Academic Year

2021 - 2022

Localization with LoRa on 2.4 GHz

Providing the infrastructure

Louis de Looze

Bachelor's thesis to obtain the degree of Bachelor of Science in

Electronics and ICT Engineering Technology

Promotor

Prof. Maarten Weyn

Copromotor

Michiel Aernouts

University of Antwerp

Faculty of Applied Engineering

Table of Contents

Abstract: Semtech recently launched their new chip, the SX1280. This is not only a low range and a low power transceiver, but this module also provides ranging capabilities. By operating on the 2.4 GHz frequency band, the chipsets are region independent and allow for a 100% duty cycle.

In this paper, we can localize *Nodes* with multilateration by using multiple SX1280 transceivers. The developed system is a (working) framework, allowing for scalability and modularity in the future.

The infrastructure has been tested outdoors with minimal reflections and inside a warehouse. The conclusion is that this technology works better outdoor, because of the lower reflections. It has more practical use cases indoors, but it would need more complex algorithms for localization that could be based on machine learning (ML) or external sensor data.

Used technologies: Espressif ESP32, Arduino, Redis, Supabase (PostgreSQL), Python, Svelte (+ Tailwind), Docker
Deployed on: DigitalOcean, Netlify

Keywords: Multilateration, Triangulation, Localization, LoRa 2.4 GHz, Ranging
Github Repository: https://github.com/alphaLeporis/LoRa_2G4_localization

1 Preface

1.1 What is LoRa on 2.4 GHz and why use it for localization?

The SX1280 transceiver made by Semtech [1] uses the 2.4 GHz band to provide long range communication. The chipset also integrates time-of-flight functionality, allowing for ranging. Because of the low power consumption, the chipset can be integrated into small and battery-powered devices.

The radio operates in the ISM (industrial, scientific and medical) band, an unlicensed band, which means that users do not need a license and there are no restrictions on its use. Another great advantage is the global interoperation that this band provides.

According to Semtech, it would withstand heavy interference. Making it a great solution for robust and reliable wireless solutions. Research has been done by Semtech to analyse Wi-Fi [2] and Bluetooth immunity [3].

But why would we use LoRa for localization instead of well-known solutions? The greatest disadvantage of any global navigation satellite system (GNSS) is the fact that an open sky is needed. Therefore, the warehouse industry is investing in other localization systems. The cost could also be an important factor. As shown in Table 1, a GNSS module would be almost five times the price of a LoRa transceiver. Another reason could be the power consumption; the power consumption of a LoRa transceiver and a GNSS-receiver would be around the same value, but a communication layer would be needed to propagate the GNSS data.

More recent localization techniques like Wi-Fi or BLE could be compared. These technologies have a lower range, meaning there are more anchors needed in comparison to LoRa. A case study has been done by Danalto comparing both technologies, switching technologies in an environment would reduce complexity and cost [4].

	Range	**Cost**	**Power consumption**
LoRa	Up to 3km LOS	± 6 EUR	TX: $140mA$ RX: $10mA$
GNSS	Open sky needed	± 25 EUR	Acquisition: $55mA$ Tracking: $52-13mA$

Table 1: Comparison between LoRa [5] *and GNSS* [6]

2 Overview of the system

To evaluate the new chipsets by Semtech, we made a multi-node localization framework. This framework can be used as-is, but because of our modular design, every module can be interchanged to improve the overall performance.

2.1 High-level

As shown in Figure 1, in our design there are three roles in the network: *Gateway (GW)*, *Anchors* and *Nodes*.

- The *Gateway* publishes the LoRa packets to a server for further processing.
- The *Anchors* are fixed and have a cartesian coordinate saved in the database.
- The *Nodes* are mobile, and their location must be determined.

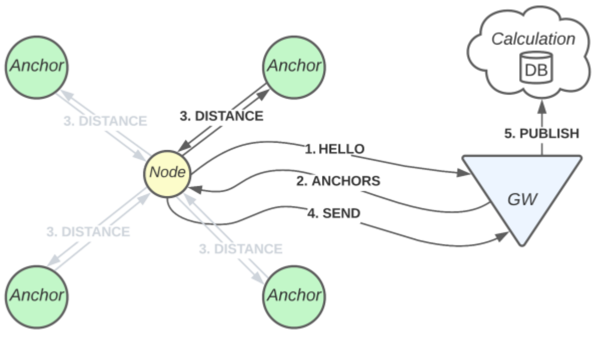


Figure 1: A high-level overview of the architecture

The following procedure is used:

1. A new *Node* joins the network and contacts the *Gateway* for information about the network.
2. The *Gateway* answers with the *Anchors* that should be on the network for ranging.
3. The *Node* stores this information in volatile memory.
4. The *Node* will do a ranging operation on the first *Anchor.*
    a. If the *Anchor* answers, the response holding multiple distances is sent to the gateway.
    b. If the *Anchor* does not answer, no response is sent.
5. The *Gateway* publishes this response on the Redis instance, including information about the *Anchor* and the *Node*.
6. A Python-based listener bundles the data and calculates the location.
7. The calculated location is stored on a persistent PostgreSQL database (Supabase [7]).

2.2 Data packets

For simplicity we will use the same packet structure during every communication:

Packet type (8b)	**Node ID** (8b)	**Anchor ID** (8b)	**Payload** (32b * [5])	**Identifier** (8b)

The packet type can be one of the following:

Type	**Name**	**Payload**	**Meaning**
0	Hello	/	*Node* tells the *GW* that it booted up.
1	Localization	5 measurements	*A node* sends these measurements to the *GW*.
2	Hello Answer	Anchor IDs	*GW* tells what *Anchors* are online and can be split into multiple packets. The Anchor ID field tells how many there are.
3	Calibrate	Real distance	Future reserved: Received by the *Anchor* to calibrate itself with another *Anchor.*

2.3 Hardware

The hardware was provided by the University of Antwerp and each box consists of an ESP32 and an Ebyte E28 (SX1280) visible in Figure 2. These components are enclosed in a sealed container with an internal battery for a full wireless deployment.

The library we used to communicate with the SX1280 chipsets is built and maintained by Stuart Robinson [8]. It is designed to work with the Arduino IDE and is (mostly) compatible with the ESP32 but is originally programmed for Atmel microcontrollers.

2.3.1 Power Consumption: LoRa chipset

To calculate the power consumption of the radio module, we will make use of the tool provided by Semtech. This tool will give an indication of the Time on Air [9]. The current draw of the different states can be found in the datasheet of the used chipset. We can map the current draw over time and calculate the weighted average.

Our network is built with two mobile *Nodes*, each waiting $100ms$ between ranging transmissions. Also, four *Anchors* are being used.

The calculations have been done with the parameters shown in Table 2.

Radio Parameters	Ebyte e28 (SX1280) [5]
Spreading Factor: 8 Bandwidth: 812kHz Code Rate: 4/5 Preamble: 8 TX Power: 10dB Packet length (struct): 24 bytes (192 bits)	TX current draw: 140mA RX current draw: 10mA

Table 2

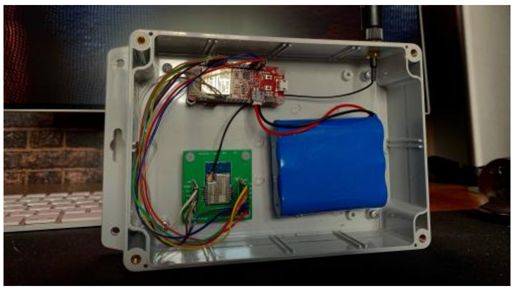


Figure 2: The internals of a LoRa chipset + microcontroller box

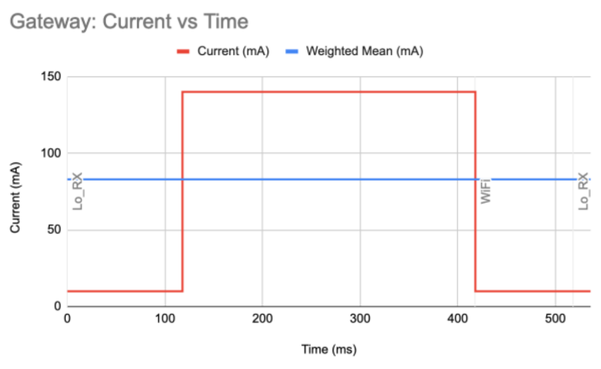
Important: Changing the Spreading Factor by one, changes the Time on Air (TOR) with factor two. Using other parameters to improve range, will have a longer Time on Air value and thus increasing the current draw.

Note: We are only working with the power consumption of the radios, excluding the controller.

2.3.1.1 Gateway (during normal operation)

The *Gateway* only receives LoRa packets, making the average power consumption of the LoRa chipset: $10mA$. Sending data over the Wi-Fi module with the ESP32 [10] consumes $140mA$.

The weighted average is 83mA and the power profile can be seen in Graph 1.



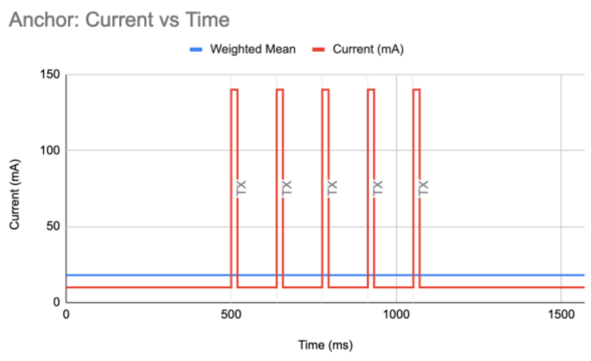
Graph 1: Power consumption over time of the Gateway

Timings
RX packet TOR: $18ms$ (calculated in Figure **4**) Delay between packets: $200ms$ Sending a packet over Wi-Fi: $300ms$

2.3.1.2 Anchor

The *Anchor* is continuously listening for arriving packets. When a ranging packet is received, an automatic reply is sent (inside the SX1280 chipset). There are five ranging communications, each with a delay of $100ms$.

The weighted average is $18mA$ and the power profile can be seen in Graph 2.



Graph 2: Power consumption over time of the Anchor

Timings

RX/TX ranging TOR: $19ms$ (calculated in Figure 3)

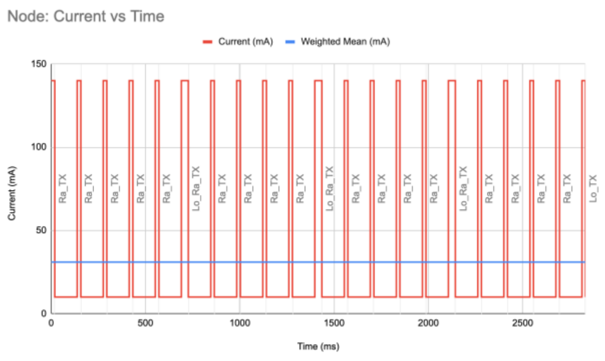
Delay between ranging packets: $100ms$

Delay between ranging sets: $1000ms$

2.3.1.3 Node (during normal operation)

The *Node* sends two types of packets: ranging and data.

The weighted average is $31mA$ and the power profile can be seen in Graph 3.



Graph 3: Power consumption over time of the Node

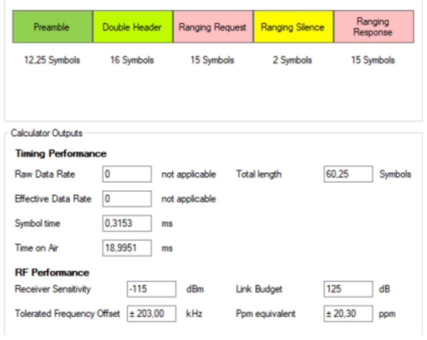
Timings

RX/TX ranging TOR: $19ms$ (calculated in Figure 3)

TX packet TOR: $18ms$ (calculated in Figure **4**)

Delay between ranging packets: $100ms$

Delay between full ranging round: $0ms$



Preamble	Double Header	Ranging Request	Ranging Silence	Ranging Response
12,25 Symbols	16 Symbols	15 Symbols	2 Symbols	15 Symbols

Calculator Outputs

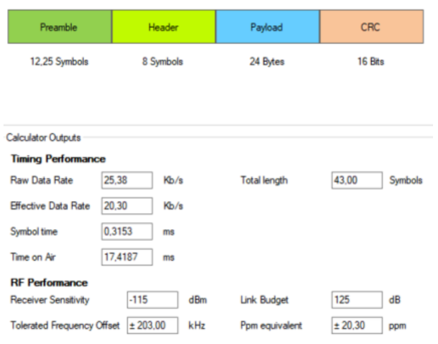
Timing Performance

Raw Data Rate	0	not applicable	Total length	60,25	Symbols
Effective Data Rate	0	not applicable			
Symbol time	0,3153	ms			
Time on Air	18,9951	ms			

RF Performance

Receiver Sensitivity	-115	dBm	Link Budget	125	dB
Tolerated Frequency Offset	± 203,00	kHz	Ppm equivalent	± 20,30	ppm

Figure 3: Semtech SX1280 tool for Ranging



Preamble	Header	Payload	CRC
12,25 Symbols	8 Symbols	24 Bytes	16 Bits

Calculator Outputs

Timing Performance

Raw Data Rate	25,38	Kb/s	Total length	43,00	Symbols
Effective Data Rate	20,30	Kb/s			
Symbol time	0,3153	ms			
Time on Air	17,4187	ms			

RF Performance

Receiver Sensitivity	-115	dBm	Link Budget	125	dB
Tolerated Frequency Offset	± 203,00	kHz	Ppm equivalent	± 20,30	ppm

Figure 4: Semtech SX1280 tool for LoRa

2.3.2 Improvements: Microcontroller

An important consideration is the choice of the microcontroller. Right now, all hardware is identical, but we would advise altering the hardware configuration to lower the cost, complexity, and power consumption.

- On the *Anchors*, all the processing is done on the SX1280 chip. This means that almost no processing needs to be done on the microcontroller. An ESP32 is thus plenty powerful, an ATTiny85 would be enough in this case.
- On the *Node*, some processing is done. But no other connectivity is needed, so an ATMega328 would be a great choice. It has more connectivity than the ATTiny85, allowing for external sensors (to improve the localization).
- The *Gateway* needs the most processing power. It processes the LoRa packets and sends them asynchronous over Wi-Fi. For this reason, an ESP32 is a great choice because of the two cores.

A comparison has been done in Table 3.

	Original	**Proposal**
Gateway	ESP32	/
Anchor	ESP32	ATTiny85 ($8\ MHz - 3.3V$)
	$20mA \sim 31mA$	$3mA \sim 4mA$
Node	ESP32	ATMega328/168 ($8\ MHz - 3.3V$)
	$20mA \sim 31mA$	$3mA \sim 4mA$

Table 3: Comparison between the original hardware and our proposal.

An improvement that could be done with the gateway is adding an ethernet port with Power over Ethernet (PoE). This would improve the latency and stability and the device would get its power through ethernet. PoE switches are becoming very ubiquitous, this would allow for fast deployment.

When running the *Anchor* on a 6600mAh Li-Po battery, it would be able to run for about 5 days. With our proposed controller, the anchor could run for 12 days.

If the goal is to make the *Node* a wearable, a small Li-Po battery (350mAh) must be used. The battery life would be 5 hours. With our proposed controller it would be 9 hours (an entire working day).

2.4 Software

Our goal is to create a framework where each module can be interchanged without designing a completely new system. This section will be an overview of every module that is needed to provide the whole localization service. As shown in Figure 5, the system is split into three parts.

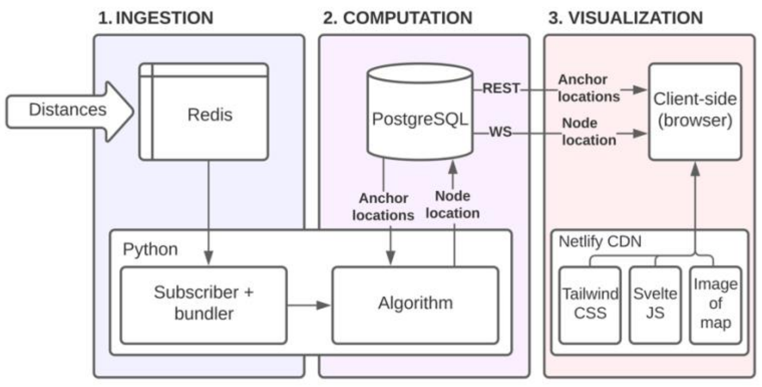


Figure 5: Overview of the software stack.

2.4.1 Data Ingestion

The Data Ingestion system is based on Redis [11]. It is an in-memory storage system with a message broker. Another often used pub-sub protocol is Message Queuing Telemetry Transport (MQTT) [12], but because of our prior knowledge of Redis, we chose this route. The Redis instance is deployed inside a Docker container that can be deployed locally or in a Virtual Private Server (VPS). During testing, we used a Digital Ocean Droplet for deployment.

The *Gateway* will "publish" the distance readings on the Redis Instance, there they will not be stored. Only the currently subscribed listeners will be able to capture the messages. Alongside Redis, there is a Python script "subscribing" and allowing it to compute the captured messages.

2.4.2 Data Computation

When there is a desire to improve the localization results, this is the module where a lot can be done. Right now, there is a pure mathematical multilateration algorithm that calculates the location based on the new data. The algorithm used is developed by Kamal Shadi [13]. Our goal was to create a framework for localization, tough in the future, ML-based algorithms could be used to improve the correctness of the results.

All the computation happens inside of a Python script. This stage is composed of three steps, visualised in Figure 6.

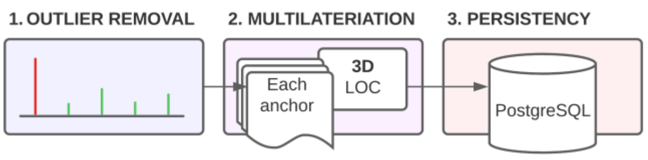


Figure 6: Overview of the data computation module

- **Outlier Removal**: A Standard Deviation outlier removal is being used to remove wrong results. The resulting distances will be averaged out to minimize errors.
- **Multilateration**: The data from each (in range) node is used to calculate a cartesian position.
- **Persistency**: The final position is saved inside the PostgreSQL database (Supabase [7]).

2.4.3 Data Visualization

The visualization happens client side in the browser and is written in Svelte [14], a lightweight JavaScript framework.

This system is designed to run serverless, meaning that the cloud provider is responsible for executing the code by dynamically allocating the resources. Because of this architecture, this code was compiled and deployed on Netlify and available online.

To make the data visualization live, we made use of WebSockets [15]. When data is added to the database, it will be directly delivered to the browser. When the node moves, the location can be tracked live without reloading the page. A screenshot of the page is shown in Figure 7.

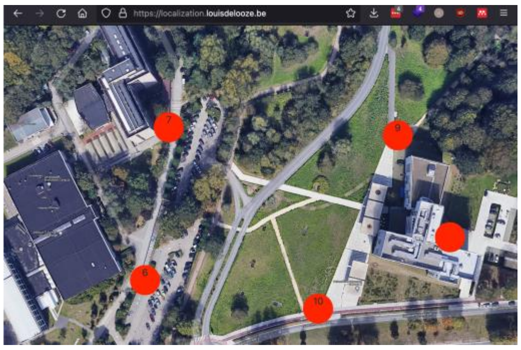


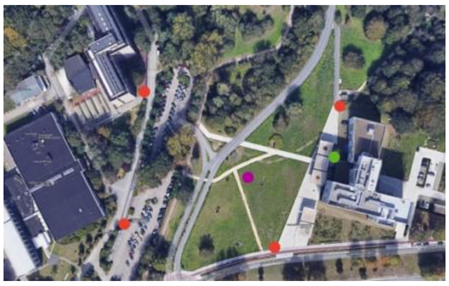
Figure 7: The visualization as a web page

3 Results

All experiments have been done with the Radio parameters from Table 2.

3.1 Outdoor experiment

This experiment has been done on the field of Campus Groenenborger between Z – V and has been done two times.



Area: around 150x130m

Line Of Sight (LOS): mostly

Anchors (Red): fixed 2m in the air (near metal)

Node (Magenta): walked around and held high in the air

Gateway (Green): static, connected to Wi-Fi

Calibration: compared reading with a known distance

Because the experiment is done outdoors GNSS can be used. We used an Apple Watch to track our movements, this is shown as a red line in Figure 8 and Figure 9. The yellow squares show the locations where we stood still for a while, allowing for better localization.

The magenta triangles visualize the calculated locations found with LoRa.

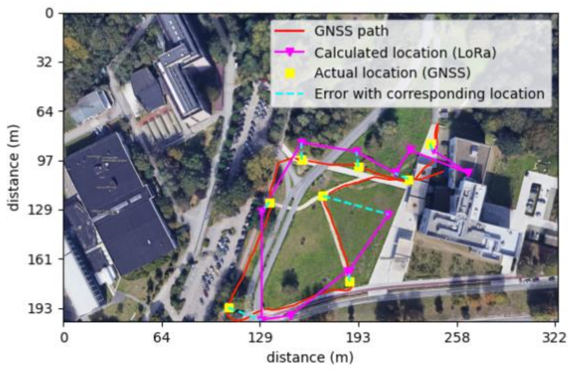


Figure 8: Experiment 1

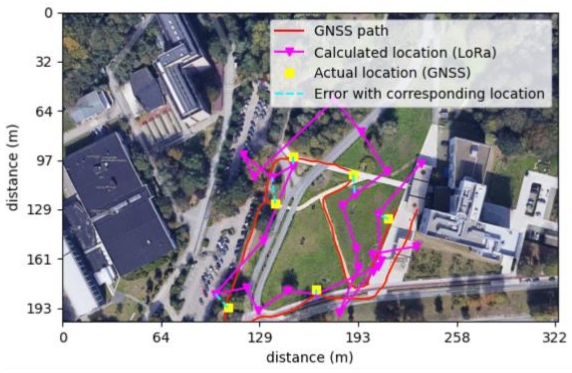


Figure 9: Experiment 2

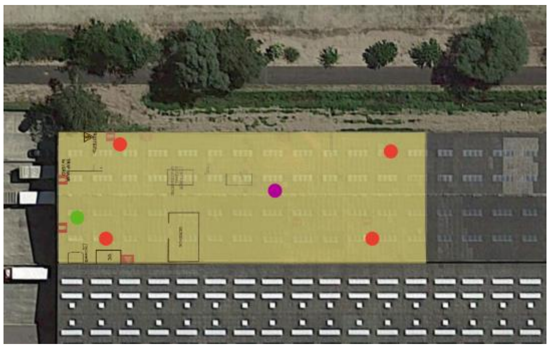
The average error in Figure 8 between a yellow square and a magenta triangle is $14.8m$, in Figure 9 the average error is $9.8m$. Visually we could conclude that the results of experiment 2 seem worse than the first, this is because we used a smaller delay between transmissions. This leads to more tracking points than reference points, that we cannot compare.

We can conclude that the localization works well. This is mostly due to the large distances and near LOS. Some optimisations can be done:

- Testing different radio parameters.
- Putting the *Anchors* on poles, with less interference from surrounding metal structures.
- Improving the algorithm by looking at the previous points and checking if the new point is reliable.

3.2 Indoor experiment

The indoor experiment has been done in a furniture warehouse.



Warehouse size: 25m x 75m

LOS: No, a lot of boxes + metal structures

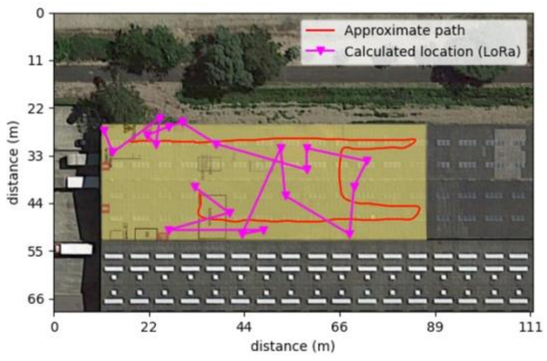
Anchors (red): 1m or 3m in the air

Node (magenta): walked around and held high in the air

Gateway (green): near the Wi-Fi AP

Calibration: None (same as outdoor)

Because the experiment is done indoors GNSS cannot be used. We measured the distances of my itinerary and marked them on the map. The magenta triangles visualize the calculated locations found with LoRa.



Here, the results are less promising. Calculating the average error is also harder because of the lack of ground truth. There are multiple reasons for these results:

- The warehouse is small. This means that the error will be larger on the distances as found by Stuart Robinson: “A variation of ±10m at 100m distance is not unusual” [8]
- The placement of the antennas was not optimal, this had an impact on the RF performance.
- We used the same radio settings as outdoors. Using other settings might give better results.
- Changing the localization algorithm to a fingerprinting based one might give more accurate results. Primarily because of the number of reflections.

4 Conclusion

This technology has a lot of potentials. Because of the large range and low power consumption it can be deployed very easily in any environment. Using different controllers for each role in the network would drastically improve the deployment cost and complexity. If the initial infrastructure has been installed, adding a new mobile *Node* is easy and low-cost.

But the ranging is not accurate enough at small distances, leading to large errors. This could be solved by using other (more advanced) localization algorithms. At greater distances, the results are more usable and could be improved with better calibration. Data rates are also too slow for person or vehicle tracking, but asset tracking would be a great use case. This was not the goal of our research though, setting up a framework was.

The fundamentals of a localization system have been made and our research has proven that further development is worth the time/cost investment.

5 Bibliography

[1] "SX1280 | Long range, low power 2.4GHz RF Transceiver | Semtech." https://www.semtech.com/products/wireless-rf/lora-24ghz/sx1280 (accessed May 21, 2022).

[2] "Wi-Fi Immunity of LoRa at 2.4GHz," *Semtech*, 2017. https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/44000000MDcO/Ll4bon.4HPwcyXv9fegcfcgbpvLYd7Lx_aZLMzYNLIQ (accessed May 24, 2022).

[3] "Bluetooth Immunity of LoRa at 2.4GHz," *Semtech*, 2018. https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R000000HSNz/HeroVQVfOkV2KaLGS7BiPaFWWoCXVsrP3rFKf9hs6JQ (accessed May 24, 2022).

[4] "Location Tracking: Out With the Old and In With LoRa® 2.4 GHz - danalto." https://www.danalto.com/location-tracking-new-vs-old/ (accessed May 24, 2022).

[5] Ebyte, "Datasheet Ebyte E280-2G4T12S SX1280," 2021, Accessed: May 21, 2022. [Online]. Available: https://www.cdebyte.com/Uploadfiles/Files/2022-1-12/2022112139253269.pdf

[6] OriginGPS, "OriginGPS Hornet Datasheet," 2017. https://origingps.com/wp-content/uploads/2022/03/Multi-Micro-Hornet-ORG1510-R02-Datasheet-Rev-2.1.pdf (accessed May 21, 2022).

[7] "The Open Source Firebase Alternative | Supabase." https://supabase.com/ (accessed May 24, 2022).

[8] "GitHub - StuartsProjects/SX12XX-LoRa: Library for SX12XX LoRa devices." https://github.com/StuartsProjects/SX12XX-LoRa (accessed May 23, 2022).

[9] Semtech, "Using the SX1280 in Low Power Applications," 2018. https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R000000HSO9/vmLAj4haZE78QozA8W0mnLRc1WY.SDQ7RYewLFF3k8k (accessed May 23, 2022).

[10] "ESP32 Series Datasheet Including," 2022, Accessed: May 23, 2022. [Online]. Available: https://www.espressif.com/en/support/download/documents.

[11] "Redis." https://redis.io/ (accessed May 24, 2022).

[12] "MQTT - The Standard for IoT Messaging." https://mqtt.org/ (accessed May 24, 2022).

[13] "GitHub - kamalshadi/Localization: Multilateration and triangulation for target localization." https://github.com/kamalshadi/Localization (accessed May 23, 2022).

[14] "Svelte • Cybernetically enhanced web apps." https://svelte.dev/ (accessed May 24, 2022).

[15] "The WebSocket API (WebSockets) - Web APIs | MDN." https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (accessed May 24, 2022).