased the sampling rate to 88kHz, which is four times the highest frequency we wanted to observe (22kHz). This solved the issue and gave us much clearer and more accurate results.

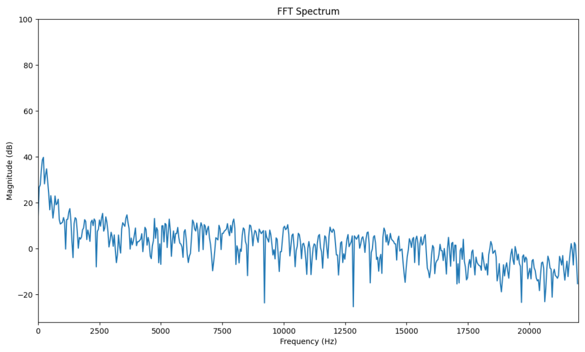


Figure 5.3: Aliased FFT Power Spectrum (The right part is mirrored at 12kHz)

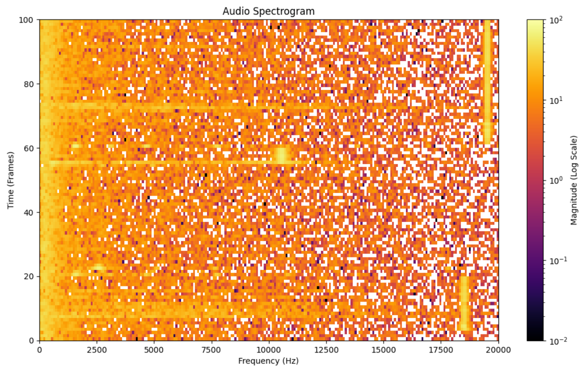


Figure 5.4: FFT Aliasing Correction: No more mirroring due to high sampling rate

FFT Observations

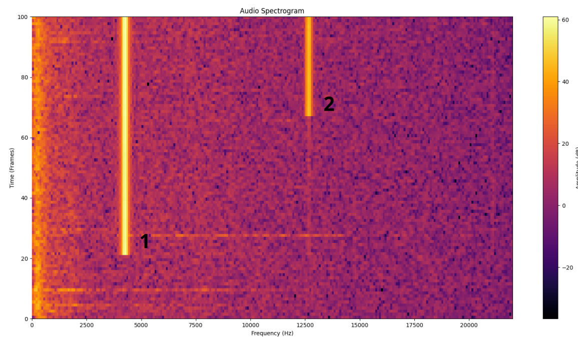


Figure 5.5: Harmonic Observations

After generating a sinusoidal signal **(1)** with a frequency of 4.2 kHz, we can clearly see a "vertical line" at this same frequency. However, we also notice faint traces of odd harmonics of order 3 and 5, visible at 12.5 kHz and 21 kHz, which could correspond to the 3rd and 5th harmonics. The harmonics should not be visible if the sinusoid was perfect.

The Fourier series states that a periodic signal can be expressed as a sum of harmonics.

$$x(t) = a_0 + \sum_{n=1}^{\infty} [a_n \cos(2\pi n f_0 t) + b_n \sin(2\pi n f_0 t)]$$

For a *pure sinusoid* like $x(t) = A\sin(2\pi f_0 t)$ it already matches the form of a single term in the series: $b_1 = A$, and all other coefficients a_n and b_n for $n > 1$ are zero.

Thus, a pure sinusoid has no harmonics because only one term (the fundamental frequency) is present. The harmonic observations on the supposed pure sinusoid is probably due to the fact that some quantification noise or imperfections in the speaker may have distorted the signal.

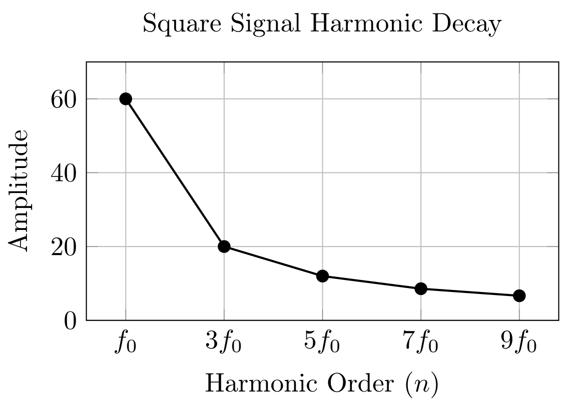
In **(2)**, we observe the 3rd harmonic when generating a square wave signal at 4.2 kHz (replacing the sinusoid in the test scenario). Square wave signals create multiple harmonics.

The Fourier series for a periodic square wave with frequency f_0, amplitude A, and 50% duty cycle is:

$$x(t) = \frac{4A}{\pi} \sum_{n=1,3,5,\ldots}^{\infty} \frac{1}{n} \sin(2\pi n f_0 t)$$

The square signal is odd, hence $a_n = 0$ and all cosine parts equals zero. The summation run over odd harmonics $n = 1, 3, 5, \ldots$ with each term decreasing in amplitude as $\frac{1}{n}$. In a nutshell, square waves inherently generate odd harmonics of the fundamental frequency.

In our demonstration, $f_0 = 4.2\,\text{kHz}$, the harmonics should appear at $3f_0 = 12.6\,\text{kHz}$, $5f_0 = 21\,\text{kHz}$, and so on.



This validates that our FFT implementation works well, as it successfully detects the fundamental frequency and harmonics of both sinusoidal and square wave signals.

5.2 Data Compression

5.2.1 Problem: Data Compression for Efficient LoRa Transmission

In simple terms: the full FFT output consists of an array of floats values representing the magnitude data. For our setup, with $N_{\text{SAMPLES}} = 1024$, the FFT produces 512 floating-point values. Each float on the ESP32 is 4 bytes (32 bits), resulting in a total size of:

$$\text{FFT Size} = 512 \times 32\,\text{bits} = 16,384\,\text{bits} = 2,048\,\text{bytes}$$

However, the maximum packet size for LoRa communication is only 256 bytes, making it impossible to transmitt the whole FFT in a single packet and packet fragmentation being a known problem catalyst and an overall bad practice.

5.2.2 Our Solution: The Naive Approach

With no background in compression and limited time, we implemented a simple, suboptimal approach that reduces data size sufficiently to meet LoRa's constraints.

To address the problem stated earlier, we use a combination of data reduction and lossy compression techniques to bring the size of the FFT data down to 128 bytes, achieving a total compression ratio of 16 : 1. This approach use the specific characteristics of the signals we measure and their frequency range to optimize the data transmission.

Some parameters of our setup plays in our favor. Since we analyze acoustic signals within the 0 Hz to 22 kHz range, we sample at a rate of 88.2 kHz to prevent aliasing, respecting the Shannon-Nyquist criterion with a margin factor of 4 (as 2 times the criterion leads to severe signal distortion). The FFT output, which ranges from 0 Hz to 88.2 kHz, includes significant portions that are either aliased or distorted. From experience, only the first quarter of the FFT (0 Hz to 22 kHz) is usable.

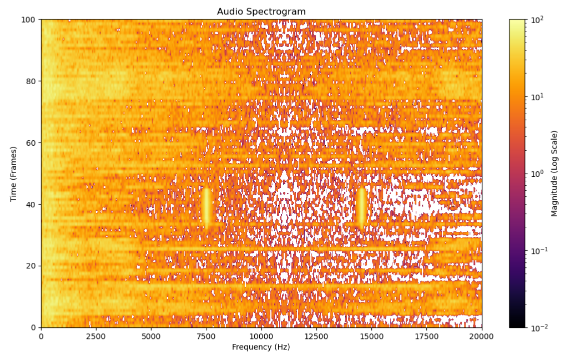


Figure 5.6: Sample Rate 44kHz Results in Aliasing

This allows for a preliminary data reduction: from 512 frequency bins (representing the full 88.2 kHz range) to 128 bins (covering the reduced 22 kHz range). This reduction alone decreases the data size to: $128 \times 32\,\text{bits} = 4,096\,\text{bits} = 512\,\text{bytes}$. Resulting in a much more convenient 4 : 1 compression ratio.

5.2.3 Spectral Analysis of High Pressure Water Leaks

If we look at the waveform of the acoustic signal of a high pressure water leak, we cannot deduce anything. However looking at the spectrometer things begin to make sense.

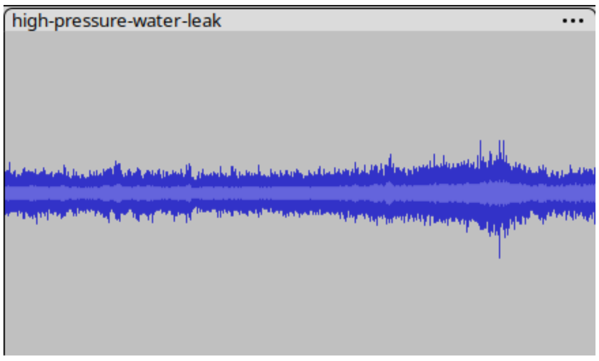


Figure 5.7: Waveform View of High Pressure Water Leaks Acoustic Signal

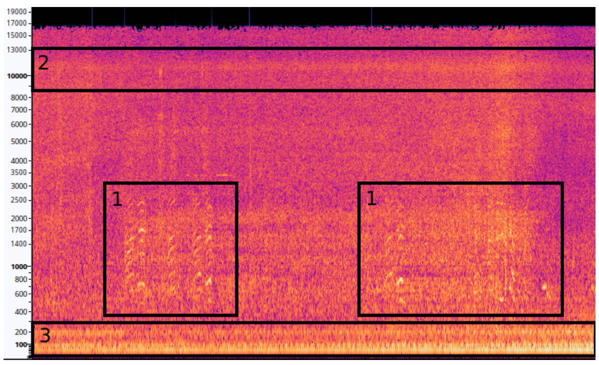


Figure 5.8: Spectrogram View of High Pressure Water Leaks Acoustic Signal

In (1) we obeserve typical voice patterns ranging from 300 to 3400 Hz, it must be filtered in the future. In (2) we observe the actual usefull high pressure leak spectral pattern. The ligh trace around 11kHz in (2) correspond to the high frequency component of the leak. The two marked traces in (3) are unidentified but it might be machinery or low frequency component. If we look carefully we can see very subtle traces burried in the noise in the mid-range frequencies.

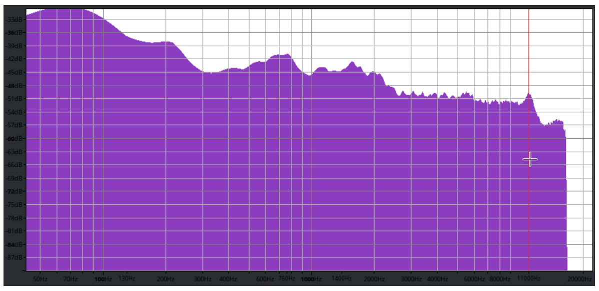


Figure 5.9: Frequency Spectrum of High Pressure Water Leak Sample 1

By looking at a different water leak audio sample we observe the same patterns. In the following case the high frequency component is shifted to 8kHz.

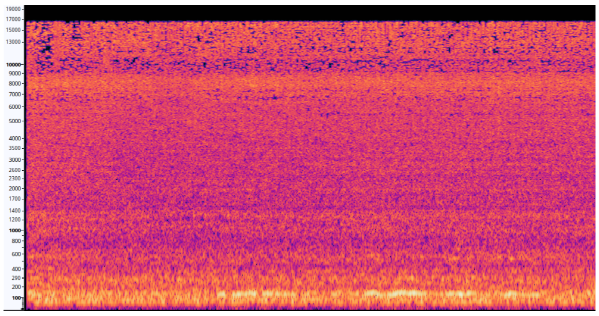


Figure 5.10: Spectrogram High Pressure Water Leak Sample 2

On the following Frequency Spetrum we can clearly identify the 8kHz leak signature.

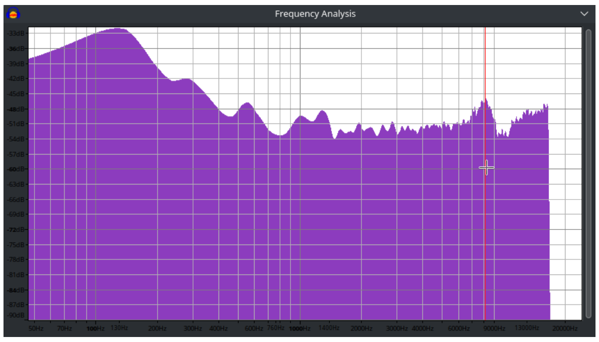


Figure 5.11: Frequency Spectrum of High Pressure Water Leak Sample 2

Looking at the two frequency spectrum side by side we can identify some similarities and try to grasp the spectral characteristics of high pressure leaks.*

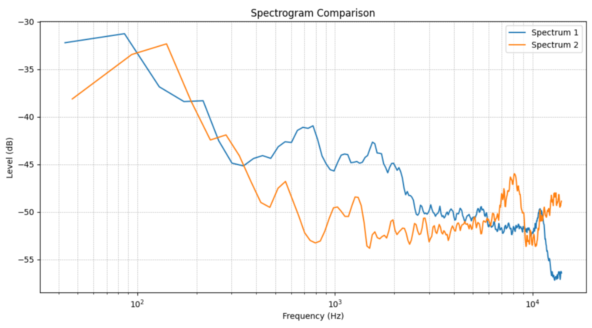


Figure 5.12: Similarities between the two Spectrograms

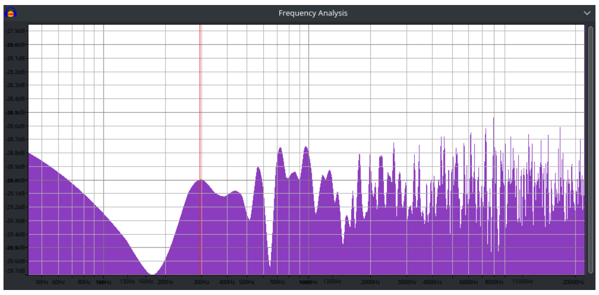


Figure 5.13: Frequency Spectrum of White Noise for Comparison

Using Dynamic Time Warping we can further confirm the similarities. If the alignment path is diagonal, the two spectrograms are identic.

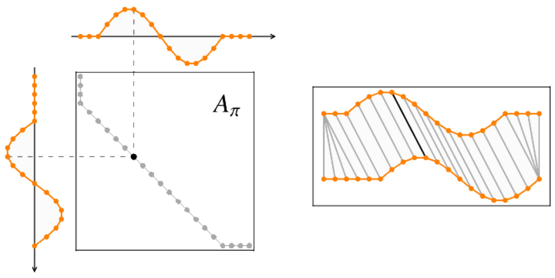


Figure 5.14: DTW Alignement Path Principle[15]

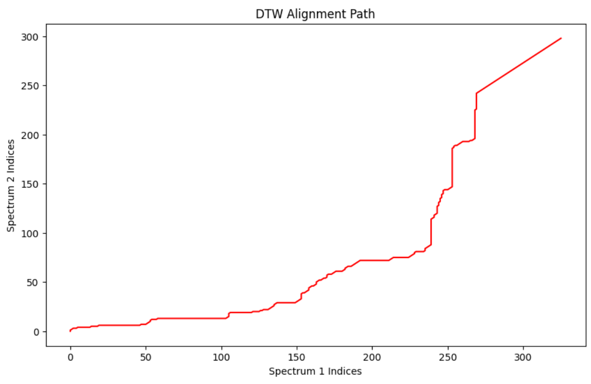


Figure 5.15: Spectrogram Alignement Path (Sample 1 and 2)

5.2.4 Filtering

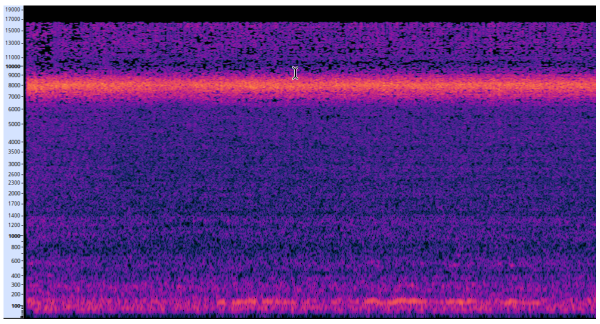


Figure 5.16: Isolated High Frequency Leak Component

5.3 Data Transmission

Each data that are gathered by the sensors and computed can be sent to different packets that have been described in Chap. 4. However we now have to take into account the physical layer and be sure that it can be sent flawlessly through the pipe network.

5.3.1 LoRa Module: SX1278

The choice we made for the LoRa module is plain and simple, *Semtech* is a company well-known for its `SX127X` line of product which can produce at different frequencies a PHY LoRa signal. Using one of their product coupled with an antenna at $433MHz$ can do easily the job. This module can send up to 256 bytes of data, which is quite enough for our application. Moreover because it is a well-known product its price can be really low.

For the developpement side, we thought that this module was so known that couple of libraries where available, even for the "ESP-IDF" framework. But no, we had to retro-engineer some of the libraries made by others for other μC. Overall, this took some time but the system is really robust and work flawlessly.

5.3.2 Transmission Mechanism: How It Works

We can send 256 bytes maximum, with a ToA *(Time over the Air)* of approximately 14 seconds maximum because we are on the 433 MHz frequency which allows up to 10% of the duty cycle. Here is how we setup each parameters of our communication:

- CR (Coding Rate): $\frac{4}{8}$
- BW (Bandwidth): $125KHz$
- SF (Spreading Factor): 10
- PS (Preamble size): 8
- DE (Low Data Rate Optimize): 1 *(enabled)*
- IH (Header): 0 *(no header)*
- PL (Payload Size): 140 *(maximum payload we can send)*
- CRC (Cyclic Redundancy Check): 1 (activated)

From the data-sheet of the `SX127X` [16] we get the following formulas:

$$Ts = \frac{1}{Rs}$$

$$Tpreamble = (PS + 4.25)Ts$$

$$Payload = \frac{8PL - 4SF + 28 + 16CRC - 20IH}{4(SF - 2DE)}$$

$$n_{payload} = 8 + max(ceil[Payload](CR + 4), 0)$$

$$T_{payload} = n_{payload} \times T_S$$

$$T_{packet} = T_{preamble} + T_{payload}$$

Via a numerical application we get the following for 140 bytes, which is the maximum we can send in our tests:

$$AN : Payload = \frac{8 \times 140 - 4 \times 10 + 28 + 16}{4(10 - 2)} \Rightarrow 288\ Bytes$$

$$AN : n_{payload} = 8 + 288 \Rightarrow 296\ Bytes$$

$$AN : Ts = \frac{2^{10}}{125000} \Rightarrow 8.192ms$$

$$AN : T_{payload} = 0.0082(296 + (8 + 4.25)) \Rightarrow 2.52s$$

So it takes approximately $2.5s$ to send this data over the air. Which is in our case too much and may not respect all the LoRa specifications, however this is for testing purposes and we can send with no issue with a time interval of around $25s$ *(10% of the duty cycle)*

Now that we know how much data we can send to each node, how much time it takes to send the data and how much we need to wait between packets. Remember that this is only for testing purposes and that we may need to tweaks some parameters. For example the *Coding Rate* could be lower, however it could cause some issues in term of distance. Same for the bandwidth used.

Each node will be connected in a linear manner for the public pipeline and may be in a star disposition for private pipelines. In this case we had to think in a way to send data over the network. In this case we send each packet to the closest node possible using its signal strength and a set of payloads that will map the network. Once this setup is finished, each node can send to its nearest neighbor and the following node will do the same until it reaches the Gateway. Here is an example schema of the disposition:

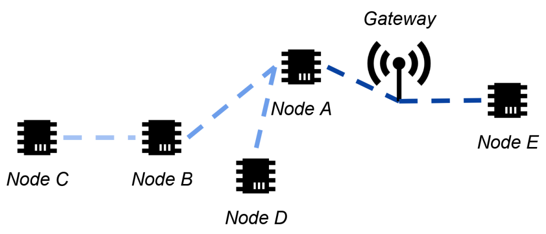


Figure 5.17: Nodes-to-Gateway Mechanism

5.3.3 Real-World Testing: Performance & Range

To test out the connectivity between nodes, we wanted to do a simple setup. In pipelines there will be multiple layers of concrete between nodes, we have to make sure that the connectivity remains intact when each node is disposed. We did a little protocol that will follow these statements:

- A node sending data is placed in my home, connected to "the pipeline".
- A master node will be connected with a battery and brought by me in my parking lot.
- The floor of the parking lot is beneath my home, and between multiple layers of walls.
- I will walk until the signal cannot be reached and write down the position.

Here is how the setup have been placed:

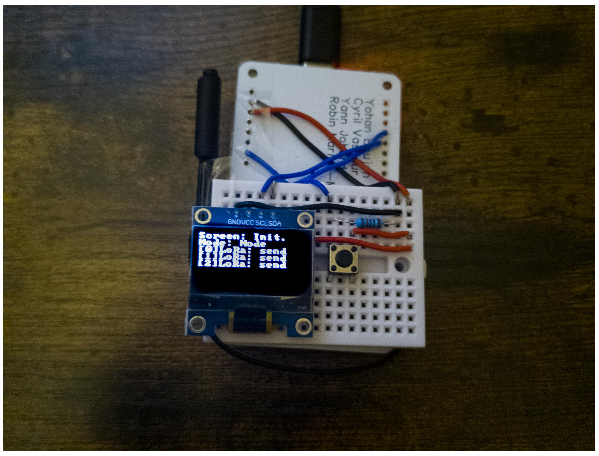
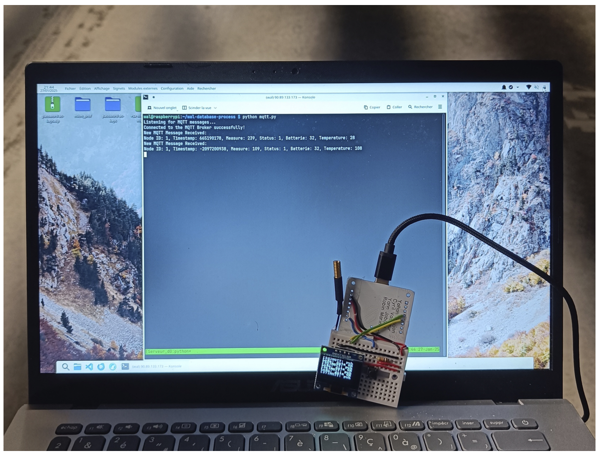


Figure 5.18: Home station: The Node sending through LoRa



```
wal@raspberrypi:~/wal-database-process $ python mqtt.py
Listening for MQTT messages...
Connected to the MQTT Broker successfully!
New MQTT Message Received:
Node ID: 1, Timestamp: 665190178, Measure: 239, Status: 1, Batterie: 32, Temperature: 28
New MQTT Message Received:
Node ID: 1, Timestamp: -2097200938, Measure: 109, Status: 1, Batterie: 32, Temperature: 108
```

Figure 5.19: Anywhere in the pipeline: Master Node transferring data

I received no signal after around 85m which is pretty good considering the price of each node which is quite low. We can therefore send many data and could tweak the parameters even more to have a better bit-rate overall.

5.3.4 Internet Link: LoRa to MQTT

As you could have seen there was data on the screen in the Fig. 5.3.3, this is because the Master Node is constantly converting the data received from the nodes to an MQTT packet in the `.json` format that will be explained in the Sec. 7.2.1. Here is how the data is converted:

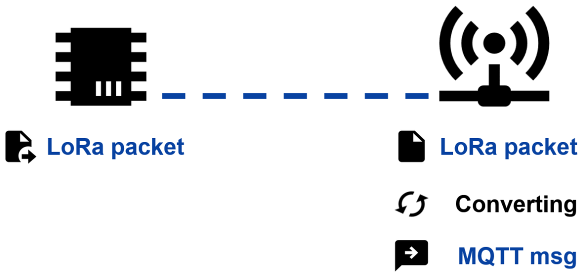


Figure 5.20: Converting between a LoRa packet to MQTT

In our example we only send a single payload with a lot of data, but in the future there will be multiple `.json` format to choose from. For now the function used creates a static json that cannot be modified in the future, but it will be rewritten to accept any modification we could do in the future. The server will be rewritten too in a much lower overhead language like C or Rust. The use of Python for now is only for demonstration purposes, and next there will be a daemon running in the background for ease of installation and use.

5.4 Supervised Approach: Support Vector Machine

Link to the repository: https://github.com/what-a-leak/wal-machine-learning

Support Vector Machine Principle

Support Vector Machine or SVM is a supervised machine learning technique used to classify data by finding the best *hyperplane* that separates different categories (class). The main idea behind SVM is it aim to maximize the distance (margin) between the hyperplane and the nearest data points, this helps ensure that the classification is robust because the data is well split. SVM works very well for linear problems but can also handle non linear ones by using *kernels* which are essentially mathematical functions that map data into a higher dimension where it becomes easier to separate.

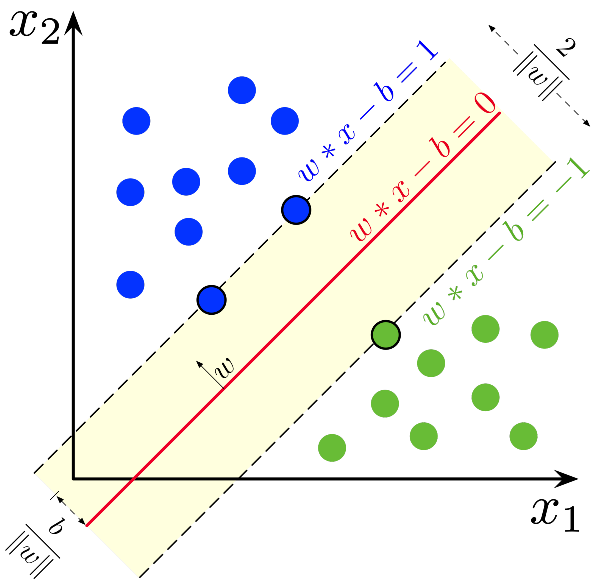


Figure 5.21: SVM Principle[17]

In a nutshell, SVM tries to draw the most optimal line or surface that divides the data into categories while keeping the line as far away as possible from the closest point of each cluster of category.

Mathematical Hints

Equation of the Hyperplane In 2D, a line can be written as:

$$y = wx + b$$

where w is the slope, and b is the y-intercept. This principle generalizes to a hyperplane in higher dimensions, but for obvious reasons a line is simpler to visualize (one can only dream of visualizing a 42 dimension hyperplane). The goal of SVM is to find the best w and b that maximizes the separation margin.

Margin Maximization The margin is the distance between the hyperplane and the closest points from each class (the support vectors). SVM maximizes this margin which can be expressed as:

$$\text{Margin} = \frac{2}{\|w\|}$$

where $\|w\|$ is the amplitude (length) of the vector w.

Constraint for Classification SVM ensures that every data point are on the correct side of the margin. For a data point x_i with a label $y_i \in \{-1, 1\}$, the constraint is:

$$y_i\,(w \cdot x_i + b) \geq 1$$

This means that points labeled $+1$ ($y_i = 1$) is on one side of the hyperplane, and points labeled -1 ($y_i = -1$) is on the other side.

Kernel when Things Goes Non-linear For non-linear problems, SVM applies a *kernel function* to transform the data into a higher-dimensional space where a hyperplane can separate them. For example, a simple kernel function might be:

$$K(x_i, x_j) = (x_i \cdot x_j)^2$$

In our case, to summary:

- The hyperplane is defined as $w \cdot x + b = 0$.
- All points satisfy the condition $y_i\,(w \cdot x_i + b) \geq 1$.
- The margin $\frac{2}{\|w\|}$ is maximized when training the model to obtain the best classification.

Implementation: Collect the Samples

In order to train the model, we created two distinct datasets: one representing leak acoustic signals and the other one without acoustic signals. Each dataset were composed of a 10s spectrogram sample under the two scenarios. The data was collected using a serial connection to a WAL Node, and the raw FFT data was saved into CSV files.

Once the data collected we could display it in a friendly spectrogram manner as follow to check for data integrity:

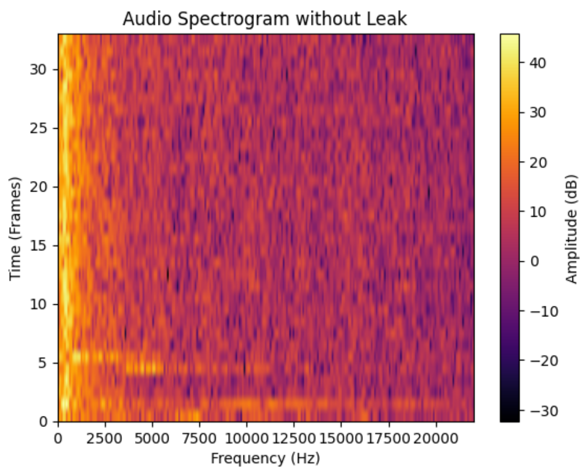


Figure 5.22: No Leak 10s Spectrogram Dataset

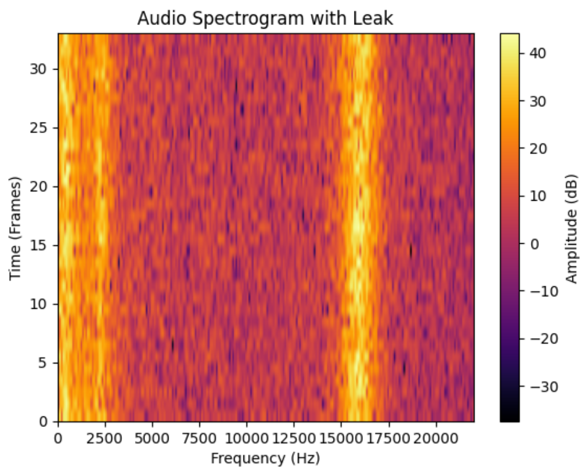


Figure 5.23: No Leak 10s Spectrogram Dataset

Training the Model

We implemented a SVN to classify audio data into two categories: "Leak" and "No Leak". After collecting the data we loaded it, added labels, and combined the datasets for later processing.

We split the combined dataset into training and testing sets to evaluate the model's performance. The features were standardized using a `StandardScaler` to ensure that all features contribute equally to the model. We trained a Support Vector Machine (SVM) classifier with a linear kernel on the training data.

After training, we made predictions on the test set and evaluated the classifier's performance using the following classification report and confusion matrix.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	1.00	1.00	1.00	7
1	1.00	1.00	1.00	7
Accuracy		1.00		14
Macro Avg	1.00	1.00	1.00	14
Weighted Avg	1.00	1.00	1.00	14

Table 5.1: Classification Report

The classification report shows very good precision, recall, and f1-score for both classes indicating that the model correctly identified all instances of "Leak" and "No Leak" in the test set. This shows that the model is highly accurate and performs well on the dataset, but will the performances still be good under real conditions? (c.f. next subsubsection).

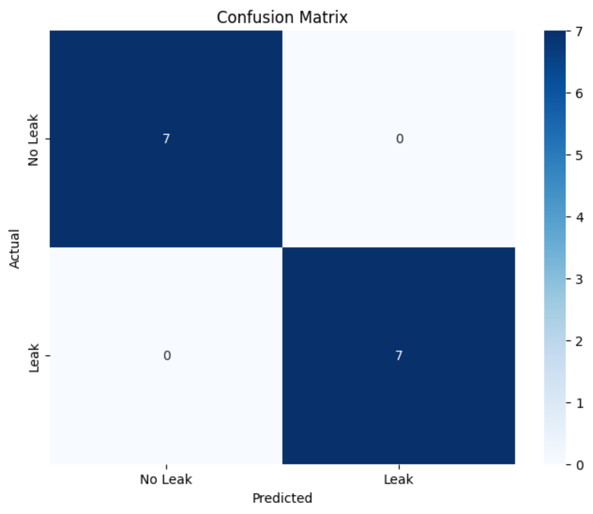
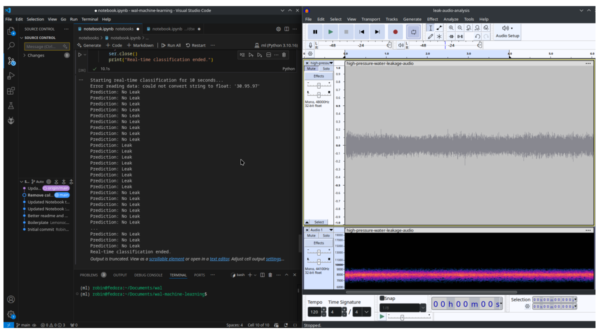


Figure 5.24: Confusion Matrix of the Trained SVN Model

Testing The Model



```
ser.close()
print("Real-time classification ended.")
```

```
Starting real-time classification for 10 seconds...
Error reading data: could not convert string to float: '30.95.97'
Prediction: No Leak
Prediction: No Leak
Prediction: No Leak
Prediction: No Leak
Prediction: No Leak
Prediction: No Leak
Prediction: No Leak
Prediction: No Leak
Prediction: No Leak
Prediction: Leak
Prediction: Leak
Prediction: Leak
Prediction: Leak
Prediction: Leak
Prediction: Leak
Prediction: Leak
Prediction: Leak
Prediction: No Leak
Prediction: No Leak
Prediction: No Leak
Prediction: No Leak
Prediction: No Leak
Prediction: No Leak
...
Prediction: No Leak
Prediction: No Leak
Prediction: No Leak
Real-time classification ended.
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

```
(ml) robin@fedora:~/Documents/wal
(ml) robin@fedora:~/Documents/wal-machine-learning$
```

Figure 5.25: Left: Testing Script Right: Audacity with Leak Audio

The model performs very well, it detects leaks even with very low amplitudes.

Security and Energy Management

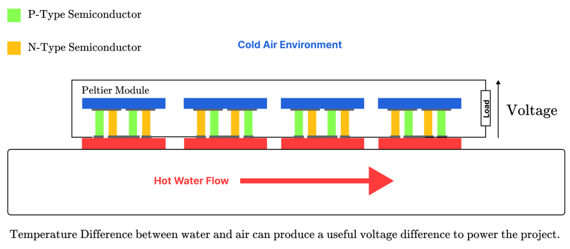
6.1 Energy harvesting

6.1.1 Reflections

Due to a lack of time, we were unable to implement a wireless energy solution. However, we would like to share our reflections on this topic.

To begin with, it is important to note that energy harvesting from the environment is very complex in our case. The nodes, which are equipped with batteries, are buried underground, embedded in concrete or installed on water pipelines. These environments are not subject to significant thermal variations and are stationary, making energy harvesting highly challenging. Nevertheless, we explored several methods:

Thermal Energy Recovery: We examined the possibility of harvesting thermal energy by exploiting the temperature difference between hot and cold water. Hot water pipes, in theory, could provide a heat source usable with thermoelectric materials to generate electricity. However, this method is limited to hot water pipes, which represent a small fraction of the total pipeline system.



🟩 P-Type Semiconductor
🟧 N-Type Semiconductor
Cold Air Environment
Peltier Module
Load
Voltage
Hot Water Flow
Temperature Difference between water and air can produce a useful voltage difference to power the project.

Figure 6.1: Thermal Harvester Principle

Vibrational Energy Recovery: Another option considered was harnessing vibrations caused by water flow or pressure variations in the pipes. Unfortunately, this approach proved unsuitable for our pipelines, as the vibrations were too weak to generate usable energy.

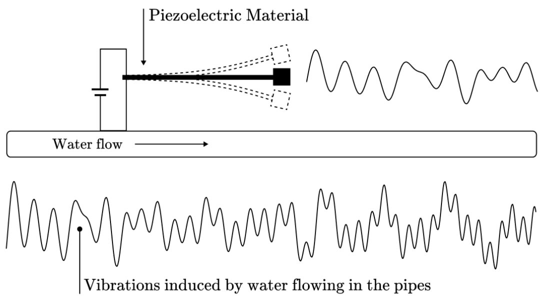


Figure 6.2: Piezoelectric Harvester Principle

Electrochemical Battery: We explored the use of electrochemical batteries that generate energy from the soil surrounding the pipelines. This method relies on reactions between two electrodes, such as copper and zinc, with the soil acting as an electrolyte. While promising for underground nodes due to its simplicity and continuous operation, its performance depends on soil composition, moisture, and electrode durability, which require further optimization.

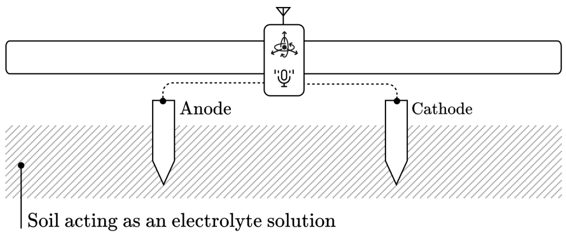


Figure 6.3: Electrochemical Battery Principle

We have also considered a solution for recharging the batteries:

Induction Charging: One practical solution would be to recharge the nodes via induction when their battery levels become critically low. This approach would extend the lifespan of the nodes. This approach, however, comes with several challenges, such as penetrating multiple layers of material, precisely locating the node, and integrating a charging system directly onto the nodes.

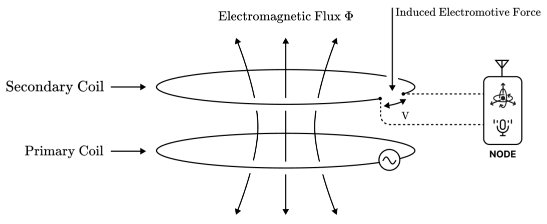


Figure 6.4: Induction Charging Principle

Solar Powering: We can imagine powering the nodes with tiny solar cells at the surface. The cells would be directly connected to the node with thin electrical wires. This solution has the potential to provide an extended (if not nearly unlimited) lifespan for the nodes. However, this solution is not practicable in our case as it would require extensive work to power all the individual nodes in the scaled network.

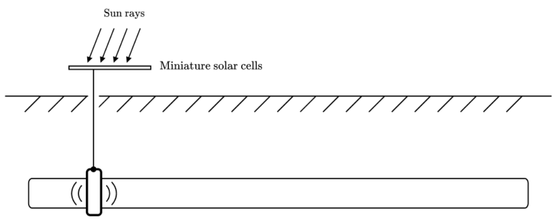


Figure 6.5: Solar Powering Principle

Although we were unfortunately unable to implement any solutions, we believe that, given more time, the first solution to attempt would be induction charging as its very interesting and to our knowledge induction charging through concrete has never been achieved.

6.1.2 Energy Consumption of one node

We conducted energy consumption measurements on the MARK1 prototype under various processing scenarios to evaluate its performance.

All measurements were performed under a strict protocol to ensure data accuracy and reliability. Each test was powered by a stable 5V power source to maintain consistent conditions and eliminate potential variations due to power instability.

The source code for the different scenarios used during testing is available here.

Scenario	Consumption in mA
IDLE	21.4mA
FFT_COMPUTE	34mA
LORA_RECEIVE	30.6mA (idle) 31.1mA (rx)
LORA_SEND	104.5mA
DEEP_SLEEP	3.9mA

Table 6.1: Consumption of MARK 1 Prototype

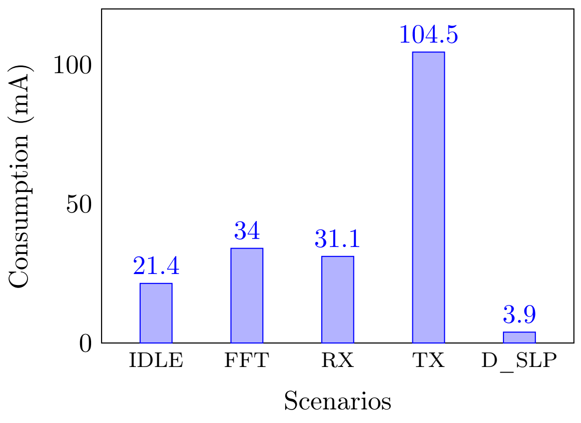


Figure 6.6: Scenario Energy Consumption Comparison

We decide to go with the acu INR18650-25R of samsung, it's capacity is 2500 mA. And the node will have two acu in parallel.

$$5000\,mA/4mA = 1282\,hours$$

Around 52 days of autonomy in deep sleep Take in account that we are only on prototyping so the microcontroller may change and the consumption too.

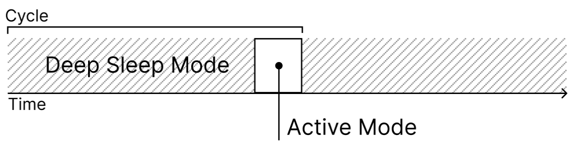


Figure 6.7: Scheduling Example

6.2 Security Features

6.2.1 Security Concern

Security is a crucial aspect of all projects involving some sort of communication, especially in a Wireless Sensor Network where data integrity and confidentiality are essential.

Context and Challenges

Our LoRa module (SX1278) operate on frequencies ranging from 410MHz to 525MHz and is designed for low power consumption and limited bandwidth. Without any security features each sensor node is vulnerable to every attack imaginable (replay attacks, data tempering, eavesdropping, overloading...). Consequently it is necessary for the project to implement a security solution that protects data integrity and confidentiality without introducing too much overhead.

6.2.2 Solution

We propose a lighweight security mechanism based on pre-shared symmetric keys that use AES-128 encryption: each sensor encrypts its data using AES in Cipher Block Chaining (CBC) mode with a unique and randomly generated Initialization Vector (IV) for every message. The IV is then transmitted with the actual message.

The issue with this method is that we have to transmitt the Initialization Vector with each message. Depending on the IV size (16 bytes usually) this can add significant overhead, and with LoRa (maximum packet size 256 byte) we can't afford it. This led us to think a new solution.

Instead of sending the whole Initialization Vector we can store a static decimated 14 byte version on each nodes and only send two randomly generated bytes to complete the IV. This solution alter AES security but this tradeoff is acceptable for the application.

Moreover the Initialization Vector goal is to introduce some randomness in the cryptographic process in order to prevent replay attacks, but fortunately our application transmitt FFT data in its payload, but Fast Fourier Transformations induce significant chaos and randomness which results in the extremely low probability of message duplicata thus naturally tampering potential replay attacks.

6.2.3 Implementation

In order to implement AES on the microcontroller we first thought about implementing the algorithm in bare-metal without library as its not a particularly complex algorithm but we found TinyAES (https://github.com/kokke/tiny-AES-c), a lightweigh cross platform library.

It is important to note that AES can only encrypt data in blocks of 16 bytes, so padding is necessary to ensure that the data fits this requirement.

This is why we have incorporated PKCS#7 padding to handle data whose length is not a multiple of the 16-byte AES block size. With this padding scheme, we add extra bytes to the last block to align the data lenght with the required block size.

Source code of the implementation: light-encrypt

```
[Test Task] Original Message: Hello World
[Test Task] Encrypted Data:
04 b4 d0 b0 1a f1 c4 a6 94
95 b3 f2 6c 4a ee fd 82 2c
[Test Task] Decrypted Message: Hello World

[Test Task] Original Message: Hello World
[Test Task] Encrypted Data:
78 d8 f3 74 04 7b fb 13 49
d2 f7 a8 65 36 a3 01 39 cc
[Test Task] Decrypted Message: Hello World
```

In the demonstration above we can clearly see the entropy implemented by the IV: while the clear message is identical, the encrypted output is absolutely different. This mechanism effectively prevents replay attacks.

Server Side Data Handling

7.1 Principles

In this section, we will explain the data handling system. Starting with the reception of the data on the main server, to its local storage on a database and finally, its processing to be handled by the user interface system.

7.1.1 Generalities

This part has two main objectives : providing an history and preparing the data for UI display. Maintaining a history of sensor values is crucial in many applications as it allows for trend analysis, anomaly detection, and decision-making. By tracking past data, users can identify patterns, predict future behavior, and take preventive or corrective actions. In our case, this historical insight is very valuable as it can help us in optimizing our system performance, considering that the analysis of the data relies on Machine Learning. The more we gather data, the more we can train our model.

Our concept/product isn't yet to be sold, but we still explored the system up to the user experience. As so, we estimate that providing data to the user is important because humans tend to enjoy applications that provide personalized, non-essential insights, like reminders to water a plant or calorie tracking. These features foster engagement and satisfy the psychological need for awareness and control. We consider that this processing of the data in view of displaying it is an legitimate part of the project because after all, the social scale of innovation cannot be neglected.

7.1.2 Processing Overview

The data processing workflow on the server follows a structured sequence to ensure smooth handling. Firstly, the data is received from gateway and passed to the Mosquitto broker, which facilitates reliable communication between the devices and the server. Secondly, the server processes the incoming data, performing validation checks to ensure the information is complete and adheres to the expected format. Once validated, the data is written into an SQLite database, which serves as a secure and organized storage solution. Thirdly, the stored data is retrieved and formatted into a structured JSON file, making it easy to access and compatible with other tools or systems. Finally, this JSON output is ready to be used by the User Interface application (Mobile Application and Website) . By following these steps, the server ensures that data is systematically received, stored, and prepared for visualization.

Here is a small schema detailling the process :

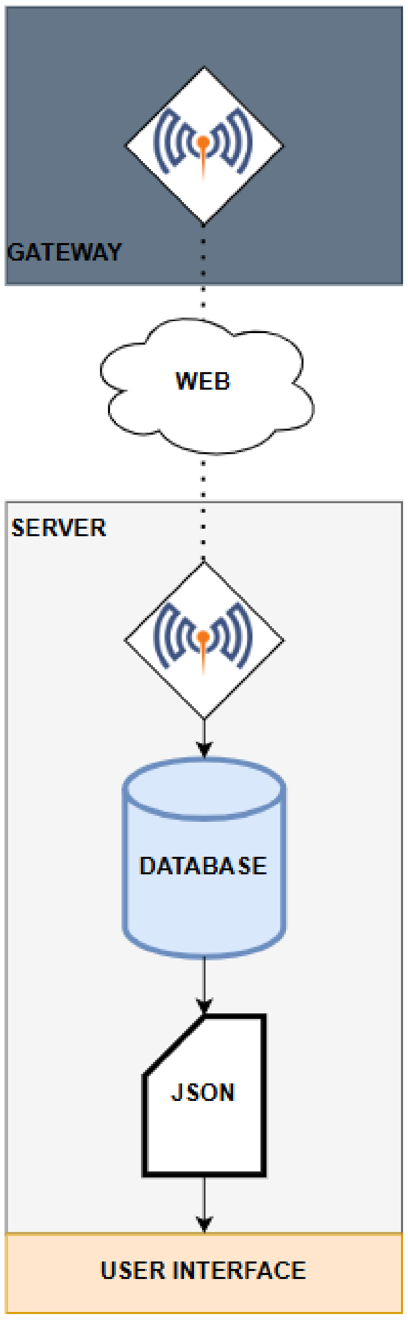


Figure 7.1: Data Handling

7.2 Data Storage

To store the data, we've chosen to use a Database. It allows for a precise manipulation of the data. For this proof of concept, we've chosen to use SQLite for it's simplicity and lightweight for the Raspberry Pi.

7.2.1 Data Format

The expected data format to be received from the broker is the following :

```
{
    "node_id": INT,
    "mesure": STRING,
    "status": BOOLEAN,
    "timestamp": EPOCH(32),
    "battery": INT,
    "temperature": INT
}
```

7.2.2 SQL Tables

In our case, we mainly focused on one client's experience to prove our concept. As so, our database architecture was made as simple as possible. We have two tables dedicated to our client, one to contain static data and the other for dynamic data. In fact, the logic behind it is that the nodes have to upstream as little data as possible. We identified this bare minimum as the node's ID for it to be correctly identified, and the data of the measure and the metadata surrounding it : measurements, status of the node, battery, temperature and timestamp. Those data would be the dynamic data as opposed to the static data, which are the node's ID and its localization, those data are not expected to change regularly.

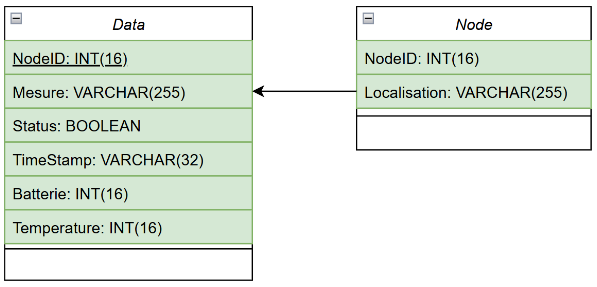


Figure 7.2: SQL Tables

7.2.3 Database Improvements

This system can be improved on a full-scale architecture depending on the client's need. We can imagine that a large public or private structure owning a large network of nodes, and needing more confidentiality and security about those strategic data, could be interested in his private database, possibly secured on a separated virtual machine. Whereas a private household owning a few nodes would cost less resources to have its data stored in a shared database with other private individuals. The improvements database side focus would probably be more oriented on user experience, such as more static parameter for personalization : nicknaming, alarm parameters, etc.

7.3 Server

In this section, we will explain how the server processes the data. We will start with it's asynchronous reception through the Web, and will follow with it's automatized JSON generation. It is important to note that for studying purposes, we have chosen to use a Raspberry Pi for the free use it offers us. However, for a large scale deployment of our application, a bigger server would be appreciable, allowing for a more reliable experience, a better computing speed, and a larger storage.

7.3.1 Data reception

To ensure an asynchronous data reception, we have chosen to use Mosquitto (MQTT), because it offers an open 24/24 port awaiting for data, while being very lightweight and remarkably easy to configure. On a side note, we ended up studying it in a courses at INSA, strengthening the trust we gave it. It also has the benefit of offering an included security measure that we will detail on the following part.

7.3.2 MQTT Security and Configuration

To secure the connection to our server, we came up with a few idea of protection around our MQTT broker.

Starting with the simple yet efficient channel credentials. This is a built-in feature of MQTT. The channel (topic) we've chosen for the communication "sensors/data" can only be reached if the sender include a username/password combination that matches the one in the configuration file of the server. Thus canceling potential pirate emission.

Our second security choice, was to configure the firewall of the host of the raspberry pi to close other ports than MQTT's one. By knowing which precise port is open and what for, we can more easily determine if some event isn't legitimate.

On data arrival, the JSON file that went through the two precedent securities is now analyzed to ensure it's format is compliant with the system. If one of the data contained within has an unexpected format, the packet will be thrown away. Thus limiting the options for a potential attacker.

7.3.3 User output Data formatting

To ensure a smooth user experience, as the data is getting stored in the database, the server starts updating the JSON file used to display the data in the user interface. It will choose the nodes that are correspond to the user, and get all the datas collected by this node to put it in the JSON.

This operation can be adapted for depending of the usage that is awaited of the data. If the client thinks that data's which are more than three months old are not relevant, they wont be added anymore to the JSON, will staying in the database. This part really depends on the user's feedback.

7.3.4 Possible improvements

There are several improvements possible for the Server. As mentionned before, a material improvement is a must if the project comes to being commercialized. Some proper infrastructure would be way more relevant than an micro computer in pretty much every aspect.

However, on the functionality side, there also some improvements that would definitely greatly improve our system. The most important one is definitely to offer a more important role to the server in determining if weather or not there is a leak, or if there is an abnormal behavior. To achieve that goal, with the consent of our customers, the pooling of the data in a common dataset would be beneficial to our machine learning model greatly. With a more diverse dataset on the water usage of different user, the model would have more information to recognize a leak's pattern. This is important because depending on the material, the size of the pipe or the distance between two nodes, the results while offering similarities, would be very different in term of absolute data. Thus, strengthening our model even more, and possibly re-deploy it on the current nodes as a software update.

This is one possible improvement, bu many more are probably yet to be found.

User Interface

8.1 Website

In this part, we will explain how the website work. We decide to implement one to follow the course of our applications and see the collected data from a laptop everywhere in the world.

8.1.1 Why did we choose maven

Maven is an essential tool for development projects due to its ability to efficiently manage dependencies, standardize project structures, and automate processes such as compilation, testing, and deployment. In our fifth year, where projects become more complex and collaborative, we had the opportunity to use Maven and become familiar with it. This explains why we chose to use it for our project.

8.1.2 Springboot and maven

Spring Boot is an essential framework for developing modern Java applications, as it simplifies configuration and provides a ready-to-use environment to quickly start a project. In our fifth year, where we worked on projects requiring complex features like REST API management or database connections, Spring Boot allowed us to save time. This is why we used it in our project, applying the skills we have acquired.

8.1.3 Database Connection

To link with the database that retrieves node values, we use a JSON file generated by the data management system. This file includes all the relevant tables and is parsed by the web server to extract the required information.

8.1.4 User Identification

In terms of security, we decided to use a user and password system. This system was designed to make the website robust against SQL injection, thanks to the java code in the website's backend. The user and password are propagated on each page thanks to a backup system that removes access if the web page is left. The credentials are stored internally and are only accessible to the server and therefore the administrator.

Below you can find our user identification page.

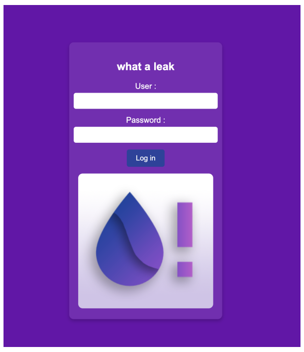
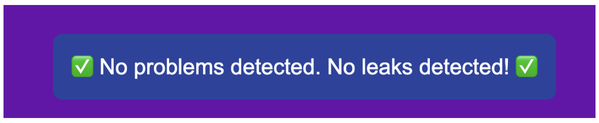


Figure 8.1: Identification feature

8.1.5 Which feature has to be implement

Focusing on the core principles of our system, we first implemented essential functions to inform users about potential leaks. This includes displaying a message indicating whether a leak has occurred and specifying the affected node, as shown in the screenshot below.



✅ No problems detected. No leaks detected! ✅

Figure 8.2: Leak Detection

With this functionality in place, we decided that a graph visualizing the FFT (Fast Fourier Transform) would be a useful addition. It helps with debugging and provides a means to monitor the data effectively. Thus, we implemented the Graph Visualization page.

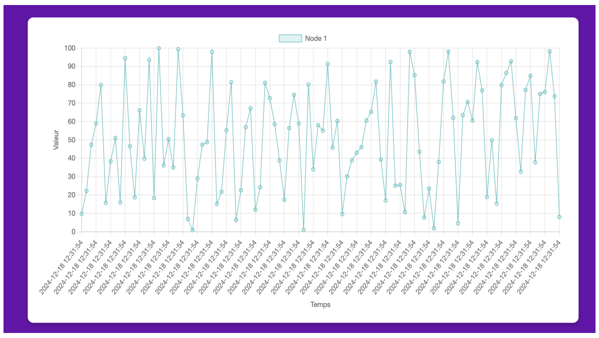
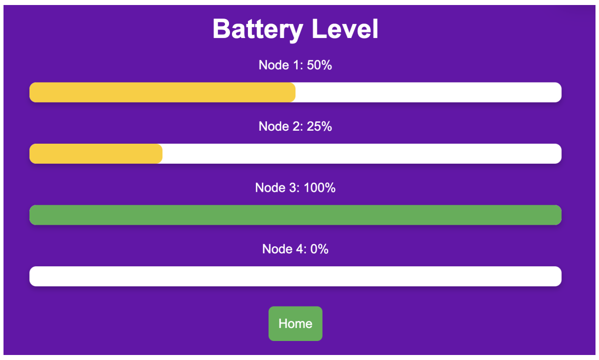


Figure 8.3: Graphic display

Next, we thought it would be beneficial to track the battery levels of each node to facilitate quicker interventions in case of failures. This idea led to the creation of the Battery Level page.


Battery Level

Node 1: 50%

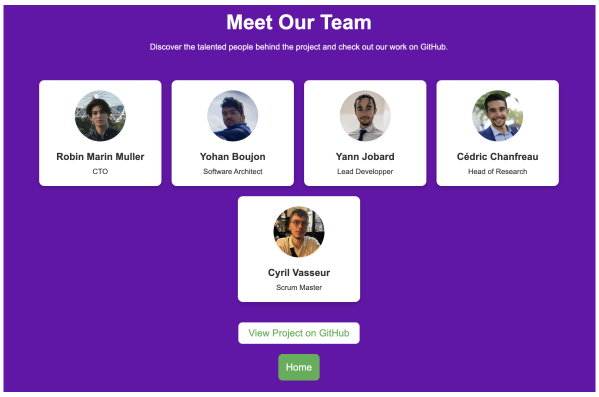
Node 2: 25%

Node 3: 100%

Node 4: 0%

Figure 8.4: Battery level

For convenience and as a reminder, we also included a link to our GitHub repository, where the entire project is hosted.


Meet Our Team

Discover the talented people behind the project and check out our work on GitHub.

Robin Marin Muller
CTO

Yohan Boujon
Software Architect

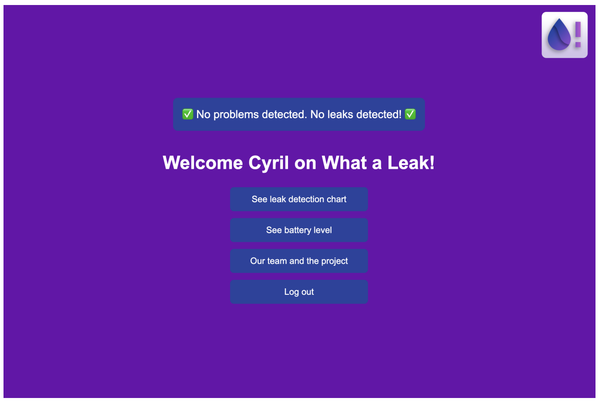
Yann Jobard
Lead Developper

Cédric Chanfreau
Head of Research

Cyril Vasseur
Scrum Master

Figure 8.5: Team presentation

Finally, we designed a Home Page to simplify navigation across all the pages of our application.


✅ No problems detected. No leaks detected! ✅

Welcome Cyril on What a Leak!

Figure 8.6: Home for the website

8.1.6 Wesbsite online how to?

The web server runs locally on the Raspberry Pi. It is accessible externally through port forwarding, which maps a public IP address to the Raspberry Pi's private IP address.

8.1.7 Limit and improvement

For now, the system operates locally on a Raspberry Pi. If the project progresses beyond the prototype stage, it would be beneficial to acquire a domain name and a dedicated server. For example, the domain name could be something like www.wal.com.

One of the main limitations of the system is that the REST architecture behind the website is quite challenging to maintain without breaking the web page. It might be worth considering redesigning the structure using a microservices architecture, which would make maintenance and updates easier.

8.2 Mobile Application

8.2.1 Choice of React Native and Expo

React Native and Expo were chosen for the development of our mobile application due to their ability to simplify cross-platform development.

React Native allowed us to write code in JavaScript and React that can be compiled into native code for both iOS and Android platforms. One of the benefits is that it reduces development time, as there is no need to maintain separate code bases for each. Moreover, React Native offers a good library of pre-built components and a large community of developers, which facilitates rapid development and troubleshooting.

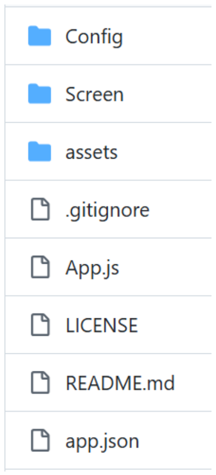
Expo simplified the initial setup of What a Leak app by providing a suite of tools and services that streamline the development process. With Expo, we quickly set up a new project, with broad access to native APIs. Its managed workflow handles complexities like build configurations and dependency management, allowing us to focus on application functionality.

8.2.2 Architecture of the Application

The application architecture is designed with a clear separation of each page. The main components of the architecture include:

- **assets**: Contains static resources such as images, icons, and other visual assets.
- **Config**: Stores and manages node data, including battery, vibration, sound levels, and leak status in a specific positions.

- **Screen**: Houses the main screens of the application, such as the Dashboard, Profile, Leak Detection, and Statistics. Each screen focuses on a specific feature and handles its corresponding UI and logic.
- **App.js**: Serves as the entry point of the application. It manages the navigation between screens by calling Navigation.js.
- **app.json**: Defines the app configuration, including metadata such as name, version, and assets bundling.



Config
Screen
assets
.gitignore
App.js
LICENSE
README.md
app.json

Figure 8.7: Application Workspace

8.2.3 Screens and User Interface

The main features of the application are implemented on several screens, each designed to provide a smooth user experience:

- **WelcomeScreen**: Displays a welcome message and the app logo when the app is launched. It transitions to the Home Screen after the mobile application loads.
- **HomeScreen**: Provides options to log in or close the app. It serves as the main entry point for users.

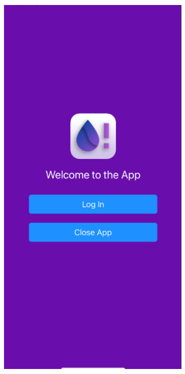


Figure 8.8: Home Screen

- **DashboardScreen**: Overview of the nodes, including their battery levels and leak detection status. It also includes a weekly statistics graph.

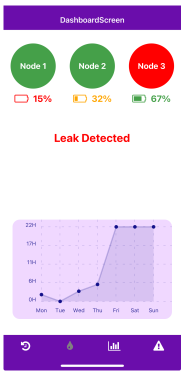


Figure 8.9: Dashboard Screen

- **HistoricScreen**: Shows the history of node activities and their positions on a map. Users can navigate to detailed node information.

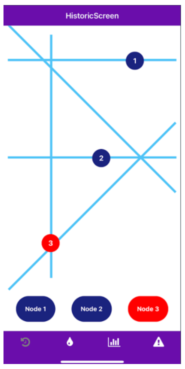
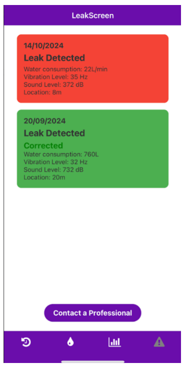


Figure 8.10: Historic Screen

- **LeakScreen**: Lists the history of detected leaks with details such as date, water consumption, vibration level, sound level, and location. It also provides a contact button to reach a professional.



LeakScreen

14/10/2024
Leak Detected
Water consumption: 22L/min
Vibration Level: 35 Hz
Sound Level: 372 dB
Location: 8m

20/09/2024
Leak Detected
Corrected
Water consumption: 760L
Vibration Level: 32 Hz
Sound Level: 732 dB
Location: 20m

Contact a Professional

Figure 8.11: Leak Screen

- **StatisticsScreen**: Provides detailed statistics on water consumption, with options to view data by week, month, or year.

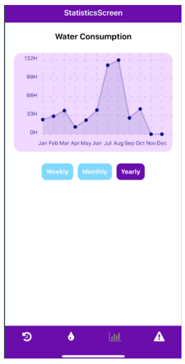


Figure 8.12: Statistics Screen

8.2.4 Features implemented

The application reads the battery level, vibration level, sound level, and leak detection status from the database JSON file located in the Config directory. This JSON file is synchronized with the server to ensure that the data is always up-to-date. The application displays this information on various screens, such as the Dashboard and Node Info screens, providing users with real-time insights into the status of their nodes.

8.2.5 Future integration

Future integration plans include enhancing the application's ability to handle and process more data stored in JSON format. We plan to integrate more detailed analytics and reporting features that will utilize data to provide users with deeper insights into their water usage patterns and leak detection history. Moreover, we aim to implement real-time notifications to alert users immediately when a leak is detected, ensuring prompt action can be taken. User authentication will be integrated to allow multiple users to manage their own profiles and data securely. Advanced analytics features will be added to provide deeper insights into water usage patterns and potential areas for improvement, helping users to optimize their water consumption and detect issues more effectively.

8.2.6 Development and Deployment

The development and deployment process involves generating an APK (Android Package) or IPA (iOS App Store Package) for the application. This APK/IPA is created using Expo's build tools, which package the application for installation on Android or iOS devices. Each time a modification is made to the application, a new generation is required to include the latest changes. This ensures that the application is always up-to-date with the most recent features and fixes before being deployed to users.

8.2.7 Challenges and Solutions

During the development of the application, several challenges were encountered and addressed:

- **Cross-Platform Compatibility**: Ensuring the application works seamlessly on both iOS and Android devices. This was addressed by using React Native's cross-platform components and Expo's managed workflow.
- **Data Synchronization**: Keeping the local data in sync with the remote server data.
- **Performance Optimization**: Ensuring the application performs well, especially when handling data. This was addressed by optimizing the data fetching and rendering processes.

In summary, the mobile application leverages React Native and Expo to provide a robust and efficient solution for monitoring and managing water usage and leak detection, with future plans to enhance functionality and user experience.

Results and Discussion

9.1 Experimental Results

The experimental phase of the project provided significant insights into the performance and limitations of the system. Below are the key findings:

Achievements

- **Leak Detection and Signal Classification**: Controlled tests confirmed accurate leak detection and effective signal classification, meeting the initial project objectives.
- **Communication Reliability**: The system demonstrated successful LoRa communication, even in noisy environments, ensuring robust data transmission between nodes.
- **Integration with Applications**: Effective communication between nodes and the website/mobile application was established, showcasing a functional end-to-end solution.
- **Scalability**: The modular architecture allows for the integration of additional sensor nodes or functionalities in the future.
- **Security Concerns**: Steps were taken to anticipate for potential cybersecurity attacks on different steps of the processing.
- **Advanced Machine Learning on the Edge**: Successfully implemented on-device machine learning models to enable real-time data analysis and decision-making without relying on server processing.
- **Project Development**: The Agile methodology facilitated steady progress, resulting in a fully operational Mark 1 prototype, with Mark 2 currently in development.
- **Teamwork and Collaboration**: The project benefited greatly from excellent teamwork and collaboration, enabling efficient problem-solving and innovation.

Limitations

- **Battery Management**: The current prototype lacks an integrated battery management system. While inductive charging and optimized consumption were considered, they remain unimplemented.
- **Signal Interference**: High-noise environments introduced signal interferences that occasionally impacted communication reliability.
- **Real-World Testing**: The prototype has not been tested under real-world conditions, limiting the evaluation of its performance in practical applications.
- **Physical Model Efficiency**: The current physical prototype is less effective than anticipated, requiring redesign or professional assistance to enhance its performance.
- **Network Dependency**: The Gateway to Server communication relies on a stable network, which could be a limitation in remote or unstable environments.

Future Improvements that can be implemented

To address the identified limitations and further refine the system, the following steps are recommended:

- **Battery Management**: Optimize energy consumption to increase node autonomy.
- **Improved Signal Processing**: Refine the AI models for better accuracy in signal classification and noise handling.
- **Expanded Testing**: Install the network in a larger and more realistic prototype model to evaluate its performance in real-world scenarios.
- **Accurate Leak Localization**: Improve leak localization to centimeters scale.
- **Physical Model Redesign**: Redesign and build a more effective physical model to support reproductible testing.
- **User Data Pooling**: Create a common dataset made of every user's data to improve the machine learning models.
- **IPv6 6Lo-WPAN for Integration for Large Scale Deployment**: Integrating IPv6 6Lo-WPAN could facilitates seamless integration with existing IP-based networks, which is more suitable for large-scale deployments

Bibliography

[1] *Leaks Mean One in Five Litres of Water in France*. https://www.connexionfrance.com/news/leaks-mean-one-in-five-litres-of-water-in-france-is-wasted-says-study/166375. Accessed: 2025-01-04. 28 June 2023.

[2] O. Hunaidi and W. T. Chu. "Acoustical characteristics of leak signals in plastic water distribution pipes". In: *Applied Acoustics* 58 (1999), pp. 235–254. DOI: 10.1016/S0003-682X(99)00013-4.

[3] Yan Gao and Yuyou Liu. "Theoretical and experimental investigation into structural and fluid motions at low frequencies in water distribution pipes". In: *Mechanical Systems and Signal Processing* 90 (2017), pp. 126–140. DOI: 10.1016/J.YMSSP.2016.12.018.

[4] *Find Leaks - Subsurface Leak Detection*. https://www.subsurfaceleak.com/find_leaks.html. Accessed: 2025-01-28.

[5] Konstantinos Marmarokopos et al. "Leak Detection in Plastic Water Supply Pipes with a High Signal-to-Noise Ratio Accelerometer". In: *Measurement and Control* 51 (2018), pp. 27–37. DOI: 10.1177/0020294018758526.

[6] Majid Ahadi and M. S. Bakhtiar. "LEAK DETECTION IN WATER-FILLED PLASTIC PIPES THROUGH THE APPLICATION OF TUNED WAVELET TRANSFORMS TO ACOUSTIC EMISSION SIGNALS". In: *Applied Acoustics* 71 (2010), pp. 634–639. DOI: 10.1016/J.APACOUST.2010.02.006.

[7] Mati-Ur-Rasool Ashraf Virk et al. "Leak Detection Using Flow-Induced Vibrations in Pressurized Wall-Mounted Water Pipelines". In: *IEEE Access* 8 (2020), pp. 188673–188687. DOI: 10.1109/ACCESS.2020.3032319.

[8] Suan Lee and Byeonghak Kim. "Machine Learning Model for Leak Detection Using Water Pipeline Vibration Sensor". In: *Sensors (Basel, Switzerland)* 23 (2023). DOI: 10.3390/s23218935.

[9] Jungyu Choi and Sungbin Im. "Application of CNN Models to Detect and Classify Leakages in Water Pipelines Using Magnitude Spectra of Vibration Sound". In: *Applied Sciences* (2023). DOI: 10.3390/app13052845.

[10] J. Gong et al. "Detection of Emerging through-Wall Cracks for Pipe Break Early Warning in Water Distribution Systems Using Permanent Acoustic Monitoring and Acoustic Wave Analysis". In: *Water Resources Management* 34 (2020), pp. 2419–2432. DOI: 10.1007/s11269-020-02560-1.

[11] Y. Wen et al. "Mobile instrument for intelligent leak detection and location on underground water supply pipelines". In: 5758 (2005). DOI: 10.1117/12.599452.

[12] *ICS-43434 MEMS Microphone Datasheet*. URL: https://invensense.tdk.com/wp-content/uploads/2016/02/DS-000069-ICS-43434-v1.2.pdf.

[13] *LSM6DS3TR MEMS IMU Accelerometer Gyroscope Datasheet*. https://www.st.com/resource/en/datasheet/lsm6ds3tr-c.pdf.

[14] *FFT Principle Explained*. https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft. Accessed: 2025-01-28.

[15] *Dynamic Time Warping Principle*. https://rtavenar.github.io/blog/dtw.html. Accessed: 2025-01-04.

[16] *SX1278*. en. URL: https://www.semtech.fr/products/wireless-rf/lora-connect/sx1278 (visited on 01/28/2025).

[17] Wikipedia. *Machine à vecteurs de support — Wikipedia, The Free Encyclopedia*. http://fr.wikipedia.org/w/index.php?title=Machine%20%C3%A0%20vecteurs%20de%20support&oldid=222311199. [Online; accessed 28-January-2025]. 2025.

[18] Yehia Khulief et al. "Acoustic Detection of Leaks in Water Pipelines Using Measurements inside Pipe". In: *ASCE JOURNAL OF PIPELINE SYSTEMS ENGINEERING AND PRACTICE* 3 (May 2012), p. 47. DOI: 10.1061/(ASCE)PS.1949-1204.0000089.