TECHNICAL UNIVERSITY OF CRETE

DIPLOMA THESIS

Multi-static localization with commodity Software Defined Radios (SDRs)

Author:
Dimitrios ANGELOU

Thesis Committee:
Prof. Aggelos BLETSAS
Prof. Athanasios LIAVAS
Prof. George KARYSTINOS

A thesis submitted in fulfillment of the requirements
for the diploma of Electrical and Computer Engineering

in the

School of Electrical and Computer Engineering
Telecommunications Laboratory

January 8, 2025

TECHNICAL UNIVERSITY OF CRETE

Abstract

School of Electrical and Computer Engineering

Electrical and Computer Engineering

Multi-static localization with commodity Software Defined Radios (SDRs)

by Dimitrios ANGELOU

Passive radio frequency identification (RFID) tag localization has been mostly examined for monostatic and synchronized setups. This work takes it a few steps further and performs real-time, distributed, multistatic localization of such tags. It succeeds by exploiting the recently-proposed elliptical direction-of-arrival-based (EllDoA) localization algorithm, designed for multi-antenna setups, with however, synchronized radios at the carrier level. This work also employs a carrier frequency offset (CFO) mitigation method and demonstrates its real-time feasibility even with cheap and distributed, unsynchronized software defined radios (SDR). The inherent carrier phase offset (CPO) problem of such setups is also addressed, and it is experimentally shown that the calibration step needs to be done only once, and not for further reruns of the experiment, rendering it ideal for practical applications. As a collateral dividend, a complete GNURadio-based software solution was developed for distributed, unsynchronized and real-time localization. It was also shown that an unsynchronized setup with ultra-low cost SDRs is possible to perform localization with a 15-20% impact on accuracy, compared to the centrally-synchronized case.

TECHNICAL UNIVERSITY OF CRETE

Abstract

School of Electrical and Computer Engineering

Electrical and Computer Engineering

Multi-static localization with commodity Software Defined Radios (SDRs)

by Dimitrios ANGELOU

Ο εντοπισμός παθητικών ετικετών RFID έχει μελετηθεί κυρίως για μονοστατικές και συγχρονισμένες τοπολογίες. Αυτή η πτυχιακή έχει σκοπό να προχωρήσει την έρευνα σε αυτό τον τομέα ερευνώντας το κατανεμημένο, πολυστατικό εντοπισμό RFID ετικετών σε πραγματικό χρόνο. Το επιτυγχάνει χρησιμοποιώντας τον αλγόριθμο εντοπισμού με βάση την ελλειπτική κατεύθυνση άφιξης (EllDoA), ο οποίος έχει σχεδιαστεί για διατάξεις πολλαπλών κεραιών, αλλά συγχρονισμένων σε επίπεδο φέροντος. Το έργο αυτό χρησιμοποιεί μια μέθοδο αντιμετώπισής της ολίσθησης συχνότητας φέροντος (CFO) και αποδεικνύει τη δυνατότητα υλοποίησής της σε πραγματικό χρόνο ακόμη και με φτηνά, κατανεμημένα, μη συγχρονισμένα ραδιόφωνα ελεγχόμενα από λογισμικό (SDR). Το εγγενές πρόβλημά της απόκλισης φάσης φέροντος (CPO) αντιμετωπίζεται επίσης, και αποδεικνύεται και πειραματικά ότι το βήμα βαθμονόμησης αρκεί να εκτελεστεί μόνο την πρώτη φορά, και όχι ξανά σε επόμενες επαναλήψεις του πειράματος. Ως παράπλευρο κέρδος, υλοποιήθηκε ένα πλήρες πακέτο λογισμικού γύρω από το GNURadio για μη συγχρονισμένο εντοπισμό σε πραγματικό χρόνο. Τελικά, φάνηκε ότι το πείραμα στην περίπτωση των ασυγχρόνιστων ραδιοφώνων πολύ χαμηλού κόστος ερχόταν με κόστος ακρίβειας της τάξης των 15-20% σε σχέση με τα συγχρονισμένα ραδιόφωνα.

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Prof. Bletsas for giving me the opportunity to work in such an interesting field and project. By his side I learned how true research is conducted, gained access to equipment I wouldn't have been able to otherwise, but most importantly had a person I could trust all the way through my thesis. His approachability, guidance, and academic knowledge were invaluable throughout this journey. He had the perfect balance of knowing when to let me work at my own pace and when to challenge me to push beyond my limits.

Dr. Bletsas also gave me the opportunity to get a 10-week internship at WINLAB, Rutgers in NJ. An amazing experience, not only academically but also personally, allowing me to greet the academic and cultural world of the USA, where I met amazing people that I will forever remember.

Moreover, I would like to thank my thesis committee, Dr. George Karystinos, and especially Dr. Athanasios Liavas, who stepped in at the very last moment to officially take on the role of my academic supervisor due to Dr. Bletsas's leave in the United States.

I also want to thank my parents Maria and Thomas, that lovingly supported me throughout all this time every way possible. My close friends and partners in crime Rafael, Odysseas, Alex, Alexandros, Sif, Stavros, with whom we shared the greatest experiences all these years and made 5 years feel like 5 seconds. My girlfriend Ioanna, whose unwavering understanding and support meant the world to me. The lab-rats Anastasis and Giorgos that made every morning, afternoon and evening at the lab certainly not boring. Vougioukas, who is a true inspiration for me, and every person I met through this journey that gave me something to hold for the rest of my living life. Thank you.

Angelou Dimitris, Chania, 2024

Contents

Dedicated to curiosity, persistance and knowledge...

Chapter 1

Introduction

1.1 RFID technology

Radio Frequency Identification (RFID) technology has emerged as a critical component in a wide array of applications, ranging from supply chain management and inventory control to security and healthcare. RFID systems are designed to identify and track objects, animals, or people automatically through the use of radio frequency waves. This technology leverages small electronic devices known as RFID tags, which store data that can be remotely accessed by RFID readers. The data stored in RFID tags can include product information, location details, and unique identification numbers, enabling seamless data exchange and real-time monitoring.

1.1.1 Components of RFID Systems

An RFID system typically comprises three main components:

1. **RFID Tags**: These are the primary data carriers within the system. RFID tags are further categorized into three types based on their power source: active, semi-passive, and passive tags.
2. **RFID Reader**: Also known as an interrogator, the reader emits radio waves to communicate with the tags. The reader captures the data from the tag and transmits it to a computer or database for processing.
3. **Antenna**: The antenna facilitates communication between the reader and the tag by emitting and receiving radio frequency signals. Depending on the configuration, antennas can be integrated within the reader or be externally positioned.

1.1.2 RFID Tags: Types and Characteristics

Passive RFID Tags

Passive RFID tags do not have an internal power source. Instead, they derive their power from the electromagnetic field generated by the RFID reader. When the reader emits a radio frequency signal, it induces a small current in the tag's antenna, which powers the tag's microchip to send back the stored data. Passive tags are typically less expensive and smaller compared to active tags, making them suitable for a wide range of applications, particularly in inventory tracking and asset management. However, they have limited read ranges, typically from a few centimeters to several meters, depending on the frequency used.

Active RFID Tags

Active RFID tags are equipped with their own internal power source, usually a battery, which enables them to broadcast signals autonomously. This capability allows active tags to have a much longer read range, often up to hundreds of meters, making them ideal for tracking large assets or in environments where the reader cannot be placed in proximity to the tags. Active tags can also store more data and offer higher data transmission rates. However, their larger size, higher cost, and the need for periodic battery replacement or recharging are significant drawbacks.

Semi-Passive RFID Tags

Semi-passive RFID tags, also known as battery-assisted passive (BAP) tags, occupy a middle ground between passive and active tags. These tags have an internal battery like active tags, but the battery is used to power the tag's circuitry rather than to broadcast signals. The tag still relies on the reader's signal to initiate communication. Semi-passive tags offer greater read ranges and better performance in environments with interference compared to passive tags, while still being less costly and smaller than active tags. They are particularly useful in applications that require sensors, such as temperature or humidity monitoring.

1.1.3 RFID Frequencies and Regional Regulations

RFID systems operate across a variety of frequency bands, each with its own characteristics and applications. The frequency at which an RFID system

operates significantly impacts its performance, including read range, data transfer rate, and the ability to penetrate materials.

Low Frequency (LF)

LF RFID systems operate at frequencies between 30 kHz and 300 kHz, with 125 kHz and 134.2 kHz being the most commonly used. LF systems have a short read range, typically up to 10 cm, but they are highly resistant to interference from liquids and metals. This makes LF RFID suitable for applications like animal tracking, access control, and industrial automation. LF RFID systems are used globally with minimal variation in frequency allocation.

High Frequency (HF)

HF RFID systems operate at 13.56 MHz and offer a read range of up to 1 meter. HF systems provide a faster data transfer rate than LF systems and are also more resistant to interference, although not as much as LF systems. HF RFID is widely used in applications such as contactless payment systems, smart cards, and library book tracking. The 13.56 MHz frequency is standardized globally, making HF RFID systems universally applicable.

Ultra-High Frequency (UHF)

UHF RFID operates between 300 MHz and 3 GHz, with 860 MHz to 960 MHz being the most commonly used range. UHF RFID systems offer long read ranges, typically up to 12 meters or more, and fast data transfer rates, making them suitable for supply chain management, vehicle tracking, and large-scale inventory management. However, UHF signals are more susceptible to interference from liquids and metals. UHF RFID frequency allocation and power limitations vary by region (as of 2024):

- **United States** 902–928 MHz @ max 4W EIRP
- **Greece** 865.6–867.6 MHz @ max 2W ERP

Microwave Frequency

Microwave RFID systems operate at frequencies above 1 GHz, typically at 2.45 GHz. These systems offer high data transfer rates and are used in specialized applications such as electronic toll collection and secure access control. However, they have a shorter read range compared to UHF and are

more affected by environmental factors.

1.1.4 RFID Configurations: Monostatic vs. Bistatic

The configuration of RFID systems can significantly affect their performance, particularly in terms of read range and signal clarity. The two primary configurations commercially used are monostatic and bistatic.

Monostatic Configuration

In a monostatic configuration, the RFID reader uses a single antenna for both transmitting the interrogation signal and receiving the response from the RFID tag. This configuration is simpler and more cost-effective, making it suitable for applications where space is limited or where the system does not require long read ranges.

Bistatic Configuration

In a bistatic configuration, separate antennas are used for transmitting and receiving. This setup can improve the system's sensitivity and read range, as the transmitting and receiving functions are optimized independently. Bistatic configurations are often used in applications where high accuracy and long-range detection are critical, such as in toll collection systems or large-scale asset tracking.

This thesis aims to dive deeper into the multistatic architecture.

Multistatic Configuration

A multistatic RFID configuration involves the use of multiple antennas for both transmitting and receiving signals, unlike monostatic and bistatic setups that use one or two antennas respectively. In multistatic systems, several transmitting and receiving antennas are strategically placed to cover a larger area and provide more robust detection capabilities.

They key differences are:

- Coverage and Range: Multistatic systems can cover larger and more complex areas, offering enhanced spatial coverage and better detection accuracy across various angles and positions.

- Complexity: The configuration and operation of multistatic systems are more complex compared to monostatic and bistatic setups. They require precise synchronization and calibration among multiple antennas to ensure effective communication and data collection.
- Cost and Infrastructure: Multistatic configurations typically demand more infrastructure, including additional antennas, processing and synchronization equipment, which can increase the overall cost and complexity of the system.

The aim of this thesis is to improve upon the multistatic configuration, exploiting it's advantages (benefiting from the range increase it offers) while also battling the core problem it introduces (added installation, infrastructure, and configuration cost).

1.2 Software-Defined Radio (SDR) Systems

Software-Defined Radio (SDR) is a transformative technology that has revolutionized the field of wireless communication by replacing traditional hardware based radio components with software-driven solutions. In SDR systems, key functions such as modulation, demodulation, filtering, and signal processing are implemented in software, allowing for a high degree of flexibility and adaptability. This flexibility makes SDRs capable of supporting a wide range of frequencies and protocols, making them an invaluable tool for research, development, and experimentation in the field of radio communications.

1.2.1 Concept of SDRs

At its core, an SDR consists of a general-purpose hardware platform that includes radio frequency (RF) front-end component, analog-to-digital converters (ADCs) and digital-to-analog converters (DACs). These components interface with software that can be programmed to perform various radio functions, such as transmitting or receiving signals across different frequency bands. Unlike traditional radios, where changes to the functionality often require physical modifications or the addition of new hardware, SDRs can be reconfigured or updated through software alone, making them highly versatile.

1.2.2 Capabilities of SDRs

SDRs come with a wide range of capabilities, varying based on the specific model and its intended use. Some SDRs are designed solely for receiving signals (RX), while others are capable of both transmitting (TX) and receiving, making them suitable for a broader range of applications. Additionally, advanced SDRs offer features such as Multiple Input Multiple Output (MIMO) capabilities, which allow for the simultaneous use of multiple antennas, improving communication performance and enabling advanced techniques like beam forming. Synchronization across multiple SDRs can also be achieved using external devices like the OctoClock, which provides precise timing for coordinated operation in complex systems.

Examples of SDR Platforms

USRP (Universal Software Radio Peripheral) Series

The USRP series, developed by Ettus Research, is one of the most widely used SDR platforms in both academic and industrial settings. USRP devices are known for their high performance and flexibility, making them suitable for a wide range of applications, including signal research, prototyping, and educational purposes.

- **USRP X310**: The X310 is a high-end SDR that offers wide bandwidth, high dynamic range, and dual MIMO capabilities. It supports a range of daughterboards, allowing users to cover frequencies from DC to 6 GHz. The X310 is often used in advanced research projects and can be synchronized with other USRPs using the OctoClock for large-scale experiments.
- **USRP N200/N210**: These mid-range SDRs offer high flexibility and performance, suitable for general-purpose applications. They support a range of RF daughterboards and can cover frequencies from DC to 6 GHz. The N200/N210 models are commonly used in academic research and teaching.
- **USRP B200/B210**: These are lower-cost, portable SDRs with integrated RF capabilities, making them ideal for mobile applications. The B210, in particular, supports full-duplex MIMO and covers a wide frequency range, making it popular in both educational and hobbyist settings.

RTL-SDR and Clones

RTL-SDR is an extremely low-cost Software-Defined Radio that originated as a simple USB DVB-T (Digital Video Broadcasting - Terrestrial) dongle intended for TV reception. However, it was soon discovered that these devices could be repurposed for general SDR use, leading to a surge in popularity, particularly among hobbyists, beginners, and those entering the world of SDR. Despite its low price, typically under 30€, the RTL-SDR has become a staple in the SDR community due to its affordability, ease of use, and wide availability.

The RTL-SDR is a single-channel receiver, meaning it can only receive one signal at a time. It does not have the capability to transmit (TX), making it suitable exclusively for receiving and analyzing signals. Its typical range is from 22MHz to 1.75 GHz, covering a wide array of bands, including AM, FM, HF, VHF, and UHF, which makes it versatile for various applications such as monitoring radio broadcasts, weather stations, and even some satellite communications. It also offers a maximum instantaneous bandwidth of approximately 2.4 MHz, which is sufficient for many common SDR applications but may be limiting for more complex or high-bandwidth requirements.

The original RTL-SDR design has undergone several iterations, with the developers continuously improving the device to enhance its performance and usability. The latest version, currently Version 4, includes various improvements such as better frequency stability, lower noise floor, and improved sensitivity.

Given the popularity of the RTL-SDR, clones and variations have emerged, with varying levels of quality. While some clones offer slight performance improvements or additional features, others suffer from poor build quality and reduced reliability. These quality issues can significantly impact the performance of the device, leading to issues such as increased noise, reduced sensitivity. Most of them don't support bias tee, have worse thermal dissipation, limited frequency range and other problems, that make them a bad choice of purchase considering that the cost is almost the same, sometimes even higher than the original platform.

LimeSDR LimeSDR is a highly flexible SDR platform that supports both TX and RX, covering a wide frequency range from 100 kHz to 3.8 GHz. It offers dual MIMO channels and is compatible with a wide range of software tools, making it suitable for a variety of applications, including cellular network research, satellite communications, and more.

BladeRF BladeRF is another versatile SDR platform that supports full-duplex operation and covers a wide frequency range from 47 MHz to 6 GHz. It is known for its high performance and is used in more demanding applications, such as LTE and GSM network development.

HackRF One HackRF One is a popular mid-range SDR that offers both TX and RX capabilities, covering frequencies from 1 MHz to 6 GHz. It is designed to be an affordable yet versatile tool for experimentation with a wide variety of radio protocols.

1.2.3 Cost Considerations

The cost of SDRs varies widely based on their capabilities and intended use. Low-cost options like RTL-SDRs can be acquired for under $30, making them accessible to beginners and hobbyists. Mid-range devices like HackRF One and LimeSDR typically cost between $200 and $500, offering more features and better performance. High-end SDRs like the USRP X310 can cost several thousand dollars, but they provide the performance and flexibility needed for advanced research and commercial applications.

1.2.4 Academic Applications and Benefits

SDRs are particularly valuable in academic settings due to their flexibility and the ability to experiment with a wide range of communication protocols and frequencies. They allow students and researchers to design, implement, and test complex wireless systems without the need for expensive and specialized hardware. SDRs can be used to simulate real-world scenarios, develop new communication algorithms, and explore cutting-edge technologies such as 5G, cognitive radio, and software-defined networking.

Moreover, the availability of open-source software tools, such as GNU Radio, further enhances the utility of SDRs in education, providing a platform for learning and experimentation. The combination of software flexibility and hardware versatility makes SDRs an essential tool for academic research, enabling innovative work in fields such as telecommunications, signal processing, and wireless communication.

1.3 GNU Radio

GNU Radio is a powerful and flexible open-source software development toolkit that provides signal processing blocks to implement software-defined radios (SDRs) and other communication systems. It enables users to design, simulate, and deploy a wide range of radio frequency (RF) systems entirely in software, allowing for the creation of sophisticated and customizable signal processing applications without the need for specialized hardware. Another key strength of GNU Radio is its cross-compatibility with Linux, Windows and macOS enabling the development and deployment of projects to almost any commercial PC.

1.3.1 How GNU Radio Works

GNU Radio operates by providing a graphical programming environment known as GNU Radio Companion (GRC) and a comprehensive library of signal processing blocks that can be connected to create complex signal flow graphs. These flow graphs define the operations and transformations applied to the input signals, which can be visualized, processed, or transmitted through connected SDR hardware. The modular architecture of GNU Radio allows users to mix and match different blocks, enabling the rapid prototyping and testing of communication systems. It is not only limited by the GUI, as flowcharts can be edited and developed directly from a code editor giving even more versatility to the programmer.

1.3.2 Out-of-Tree (OOT) Modules

A unique feature of GNU Radio is its support for Out-of-Tree (OOT) modules, which allow users to extend the functionality of GNU Radio by developing custom signal processing blocks. These modules can be written in either C++ or Python, depending on the performance requirements and developer preference.

C++ offers much greater performance (and is used in this project) while python is better suited for simpler blocks and rapid prototyping.

OOT modules are commonly shared, usually through git hosting platforms, enabling enthusiasts and developers to find interesting projects to use their SDR's and improve upon.

1.3.3 SDR Hardware Support

Another huge benefit of GNU Radio is its support for many SDR families. Both official and unofficial modules exist for providing support for any of the most common SDR's (Ettus USRPs, RTL-SDRs, HackRF, BladeRF, LimeSDR e.t.c.). Due to its architecture, if two SDRs provide similar capabilities, any project can be very easily adapted to working with another family of SDRs making easy for projects to scale and lower in cost.

Chapter 2

RFID protocol and implementation

The RFID tags we are using comply to the Gen2 EPC protocol. The Gen2 EPC (Electronic Product Code) RFID protocol, also known as EPCglobal Class-1 Generation-2 (C1G2), is a widely adopted standard for passive RFID systems. Developed and maintained by EPCglobal, defines the communication between RFID readers (interrogators) and RFID tags. It operates in the UHF (Ultra High Frequency) band, typically within the frequency range of 860 MHz to 960 MHz, depending on regional regulations. Most of the information about the protocol provided derive from the latest release 3.0 of 2024 [1].

2.1 Gen2 EPC specification

The official Gen2 protocol specifies that an interrogator can modulate an RF carrier using single-sideband amplitude shift keying(SSB-ASK), double-sideband amplitude shift keying (DSB-ASK), or phase-reversal amplitude shift keying (PR-ASK) using a pulse-interval encoding (PIE) format. Out of these, the one used in this project is DSB-ASK as it's the simplest to implement by just turning on and off the RF carrier.

The tags harvest the energy during of the RF carrier and modulate the amplitude and/or phase of the continuous wave (CW) using either FM0 or Miller encoding, selected by the interrogator which listens for the backscattered reply when transmitting the unmodulated RF carrier.

The tags operate at the 860 – 960 MHz range depending on the region. For the experiments run in Greece 866 MHz were used, and 910 MHz while in the United States in accordance to the local laws and regulations.

The interrogator to tag link uses PIE as shown in fig: 2.1 where high values represent the transmitted CW signal and the low values the attenuated

(stopping the transmission in my case). Tari is a reference time base for the Interrogator-to-Tag signaling, it must fall in the range between 6.25μs and 25μs and in my case opted for 25μs as it's on the upper limit and need the least resources.

All interrogator to tag communication begin with either a preamble or a frame sync. The preamble prepends the query command which initiates an inventory round and the frame sync prepends every other signalling.

The preamble (as shown on figure 2.2) consists of: (1) a fixed duration delimiter (12.5μs +-5%), (2) a data-0 symbol, (3) the R-T calibration symbol and (4) the T-R calibration symbol.

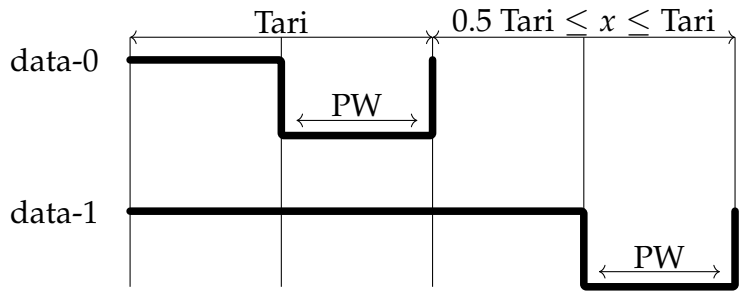


FIGURE 2.1: PIE: symbols

The RTcal must equal the duration of the data-0 plus the duration of the data-1 symbol. The tag will interpret subsequent symbols of length less than half of RTcal as data 0s and longer ones as data 1s. Anything longer than 4RTcal is considered invalid.

The TRcal is specified using the DR ratio which is specified by the interrogator in the query. The tag backscatter link frequency is defined by equation 2.1 where BLF must be one of some specific values and tolerances defined in the protocol.

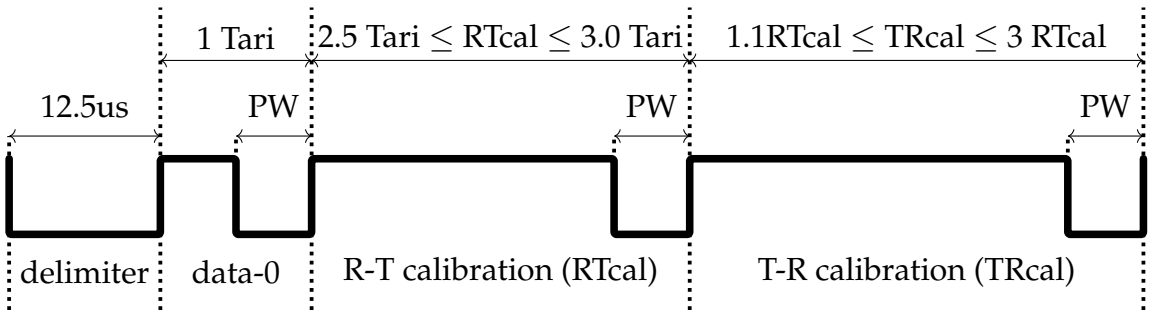


FIGURE 2.2: Interrogator preamble

The frame-sync is identical to the preamble but without the TRcal symbol, as shown on figure 2.3.

$$BLF = \frac{DR}{TRcal} \tag{2.1}$$

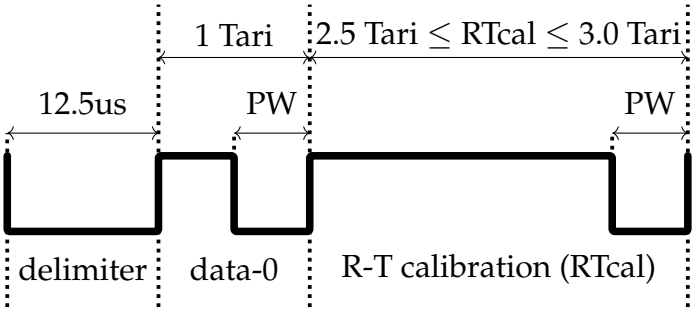


FIGURE 2.3: Interrogator Frame-Sync

A Tag shall backscatter using a fixed modulation format, data encoding, and data rate for the duration of an inventory round where the data encoding and rate are specified by the interrogator through the query command. On the figures concerning the tag transmissions, low level refers to the absorption state and high level to the reflective state.

The tag's data encoding options for the backscatter data are FM0 and Miller modulated subcarrier. This work relies on FM0 which was selected because it was slightly easier to decode and implement even though it can be more error prone.

FM0 inverts the phase at every symbol boundary, with the data-0 having an extra phase inversion in the middle of the symbol. There are four possible FM0 symbols (figure 2.5) denoted by the symbol states as shown on figure 2.4. There are 4 possible states S_1, S_2, S_3, S_4. The possible transitions are depicted on 2.4 where some transmissions are disallowed (e.g. S_2 to S_3) as they don't adhere to the rule of the phase inversion at the symbol boundary.

All tag to receiver command signalling must begin with one of two possible preambles selected by the interrogator through a bit (TRext) set in the query, a short and a longer one. In this work the preamble opted for is the shorter one (1010V1) as shown on 2.6. The preamble is really important in this work as this is what is used to detect the phase of the tag path.

The query command that has been mentioned is a 22 bit structure used to initiate the inventorying and initialize most of the communication parameters.

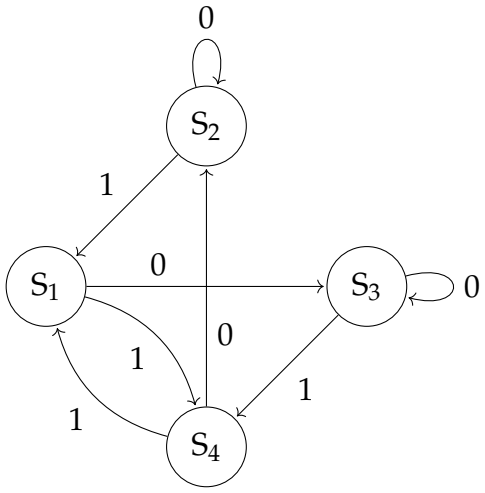


FIGURE 2.4: FM0 state diagram

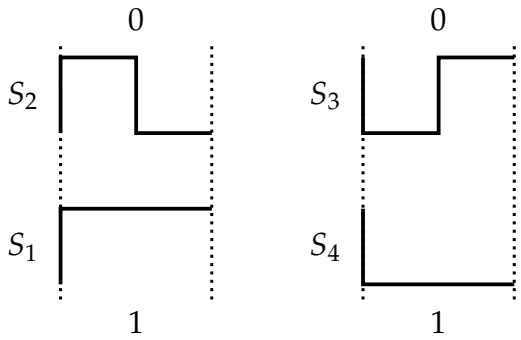


FIGURE 2.5: FM0 symbols

On figure 2.1 the parts of the query command can be seen and bolded are the value that are set in this work.

Command	DR	M	TRext	Sel	Session	Target	Q	CRC
1000	**0: DR=8** 1: DR=64/3	**00: M=1** 01: M=2 10: M=4 11: M=8	**0: No pilot tone** 1: Use pilot tone	**00: All** 01: All 10: ∼SL 11: SL	**00: S0** 01: S1 10: S2 11: S3	**0: A** 1: B	**0**-15	CRC-5 **10000**

TABLE 2.1: Query commnand

- DR defines the divide ratio of equation 2.1.
- M sets the cycles per symbol (used in Miller coding).
- TRext sets whether there is a pilot tone before the tag preamble.
- Sel chooses which Tags respond to the Query.
- Session chooses a session for the inventory round.

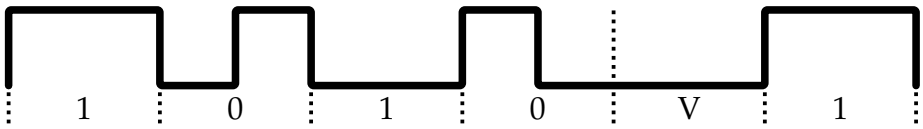


FIGURE 2.6: FM0 short preamble (TRext=1)

- Target selects from either population of tags.
- Q sets the number of slots (tags to participate) per round.
- CRC is a 5bit cyclic redundancy check based on the previous bits of the query.

There are many commands an interrogator should implement to fulfill the Gen2 protocol requirements and many possible scenarios that can play out (tag collision, communication errors, etc.) that the protocol covers. In this work only the most basic interrogation that covers just one case will be exhibited as shown in figure 2.7. An interrogator issues a Query command and the tag responds with a random 16bit number after T_1s defined by the standard. The interrogator in turn replicates the number in T_2s and if that is the same, the tag sends its unique EPC ID. After the EPC is confirmed the interrogator may send more commands to the tag.

For this work only the Query and the RN16 detection is implemented. The query is needed to configure the tag, and the preamble of the RN16 is used for the phase measurements needed for the localization.

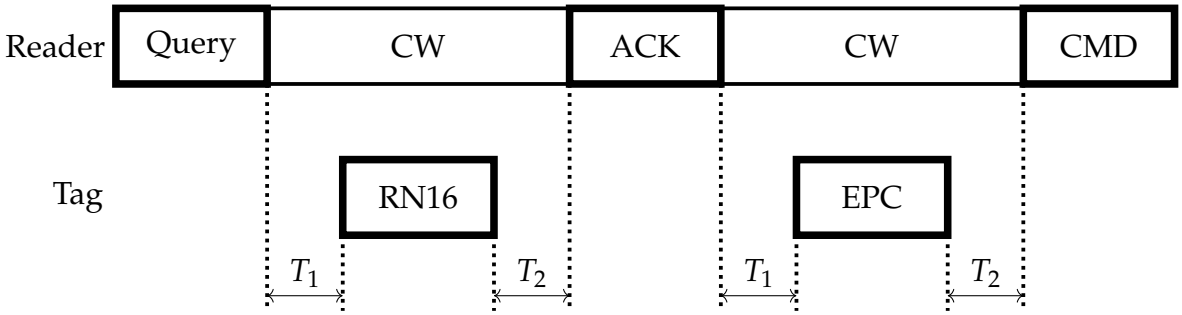


FIGURE 2.7: Single tag reply

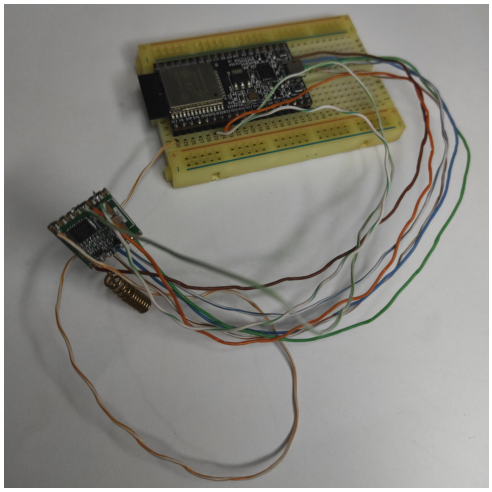
2.2 RF69HW module

The RF69HW is a high-performance, low-cost (2€) RF transceiver module widely used in wireless communication applications. Manufactured by HopeRF, the RF69HW operates in the ISM (Industrial, Scientific, and Medical)

bands, particularly in the 315 MHz, 433 MHz, 868 MHz, and 915 MHz frequencies, depending on the regional and functional requirements [2]. The different packages can be seen on figure 2.8a.

It supports 20dBm of output power, FSK, GFSK, MSK, GMSK and **OOK** modulations and an SPI interface. This makes it suitable to be used as an RFID interrogator when paired with a microcontroller as the ESP32. It's complete package measures at 19.7×16mm making very small and light, appropriate for easy large scale deployment.

(A) The different RF69HW packages



(B) The RF69HW paired with an ESP32.

The RF69HW is configured by modifying its register values. It has 59 configuration registers, but we only need to modify a subset of them to adapt it for our own use.

In my case I paired it with an ESP32 (Figure 2.8b) instead of an Arduino for two main reasons:

1. The ESP32 operates at a 3.3V logic level, which is directly compatible with the RF69 chip. In contrast, Arduinos use a 5V logic level, necessitating the use of a logic level shifter to avoid damaging the RF69.
2. The ESP32 comes with built-in WiFi and Bluetooth functionality, enabling wireless communication and large-scale deployment without the need for additional cabling or external modules.

The way I used the RF69 chip is unintended. Two operation modes are supported.

Address	Register Name	Reset (built-in)	Default (recommended)	Description
0x02	RegDataModul		0x00	Data operation mode and Modulation settings
0x07	RegFrfMsb		0xE4	RF Carrier Frequency, Most Significant Bits
0x08	RegFrfMid		0xC0	RF Carrier Frequency, Intermediate Bits
0x09	RegFrfLsb		0x00	RF Carrier Frequency, Least Significant Bits
0x19	RegRxBw	0x86	0x55	Channel Filter BW Control
0x2C	RegPreambleMsb		0x00	Preamble length, MSB
0x2D	RegPreambleLsb		0x03	Preamble length, LSB
0x2E	RegSyncConfig		0x98	Sync Word Recognition control
0x2F-0x36	RegSyncValue1-8	0x00	0x01	Sync Word bytes, 1 through 8
0x3C	RegFifoThresh	0x0F	0x8F	Fifo threshold, Tx start condition
0x37	RegPacketConfig1	0x10		Packet mode settings
0x3D	RegPacketConfig2		0x02	Packet mode settings

TABLE 2.2: Edited registers descriptions

Name (Address)	Bits	Variable Name	Value Set	Description
0x02	7	-	0	unused
	6-5	DataMode	00	Packet mode
	4-3	ModulationType	01	K
	2	-	0	unused
	1-0	ModulationShaping	10	fcutoff = BR
0x03	7-0	BitRate(15:8)	0x02	MSB of Bit Rate
0x04	7-0	BitRate(7:0)	0x80	LSB of Bit Rate
0x05	7-6	-	00	unused
	5-0	Fdev(13:8)	000000	MSB of the frequency deviation
0x06	7-0	Fdev(7:0)	0x52	LSB of the frequency devation (5kHz)
0x2C	7-0	PreambleSize(15:8)	0x00	Size of the preamble to be sent MSB (0)
0x2D	7-0	PreambleSize(7:0)	0x00	Size of the preamble to be sent LSB (0)
0x2E	7	SyncOn	0	Disable sync word
	6	FifoFillCondition	0	If SyncAddress interrupt occurs
	5-3	SyncSize	000	Size of Sync word (0 bytes)
	2-0	SyncTool	0	0 tolerated bit errors in Sync word
0x2F-0x36	7-0	SyncValue	0x00	Sync Word (0)
0x3F	8	PacketFormat	0	Fixed length
	6-5	DcFree	00	None encoding/decoding
	4	CrcOn	0	CRC calculation off
	3	CrcAutoClearOff	0	Clear FIFO on packet reception CRC fail
	2-1	AddressFiltering	00	None
	0	-	0	unused
0x3C	7	TxStartCondition	1	FifoNotEmpy
	6-0	FifoThreshold	0001000	FifoLevel interrupt (8 bits)
0x3D	7-4	InterPacketRxDelay	0000	Delay between PayloadReady
	3	-	0	unused
	2	RestartRx	0	unused in packet Rx
	1	AutoRxRestartOn	1	Rx auto-restarted after PayloadReady
	0	AesOn	0	Disable AES encryption/decryption

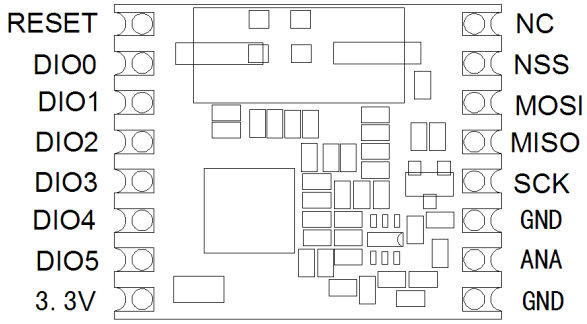


FIGURE 2.9: RF69HW pinout

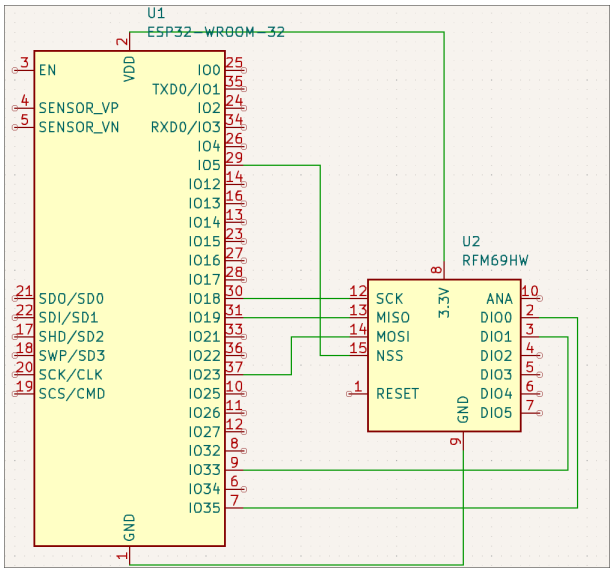


FIGURE 2.10: Connection Schematic

1. **Continuous Mode**: Each bit transmitted or received is accessed in real time at the DIO2/DATA pin. This mode may be used if adequate external signal processing is available.

2. **Packet Mode**: ser only provides/retrieves payload bytes to/from the FIFO. The packet is automatically built with preamble, Sync word, and optional AES, CRC, and DC-free encoding schemes.

The chip supports multiple modes: Sleep, Stand-by, Frequency synthesizer, Transmit, Receive and Listen. I will only use transmit mode, as I only want the chip to be used as an RFID illuminator that enables a tag.

Two RX/TX modes are supported, *continuous* and *packet*. Due to weird timing issues I was unable to use continuous mode effectively so I resorted to using the packet mode.

Specifically, `void loop()` couldn't run fast enough with `digitalWrite()` and

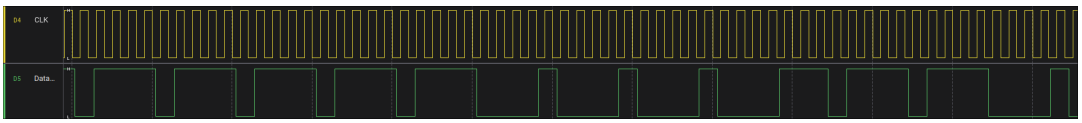


FIGURE 2.11: ESP32 timing issue

`digitalRead()`, thus the reading speed of the clock originating from the RF69 chip and the output on the GPIO pin could not sync. On figure 2.11 two signals capture with a logic analyzer can be seen. The top yellow one is the timing clock of the RF69 running at 50Khz (DIO1), and on the bottom green one the DATA output from the ESP32 (DIO2). The code was trying to invert the output on each negative pulse of the clock, but the detection speed was not fast enough to do so. Thus, continuous mode was deemed unusable on the current setup (maybe FreeRTOS instead of Arduino could fix the problem) and packet mode was chosen instead.

2.2.1 FIFO

In packet mode, data to be transmitted are stored in a configurable FIFO on chip, accessed by the SPI interface. It is 66 bytes long and 1 byte wide.

It supports multiple interrupts in the form of output pin states on the DIO pins of the package. The ones used are:

- **TxReady**/DIO0: It is used by the library to time the transmissions and avoid collisions.
- **FifoLevel**/DIO1: threshold can be programmed by FifoThreshold in RegFifoThresh. The pin goes high when the bytes in the FIFO exceed a user specified value.

Sync Word Recognition

RF69 chips support sync word on both TX and RX and is activated by setting SyncOn in RegSyncConfig. It continuously compares the incoming data with its internally programmed Sync word and sets SyncAddressMatch when a match is detected.

For my purposes, sync words are of no use, so I have disabled them.

Packet format

Three packet formats are supported:

1. **Fixed Length**: When the packet length is known in advance, its value can be set to 255 bytes if AES is not enabled else it's limited to 64 bytes. The length is set in PayloadLength register.
2. **Variable Length**: In applications where the length of the packets vary in size over time this mode enables the user to prepend the size of the message on the payload, so any receiver can easily find the boundaries of the message.
3. **Unlimited Length**: When PayloadLength is set to 0, it enables the user to transmit and receive packets of arbitrary length.

I opted for the latter as I want to be able to continuously transmit a CW intercepted by the query packets.

2.2.2 Usage

The intent was to use the RF69 module as a cheap alternative to the USRP transmitter. Unfortunately the tag would only get activated on very short distances (10-15cm), in contrast to what literature [3] describes. I could not really find the cause, but the tag response characteristics were very distinguishable, when it worked, as shown on figure 2.12. The transmission query of the tag can be seen in blue, the tag response containing the *preamble* and *RN16* in orange, and the CW is grayed out. This is a very typical query and tag response.

On figure 2.13 the *preamble* in orange and *RN16* in gray can be distinguished clearly. In this case, due to the phase of the signal path, the data seems inverted. This is later detected and mitigated by the software.

Due to the CFO between the RF69HW (TX) and the USRP (RX) the phase is constantly changing, as shown on figure 2.14, rendering the data useless for the EllDoA technique. That because it's not possible to measure the phase difference between the two states, and it drifts over time. What is needed is no phase drift and the phase offset among subsequent readings being constant.

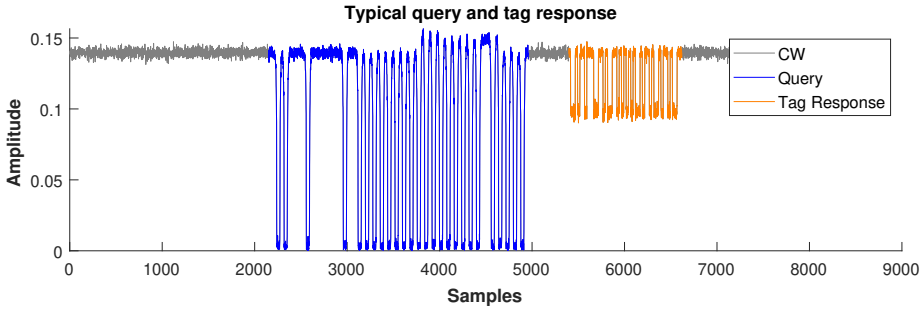


FIGURE 2.12: Query and tag response signal using the RF69HW.

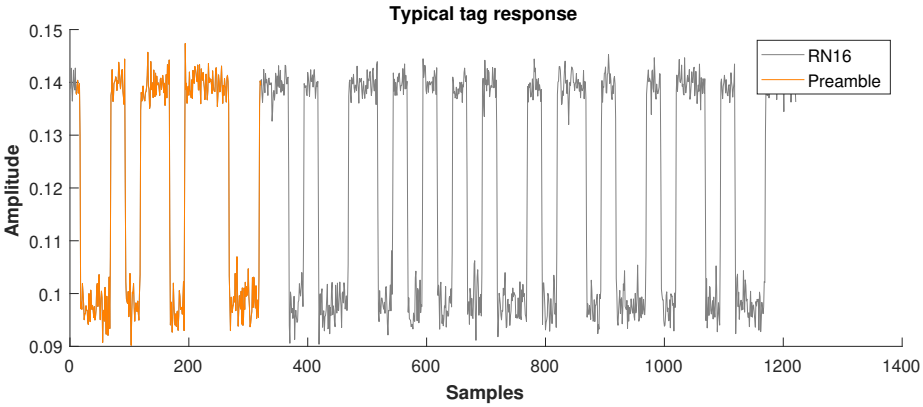


FIGURE 2.13: Typical tag response using FM0.

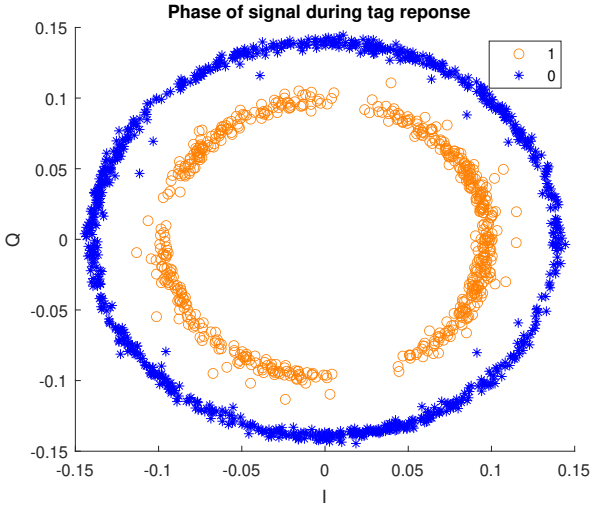


FIGURE 2.14: IQ plot of tag response.

2.3 GNURadio implementation

A GNURadio OOT module was implemented called *EPC_Gen2* to facilitate the communication between the USRPs and the tags. It supports both the RF69HW module pair with the code created for it, and most importantly the USRP. It is designed to be as modular, robust and least convoluted as I could, enabling researchers to easily further work and improve upon it. The code is written with real-time performance in mind, and no blocking states as previous work had issues where the state machine would lock easily or move to wrong states, these problems do not exist in this module.

Four blocks were created:

1. **usrp_transmitter**
2. **rf69_transmitter**
3. **receiver_impl**
4. **tx_rotator**

2.3.1 usrp_transmitter

The USRP transmitter within this module is a source block (it produces data), designed to control the USRP (or any other TX capable SDR) and establish communication with RFID tags. It functions as a state machine, handling various stages of RFID interaction. Currently, the transmitter supports three key states: CW (continuous wave), QUERY and ACK.

1. CW is the default state and sends samples with constant phase and amplitude.
2. QUERY sends a single query and then switches back to CW.
3. ACK send an ACK message back to the tag and switches back to CW.

The transmitter automatically performs FM0 encoding of the outgoing bit stream. The module is designed with the potential to support additional EPC Gen2 commands and more complex state transitions if needed.

Communication between the transmitter and other GNURadio blocks is handled through GNURadio's message passing system, utilizing message ports. However, due to the inherent scheduling of GNURadio's message handling, there are some limitations in terms of speed and latency. This current implementation serves as a proof of concept, demonstrating the feasibility of

integrating additional states and commands into the state machine. Further optimizations could enhance the performance for more complete RFID interactions.

2.3.2 receiver_impl

The receiver block is by far the most intricate component of the *EPC_Gen2* module. Its primary responsibility is to detect and decode the RFID tag responses from the USRP transmissions while extracting key phase information from the backscatter signals. It also supports communication with the transmitter block to send acknowledgment (ACK) messages as needed. The complexity of the receiver block arises from the need to handle both signal detection and decoding in real time, but most importantly frequency offset correction and extracting phase measurements for the localization.

Transmission detection

This block implements non-coherent detection of the transmissions, focusing on the amplitude of the received signal to detect pulses corresponding to the transmitter's signal and tag's response. It uses a moving average-based detection method with a window size of 1500 samples, optimized to handle the incoming data in real time. These samples are stored for later use in carrier frequency offset (CFO) cancellation.

The block operates using a state machine with three primary states: *RECEIVING_CW*, *FEEDING_RN*, and *DECODING_RN*. During the ***RECEIVING_CW*** state, the system accumulates samples while calculating the moving average in real time. This state corresponds to the time that the TX is transmitting the CW. Initially, when the list of samples is being populated, the average calculation has a time complexity of $O(n)$, where n is the number of samples in the window. Once the buffer is full, however, the system transitions to an optimized approach, where each new sample only updates the average by adding the new sample and subtracting the oldest sample—this reduces the cost to $O(1)$, ensuring real-time performance without compromising on computational efficiency.

In this implementation, a threshold is dynamically set at $(0.6 \cdot average)$ and the values of incoming samples are compared against it. This threshold can be further adjusted to account for the noise floor by introducing a minimum noise threshold, ensuring that the system remains resilient against

```
cw[]  //The samples that produce the average
cw_avg_ampl = 0  //The average value of the window

foreach(sample){
  if(!cw.full()){
    w_avg_ampl = 0
    for (i..cw.size()){
        cw_avg_ampl += abs(cw[j])
    }
    cw_avg_ampl /= cw.size()
    else{
      cw_avg_ampl = cw_avg_ampl + sample_ampl/cw.size()
        - std::abs(cw[0])/cw.size()
    }
  cw.push(sample);
  }
}
```

background noise. The state of the signal is continuously monitored using a state machine, and every time the signal crosses the threshold, it may transition between a low state and a high state, or vice versa. However, to prevent false detections caused by noise, a safeguard is implemented: if fewer than 12 samples have passed since the last state change, the detection is reset. This mechanism acts as a rudimentary noise filter, ensuring that only legitimate signal changes are detected.

If a minimum of nine state transitions have been observed, and no further transitions occur for a set period of time, the system assumes the end of the *query* transmission. At this point, the state transitions to *FEEDING_RN*.

FEEDING_RN is responsible for sequentially dividing the incoming samples with the values stored during the *RECEIVING_CW*. This performs the step of equation (4.7). After all samples have been processed and stored, the state transitions to *DECODING_RN*.

DECODING_RN performs the tag's RN16 detection, FM0 decode and phase extraction. After successfully completing this task, the state machine returns to *RECEIVING_CW* to await the next transmission. During this time the incoming samples are blocked, and everything must be done in a timely fashion to preserve the real time character of the protocol.

The detection of the tag's RN16 response is initiated by cross-correlating the first 12 preamble bits (`110100100011`, figure 2.6) based on their amplitude, to estimate the start of the transmission. Cross-correlation is computationally

expensive, so it is only performed starting on the first 100 samples. This approach is highly reliable because the tag response typically exhibits minimal timing deviation.

In this implementation, each bit of the RN16 response is sampled 25 times, which is a result of the query parameters and sampling rate. After detecting the start of the response, subsequent bits are identified at intervals of 25 samples. To distinguish between the tag's two states—absorption and reflection—the average amplitude of the signal during the RN16 transmission is calculated, and each sample is compared to this average.

One crucial issue that arises during decoding is phase inversion, as the cross-correlation and bit extraction are performed based solely on the signal's amplitude. Therefore, an additional step is required to manually check for and correct any bit inversions. Once the correct bit sequence is confirmed, the phase of the samples corresponding to the two states (absorption and reflection) is extracted.

Using Equation 4.5, the final phase of the signal is derived. This final phase can be transmitted in real-time to a UDP server for further processing, such as localization. Simultaneously, the decoded bits are forwarded to the transmission block, which generates and transmits the appropriate ACK response to the RFID tag.

An important note is that this module can work completely independently on a separate PC/host than other receivers and transmitters building towards a true multistatic and distributed localization system.

2.3.3 tx_rotator

This enables the use of multiple USRP on the transmitting side. It is a sync block, connected to an arbitrary amount USRPs and diverts the input to a single one in round-robin fashion. Thus, only one is transmitting at each point in time. To be able to differentiate between them, it supports sending the currently active USRP via UDP packets to the same server as the transmitter block broadcasting.

2.3.4 rf69_transmitter

This is a very simple block, which sends a UDP packet to as server hosted on the microcontroller controlling the RF69HW module. Each UDP packet

causes a query to be sent.

2.3.5 Usage

The code with instructions is provided on my GitHub page. XML code is provided to generate blocks for GNURadio Companion. An example can be seen on figure 2.15. Using the GUI is not recommended as iterating through versions, and expanding on a GNURadio project can be performed much faster by directly editing python files made for GNURadio. I have crated helper functions that make adding more USRPs a breeze.

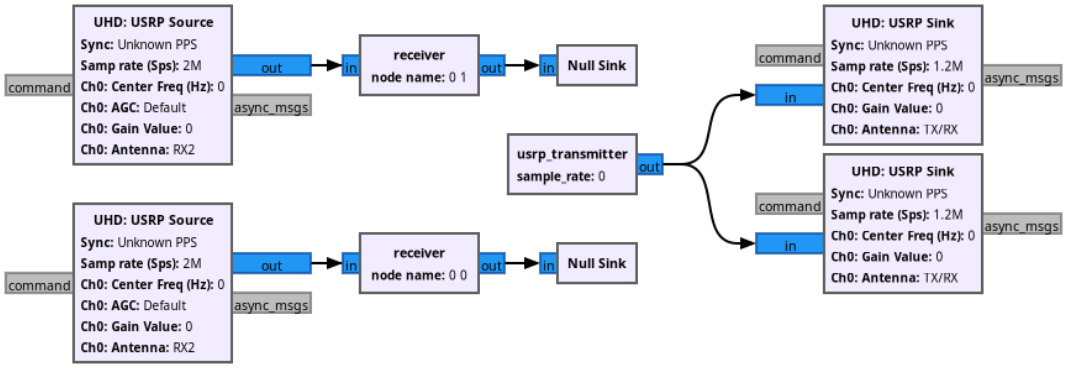


FIGURE 2.15: GRC example

2.4 GUI and server implementation

A Python-based tool was developed to facilitate the localization process and visualize its results in real-time serving both as a data processor and a visualizer.

At the core of the tool is a local server that listens for UDP packets, which was chosen due to its minimal overhead compared to TCP. UDP's is ideal for this application because packet loss is not critical, and the priority lies in speed and simplicity of setup. It supports two packets formats.

- Phase packets; The *receiver* GNURadio block sends packets in the format `x,y,phase` where `x,y` is the location of the corresponding USRP's antenna, and phase the extracted phase.
- TX update packets; `TX_POS: x y` where `x,y` is the location of the TX antenna that's currently active. This is sent from the *tx_rotator* block every time it changes it's output.

For each distinct USRP source, the tool dynamically creates an object that stores the history of its measurements. Upon receiving a new UDP packet, the tool checks whether an object corresponding to the transmitting USRP already exists. If it does, the phase information from the new packet is appended to that USRP's history; otherwise, a new object is created to store the incoming data.

The user can select pairs of USRPs to perform the DoA on, the pairs can either share a common receiver or a common transmitter. Compared to other implementations, there is no for all the nodes to be on a same line, but they can be anywhere. On all applications though everything is aligned on the y-axis, and reason is explained later. For every pair that has been selected two ellipses are created with the foci being each USRP's location and the currently active Tx antenna. The ellipses are formed based on the USRP's past measurements by averaging them, improving the precision. The larger the window the more precise it becomes, but that comes with a cost to the real time performance as it introduces a delay on the stabilization of the measurement after moving the tag.

After the ellipses are created the intersection points need to be calculated. Solving for equation in conics is a very difficult task usually solved numerically in approximation. In this application I opted for using the `scipy` package and the `fsolve` method which attempts to find the roots of linear and non-linear systems of equations.

In the degenerate case where all foci go through a single line, every pair of ellipses generates two solutions, symmetrical to that line. Thus, it's unknown from which half of the plane the true DoA is. By just using a single side of the plane, I can just ignore the second solution. So having all the USRPs on a straight, just helps on ignoring the second solution.

When multiple pairs have been created and multiple DoA lines have been found, all their intersection points can be easily found by solving the linear system they form. This constitutes a localization estimate, and it's continuously being printed on the console.

For the GUI `tkinter` was used and `matplotlib` for the visualization. I want to give attention to the way data is drawn on Matplotlib. Because redrawing is very expensive, the programmer should not redraw by calling the `plot()`, `scatter()` `etc` functions every time something updates. Instead, it's much more performant, reducing the refresh rate by a factor of 100 in my case,

to update the plots using functions such as `set_data()` and `set_offsets()` provided by `matplotlib`. Keeping track of the plots and the data that needs to be updated becomes difficult very fast, but the performance improvements make it very worth it.

The GUI offers the ability to the user to calibrate the pair. This is performed by selecting the wanted pair and pressing the calibrate button. Then the program adds an offset to one of the phase measurements so it equals the other. This is done to perform the CPO correction. By placing the tag in the midpoint of the two different foci between the pairs, (between the RXs if the TX is shared and between the TXs when the RX is shared), and pressing the calibration button, the procedure is complete, and the estimated DoA should now point directly towards the tag.

A historic plot is also offered that shows the past values that are considered for the ellipse creation of each pair. And there is an option to show all the past intersections on the plot as a trivial tracking implementation.

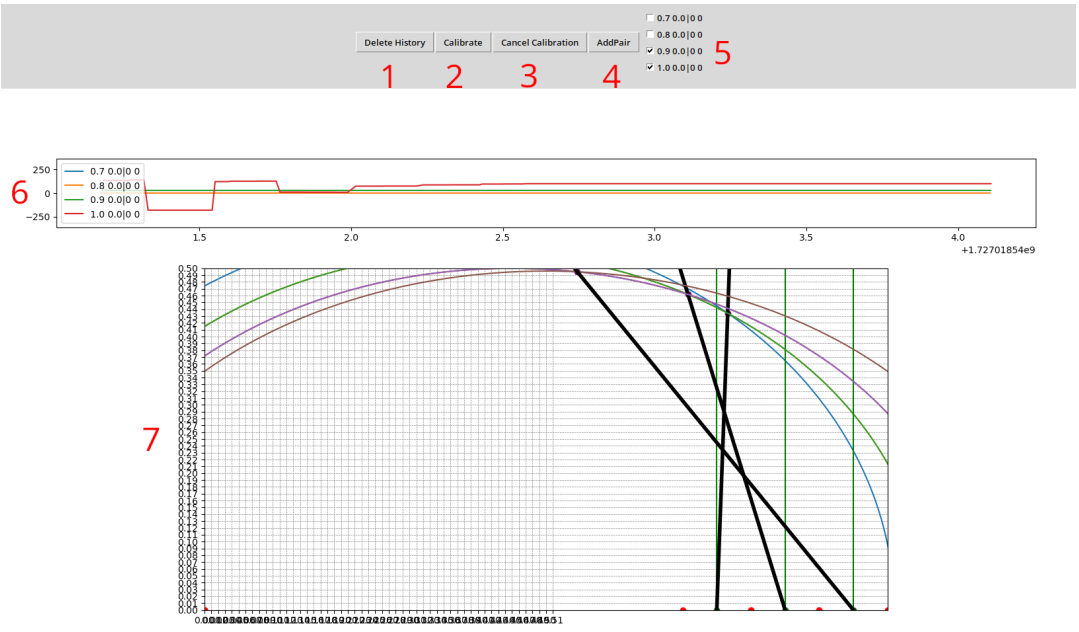


FIGURE 2.16: The GUI server

Figure 2.16 shows a screenshot of the GUI developed for visualizing and calibrating the localization.

1. The delete history buttons removes all the intersection history points from the plot.

2. The calibrate button performs the calibration on the selected pair.

3. The cancel calibration removes all calibrations to start over. This is very useful when a mistake has been made or the environment has changed significantly altering the phase measurements.
4. The add pair button adds a new pair of ellipses around the two selected USRPs. Only the intersection points between those pairs of ellipses are calculated and not between all ellipses
5. All USPRs and TX combinations are displayed and are able to select to create the pairs for 2 and 4
6. The historic values of each USRP that are considered when forming the ellipses is displayed. If a reading shows older samples than the rest, that means that the RSSI is lower.
7. The plot displays the ellipses of the pairs with different colors, the positions of the antennas as red circles (on y-axis). The estimated DoA as thick black lines and green lines in the middle between the foci of each pair.

Chapter 3

Localization

Many direction of arrival techniques exist in the literature, such as SAMV (iterative sparse asymptotic minimum variance), MUSIC (MUltiple SIgnal Classification) and EllDoA (Elliptical DoA). In my thesis, I will be using the latter to attest my results.

3.1 System Model

The localization technique employed in this work depends on transmitter (Tx) antennas and receiver (Rx) antennas, located on the same plane with the RFID tag. A single transmitter antenna and multiple receiving antennas are assumed; the methodology and the results remain the same with a setup exploiting multiple transmitter antennas and a single receiver antenna. The location of the RFID tag on this plane is represented by $\mathbf{x}_T = \begin{bmatrix} x_{tag} & y_{tag} \end{bmatrix}^T$. Similarly, the locations of the Tx antenna and the i-th Rx antenna on the plane are denoted as $\mathbf{x}_{Tx} = \begin{bmatrix} x_{Tx} & y_{Tx} \end{bmatrix}^T$ and $\mathbf{x}_{Rx,i} = \begin{bmatrix} x_{Rx,i} & y_{Rx,i} \end{bmatrix}^T$, respectively.

The complex channel gain for the path from Tx antenna to the i-th Rx antenna follows:

$$h_i = h_{CT}\, h_{TR,i} = |h_{CT}\, h_{TR,i}|\, e^{-j\phi_i} \in \mathbb{C}, \tag{3.1}$$

where h_{CT} and $h_{TR,i}$ represent the baseband complex coefficients for the paths from the Tx antenna to the RFID tag and from the RFID tag to the i-th Rx antenna, respectively. Here, $h_i \in \mathbb{C}$, $|h_{CT}\, h_{TR,i}| \in \mathbb{R}^+$, and $\phi_i \in [0, 2\pi]$. The phase shift introduced by the propagation channel h_i is denoted as follows:

$$\phi_{\text{prop},i} \equiv \phi_i \stackrel{\triangle}{=} \angle h_i = \frac{2\pi}{\lambda}(d_{CT} + d_{TR,i}) + \phi_{\text{M},i}, \tag{3.2}$$

where $d_{CT} = ||\mathbf{x}_{Tx} - \mathbf{x}_T||_2$ is the Euclidean distance between the Tx antenna and the RFID tag, $d_{TR,i} = ||\mathbf{x}_{Rx,i} - \mathbf{x}_T||_2$ is the Euclidean distance between the RFID tag and the i-th Rx antenna, λ is the wavelength of the carrier signal, and $\phi_{\mathrm{M},i}$ accounts for the phase contribution due to multipath effects.

It is very important for the SDRs to be synchronized to eliminate both carrier frequency offset (CFO) and phase offset (CPO). This thesis will later prove and demonstrate a method that when combined with the EllDoA method, eliminates this requirement.

The phase measured at each i-th receiver antenna is also influenced by the RFID tag itself. This occurs because the tag's reflection coefficient depends on its terminating load, which in turn, is affected by the received power at the tag. Various factors, including the non-linear rectification at the tag's energy-harvesting circuit, contribute to this dependency. Consequently, the tag introduces an additional phase shift ϕ_{tag}, which depends on factors such as the reader's transmit power, the positions of the reader's antennas, and the tag's location [4]. Moreover, there are fixed delays due to cabling and random factors, which contribute a constant phase offset $\hat{\phi}_0$, as well as phase noise $\hat{\phi}_{n,i}$ in the receiver chain of the i-th Rx antenna. Thus, the overall phase at the output of the i-th Rx antenna can be expressed as follows:

$$\Phi_i = \phi_{\mathrm{prop},i} + \hat{\phi}_0 + \phi_{tag} + \hat{\phi}_{n,i} \tag{3.3}$$

$$= \frac{2\pi}{\lambda}(d_{CT} + d_{TR,i}) + \underbrace{\phi_{\mathrm{M},i} + \hat{\phi}_0 + \phi_{tag} + \hat{\phi}_{n,i}}_{\tilde{\phi}_{n,i}}. \tag{3.4}$$

Experiments and processing software of this work exploit phase measurements reported within the $[0, 2\pi)$ range. Hence, the phase measured by the i-th Rx antenna is given by:

$$\phi_{Rx,i} = \Phi_i \mod 2\pi \tag{3.5}$$

$$= \left[\frac{2\pi}{\lambda}(d_{CT} + d_{TR,i}) \mod 2\pi + \tilde{\phi}_{n,i} \mod 2\pi\right]. \tag{3.6}$$

From the above expressions, it is evident that any value of sum of distances $d_{CT} + d_{TR,i}$ satisfying

$$d_{CT} + d_{TR,i} = \lambda \frac{\phi_{Rx,i}}{2\pi} + k_i\lambda, k_i \in \mathbb{N}, \tag{3.7}$$

will result in the same phase measurement, given a specific noise value, where $\phi_{Rx,i}$ is the measured phase reported from the i-th Rx antenna constrained to $(0, 2\pi]$. Therefore, phase measurements inherently introduce distance ambiguity, which must be explicitly addressed.

The received signal $y(t)$ is the superposition of the carrier wave and the tag's backscattered signal. Simplifying the system model by neglecting the structural mode of the tag antenna [5] and simplifying the notation (dropping index i of receive antenna), the received signal can be written as follows:

$$y(t) = A_{cw}\, e^{-j(2\pi f_{\text{Tx}} t + \alpha)} + x(t)\, A_{tag}\, e^{-j(2\pi f_{\text{Tx}} t + \Phi)} + n(t), \tag{3.8}$$

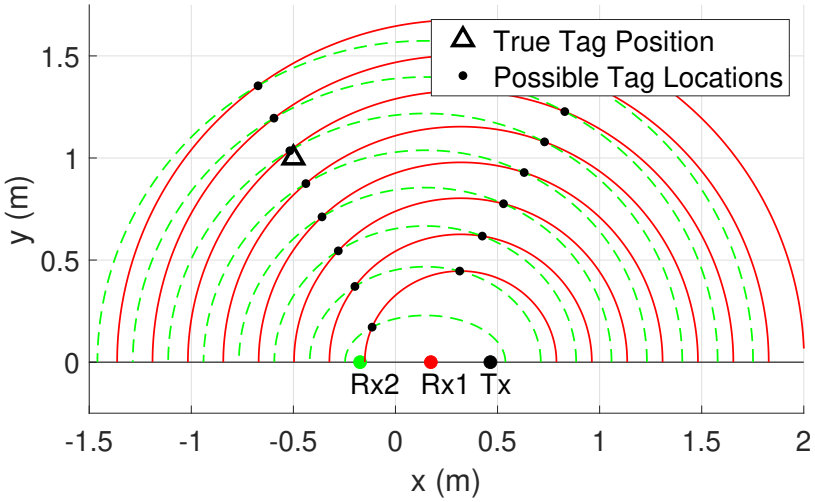
where A_{cw} represents the amplitude of the carrier wave and A_{tag} the amplitude of the tag-backscattered signal; $x(t) \in \{0, 1\}$ refers to the binary values encoded by the tag, corresponding to the reflection and absorption state; $n(t)$ denotes additive white Gaussian noise; f_{TX} is the carrier frequency of the transmitter (e.g., 910 MHz); α is the phase of the carrier wave as received by the Rx antenna dependant on the Tx-Rx channel; Φ depends on the position of the tag in space.

3.2 EllDoA

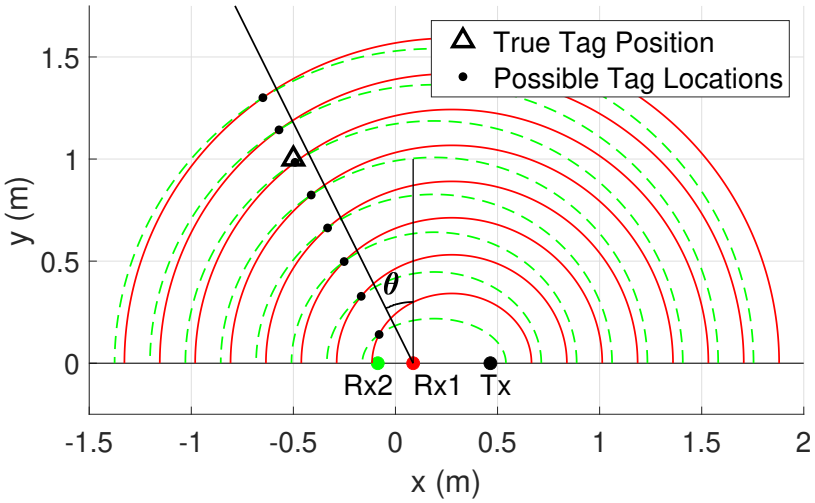
Elliptical direction of arrival (DoA) is a DoA-based technique, coined as EllDoA [6], which leverages elliptical geometry, formed in a bistatic configuration. From Eq. (3.7), the measured $\phi_{Rx,i}$ is transformed into a combined distance $d_{CT} + d_{TR,i}$ for a specific $k_i \in 0, \ldots, K$. This transformation defines the i-th set of $K + 1$ concentric ellipses on a two-dimensional plane; the foci of the i-th set of ellipses are located at the positions of the Tx antenna and the i-th Rx antenna. Fig. 3.1 offers an example for two receive antennas (i.e., two sets of ellipses); the intersection of the two set of ellipses offers estimation of DoA. Thus, EllDoA can operate with just two receive (Rx) antennas to determine the direction of arrival.

A set of two ellipses on plane can have more than one DoA solutions, introducing solution ambiguity. It was shown in [6] that by limiting the distance δx between the receive antennas below half wavelength, i.e., $\delta x < \lambda/2$, the DoA ambiguity can be resolved (Fig. 3.1).

The first step to estimate the DoA of the tag's signal is by determining the



(A) $\delta x > \lambda/2$



(B) $\delta x < \lambda/2$

FIGURE 3.1: Elliptical direction of arrival (EllDoA).

intersection points $\mathbf{p} = [p[1], p[2]]$ of the two sets of ellipses,[1] as illustrated in Fig. 3.1. The two sets of ellipses are defined by the following equations:

$$\frac{(x-x_1)^2}{a_1^2[k_1]} + \frac{(y-y_1)^2}{b_1^2[k_1]} = 1, \qquad k_1 \in 0, \ldots, K, \tag{3.9}$$

$$\frac{(x-x_2)^2}{a_2^2[k_2]} + \frac{(y-y_2)^2}{b_2^2[k_2]} = 1, \qquad k_2 \in 0, \ldots, K, \tag{3.10}$$

where a_i and b_i represent the major and minor axes of the ellipses and (x_1, y_1) and (x_2, y_2) are the centers of the two sets of ellipses, respectively.

[1] In this work the intersection points are determined by finding the roots of the non-linear equations defined by the ellipses using the *scipi* package and *fsolve*.

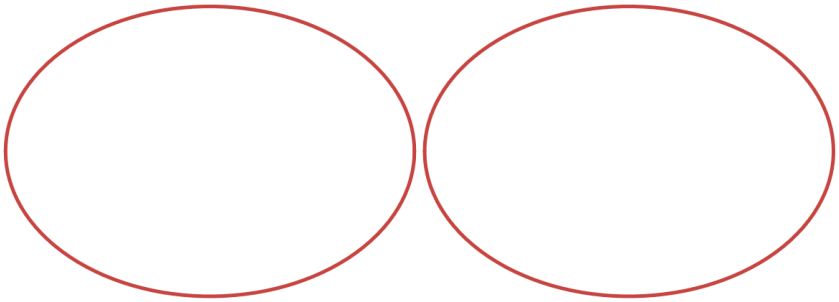
By the ellipsis definition, $d_{CT} + d_{TR,i}$ is constant for any point (x, y) on the ellipsis. Thus, the major axe can be found as follows:

$$2a_i[k_i] = d_{CT} + d_{TR,i} = \lambda \frac{\phi_{Rx,i}}{2\pi} + k_i\lambda. \tag{3.11}$$

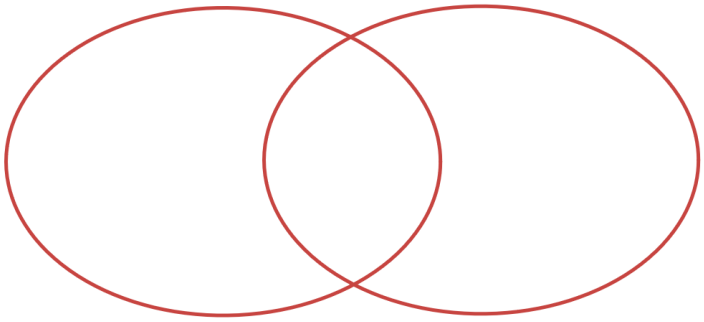
Similarly, the minor axe is directly calculated through the ellipsis eccentricity:

$$b_i[k_i] = \sqrt{a_i[k_i]^2 - \left(\frac{||\mathbf{x}_{Tx} - \mathbf{x}_{Rx,i}||_2}{2}\right)^2}. \tag{3.12}$$

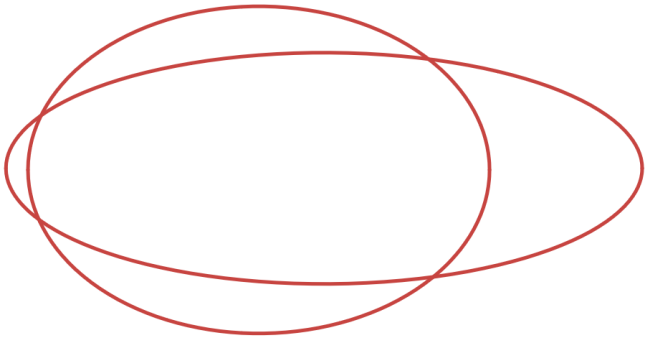
A set of two ellipses on plane can have either 0,2 or 4 solutions as depicted on figure 3.2. This introduces an ambiguity as a single solution is needed to determine the DoA. It has been proven that by limiting the distance between the foci of the different ellipses so that the non-common foci have a distance $\delta x < \lambda/2$. This limits the solutions to either 0 or 2. In the case of two solutions, one can be ruled out manually by excluding parts of the plane if prior knowledge of the tag location is known. Another way of conquering this problem is using more receivers and or transmitters. 11



(A) 0 solutions case



(B) 2 solutions case



(C) 4 solutions case

FIGURE 3.2: Ellipse intersection cases on a plane

Chapter 4

Real-time Carrier Synchronization (CFO, CPO)

4.1 Carrier Frequency Offset (CFO)

Carrier frequency offset (CFO) arises when the local oscillator frequency of the receiver and the transmitter don't match. This is the case of bistatic and multistatic setups, where the radios do not share a common clock, as opposed to monostatic setups or synchronized radios. This constant frequency drift is a very serious issue in telecommunications and especially localization techniques. EllDoA, as many others, is a phase based method for detecting the direction of arrival of a signal by using the phase difference among two or more receivers. CFO renders such techniques useless and must therefore be dealt with.

4.1.1 CFO model

The received signal (Eq. (3.8)) is down converted by the local oscillator on the receiving end and the received signal, after filtering, is given as follows:

$$S(t) = A_{cw}e^{-j(2\pi\Delta f t+\alpha)} + x(t)A_{tag}e^{-j(2\pi\Delta f t+\Phi)}, \tag{4.1}$$

where Δf is the CFO; the first term is the self-interference, which must be removed, in order to decode the tag's signal. It is obvious from the above that due to CFO, the phase changes at a constant rate of $2\pi\Delta f$ and thus, phase-based localization (as in EllDoA) is impossible without *accurate* CFO removal.

(A) (B) (C)

FIGURE 4.1: (a) Sampled IQ data (b) IQ data after CFO cancellation (c) IQ data after CFO and self-interference cancellation.

4.1.2 CFO cancellation

Many methods of detecting and canceling the carrier frequency offset exist. Detecting the frequency offset can be done using FFT. The sampling rate used in this thesis is 2MHz, to achieve good enough granularity would be very costly, and even then there would be very noticeable residual frequency offset. Another method is using specific signals such as Zadoff–Chu sequences. This entails transmitting sequences that cyclically shifted versions of themselves that are orthogonal to one another. A proof of concept was implemented that lead to frequency offset detection. After the detection, a multiplication of the received signal with a sinusoid of the CFO frequency and a baseband filtering would result in the corrected signal. This technique imposed some problems, mostly concerning the cost, the query rate (as extra calibrating sequences inhibit more frequent queries) and finally the accuracy, that left more to be desired as there was still a residual frequency offset, and a phase offset introduced that would interfere with the DoA results.

Therefore, other solutions had to be considered. CFO cancellation in this work is based on a real-time methodology, proposed in [3]. In the short timeframe of tag interrogation, in the order of a few milliseconds, the observed CFO remains constant. The core idea is to exploit the CFO during transmission of the continuous wave (i.e., when the tag does not backscatter), to cancel out the CFO during RFID tag backscattering, by dividing these sample sets bettween them sequentially. To do so, the received signal before tag transmission and the tag response are both captured and stored in memory. The received signal before tag transmission is denoted as $CW = A_{cw}e^{-j(2\pi\Delta f t+\alpha')}$, where α' is an unknown phase, accounting for the unknown time instant when signal capturing was initialized. The received signal during tag backscattering is divided by the stored signal:

$$S'(t) = \frac{S(t)}{CW} \tag{4.2}$$

$$= \frac{A_{cw}\, e^{-j(2\pi\Delta f t+\alpha)} + x(t)\, A_{tag}\, e^{-j(2\pi\Delta f t+\Phi)}}{A_{cw}\, e^{-j(2\pi\Delta f t+\alpha')}} \tag{4.3}$$

$$= \underbrace{e^{-j(\alpha-\alpha')}}_{\text{self-interference}} + x(t)\, \frac{A_{tag}}{A_{CW}}\, e^{-j(\Phi-\alpha')}. \tag{4.4}$$

The offered signal $S'(t)$ is not affected by the CFO anymore; however, two problems arise: 1) self-interference and 2) additional phase offset α' in the backscattered signal. Given that the self-interference term is a constant, averaging can be adopted for its estimation; $x(t)$ as stated before, represents the states of absorption and reflection of the tag. By taking samples during the absorption state ($x(t) = 0 \Rightarrow S'(t) = e^{-j(\alpha-\alpha')}$) and averaging them, a precise estimate of the self-interference component can be provided. Thus, CFO and phase offset can be removed as follows:

$$S_{tag} = \frac{S'(t) - \overbrace{e^{-j(\alpha-\alpha')}}^{estimate}}{e^{-j(\alpha-\alpha')}} \simeq \frac{x(t)\, \frac{A_{tag}}{A_{cw}}\, e^{-j(\Phi-\alpha')}}{e^{-j(\alpha-\alpha')}}$$

$$= x(t)\, \frac{A_{tag}}{A_{cw}}\, e^{-j(\Phi-\alpha)}. \tag{4.5}$$

The received tag phase is now $\Phi - \alpha$ instead of Φ. This is addressed through calibration, explained subsequently. Fig. 4.1 shows the sampled in-phase (I) and quadrature (Q) samples before CFO correction (a), after CFO cancellation (b), and after CFO and self-interference cancellation (c).

4.1.3 Carrier Phase Offset (CPO) Calibration

In the synchronized setup with the USRPs connected via the OctoClock, the clocks were expected to be both frequency- and phase-synchronized. However, phase alignment was not observed as expected, but a random phase offset was observed on the measurements, constant during each run. According to Ettus Research, an additional calibration step is needed to achieve precise phase alignment after synchronization. The same calibration is also needed for the unsynchronized case, as the phase offset is random.

The calibration is performed by positioning the RFID tag on the perpendicular bisector of the line between the two receiving antennas, equidistant from each. This arrangement should theoretically yield identical phase shifts at both antennas, since the reflected signal travels the same distance to each antenna. However, due to the constant phase offset (CPO), a remaining phase difference is observed.

To correct this, a phase offset is applied to one phase measurement, aligning it with the other. This adjustment compensates for the phase discrepancy caused by the CPO, ensuring synchronized measurements. Once calibrated, this offset remains stable across the experiment.

The CFO correction algorithm offers an additional advantage, as a collateral dividend: while traditional approaches require recalibration with each experimental reset, here, the calibration offset remains valid across multiple runs. Consequently, for each RX (i) node and each TX (j) node, the signal can be represented as follows:

$$S_{i,j}(t) = A_{cw}e^{-j(2\pi\Delta ft+\alpha+\alpha_{i,j})} + x(t)A_{tag}e^{-j(2\pi\Delta ft+\Phi+\alpha_{i,j})} \tag{4.6}$$

and

$$S'_{i,j}(t) = \frac{S_{i,j}(t)}{CW_{i,j}} \tag{4.7}$$

$$= \frac{A_{cw}\, e^{-j(2\pi\Delta ft+\alpha+\alpha_{i,j})} + x(t)\, A_{tag}\, e^{-j(2\pi\Delta ft+\Phi+\alpha_{i,j})}}{A_{cw}\, e^{-j(2\pi\Delta ft+\alpha'+\alpha_{i,j})}} \tag{4.8}$$

$$= e^{-j(\alpha-\alpha')} + x(t)\,\frac{A_{tag}}{A_{CW}}\, e^{-j(\Phi-\alpha')} \tag{4.9}$$

$$\equiv S'(t). \tag{4.10}$$

Thus, even though the phase offset $\alpha_{i,j}$ is different across multiple runs, the CFO correction algorithm above manages to eliminate it at each run.

Chapter 5

Results

5.1 Experimental Synchronous and Asynchronous Setup

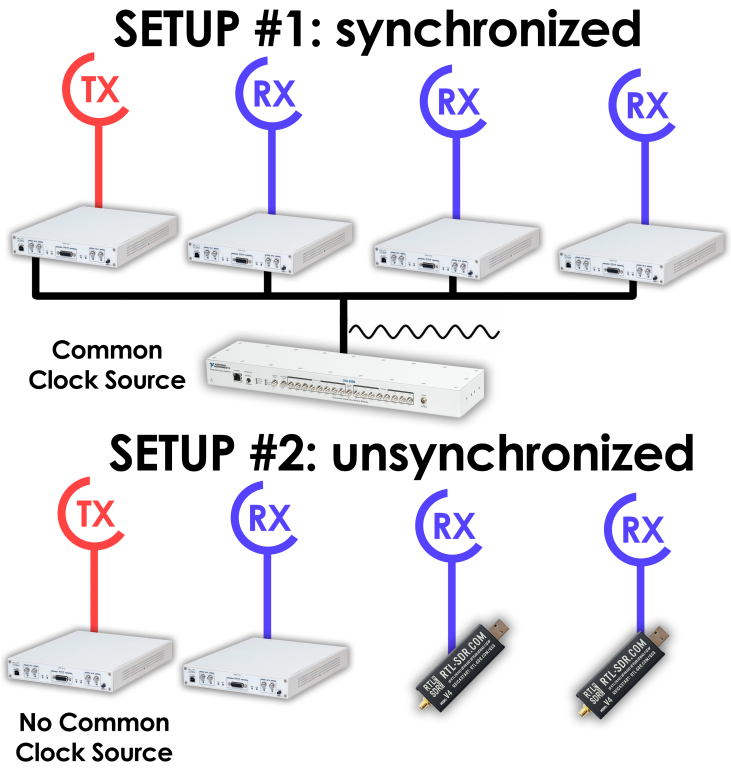


FIGURE 5.1: Carrier synchronized and unsynchronized setups.

The experiment was conducted with two setups; the first utilized as baseline reference, with 4 SDRs synchronized externally with a common carrier and clock pulse (Fig. 5.1-top); the second utilized SDRs that were not synchronized; instead, they utilized their internal clocks, introducing both CFO and CPO (Fig. 5.1-bottom).

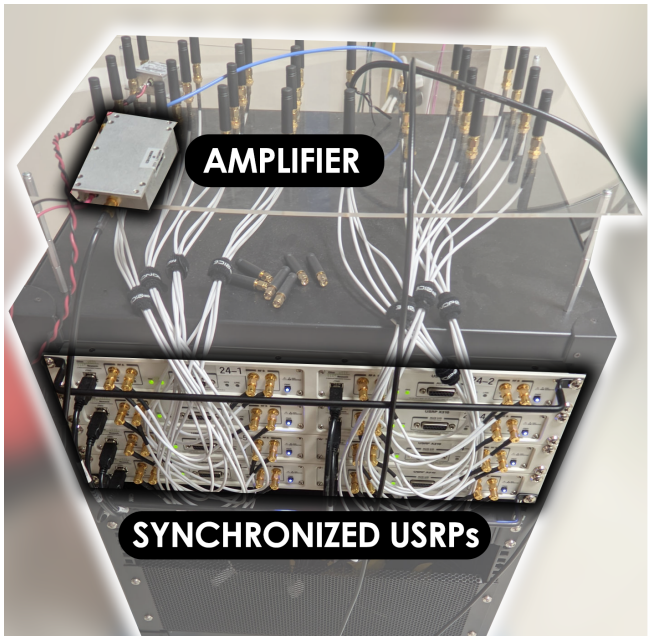


FIGURE 5.2: Synchronized USRP rack.

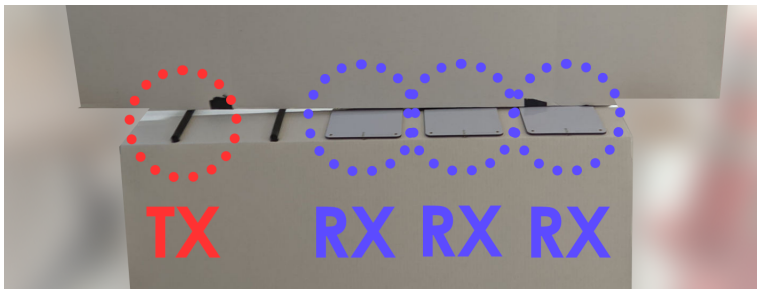


FIGURE 5.3: Antennas placement.

The first setup used USRP x310s as the SDRs of choice, all synchronized with the Octoclock distribution module (Fig. 5.2). The second setup removed the Octoclock synchronization module, and the SDRs connected to the two right-most Rx antennas (Fig. 5.3) were replaced with two RTL-SDR (v4), providing a testbed which introduces CFO between all nodes. It should be noted that in both setups there is a CPO introduced that can't be avoided and needs to be addressed.

For both setups, three SDRs were utilized as RX and one as TX. The RX antennas were placed starting 34 cm away from the TX one, and spaced apart by 18 cm (Fig. 5.3). The TX antenna was an omni-directional 3 dBi ETTUS VERT900 and each RX antenna was a 3.4 dBi Vulcan RFID p11 UHF wide-beam directional antenna. They were all placed on the edge of a table with the directional antennas pointing upwards and each was connected with its respective SDR.

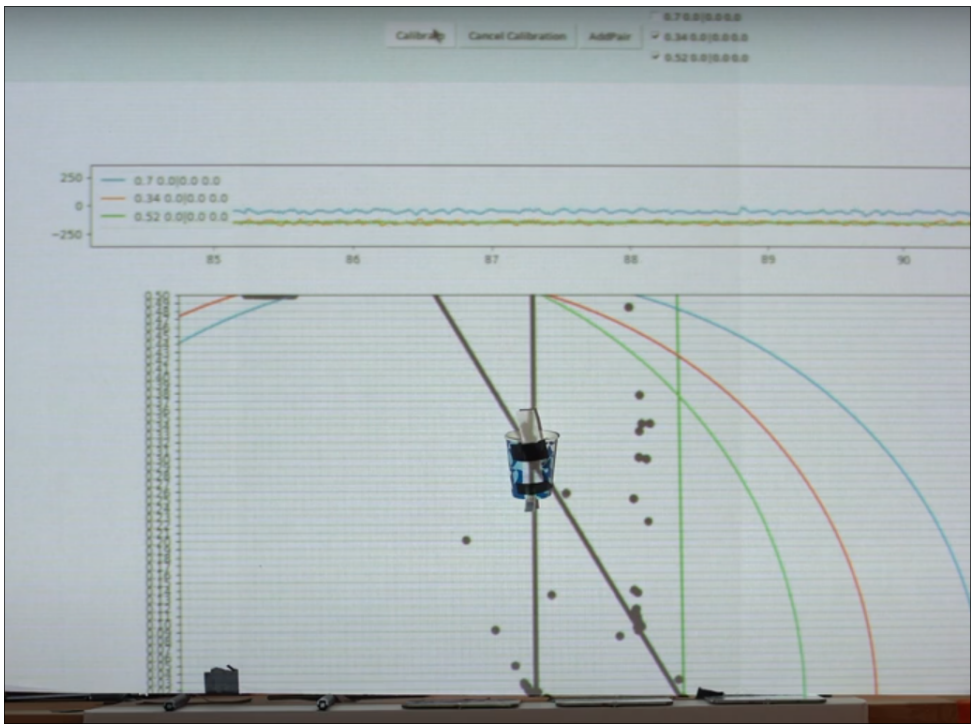


FIGURE 5.4: Taking measurements.

An operating carrier frequency of 910 MHz was chosen. As the antennas were facing upwards, a cardboard backing was placed vertically, allowing placement of the RFID tag on various positions. A projector was placed in front of the cardboard, calibrated to show a 2d grid with the real world coordinates and the DoA estimations as an overlay on top of the actual experiment.

One more setup was tested, using 2 TXs and 2 RXs; and the results were similar.

On Fig (5.4) a snapshot from my videos demonstrating real-time localization. The video links are provided. The first[1] utilizes the 4 different USRP x310s in a synchronized multistatic topology. And the second [2] utilizes the unsynchronized USRPs and RTL-SDRs.

5.2 Numerical Results

The experimental setups, described in Section 5.1, involve a synchronized setup (with external, shared synchronizer) for benchmark reference and the unsynchronized setup of interest, where the system will be evaluated. For

[1] Synchronized EllDoA | https://youtu.be/qYWwYV0dL_g
[2] Unsynchronized EllDoA | https://youtu.be/5FY_hSV6Rc8

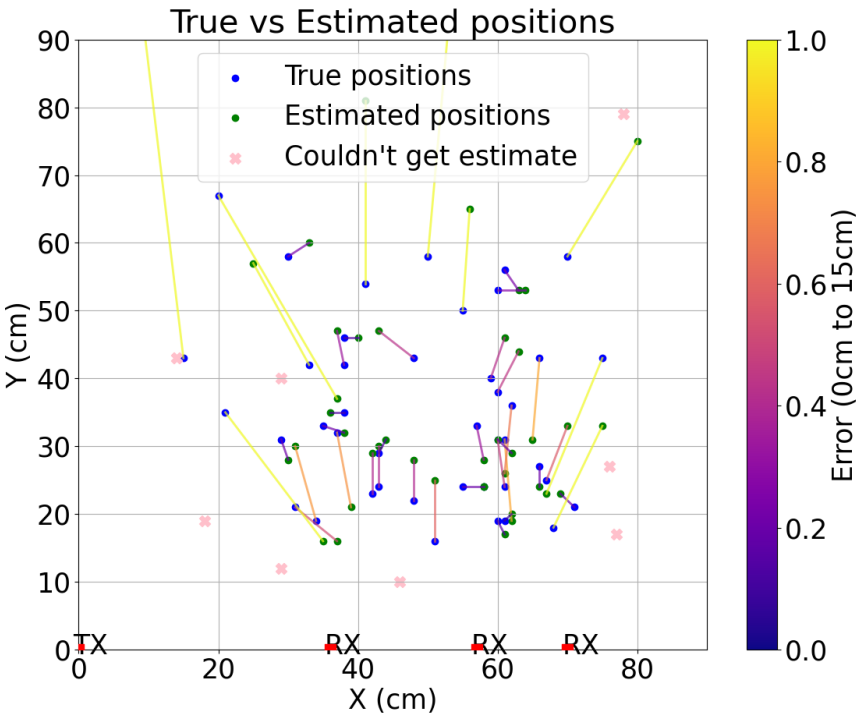


FIGURE 5.5: Errors using only USRPs.

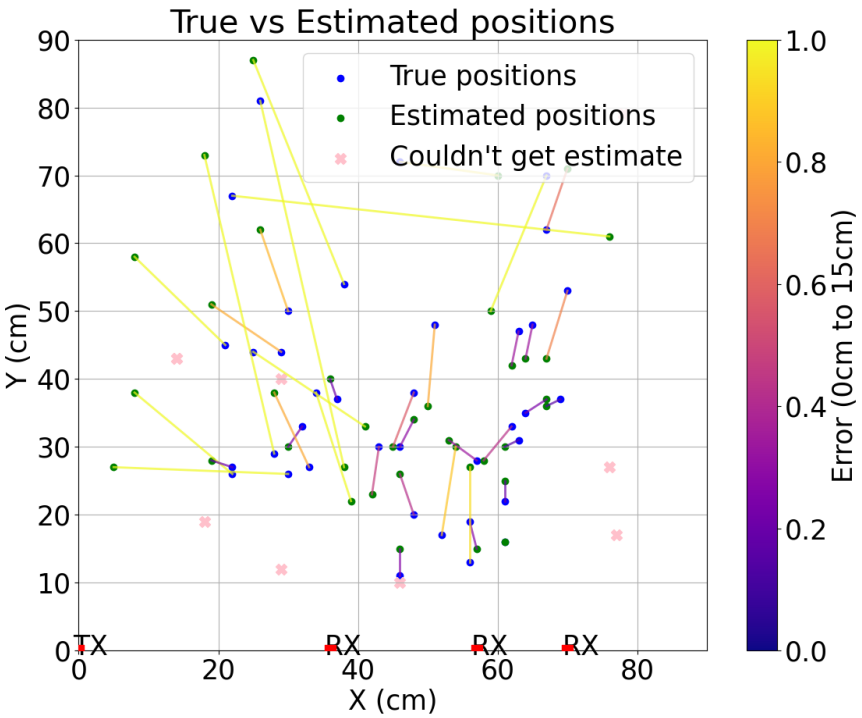


FIGURE 5.6: Errors using RTL-SDRs and USRPs.

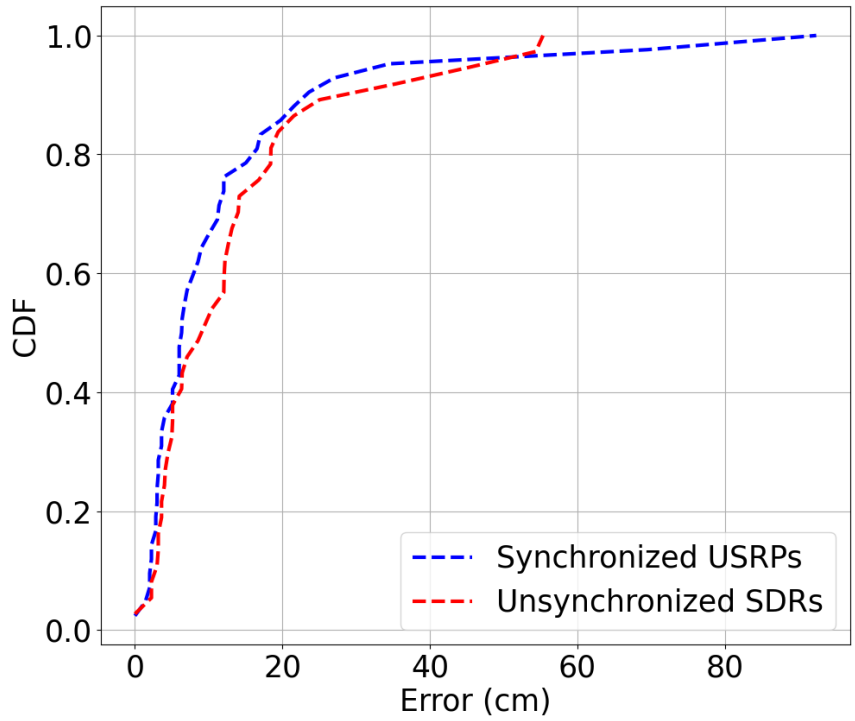


FIGURE 5.7: Comparing the CDF of the two setups.

both setups, the tag was placed at approximately 40 random locations and the average of the last 40 phase measurements at each location was used to minimize the impact of noise; increasing the number of measurements would decrease noise power but also compromise the real-time operation of the estimation; thus, the selected number of measurements provided a tradeoff between real-time estimation and robustness to noise. Due to strong multipath in the experimentation area, the tag activation range and read distance was limited inside a 80 cm $\times$ 80 cm area.

Figs. 5.5 and 5.6 depict the true and estimated position, when only USRPs (Fig. 5.5) or a mix of USRPs and low-cost RTL SDRs are utilized (Fig. 5.6); more importantly, the setup in Fig. 5.5 is synchronized, while the setup in Fig. 5.6 is unsynchronized; in the latter case, carrier-level synchronization and localization will be conducted jointly. Both figures demonstrate more accurate estimation of tags near the center of the test region, as opposed to estimation near the corners. This is due to the small SNR and respective reading distance and the large number of blind spots due to strong multipath. Nevertheless, both figures demonstrate that tags can be localized, even when SDRs are unsynchronized at the carrier level and of low-cost, as in the case

of Fig. 5.6.

Fig. 5.7 offers the cumulative distribution function (CDF) of the localization error of both (synchronized and unsynchronized) setups. Interestingly, it is observed that using the unsynchronized setup comes with a small cost compared to the synchronized setup for real-time applications; 80% of the localization errors were under 16 cm for the synchronized setup and under 18.5 cm for the unsynchronized setup; similarly, 90% of the localization errors were under 23.2 cm and 28.2 cm for the synchronized and unsynchronized setup, respectively.

Chapter 6

Conclusion

6.1 Conclusion

This work addressed the challenging problem that arises when non synchronized radio modules are combined; carrier frequency offset and carrier phase offset. If any of these exists, phase based localization techniques cannot be performed. This work managed to implement a method of utilizing such setups for RFID tag localization by combining the EllDoA technique with a smart CFO mitigation algorithm. It was also found that these two techniques, combined, mitigate the need to perform the CPO calibration step on each experiment rerun. A single calibration on the first run can be stored and be reused later as is.

The proposed solution can utilize any GNURadio-compatible SDR transmitter and receiver to perform real-time RFID tag localization. This system is designed to be modular, robust, and easily scalable, paving the way for practical real-world applications.

It was exhibited that the unsynchronized setup utilizing cheap RTL-SDRs comes with a 15-20% accuracy cost in the 80th and 90th percentile compared to the synchronized USRP setup. These results make the future of low-cost multistatic localization look extremely promising.

6.2 Future Work

This implementation leaves things to be desired. First and foremost the experiment should be repeated on a larger area to strengthen the reliability of the results. One of the greatest limitations currently was the activation range of the tags and the many black spots in the space where the tag would not get

illuminated. This issue should be explored further, maybe the problem lies on the transmission power, the query parameters selected or the sampling rates chosen. The reason work is done on multistatic topologies on RFID implementation is their benefit on tag reading distance.

Solving that problem, next would be using more RXs to increase the accuracy. Another limitation of DoA as described is that the antennas must be spaced very closely together, but more RXs can mitigate this issue give the option of spacing the RXs further apart improving accuracy.

On the software side, the GNURadio modules are robust, and the next step would be to implement more of the EPC GEN2 specification. This would enable the confirmation of the EPC, the ability to track more than one tag at a time, select a specific tag to track and use other modulation schemes. Also, the accuracy can probably further be improved by increasing the amount of samples that are being taken into consideration during the phase measurement step.

It would also be interesting to see follow-up work on the RF69HW module which would enable very low-cost, power-efficient, space-efficient and modular implementation of the setup.

Bibliography

[1] *EPC® Radio-Frequency Identity Generation-2 UHF RFID Standard: Specification for RFID Air Interface Protocol for Communications at 860 MHz – 930 MHz*, GS1 AISBL Std., January 2024, © 2024 GS1 AISBL. [Online]. Available: https://ref.gs1.org/standards/gen2/

[2] L. Hope Microelectronics Co., *RFM69HW ISM Transceiver Module v1.1 Specification*, Hope Microelectronics Co., Ltd., 2006. [Online]. Available: http://www.hoperf.com

[3] Y. Wang, J. Cao, and Y. Zheng, "Toward a low-cost software-defined UHF RFID system for distributed parallel sensing," *IEEE Internet Things J.*, vol. 8, no. 17, pp. 13 664–13 676, 2021.

[4] A. Bletsas, A. G. Dimitriou, and J. N. Sahalos, "Improving Backscatter Radio Tag Efficiency," *IEEE Trans. Microwave Theory Tech.*, vol. 58, no. 6, pp. 1502–1509, Jun. 2010.

[5] A. Bletsas, A. G. Dimitriou, and J. Sahalos, "Improving backscatter radio tag efficiency," *IEEE Trans. Microwave Theory Tech.*, vol. 58, no. 6, pp. 1502 – 1509, Jun. 2010.

[6] K. Skyvalakis, E. Giannelos, E. Andrianakis, and A. Bletsas, "Elliptical doa estimation & localization," *IEEE Journal of Radio Frequency Identification (J-RFID)*, vol. 6, pp. 394–401, 2022.