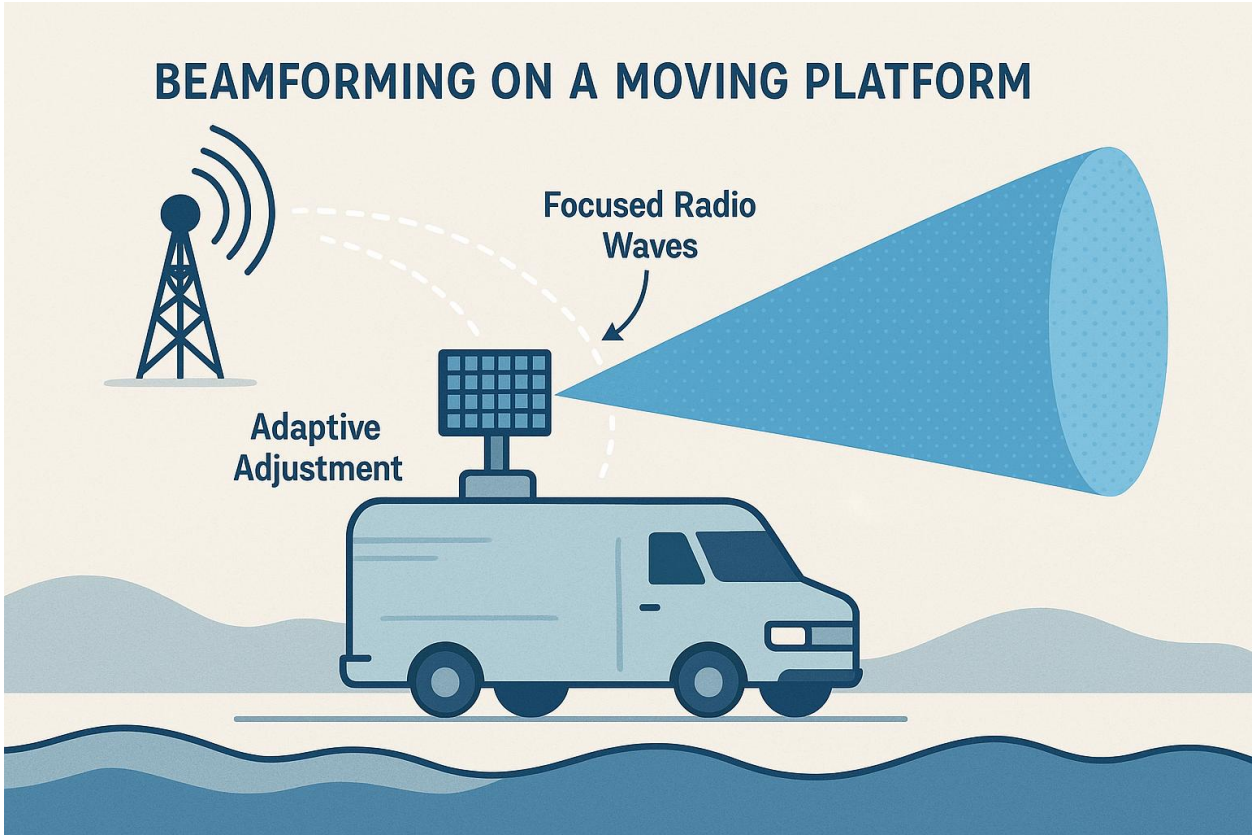
CHALMERS
UNIVERSITY OF TECHNOLOGY



Inertial measurement data collection for beamforming on a moving platform

Master's thesis in Embedded Electronic System Design

Ana Crkvenjas

Department of Microtechnology and Nanoscience
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025

Master's thesis 2025

Inertial measurement data collection for beamforming on a moving platform

Ana Crkvenjas

Department of Microtechnology and Nanoscience
Chalmers University of Technology
Gothenburg, Sweden 2025

Inertial measurement data collection for
beamforming on a moving platform
Ana Crkvenjas

Supervisor: Lars Svensson, Department of Microtechnology and Nanoscience
Company advisor: Henric Broström, Ericsson
Examiner: Per Larsson-Edefors, Department of Microtechnology and Nanoscience

Master's Thesis 2025
Department of Microtechnology and Nanoscience
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of beamforming on a moving platform

Gothenburg, Sweden 2025

Inertial measurement data collection for
beamforming on a moving platform
Ana Crkvenjas
Department of Microtechnology and Nanoscience
Chalmers University of Technology

Abstract

This project demonstrates how to maintain the beam of a wireless antenna system in the right direction while moving. Beamforming typically performs best when the antenna is stationary. However, for an antenna mounted on something like a drone or a car, it tends to tilt and rotate, which can disturb the signal. We solve this by measuring how the antenna moves its roll, pitch, and yaw using a small sensor called a 9-DoF IMU (nine degrees of freedom inertial measurement unit), which for the record includes an accelerometer and a gyroscope. Then we use a filter to combine these data and determine the exact orientation of the antenna at each moment. With that information, we update the direction of the signal (beam) so that it stays pointed at the target, even while the platform is moving. We tested this using a computer simulation that shows how the signal beam behaves in 2D (two dimensions) and 3D (three dimensions). The results show that the use of IMU data helps the antenna system maintain a strong and accurate signal, even during motion.

Keywords: Beamforming, wireless antenna, tilt, rotate, roll, pitch, yaw, intertial measurement unit

Acknowledgements

I want to thank my university supervisor, Professor Lars Svensson, for his invaluable guidance, constant support, and feedback regarding all aspects of the project. I also appreciate my company supervisor, Henric Broström, for his mentorship and support during my time at the company. His insights, encouragement, and practical suggestions were of great help.
Thank you so much, family and friends, for always motivating and patiently standing by me throughout my studies. Your support gave me strength and certainty, without which I could not have completed this journey.

Ana Crkvenjas, Gothenburg, August 2025

List of Acronyms

IMU	Inertial measurement unit
9-DoF IMU	Nine degrees of freedom inertial measurement unit
2D	Two dimensions
3D	Three dimensions
ISAC	Integrated Sensing and Communication
5G	Fifth generation of cellular network technology
6G	Sixth generation of cellular network technology
LoS	Line of sight
TX	Transmitters
RX	Receivers
UEs	User Equipment
OFDM	Orthogonal Frequency Division Multiplexing
PAPR	Peak to Average Power Ratio
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
IFFT	Inverse Fast Fourier Transform
DAC	Digital to Analog Converter
ADC	Analog to Digital Converter
MIMO	Multiple Input Multiple Output
CSI	Channel State Information
GPS	Global Positioning System
MPU-6050	6-axis motion tracking sensor
I2C	Inter-Integrated Circuit Protocol
SPI	Serial Peripheral Interdace Protocol
UART	Universal Asynchronous Receiver/Transmitter Protovol
MIO	Memory-Mapped Input/Output
GPIO	General Purpose Input/Output
IP	Intellectual Property Core
SCP	Secure Copy Protocol
SFTP	Secure File Transfer Protocol

Contents

1

Introduction

In the next generation of networks, 6G will revolutionize wireless communication by combining ultra-fast speeds, intelligent connectivity, and advanced sensing capabilities. In 6G networks, mobile devices will use advanced antennas and AI-driven beamforming to maintain fast and stable connections. Unlike 5G, which reacts to signal changes, 6G will predict and adjust in real-time, reducing delays and improving data speeds. It will also use terahertz (THz) frequencies, enabling ultra-fast wireless communication with minimal interference. One of the biggest advances in 6G is its ability to sense and map the environment. It will use wireless signals not only for communication but also for detecting objects, similar to radar. For example, at an intersection, 6G can identify all vehicles, pedestrians, and obstacles, then share this data with nearby cars to prevent accidents. It will also measure the location, speed, and direction of objects, even if they don't have sensors or transmitters. These features will be crucial for self-driving cars, smart cities, and futuristic applications like holographic communication and fully immersive augmented reality (AR). By combining communication, sensing, and AI, 6G will create a highly intelligent and connected world [1].

In bistatic integrated sensing and communication (ISAC) systems, platform motions such as acceleration, yaw, pitch, and roll introduce modulation effects that degrade sensing accuracy. Understanding and compensating for these effects requires an analytical approach to modeling platform movement, inertial measurement data, and beamforming techniques. This chapter provides an introduction to motion compensation in ISAC systems, covering kinematics, IMU (inertial measurement unit) modeling, beamforming theory, and platform stabilization methods [2]-[3]. Beamforming on a moving platform presents significant challenges due to the dynamic nature of both the transmitter and receiver, as well as the surrounding environment [4]. In an urban setting such as the one in Figure 1.1, signals often encounter obstacles such as buildings, vehicles, and other structures that cause reflection, diffraction, and scattering. These phenomena disrupt the direct path between the transmitter and receiver, leading to signal degradation and multipath interference. When a moving platform, such as a vehicle, is involved, the problem becomes more complex because the optimal beam direction must be continuously adjusted to maintain a reliable connection. The constantly changing environment introduces rapid variations in signal strength. Additionally, in modern communication systems like 5G, mobile phones act as both transmitters and receivers while base stations are connected to a centralized server, real-time tracking and beam alignment become crucial for optimizing communication performance. In 5G networks, each mobile phone con-

tains multiple antennas that enable beamforming. These mobile phones themselves act as moving platforms, continuously adjusting their beam direction to maintain a strong connection with the nearest base station. As users move, their devices must rapidly adapt to changes in signal conditions, avoid interference, and optimize data transmission [5]-[6].

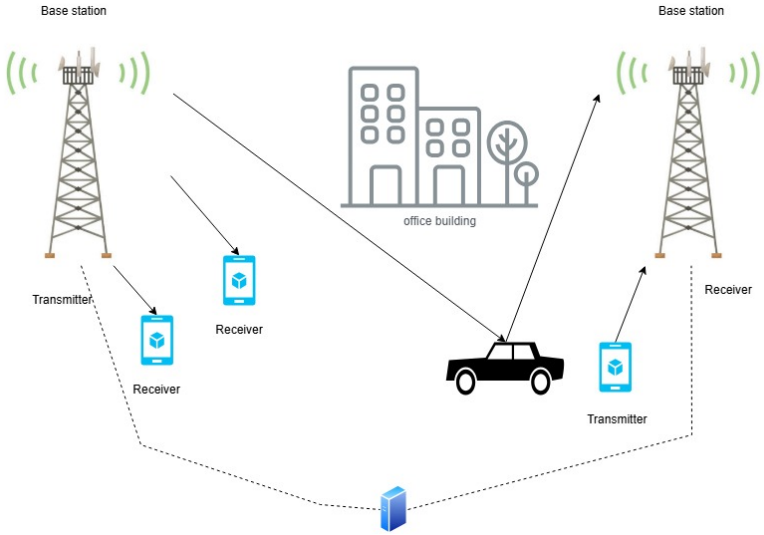


Figure 1.1: Environment for sensing in 5G network

1.1 Related work

Recently, much effort has been put into researching the area of motion compensation for ISAC Systems on mobile platforms like vehicles or drones. Most of the studies centered around kinematic platform motion modeling techniques with IMU data to mitigate the effects of acceleration, yaw, pitch, and roll on signal quality. Signal processing techniques for beamforming have been further developed to respond better to motion and environmental changes so that signal direction and strength are enhanced. Other researchers studied the division of labor between sensing and communicating where some radar-like functions are able to identify obstacles and modify the communication accordingly. In 5G systems, AI with real-time feedback has been employed in beam management and alignment and for 6G initial works are focusing on combining terahertz communication with intelligent beamforming and environmental awareness [3]. Addressing the issues of mobility, multipath interference, and dynamic beam control have been assisted by existing work towards creating motion-aware ISAC systems. Various studies have explored the impact of platform motion on radar and communication systems, addressing issues such as Doppler effects, phase errors, and adaptive beamforming techniques. Recent advances in ISAC systems have emphasized the need for accurate motion compensation models that leverage IMUs and digital beamforming algorithms. For example, previous work has demonstrated the effectiveness of Kalman filtering and sensor fusion for real-time motion estimation. Our work extends these approaches by integrating

a 6DoF (six degrees of freedom) IMU-based model with adaptive beamforming on a System-on-a-Chip (SoC) platform [7].

1.2 Purpose and goal

The purpose of this project is to improve the accuracy of sensing in dynamic environments by compensating for platform-induced motion errors. Specifically, the project aims to develop a mathematical model that maps 9DoF IMU data onto a steering vector for digital beamforming. The primary objectives are:

- To analyze and model the effects of acceleration, yaw, pitch, and roll on target detection and clutter.
- To design an algorithm that translates IMU data into motion compensation parameters for beamforming.
- Implement and validate the proposed model on PYNQ Z1 and Raspberry Pi platform integrated with an IMU sensor.

A successful implementation of this model will enhance the reliability of bistatic-implemented ISAC systems, improving performance in real-world applications.

1.3 Thesis outline

This thesis is structured as follows.

- **Chapter 1: Introduction** – Provides an overview of the problem statement, objectives, and importance of motion compensation in ISAC systems.
- **Chapter 2: Technical Background** – Covers the theoretical concepts, including platform kinematics, IMU-based motion modeling, Doppler and phase modulation effects, and beamforming principles.
- **Chapter 3: Methodology** – Details the proposed approach, including mathematical modeling, signal processing techniques, and hardware implementation.
- **Chapter 4: Implementation** – Describes the integration of the model with the PYNQ Z1 platform and IMU, along with considerations of software and hardware.
- **Chapter 5: Results** – Presents experimental results, performance analysis, and validation of the proposed approach.
- **Chapter 6: Discussion** –Interprets, analyzes, and reflects on the results and findings.
- **Chapter 7: Conclusion** – Summarizes key findings and suggests directions for future research.

2

Theory and technical background

This chapter will explain the key technical and theoretical concepts that are the basis of this thesis. Understanding these principles is important to fully understand the design, implementation, and results of the study. We will cover the main ideas and technologies that support the work, providing the necessary background to understand the later sections of the thesis.

2.1 Sensing topologies

Sensing technologies are important because they help the communication system detect objects, track movement, and improve the reliability of communication. By integrating sensing with communication, 6G can support applications such as smart cities, autonomous vehicles, and advanced security. Different sensing topologies affect how well the system can gather and use this information, making them a key part of ISAC networks. Sensing topologies define how transmitters and receivers are arranged within an ISAC system, significantly impacting performance, deployment complexity, and adaptability. The primary sensing topologies considered in the context of ISAC systems include monostatic, bistatic, and multistatic configurations[8].

In a monostatic topology, the transmitter and receiver are co-located, making it the simplest and most commonly used configuration. This setup benefits from straightforward synchronization and strong signal correlation, but suffers from self-interference, which can limit detection capabilities, especially in cluttered environments. Bistatic sensing, on the other hand, involves separate locations for the transmitter and receiver, providing unique advantages in detecting stealth targets and reducing vulnerability to jamming. However, bistatic systems require precise synchronization and advanced signal processing due to the varying bistatic angle and non-uniform propagation paths [9]-[10]. Multistatic sensing extends the concept of bistatic by incorporating multiple transmitters and/or receivers, offering improved target tracking, improved spatial coverage, and greater resilience against interference. Although multistatic systems provide superior sensing accuracy and robustness, they introduce significant challenges in terms of network coordination and data fusion[11], see Figure 2.1.

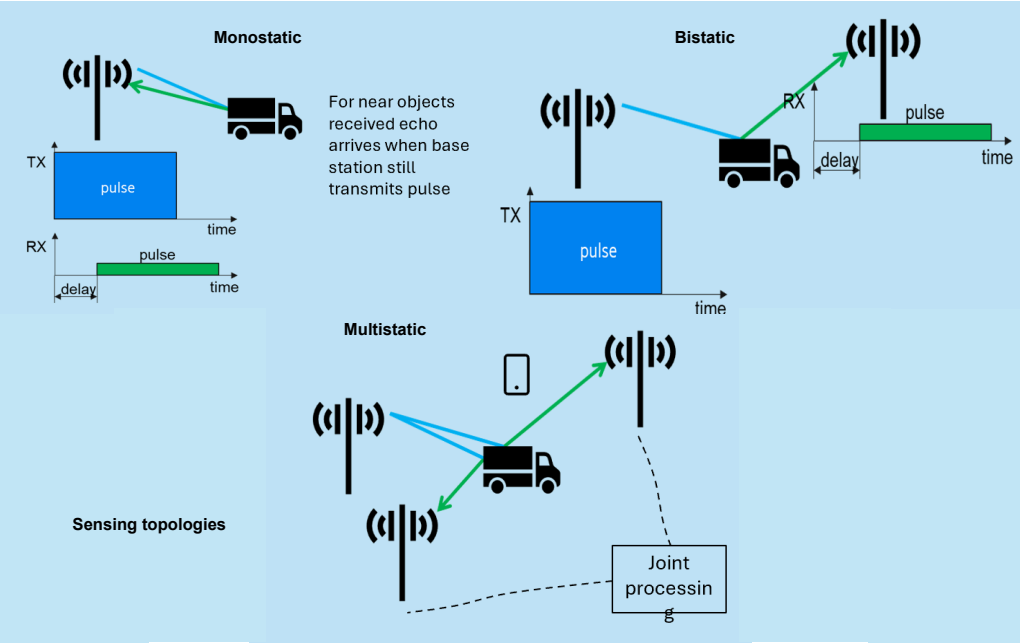


Figure 2.1: Sensing topologies

To optimize sensing performance under different conditions, ISAC systems can implement a topology-switching framework that dynamically transitions between these configurations based on real-time network and environmental factors. Simulations have shown that bistatic sensing often delivers the best trade-off between performance and deployment feasibility, making it a strong candidate for future ISAC implementations in 6G networks [8].

2.2 User equipments (UEs) in sensing systems

In the sensing and communication systems, different user equipment is included, such as smartphones and other connected devices. These devices can act as transmitters (TX) and/or receivers (RX) within a multi-static radar setup. However, for this to work effectively, UEs need to be properly synchronized, and their location and orientation must be accurately known. Some UEs may also be deployed by network operators and remain stationary, which can help address synchronization and accuracy issues that arise from mobile devices. By strategically placing stationary UEs performance and sensing capabilities of a radar system can be significantly improved [12].

One of the key limitations in radar sensing involves the base stations themselves. Depending on the specific environment, a base station might face interference that prevents it from effectively receiving radar signals. This interference can come from a variety of sources, including physical obstacles like buildings, trees, or even other electronic signals, which can all distort or block radar waves. As a result, radar-based sensing might be unreliable in areas with a lot of obstructions or noise. In practical terms, this means that while a base station may handle communication tasks efficiently, it may not be effective for radar sensing in certain situations, lim-

iting its utility for tracking or monitoring objects [13].

To overcome this limitation, multistatic radar setups require multiple radar nodes to maintain a clear line-of-sight (LoS) to the target object. In these systems, radar signals are sent and received from several different points, and each object must be within the LoS of both transmitters and receivers to be properly detected. However, in typical cellular networks, the goal is to minimize the overlap of cells to reduce interference between different network regions. While this approach is effective for communication purposes, it can pose a challenge for radar sensing. The need for clear LoS connections between multiple radar nodes means that overlapping cells might actually be beneficial for radar systems, creating a potential conflict between optimal network design for communication and the requirements for effective radar sensing [12].

To improve the accuracy of radar sensing, additional sensors can be added to the network. These sensors, often receivers (RX), are strategically placed in known locations with precise orientations. By incorporating these fixed sensors into the existing communication infrastructure, radar sensing systems can significantly improve their accuracy. With the exact position and angle of the sensors determined beforehand, the system can more reliably track objects and gather data, reducing errors caused by environmental factors or misalignment. This approach ensures that the radar system is not entirely dependent on the base station or mobile devices, but instead leverages fixed, highly accurate sensors to enhance the overall sensing performance [13].

UEs, such as smartphones or other connected devices, also have the potential to contribute to radar sensing by acting as both transmitters and receivers. These UEs can be integrated into the radar system to create a more dynamic and distributed sensing network. However, for this to work effectively, synchronization is crucial. UEs must be carefully coordinated in terms of timing, and their location and orientation must be known with high accuracy. This is especially important since the mobility of UEs could introduce errors if their position is not accurately tracked. Without proper synchronization, the radar system could face challenges in determining the exact location of objects or estimating distances, which would affect its overall performance.

To address the challenges posed by the mobility of UEs, network operators can deploy stationary UEs that remain fixed in one location. These stationary devices help solve the problems related to the synchronization of mobile devices, as they do not move and therefore do not require constant updates to their position or orientation. By strategically placing these stationary UEs within the network, operators can create a more stable and accurate radar system. This approach allows the radar network to rely on fixed sensor nodes, improving the overall sensing capability and reducing errors that might arise from the constant movement of mobile devices [12]-[14]. Usage of user equipment in a sensing topologies is given in the Figure 2.2.

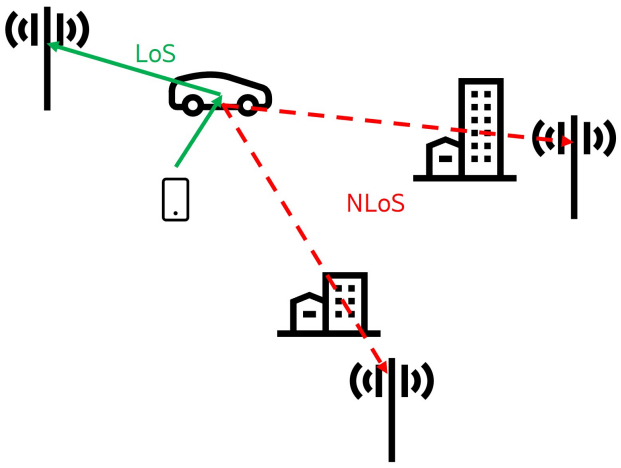


Figure 2.2: UEs in sensing topologies

In an ISAC system, in addition to stationary nodes that were described above, we also have moving nodes, which introduces several challenges that affect both communication and sensing functions. Unlike static deployments, moving nodes cause dynamic variations in channel conditions, beamforming requirements, and synchronization, leading to increased system complexity. In Figure 2.3,a node in motion is illustrated, demonstrating the dynamic behavior of the system.

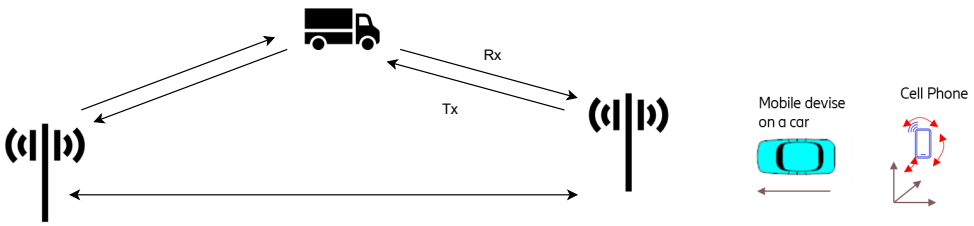


Figure 2.3: Moving node in an ISAC deployment

2.2.1 Impact of mobility on wireless communication

When a transmitter or receiver is in motion, the received signal experiences a Doppler shift which changes the carrier frequency. This can be expressed as:

$$f_d = \frac{v \cdot f_c}{c} \cos(\theta) \tag{2.1}$$

where f_d is a Doppler shift (the change in frequency), v is the speed of moving device, f_c is the original frequency of the signal, c is the speed of light and θ is the angle between the direction of the movement and the signal path. When the device moves towards the signal, the frequency increases (higher pitch), and when it moves away, the frequency decreases (lower pitch). This change in frequency can cause problems

like signal distortion and communication errors. To fix these issues, methods like adaptive equalization and channel estimation are used. Adaptive equalization helps adjust the signal in real-time to correct frequency shifts, while channel estimation predicts the changes and adjusts the transmission settings, ensuring clear and reliable communication even with movement. These techniques are essential to keep the signal accurate and prevent errors in dynamic situations [15].

As the node moves, multipath interference causes fast fading, which results in rapid fluctuations in the received signal strength. Fading means that the signal can take multiple paths to reach the receiver, each with a different delay and phase shift. The channel response, which describes how the signal behaves over time, can be modeled using a time-varying impulse function:

$$h(t,\tau) = \sum_{i=1}^{N} a_i(t) e^{j\phi_i(t)} \delta(\tau - \tau_i(t)) \tag{2.2}$$

In this equation, $a_i(t)$ represents the amplitude of the i-th path, $\phi_i(t)$ is the phase shift of the i-th path, and $\tau_i(t)$ is the delay of the i-th path. From provided formula we can see that as channel impulse response changes signal is fluctuating [16].

2.2.2 Impact of mobility on sensing capabilities

In ISAC systems, beamforming is used to direct signals toward specific targets, and this is crucial for sensing. When the transmitter and receiver are moving, the direction of the signal (beam) must constantly be adjusted to maintain a strong connection. The required beam angle θ_b is calculated using the following formula:

$$\theta_b = \tan^{-1}\left(\frac{y_t - y_r}{x_t - x_r}\right) \tag{2.3}$$

If the beam direction isn't updated in real-time as the transmitter or receiver moves, the signal will get misaligned. This misalignment reduces the strength of the signal and makes sensing less accurate. To solve this problem, techniques like AI-based beam tracking may be used, which predict the movement patterns of the devices and adjust the beam direction accordingly. Additionally, data from IMUs can be combined to compensate for motion, helping keep the beam properly aligned despite movement. These solutions ensure that the system maintains accurate sensing and communication even when the nodes are in motion [17].

Moving nodes introduce timing synchronization errors, which can negatively impact both sensing and communication. As the nodes move, the distance between them changes, which in turn affects the propagation delay. The delay at any given time, $\tau(t)$, is given by the formula:

$$\tau(t) = \frac{d(t)}{c} \tag{2.4}$$

where $d(t)$ is the time-dependent distance between the nodes, and c is the speed of light. As the distance between the nodes changes, the delay also fluctuates, causing synchronization issues. To address these issues, the system must continuously adjust its time synchronization to account for these changes. Additionally, predictive

algorithms are needed to compensate for the variations in delay, ensuring that both sensing and communication remain accurate and reliable despite the movement of the nodes [17].

2.3 OFDM framework

Orthogonal Frequency Division Multiplexing (OFDM) is already used in 5G networks, and it will likely continue to be used in 6G due to its ability to deliver high data rates and robustness against interference and signal fading. OFDM is a method of transmitting data by dividing it into multiple smaller data streams that are sent simultaneously over different frequencies. These data frequencies are orthogonal, meaning they don't interfere with each other, allowing for highly efficient use of the frequency spectrum. 6G networks will demand high transmission rates, low latency and the ability to handle massive amount of data. Using OFDM has many challenges as well such as frequency offset and PAPR (peak to average power ratio). Frequency offset occurs when there is a mismatch between transmitter and receiver frequencies. This can lead to errors in the signal reception and data loss. It is particularly problematic in high-speed, high-frequency 6G environments. Peak to average power ratio refers to the difference between the peak power and the average power of a signal. In OFDM, the signal's peak power can be much higher than its average power. In 6G network the complexity of communication system will increase and machine learning techniques will be used to help optimize and solve some of the issues OFDM faces. Machine learning algorithms can be used to predict and correct frequency offset by analyzing the patterns in the communication systems [18].

An OFDM transmitter uses a QAM (Quadrature Amplitude Modulation) modulator, which takes the incoming data and converts it into symbols that can represent multiple bits at once. This allows the system to send more data in a given time. Next, a serial-to-parallel converter splits this data into several parallel streams because OFDM transmits data over many different frequencies simultaneously. After that, the IFFT (Inverse Fast Fourier Transform) is applied to the parallel data streams, which changes the data into a format suitable for transmission over multiple subcarriers, essentially making it ready to be sent over the air. To make sure the signal can handle any delays or interference during transmission, a cyclic prefix is added. This is a small copy of the signal's beginning portion that's repeated at the start of each symbol, which helps prevent issues like echoes or multipath interference. Finally, the signal is sent through a Digital-to-Analog Converter (DAC), converting the signal from digital form to an analog signal, which can be transmitted through the communication channel, such as over the air in wireless communication. The signal will then travel through the channel, where it might face some noise or interference before reaching the receiver.

In Figure 2.4,a basic block diagram for OFDM system is used. For systems with multiple antennas, digital beamforming is applied after the serial-to-parallel conversion stage and before the IFFT stage. Digital beamforming involves adjusting the data for multiple antennas by applying a precoding matrix to optimize signal transmission across antennas and spatial layers. After precoding, the symbols are ready

for OFDM modulation. But before modulation (IFFT), they need to be placed onto the time-frequency grid to define exactly where each symbol will be transmitted in the frequency and time domains. Once this mapping is done, the signal is ready for IFFT (Inverse Fast Fourier Transform), which will turn it into the actual waveform for transmission [18]-[19].

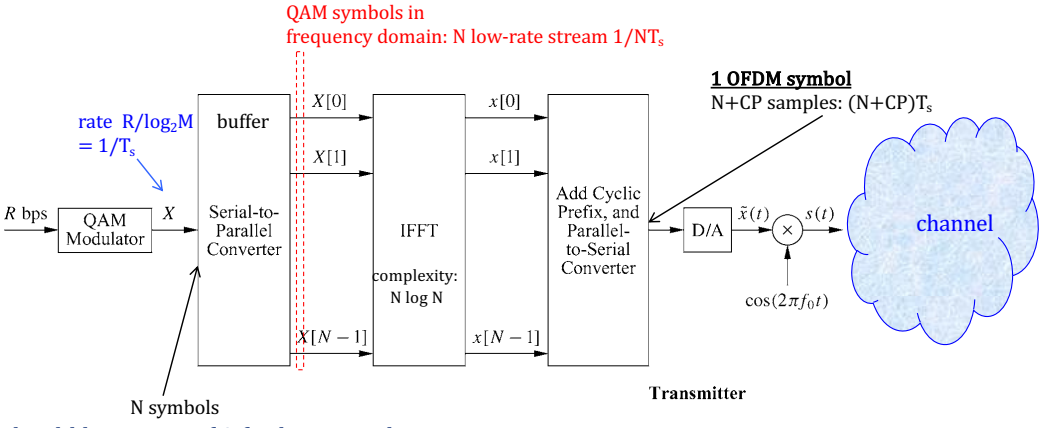


Figure 2.4: OFDM transmitter [18]

The OFDM receiver works by reversing the steps taken by the transmitter to recover the original data. First, it receives the signal through the channel, which may have added some noise or distortion. The receiver then removes the cyclic prefix, which was added at the transmitter side to prevent issues from multi-path interference. After removing the cyclic prefix, the receiver uses FFT (Fast Fourier Transform) to convert the received signal from the time domain back into the frequency domain. This step is essential because OFDM transmits data on multiple sub-carriers, and FFT helps separate the data from each individual sub-carrier, allowing the receiver to analyze them independently. Once the data is separated, parallel-to-serial conversion is used to combine the parallel streams back into a single serial data stream, just as they were originally split at the transmitter. Finally, the QAM demodulator is used to decode the modulated symbols, such as QPSK (Quadrature Phase Shift Keying) or 16QAM, and translate them back into bits, effectively recovering the original message sent by the transmitter. In short, the receiver takes the distorted signal, removes the unnecessary parts, processes it to separate the data, and then decodes it to retrieve the original information[19], see Figure 2.5.

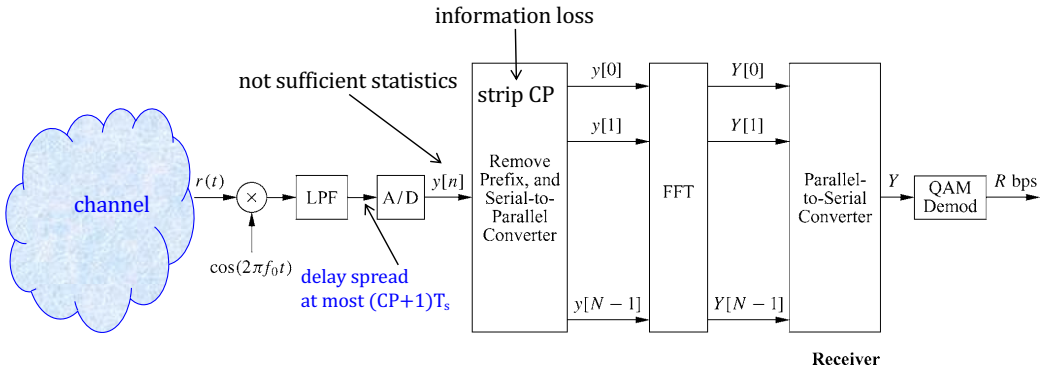


Figure 2.5: OFDM receiver [18]

Inside each base station there is a centralized transmitter or receiver or both that takes the data(video, text, internet traffic), splits it into multiple subcarriers (different frequency channels), modulates the sub-carries to prepare them for transmission. Then, the MIMO (Multiple Input Multiple Output) system takes these OFDM-modulated signals and transmits them through multiple antennas. The OFDM transmitter is not inside each antenna. Instead, the OFDM transmitter processes the signal first, and then the MIMO antennas transmit the processed signal over the same frequency band. This enables spatial multiplexing, where each antenna can transmit different data, increasing the overall capacity of the communication system. Each antenna receives a slightly different version due to reflections, obstacles, or different transmission paths [18].

2.4 Fundamentals of antennas and beamforming

Antennas are essential components in modern wireless communication and radar systems that enable transmission and reception of electromagnetic waves. Although a single antenna can transmit signals in all directions, antenna arrays offer greater control over signal directionality, enabling enhanced performance through beamforming. Beamforming is a signal processing technique that steers signals in specific directions by adjusting the phase and amplitude of signals across multiple antenna elements. This technique is used in 5G and 6G networks, radar systems, and satellite communications, where precise signal directionality is required to improve efficiency, reduce interference, and improve coverage. In this section, we review the fundamentals of antennas, the principles of beamforming, and the different beamforming techniques that are essential to achieve high-performance wireless communication systems [20].

2.4.1 Definition of beamforming and its importance

As already mentioned, beamforming is a method that makes signals clearer by focusing them in specific direction. It helps increase data capacity between network towers and user devices by making communication more efficient. Instead of send-

ing signals in all directions, beamforming focuses them directly toward intended receiver. This reduces interference and allows multiple users to receive stronger, clearer signals at the same time. As a result, more data can be transmitted without delays or disruptions, improving internet speed and reliability [20]-[21].

Beamforming began as a straightforward technique and has evolved significantly over time. In its early stages, it used fixed phase shifters to create beams at a single frequency. These simple phase shifters would adjust the signal phase in a specific pattern, resulting in a directional signal. As technology progressed, a switching system was introduced, incorporating multiple phase shifters that allowed for more flexibility in beamforming. This was a significant step, as it enabled the system to switch between different beams, improving performance. Eventually, the technology advanced even further, and fully adjustable phase shifters were integrated into each antenna element. This development made beamforming much more dynamic, enabling the creation of adaptive beams that could be steered in any direction as needed. The simplified analog beamforming framework is introduced in Figure 2.6. In this setup, the main focus is on explaining how the entire system works [22].

Analog beamforming is a way to control how a radio signal is sent out from multiple antennas. First, the signal, which carries data like voice or internet information, is converted into a radio wave (called a radio frequency (RF) signal) so it can travel wirelessly. This signal is then sent through feedlines, which are like wires that deliver the signal to each antenna in the system. However, before the signal reaches each antenna, it is delayed slightly using phase shifting. Phase shifting means changing the timing of the signal so that some antennas send it a little earlier and some a little later. This small delay is very important because it controls how the waves from all the antennas combine in the air. When done correctly, the waves add up in one direction (making the signal stronger there) and cancel out in other directions (reducing interference). This allows the system to steer the signal toward a specific location without physically moving the antennas. This makes communication clearer, stronger, and more efficient by sending the signal exactly where it is needed while avoiding unwanted areas. When the signal is received, it is adjusted, and filtered and converted back into a digital signal. Analog beamforming has limitations as well like precise control and difficulty to handle multiple users [21]-[22].

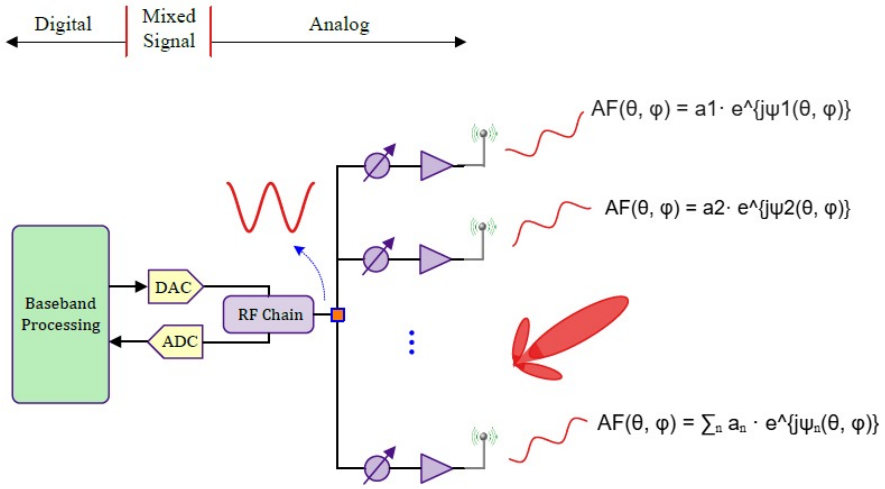


Figure 2.6: Analog beamforming

Digital beamforming is a more advanced way of directing wireless signals compared to analog beamforming. It helps solve the problems of scalability, especially when working with many antennas. In digital beamforming, each antenna has its own Digital-to-Analog Converter (DAC) and Analog-to-Digital Converter (ADC), which allows multiple beams to be sent and received at the same time, see Figure 2.7. Unlike analog beamforming, which relies on adjusting the timing (phase) of signals to steer the beam, digital beamforming processes signals mathematically using a technique called precoding. Precoding adjusts the signals before they are sent out so that they combine in a way that creates beams in specific directions. This means the system can send signals to multiple users at once without interference, making it especially useful for Multi-User MIMO (MU-MIMO). Precoding typically requires knowledge of the channel conditions, known as Channel State Information (CSI), which can be obtained through feedback from the receiver or estimated at the transmitter. However, digital beamforming requires more power and is more expensive because of the extra processing involved [22].

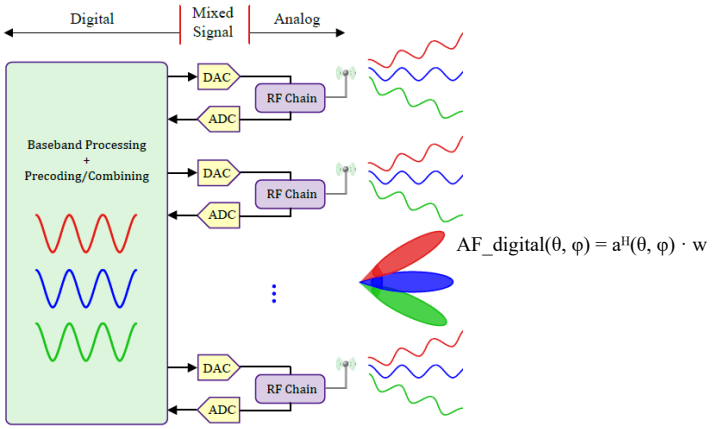


Figure 2.7: Digital beamforming

2.4.2 Introduction to antenna arrays

An antenna array is a set of N spatially separated antennas. The number of antennas in an array ranges from two to several thousand. An antenna array helps improve the signal by making it stronger, reducing noise, and focusing the signal in certain directions. This leads to better reception, helps find where signals are coming from, and makes the connection clearer overall. The simplest form of an array is the one that has two elements, where two identical antennas are positioned along an axis. The total signal from the antenna array is the result of adding up the signals from each individual antenna, taking into account their strength and direction. [23]. The total electric field is just the sum of the fields from each antenna.

$$E_t = E_1 + E_2 \tag{2.5}$$

For a two-element array positioned along the z-axis, the radiated electric field in the far-field region is given by:

$$E_t = \frac{a^{\theta} j\eta}{4\pi r} k I_0 l e^{-jkr} \cos\theta \left[e^{+j\frac{(kd\cos\theta+\beta)}{2}} + e^{-j\frac{(kd\cos\theta+\beta)}{2}} \right] \tag{2.6}$$

This equation shows how the radiated field is influenced by the element spacing, phase shifts, and the direction of observation where $k = \frac{2\pi}{\lambda}$ is the wave number, d represents the spacing between the two antenna elements, β is the phase difference between them, and θ is the observation angle [23].

From this, we define the Array Factor (AF) as:

$$AF = 2\cos\left[\frac{1}{2}(kd\cos\theta + \beta)\right] \tag{2.7}$$

or in normalized form:

$$(AF)_n = \cos\left[\frac{1}{2}(kd\cos\theta + \beta)\right] \tag{2.8}$$

The array factor is a function of the geometry of the array and the excitation phase. By varying the separation d and/or the phase β between the elements, the characteristics of the array factor and of the total field of the array can be controlled [24].

Now if we want to expand the concept with N uniformly-spaced elements, we can assume that phased array consists of multiple antennas outputs $X_1, X_2, X_3, X_4, \ldots, X_n$ arranged in a defined geometry. Each element in the array is assigned a weighting factor that adjusts its amplitude and phase. The output signals from all elements are summed to form the overall radiation pattern. The sum of the output signal of the phased array is shown in Figure 2.8.

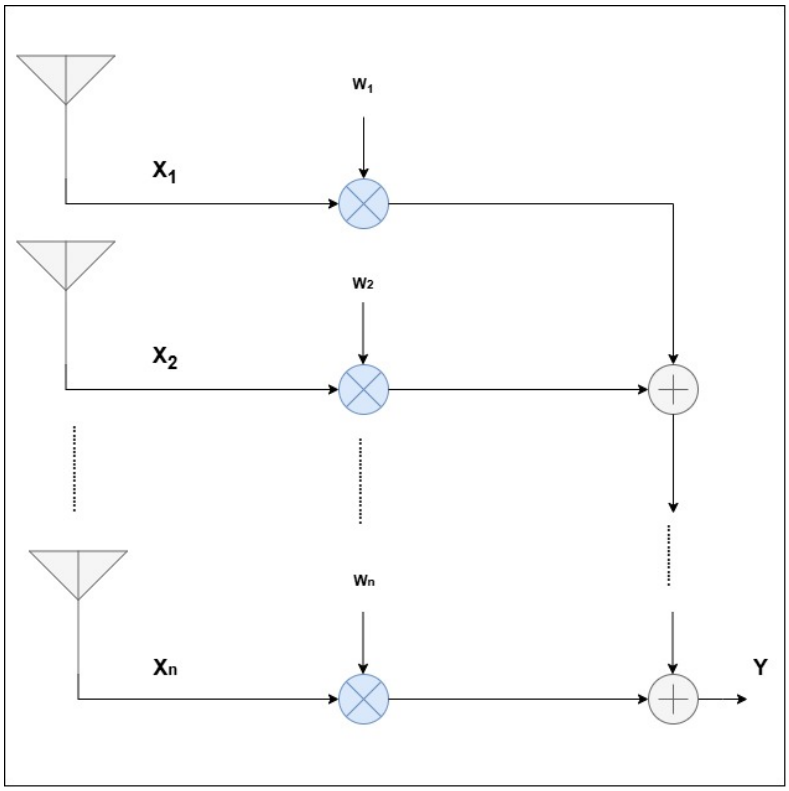


Figure 2.8: Summing of the signals to form output in phased array

In reference to the figure 2.8, the output of the antenna array can be written as:

$$Y = \sum_{n=1}^{N} w_n E_n \tag{2.9}$$

For an N-element uniformly spaced linear array, where elements are separated by distance d, the array factor is given by:

$$AF(\theta) = \sum_{n=0}^{N-1} e^{jn(kd\cos\theta+\beta)} \tag{2.10}$$

By changing the phase β or the distance d between the antennas, we can control the main direction of the signal, reduce unwanted signals (sidelobes), and adjust the spots where there's no signal (nulls), which helps in steering the beam and reducing interference.

Previously it was shown how uniformly-spaced antenna array is calculated, but if we want to steer the array towards the desired direction selected weight can be calculated by using this formula :

$$w = e^{j(kdN\cos\theta_d)} \tag{2.11}$$

In this equation, k represents the wave number, and d is the spacing between adjacent array elements. These weights adjust the phase and amplitude of the signal

at each antenna element, helping to focus the beam towards the target direction. However, a problem can occur in uniformly spaced arrays when the spacing between the antenna elements is too large. In this case, the array may also radiate strong signals in other directions, creating grating lobes, which are unwanted beams that appear due to the antenna elements being spaced too far apart, causing the array to unintentionally radiate energy in directions other than the desired one. These additional radiation beams can lead to interference, changing the direction and efficiency of the main beam [20].

To address the issue of grating lobes and improve beam steering capabilities, phased array antennas are designed to dynamically adjust the phase and amplitude of individual antenna elements in a controlled manner. Unlike traditional antenna arrays with fixed element properties, phased arrays incorporate adaptive beamforming techniques to optimize radiation patterns in real-time. A phased array antenna consists of multiple radiating elements, where each element's signal is controlled through electronically adjustable phase shifters, which means that direction of the signal is changed by adjusting the phase of each antenna. This allows the overall radiation pattern to be modified without mechanically moving the array, making phased arrays highly flexible for various applications, including radar, wireless communications, and satellite systems. [20]. The formula below calculates the phase shift for each antenna element in a phased array to steer the main signal beam in the desired direction:

$$w_n = e^{j(kdn\cos\theta_d+\phi_n)} \tag{2.12}$$

By carefully selecting ϕ_n, phased array antennas can achieve beam steering, null steering, and beam shaping, allowing for greater control over signal directionality and interference suppression.

2.5 Impact of boat motion on antenna array

Even though beamforming was described in previous sections, in this section the entire problem will be presented with more details. Imagine a boat sailing across the vast Atlantic Ocean, equipped with an array of antennas. These antennas help the boat stay connected by communicating with satellites and other vessels. They are used for navigation, weather updates, and sending signals over long distances. Even in the middle of the ocean, the antenna array ensures the boat can receive important information and stay in touch with the outside world. As the boat speeds through the sea, its antenna array is constantly affected by six different types of motion. It surges forward and backward along the x-axis, swayed side to side along the y-axis, and heaved up and down along the z-axis due to the waves. Additionally, the boat experienced rotational movements: rolling from side to side (ϕ), pitching forward and backward (θ), and yawing left and right (ψ). Each of these motions influences how the antennas transmit and receive signals, making precise beamforming adjustments necessary to maintain stable communication [25]. An illustration is given in Figure 2.9.

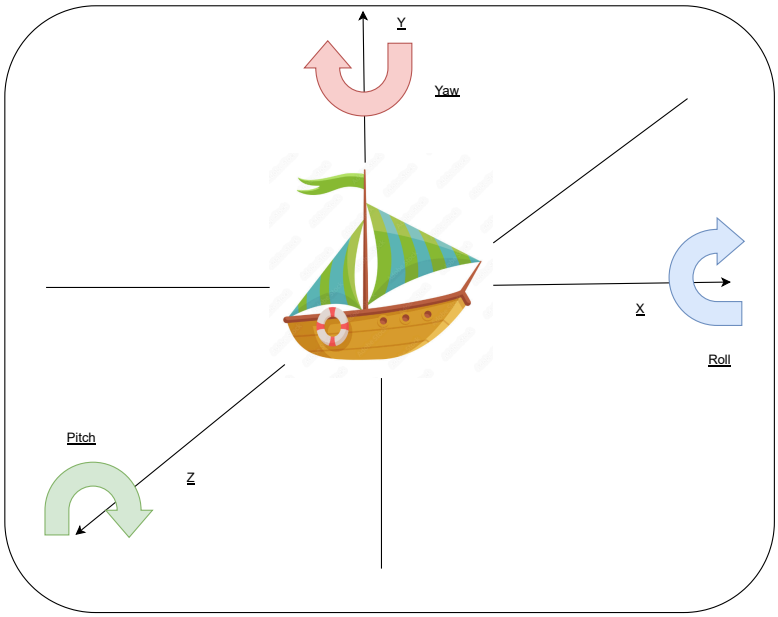


Figure 2.9: Movements of a floating boat

As the boat is moving through the sea, the antennas are not staying in the fixed positions. Normally, the position of each antenna in the array was defined by fixed coordinates (x_n, y_n, z_n). However, because the boat was in motion, the actual position of each antenna at any given time changed. This movement was influenced by two factors: rotation and velocity. The rotation, represented by rotation matrix, explained the tilting motions of the boat, roll, pitch, and yaw. Meanwhile, the velocity vector captured the boat's linear movements—surge, sway, and heave. Together, these effects determined the true position of each antenna at any moment, which was crucial for adjusting the beamforming process and maintaining clear communication [25]. A true position of antenna is given by the formula:

$$p_n(t) = p_n(0) + R(\kappa(t))p_n(0) + v(t)t \tag{2.13}$$

Changing the position of the antenna array has great effect on antenna signals. When the ship is stationary, the antenna array works by aligning the phase of the signals received at each antenna. This alignment is mathematically represented by the formula:

$$\mathbf{a}(\theta) = \left[e^{-j\frac{(N-1)d}{c}\omega\sin\theta}, \ldots, e^{j\frac{(N-1)d}{c}\omega\sin\theta}\right]^T \tag{2.14}$$

Signal phase in this formula depends on number of antennas (N),the spacing between them (d), the signal frequency (ω), and the angle of the incoming signal (θ). The signal phases are perfectly aligned, so the system can process them easily. However, when the ship moves, everything changes. The motion of the ship causes phase shifts in the received signals, making it harder to maintain synchronization. The new signal formula for a moving ship becomes:

$$a(\theta, \kappa(t)) = e^{j\psi(t)\sin\theta} \cdot e^{j\phi(t)\cos\theta} \cdot e^{j\frac{(N-1)d}{c}(\omega\sin\theta + \dot{x}(t) + \dot{y}(t))} \tag{2.15}$$

The new formula $a(\theta, \kappa(t))$ accounts for the ship's motion. It includes additional factors that represent the yaw, pitch, roll, surge, and sway of the ship. Yaw ($\psi(t)$) causes slight shifts in the phase. Pitch (θ) and roll ($\phi(t)$) change the angle at which the signal arrives, which affects the received signal. Surge ($\dot{x}(t)$) and sway ($\dot{y}(t)$) cause Doppler shifts—frequency changes due to the motion of the ship—that distort the signal. In simple terms, the ship's movement messes up the timing and angles at which the antennas receive the signal [25].

If we want to predict which of these motions matters most for the signal received or transmitted by an array of antennas on a moving boat Taylor series expansion can be used. From this expansion we can conclude how sensitive the signal is to the changes in the boat's position and direction.

$$\Delta a(\theta, t) \approx \frac{\partial a(\theta)}{\partial \theta}\Delta\theta + \frac{\partial a(\theta)}{\partial \phi}\Delta\phi + \frac{\partial a(\theta)}{\partial x}\dot{x}(t) + \frac{\partial a(\theta)}{\partial y}\dot{y}(t) \tag{2.16}$$

2.6 Homogeneous coordinates

In Euclidean coordinates, a point in the 3D space is represented by a triple of coordinate values (x, y, z), which indicates its position along three axes. In homogeneous coordinates a fourth value is added, called weight. These extra dimensions made homogeneous coordinates useful in processes like rotation, transformation, and scaling. Standard 2D Cartesian coordinates and 3D Euclidean coordinates make some transformation (such as translation) difficult to express using simple matrix multiplication. So, a 3D point for Eucledian to homogeneous coordinates is represented as (x, y, z) $\rightarrow$ (x, y, z, w) [26].

As a type of transformation, translation refers to the movement of an object from one place to another without rotation. In homogeneous coordinate system, translation is represented as matrix multiplication with translation matrix. If we want to translate a point $P(x, y, z)$ by (T_x, T_y, T_z), we use a 4x4 homogeneous translation matrix:

$$T = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.17}$$

Multiplying the point's homogeneous coordinates $P_h = [x, y, z, 1]$ by the translation matrix gives:

$$P' = T \cdot P_h = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + T_x \\ y + T_y \\ z + T_z \\ 1 \end{bmatrix} \tag{2.18}$$

This translates the point $P(x, y, z)$ to a new position $P'(x + T_x, y + T_y, z + T_z)$.

Rotation in 3D refers to rotating an object around one or more of the coordinate axes. The rotation is represented by a rotation matrix that can be applied to any

point or vector in 3D space. There are three main types of rotations in 3D, depending on which axis the object rotates around [26] - [27]. The rotation matrix for a roll (rotation around the x-axis) is:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.19}$$

The rotation matrix for a pitch (rotation around the y-axis) is:

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.20}$$

The rotation matrix for a yaw (rotation around the z-axis) is:

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.21}$$

These rotation matrices can be combined to perform rotations around multiple axes. For example, to perform a combined rotation of pitch, roll, and yaw, the total rotation matrix is the product of the individual rotation matrices:

$$R_{\text{total}} = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi) \tag{2.22}$$

To apply the rotation to a point $P(x, y, z)$, the rotation matrix is multiplied with the point's homogeneous coordinates:

$$P' = R_{\text{total}} \cdot P_h \tag{2.23}$$

This rotates the point $P(x, y, z)$ by the specified roll, pitch, and yaw angles.

One of the transformations that will be described but not used in this master thesis is scaling. This transformation changes the size of an object in 3D space. We can scale an object uniformly (in all directions) or non-uniformly (in different directions). Scaling is represented by a scaling matrix [27].

For example, to scale a point $P(x, y, z)$ by factors s_x, s_y, s_z along the x, y, and z axes, the scaling matrix is:

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.24}$$

Multiplying the point's homogeneous coordinates $P_h = \begin{bmatrix} x & y & z & 1 \end{bmatrix}$ by the scaling matrix gives:

$$P' = S \cdot P_h = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x \cdot x \\ s_y \cdot y \\ s_z \cdot z \\ 1 \end{bmatrix} \tag{2.25}$$

This scales the point $P(x, y, z)$ to a new position $P'(s_x \cdot x, s_y \cdot y, s_z \cdot z)$, effectively changing the size of the object.

In the most of the cases, rotation, translation and scaling will be combined to manipulate the object position, orientation and size by multiplying the respective transformation matrices:

$$T_{\text{total}} = T \cdot R_{\text{total}} \cdot S \tag{2.26}$$

Then this matrix can be applied to the point P_h:

$$P' = T_{\text{total}} \cdot P_h \tag{2.27}$$

2.7 Computing the steering vector from IMU data

When the sensor moves, its orientation changes based on rotational and translational components. To determine how much sensor moved in space meaning in position x, y, z we use translation vector [28].

$$T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.28}$$

To determine whether translation has occurred, we compute the displacement from acceleration:

$$T = \int v \, dt = \int \int a \, dt^2 \tag{2.29}$$

where a is the measured acceleration from the IMU. If acceleration is zero, there is no translation. If we want to check where the sensor is pointing after rotation, we use rotational matrix.

$$R = R_z(\psi) R_y(\theta) R_x(\phi) \tag{2.30}$$

Multiplying these three matrices together gives:

$$R = \begin{bmatrix} \cos\theta \cos\psi & \cos\theta \sin\psi & -\sin\theta \\ \sin\phi \sin\theta \cos\psi - \cos\phi \sin\psi & \sin\phi \sin\theta \sin\psi + \cos\phi \cos\psi & \sin\phi \cos\theta \\ \cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi & \cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi & \cos\phi \cos\theta \end{bmatrix} \tag{2.31}$$

To account both rotation and translation, sensor position has to be updated like this:

$$P' = TP \tag{2.32}$$

In this equation T stands for homogenous transformation matrix that can be calculated as follows:

$$T = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.33}$$

The entire transformation point can be expressed as:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.34}$$

This transformation is particularly important in antenna beamforming, where the azimuth and elevation angles describe the direction in which the antenna is pointing. The azimuth angle represents the horizontal direction of the antenna's radiation pattern [28]. It is measured in the XY-plane from a reference direction, typically the positive X-axis, spans from 0° to 360°, and can be calculated like this:

$$\theta = \tan^{-1}\left(\frac{y'}{x'}\right) \tag{2.35}$$

The elevation angle denotes the vertical direction of the antenna's radiation pattern. It is measured from the horizontal plane (XY-plane) upward along the Z-axis, ranging from 0° (on the horizon) to 90° (directly overhead). This angle describes how the antenna's radiation pattern is oriented above or below the horizon [28]. For instance, an elevation angle of 45° indicates that the antenna's main lobe is directed halfway between the horizon and the zenith, and it can be calculated like this:

$$\phi = \tan^{-1}\left(\frac{z'}{\sqrt{x'^2 + y'^2}}\right) \tag{2.36}$$

Figure 2.10 shows the azimuth and elevation angles used to describe the orientation of an object in 3D space.

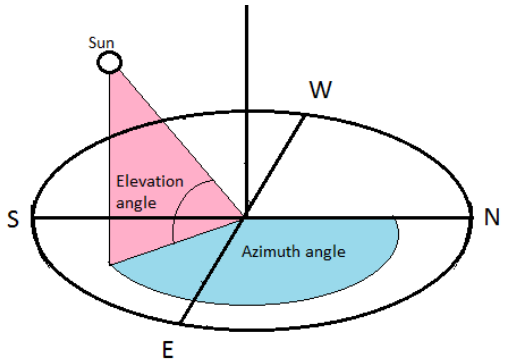


Figure 2.10: Azimuth and elevation angles illustration [29]

2.8 Preprocessing the collected data

Once the raw IMU data is acquired, it undergoes preprocessing to clean and smooth the data, which can be noisy due to factors like vibrations or electrical interference. The preprocessing step involves filtering the data using either the Kalman Filter or the Complementary Filter. The Kalman Filter is a more advanced technique that uses a recursive approach to estimate the system's state. It involves two main steps: the prediction and update steps [30]. In the prediction step, the state is predicted based on the previous state and control input:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{2.37}$$

where $\hat{x}_k^-$ is the predicted state, A is the state transition matrix, u_k is the control input, and B is the control input matrix. In the update step, the K_k is computed, and the state estimate is corrected using the measurement:

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R} \tag{2.38}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{2.39}$$

In the Kalman gain equation, P_k represents the state estimate covariance, H is the measurement matrix, R is the measurement noise covariance, and K_k is the Kalman gain that optimally blends the predicted state and measurement.

The Complementary Filter, on the other hand, is a simpler method that combines accelerometer and gyroscope data using the formula:

$$\theta_k = \alpha\left(\theta_{k-1} + \Delta t \cdot \omega_k\right) + (1-\alpha) \cdot \theta_{\text{accel}} \tag{2.40}$$

where θ_k is the estimated angle, ω_k is the angular velocity from the gyroscope, and θ_{accel} is the angle computed from the accelerometer data. The constant α determines the weight given to the gyroscope versus the accelerometer data. Both filtering techniques help eliminate noise and smooth out sudden changes in the data, ensuring that the cleaned data can be used for the next step in the process, such as the beamforming computation for the antenna array [31].

2.9 Motion compensation in beamforming

Once all data are collected and preprocessed, the next step is to do the beamforming computation with the set of formulas that were provided in previous chapter. As was mentioned, when a platform (like a boat or drone) moves, it causes the signals received by the antennas to become distorted. To correct this and keep the beamforming system working properly, we need to make adjustments based on the platform's movement. The main adjustments we have to do are: correcting the phase shifts, correcting Doppler shifts, and tracking the platform orientation [32].

As the platform moves, the antennas will experience slight phase shifts due to the relative motion. To correct these phase shifts, we will calculate the phase correction for each antenna element based on the platform's velocity and acceleration data. The phase shift correction for antenna element is given by:

$$\theta_{\text{corrected}} = \theta_{\text{original}} + \Delta\theta_{\text{motion}} \tag{2.41}$$

where θ_{original} is the initial phase calculated from the signal's angle of arrival and $\Delta\theta_{\text{motion}}$ is the phase shift caused by the platform's movement, which can be determined by the motion-induced velocity and acceleration data provided by the IMU.

The movement of the platform relative to the incoming signal causes Doppler shifts in frequency, meaning the frequency of the received signal will appear different due to the relative motion. This must be corrected to prevent distortions in the beamforming process [32]. The Doppler frequency shift can be calculated as:

$$f_{\text{doppler}} = \frac{v}{c} f_{\text{carrier}} \tag{2.42}$$

The Doppler shift correction is applied by adjusting the signal's frequency as it is received. This ensures that the received signal matches the expected frequency, regardless of the platform's motion.

The antenna array's steering vector $a(\theta, t)$ is updated dynamically based on the platform's orientation. To update the steering vector, the corrected phase will be used and calculated. To adjust the vector two angles azimuth θ and the elevation ϕ will be used [33]. The steering vector for all antennas in the array can be expressed as:

$$a(\theta, \kappa(t)) = \begin{bmatrix} e^{j(\psi(t)\sin\theta)} \cdot e^{j(\phi(t)\cos\theta)} \cdot e^{j\frac{c}{(0)d}(\omega\sin\theta+\dot{x}(t)+\dot{y}(t))} \\ e^{j(\psi(t)\sin\theta)} \cdot e^{j(\phi(t)\cos\theta)} \cdot e^{j\frac{c}{(1)d}(\omega\sin\theta+\dot{x}(t)+\dot{y}(t))} \\ \vdots \\ e^{j(\psi(t)\sin\theta)} \cdot e^{j(\phi(t)\cos\theta)} \cdot e^{j\frac{c}{(N-1)d}(\omega\sin\theta+\dot{x}(t)+\dot{y}(t))} \end{bmatrix} \tag{2.43}$$

Here, each row corresponds to the steering vector for a specific antenna element in the array, where the index n varies from 0 to $N-1$. The term $(N-1)\cdot d$ ensures the phase shift for each antenna element is adjusted based on its position within the array.

Once the phase shifts and Doppler corrections are applied, the final step is to apply the updated steering vector in real-time beamforming computations, this would

involve dynamically adjusting the weights for each antenna element to focus the beam in the desired direction.In equation 2.44, $a(\theta, t)$ is the updated steering vector and S is the vector of received signal data.

$$\text{Beamformed Signal} = a(\theta, t) \cdot S \tag{2.44}$$

In a Python implementation (using the PYNQ framework or other computational systems), this formula would be applied continuously in a loop, where you read the real-time data from the MPU6050 IMU, compute the angles θ, $\psi(t)$, and $\phi(t)$, and update the steering vector for the antenna array.

2.10 Inertial measurement unit for motion tracking

Inertial measurement unit (IMU) device tracks the movement of an object in the space. It uses a combination of sensors to detect changes in the position, orientation, and velocity of the object. IMUs are commonly used in devices like airplanes, boats, drones, and smartphones to help determine the object's motion and orientation without relying on external references (like GPS). An IMU usually contains three main sensors that work together to measure different aspects of motion: accelerometer, gyroscope, and magnetometer [3].

An accelerometer is a sensor that measures the linear acceleration of an object along three axes (x, y, and z), which corresponds to how quickly the object is speeding up or slowing down in any direction. It detects changes in velocity, whether the object is moving forward, backward, up, down, or sideways. By continuously measuring these accelerations, the accelerometer provides real-time data on the object's movement and position. In applications like drones or boats, the accelerometer helps track the object's speed, direction, and any forces acting on it, such as when the boat accelerates forward or tilts due to waves. This data allows the system to adjust and control the motion to ensure smooth navigation and maintain stability, making the accelerometer essential for tracking motion and maintaining accurate positioning in dynamic environments [34].

A gyroscope is a sensor that measures the rate of rotation, or angular velocity, of an object around its axes (roll, pitch, and yaw). It detects how fast an object is rotating in three directions: roll (side-to-side tilt), pitch (up-and-down tilt), and yaw (turning left or right). By continuously tracking these rotations, the gyroscope provides real-time data on the object's orientation, helping to maintain stability and control. In applications like drones, for example, the gyroscope detects if the drone is tilting or rotating unexpectedly, and this data is sent to the flight controller to make adjustments, such as powering specific motors to correct the orientation. This enables the object to maintain a desired position and direction, making the gyroscope crucial for ensuring smooth, stable motion in dynamic environments.

A magnetometer is a sensor that measures the magnetic field around an object, typically to determine its orientation relative to the Earth's magnetic field, or magnetic north. It detects changes in the direction and strength of the magnetic field along three axes (x, y, and z) [3]. By measuring the magnetic field, the magnetometer helps determine the heading or direction of the object, much like a compass. In applications like boats or drones, the magnetometer provides essential data about the object's orientation, helping to correct and maintain its course. For example, if a drone turns or drifts off course, the magnetometer helps the system understand which direction it is facing, enabling it to make adjustments to ensure the object stays on the desired path. This makes the magnetometer vital for accurate navigation and orientation, especially in environments where GPS signals may not be reliable [34].

In the figure 2.11 IMU sensor MPU-6050 that will be used during this project is shown.

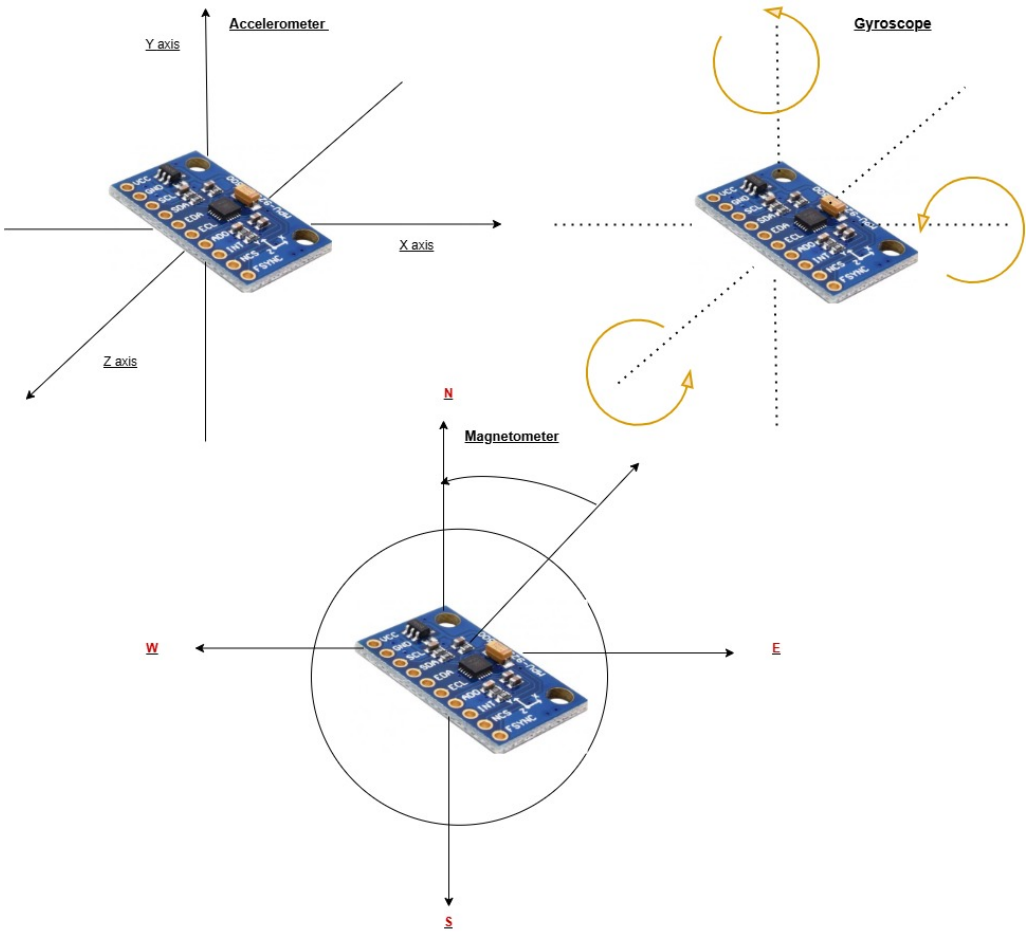


Figure 2.11: IMU sensor MPU-6050

2.11 PYNQ-Z1 board technical background

The PYNQ-Z1 (Python Productivity for Zynq) is an FPGA development board built around the Xilinx Zynq-7000 SoC (System on Chip). This chip combines a dual-core ARM Cortex-A9 processor with programmable logic (FPGA fabric) on the same die, enabling close interaction between software and hardware in a single device.

The hardware system block diagram, as shown in Figure 2.12, illustrates how different components are connected inside the FPGA. It typically includes the ARM processing system, various custom IP blocks for signal processing or control, AXI interconnects (Advanced eXtensible Interface) for data exchange, and peripheral interfaces such as I2C (Inter-Integrated Circuit), SPI (Serial Peripheral Interface), and GPIO (General-Purpose Input/Output). This diagram provides a clear overview of how hardware modules are structured and how they communicate with software, enabling efficient real-time data processing and hardware acceleration.

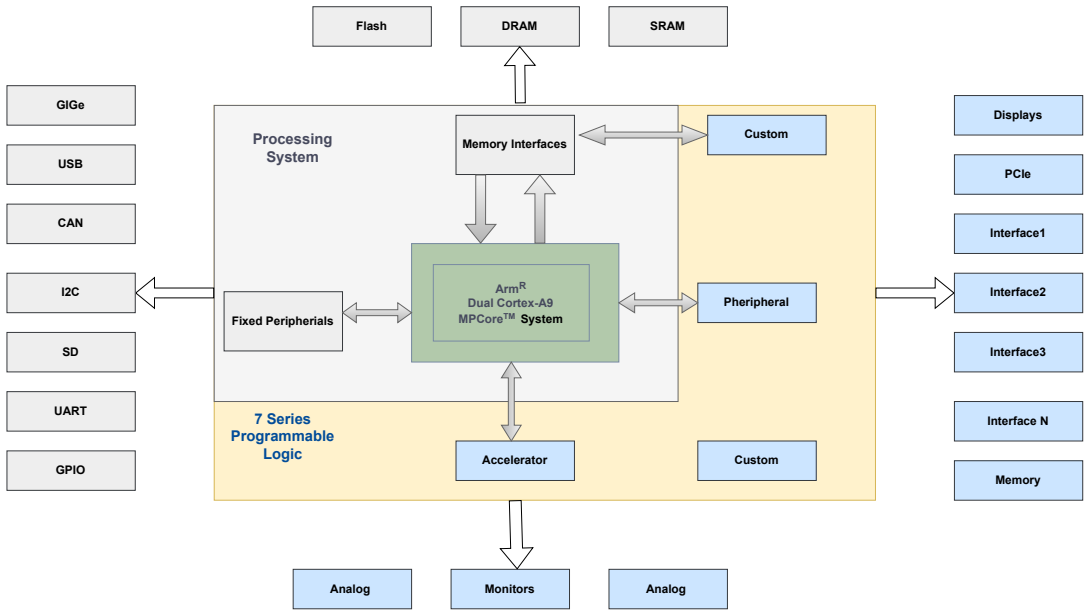


Figure 2.12: A hardware system block diagram of a PYNQ Z1

The PYNQ framework simplifies FPGA development by allowing control logic to be written in Python, typically within Jupyter Notebooks. These Python scripts run on the ARM processor inside the Zynq chip and are used to control hardware blocks, called IP cores (Intellectual Property cores), on the FPGA fabric. These IP cores are grouped into what is known as an *overlay*, which usually consists of a bitstream (`.bit`) file that programs the FPGA and a hardware description (`.hwh`) file that provides metadata about the design.

It is important to understand that Python is not used to design the hardware itself, but rather to interact with and control it. When developers need custom hardware, they write logic in C or C++ using tools such as Vivado HLS (High-Level Synthesis) or Vitis HLS. This code is then synthesized into HDL (Hardware Description

Language), packaged into an IP core, and connected within a Vivado block design. Once complete, the hardware is compiled into a new bitstream that is loaded onto the PYNQ board.

3

Approach

This chapter explains how we will implement and test beamforming on a moving platform. The main goal is to track the movement of the platform in real time, understand how this motion affects the signal received by an antenna array, and then adjust the beamforming process to correct for these disturbances. To do this, we will first use an inertial measurement unit (IMU) sensor to measure how the platform moves. This motion data will then be processed and used to adjust the way signals are combined in the antenna array. Then we need to handle these real-time calculations efficiently and adjust the phase of signals received by the antennas to maintain clear communication despite movement. By combining motion tracking and signal processing, we aim to demonstrate how adaptive beamforming can work effectively on moving platforms such as boats, drones, or vehicles.

3.1 Motion Tracking Using the MPU6050 Sensor

Sensor of movement that will be used is MPU6050 (Motion Processing Unit 6050) sensor, which is a small motion sensor. To use this sensor in an embedded system (microcontroller or FPGA), we need a way to transfer data between the sensor and the processing unit. One of the most efficient ways to do this is by using I2C communication. The MPU6050 acts as a slave device, which means that it waits for instructions from a master device. The master device plays a critical role in I2C communication by controlling the clock signal (known as SCL, or Serial Clock Line) that ensures that both devices stay synchronized, see Figure 3.1.

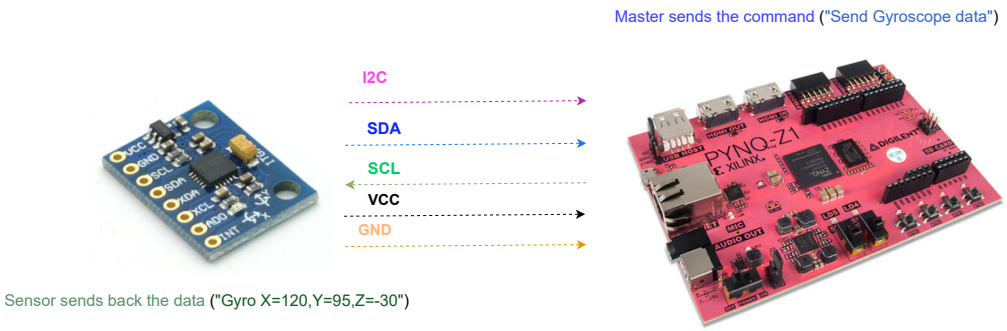


Figure 3.1: I2C communication to IMU sensor MPU-6050

3.2 FPGA Setup and Integration

To make the FPGA perform a specific task, an overlay must be created, which is a file that configures the FPGA to carry out that task. This process typically begins with designing the hardware functionality we want to implement. One efficient way to do this is by using C or C++ in a tool called Vitis HLS (High-Level Synthesis) [35]-[36]. This approach lets we describe your hardware behavior using high-level code instead of traditional hardware description languages. The goal is to turn your C/C++ code into a hardware block (known as an IP core) that can be used inside a larger system design for the FPGA. The process starts by writing algorithm or hardware logic in C or C++, such as a signal processor, data filter, or math function. Once the code is written, it is simulated to check that it works correctly. After simulation, the next step is synthesis, where Vitis HLS converts your C/C++ functions into a hardware description that can be run on an FPGA. Once the raw data is collected from the sensor and stored on the FPGA board, the next step is to process and visualize the data. This includes plotting the sensor outputs, calculating the array factor, and analyzing how it influences the beamforming performance or overall system behavior [35].

3.3 Real-time visualization

Python-based Jupyter Notebooks on the PYNQ-Z1 board are used to provide a user-friendly interface for monitoring and controlling the system. The real-time sensor data from the MPU6050 is visualized within the Jupyter Notebook, allowing users to observe the platform's motion and how the beamforming system is adjusting in response. The real-time data, including acceleration and angular velocity, can be graphed, and users can see the dynamic beamforming adjustments made by the system. This makes it easy to analyze the system's performance and verify that the compensation techniques are working as expected. For visualizing the real time sensor data and dynamic beamforming adjustments, Plotly is most recommended as it allows creation of interactive plots that users can zoom, but simpler static visualization can be performed as well by using the Matplotlib [36].

4

Design

The overall design of the system consists of several key components working in tandem to track the platform's movement and optimize the beamforming process of the antenna array. In Chapter 3, we outlined the necessary components for tracking platform movement an IMU sensor and the FPGA PYNQ-Z1 board using I2C communication. To track the platform's movement and adjust the beamforming in real-time, the sensor data is fed into a complementary filter. The filter is used to combine accelerometer and gyroscope data, providing a more accurate estimate of the platform's orientation. Based on this, the azimuth and elevation angles are calculated, which are critical for adjusting the antenna array's signal phases.This adjustment ensures that the antenna array maintains optimal signal quality even as the platform moves. Next, the array factor of the antenna array is calculated based on the adjusted signal phases. The array factor represents the radiation pattern of the antenna array, and its values are stored for analysis. We also analyze how much memory the array factor requires and plot it to visualize the system's performance. Entire system design is illustrated in Figure 4.1.

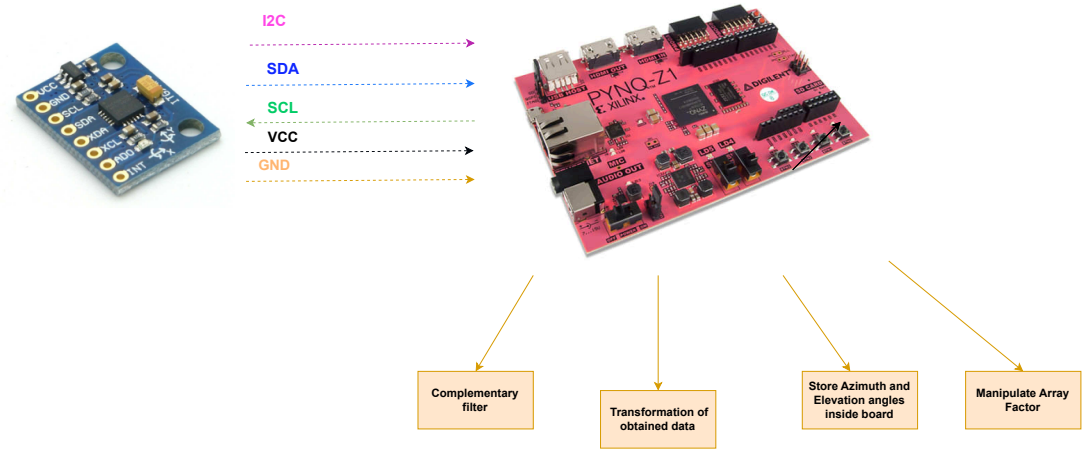


Figure 4.1: System design

To conclude, important components for this design are IMU sensor, complementary filter inside FPGA board and array factor stored inside FPGA memory. In this design, we mathematically compute the array factor and the beam that is formed based on the phase adjustments. However, when connecting a real antenna array to the system, hardware acceleration becomes essential. This is because real-time adjustments to beamforming need quick and efficient processing, which regular software can't handle, especially not in fast-changing environments. That's why in the next section, we'll discuss how hardware acceleration helps solve this problem.

4.1 Hardware acceleration

FPGA hardware acceleration may be used to handle the heavy calculations needed for beamforming, which is important in real-time applications where we need quick responses and low latency. Low latency means the system needs to process data quickly, without any noticeable delay. For example, if the platform (like a drone or robot) moves, the antenna's signal must quickly adjust to stay focused on the target. If the system is too slow, the antenna might point in the wrong direction, losing the signal or reducing performance. Beamforming involves complex tasks like phase correction and Doppler shift compensation, which are computationally heavy. Phase correction adjusts the timing of signals from different antennas to make sure they all combine in the right direction. Since each antenna might be positioned differently, this requires a lot of math to calculate for each one. Doppler shift compensation deals with changes in the signal frequency due to the movement of the platform, and correcting this also needs fast processing. All of this requires processing lots of data in real time, which is why beamforming is computationally demanding.

PYNQ Z1 board is very efficient and helps keep the system responsive. This means the antenna can be adjusted dynamically, ensuring it's always pointed in the right direction with minimal delay. This approach makes system fast, energy efficient and capable of handling large amounts of data, which is perfect for applications where real-time performance is critical, such as adjusting signals based on the movement of the platform. PYNQ-Z1 has some built-in hardware called fixed peripherals, like I2C, SPI, and UART. These are ready to use and don't need to be built in the FPGA. In order to connect a motion sensor and establish communication, one of fixed I2C peripherals must be enabled. To use them, we have to turn them on in Vivado and connect them to the right pins (MIO). If this is not set this up correctly, the processor won't be able to talk to I2C sensor, because the connection isn't made yet.

4.2 Design implementation

To start with design first step is to physically connect sensor and the board. To use the board, we usually connect it to a laptop or desktop computer using a USB cable or through a network cable (Ethernet). This lets us open and run Jupyter Notebooks that are stored inside the board. So before uploading and running our overlay, we make sure that all the devices are connected properly so the system can work as expected. Additionally, other devices can be added to make the system more useful. For example, LEDs can be used for status indication, switches for manual control, or displays to show real-time data or system status. These external devices can be connected to the PYNQ board through PMOD ports or GPIO headers, allowing interaction with the FPGA based design. By integrating such devices, the system can become more interactive and provide more feedback to the user.

Now when the board is connected we need to create a custom overlay for the PYNQ framework. For this project we need two custom overlays. The first overlay handles the I2C communication between the FPGA and an external device, such as an IMU sensor. For this overlay, the Vitis HLS code was pre-written because the I2C communication block is already designed and implemented within Vivado. The second overlay, which is done from scratch in Vitis HLS, calculates the array factor (AF) for an antenna array using an FPGA. It takes in inputs like the precoder values (which control the antenna phase shifts), the distance between antenna elements, the wavelength, and the azimuth/elevation angles for scanning. The overlay then calculates the signal strength for each direction the antenna scans and normalizes the results. By creating an overlay, the process becomes much faster, allowing the system to adjust the beamforming in real-time. This is important for applications like antenna array signal processing, where quick adjustments are needed to maintain a strong signal. After this overlay is implemented, a testbench is created to simulate and verify how the overlay behaves in different situations. The testbench helps ensure that the array factor calculations are working correctly and that the FPGA can handle real-time beamforming tasks efficiently. Block diagram is shown in Figure 4.2.

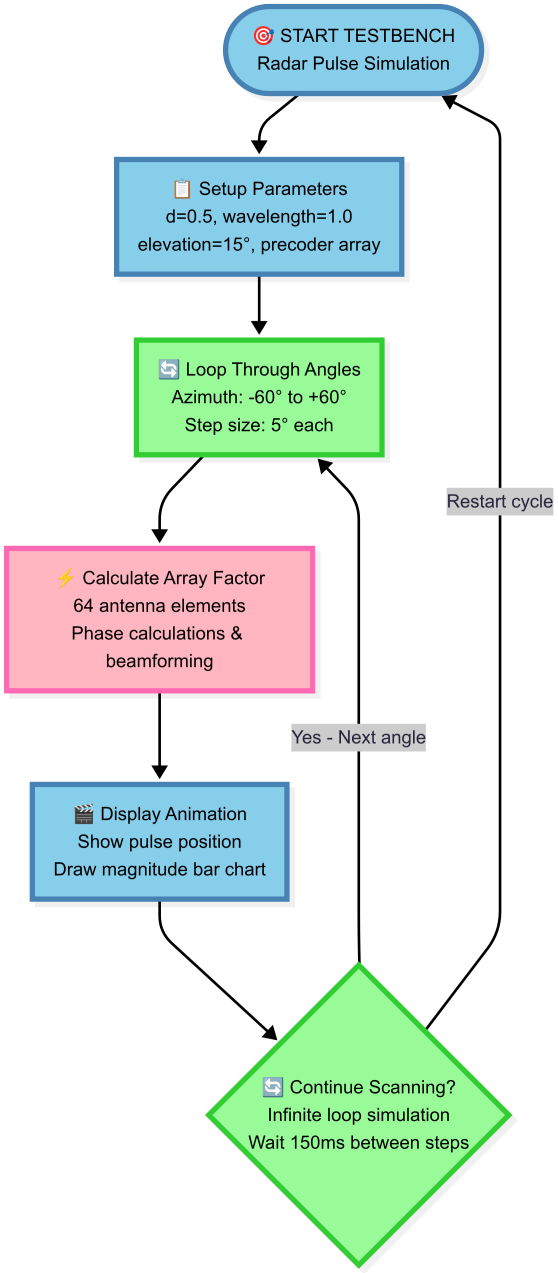


Figure 4.2: Block diagram of antenna array overlay and testbench

After completing the design in Vitis HLS, the next step is to export the RTL (Register Transfer Level) code, which is a hardware description (typically in Verilog or VHDL). This RTL code defines the hardware logic for the custom IP (Intellectual Property) core created from the high-level C/C++ code. Once exported, the RTL is imported into Vivado, where it is integrated into a block design. In Vivado, the custom IP core is connected to other system components, like the Zynq Processing System, through an AXI interface. After the design is validated and all components are correctly connected, a bitstream file (.bit) is generated, which configures the FPGA with the synthesized hardware logic. Additionally, a hardware description file (.hwh) is created, which provides the PYNQ platform with information about

the hardware structure. These files are then transferred to the PYNQ board, where the design can be used for real-time signal processing or beamforming tasks. In the Figure 4.3 the block design diagram in Vivado is shown.

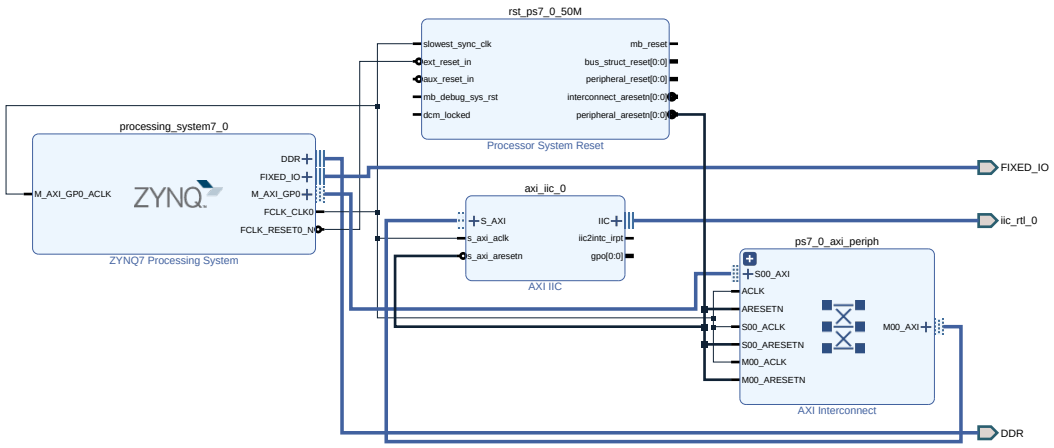


Figure 4.3: Vivado block diagram

To use the created overlay in a Jupyter Notebook on the PYNQ board, we write Python code that loads the bitstream using the Overlay class from the PYNQ library. Once loaded, custom hardware is ready to be used. If block uses AXI interfaces which this does and it's mainly used for I2C communication, we can control it from Python using libraries such as AxiGPIO for simple inputs and outputs, or MMIO (Memory-Mapped Input/Output) for direct memory access.

In summary, creating a custom overlay in PYNQ using C++ involves writing and synthesizing the design in Vitis HLS, integrating it in Vivado, generating the bitstream and configuration files, and using Python code in Jupyter Notebooks to run and test the design on the FPGA. This approach makes it much easier to build and test powerful FPGA based systems, especially for people more familiar with software programming than hardware design.

5

Results

This chapter presents the results obtained from reading raw sensor data and estimating orientation using the MPU6050 and magnetometer sensors. The data was collected via I2C communication at a sampling rate of 100 Hz for 100 seconds. The sensor was moved in a sequence of rotations yaw, followed by roll, and then pitch, to capture changes in orientation. After the rotations, the sensor was also moved around to record its translation through space. During this time, accelerometer, gyroscope, and magnetometer measurements were continuously recorded. The accelerometer data provided linear acceleration along the three axes (x, y, z), while the gyroscope measured angular velocity, and the magnetometer captured the magnetic field values. The raw sensor readings corresponding to the first ten data samples were recorded and presented in the following table for further analysis in the Table 5.1.

Index	**ax**	**ay**	**az**	**gx**	**gy**	**gz**	**mx**	**my**	**mz**
1	0.5204	0.0029	1.9999	-357	-651	1455	-29147	-28818	90
2	0.5198	0.0031	1.9999	-930	71	-284	-29147	-28818	90
3	0.5204	0.0034	1.9999	-936	79	-287	-29147	-28818	90
4	0.5192	0.0028	1.9999	-940	76	-288	-29147	-28818	90
5	0.5187	0.0026	1.9999	-935	78	-296	-29147	-28818	90
6	0.5194	0.0018	1.9999	-944	86	-301	-29147	-28818	90
7	0.5181	0.0031	1.9999	-934	82	-292	-29147	-28818	90
8	0.5183	0.0039	1.9999	-942	80	-289	-29147	-28818	90
9	0.5201	0.0039	1.9999	-939	76	-292	-29147	-28818	90
10	0.5189	0.0055	1.9999	-932	72	-290	-29147	-28818	90

Table 5.1: Sensor Data Sample without Angles

The table shows original measurements, which are helping us understand where the data started and how processing of these data changes over time and how that affects the final results. The acceleration values indicate that the device is mostly steady, with gravity acting mainly on the z-axis. The gyroscope readings show small rotational movements, while the magnetometer data remain fairly constant, suggesting a stable magnetic environment during the measurements.

If these measurements are repeated over longer period of time, somewhat different results will be obtained if different movements are performed. The variability in the repeated measurements provides a foundation for further processing, such as

applying filters or sensor fusion algorithms, to improve accuracy and robustness in orientation tracking. First, the magnetometer data needs to be calibrated to reduce noise and bias. Uncalibrated data introduces errors in sensor fusion algorithms, causing incorrect orientation or heading estimation when combining accelerometer, gyroscope, and magnetometer data.

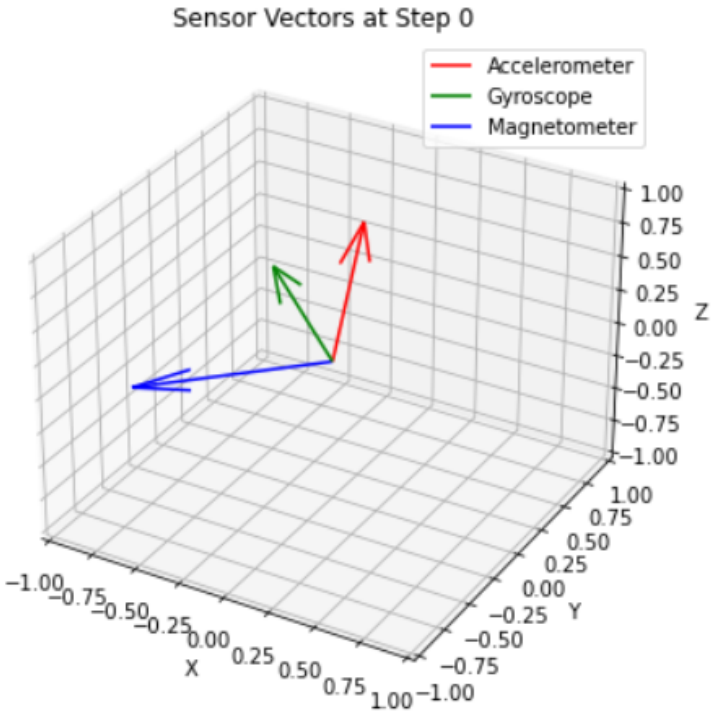


Figure 5.1: Rotation of IMU sensor

In the Figure 5.1 these values are plotted in the real time and they are changing according to the movement of the sensor. When we watch the square rotate, we are actually seeing a live, real-time animation of the sensor's orientation in space. This helps understand how the sensor (or the device it's attached to) is positioned and moving without needing to interpret raw numbers. The orientation angles (roll, pitch, yaw) give a complete description of the sensor's 3D orientation over time.

The presented image is only measurement of raw sensor data. In order to track the real sensor data, we also need to calculate linear velocity and position by integrating acceleration over time. This is important to understand how the device moves through space, allowing us to track its exact location and motion dynamics. To calculate the sensor's movement accurately, we first compute linear velocity by integrating the measured linear acceleration over time, using the formula

$$v_k = v_{k-1} + a_k \Delta t, \tag{5.1}$$

$$v_1 = v_0 + a_1 \times \Delta t = 0 + 0.5204 \times 0.01 = 0.005204 \text{ m/s} \tag{5.2}$$

From equation 5.2, for the time step of 0.01 we can conclude that the initial velocity v_1 is small, indicating that the movement at the beginning will be minimal. This suggests that the sensor will take its current position as initial. As the time interval increases and as the sensor moves further from the initial position, both velocity and position increase as well. Negative velocity and position values can be expected when the acceleration a_1 or velocity v_1 components are negative in the formula, reflecting movement in the opposite direction.

Then, we find the linear position by integrating the velocity.

$$p_k = p_{k-1} + v_k \Delta t. \tag{5.3}$$

$$p_1 = p_0 + v_1 \times \Delta t = 0 + 0.005204 \times 0.01 = 5.204 \times 10^{-5} \text{ m} \tag{5.4}$$

From equation 5.4 we observe that the displacement p_1 at time step 0.01 is very small, which confirms that the sensor has only moved a tiny distance from its initial position. This small change is consistent with the low velocity value calculated before .

These calculations convert raw acceleration data into velocity and position estimates, allowing us to track the sensor's trajectory and determine key orientation parameters. By incorporating linear velocity and position data, the system can dynamically adjust the orientation of the antenna, compensating for movement, and ensuring more precise beam steering and better overall performance. In applications like antenna arrays, accurately knowing the position and movement of the sensor is crucial to calculate the azimuth and elevation angles. These angles describe the direction of the antenna relative to a reference frame, which directly affects the array's ability to focus signals and improve reception or transmission. From Figure 5.2 we can see the full trajectory of our sensor while moving through space. At the beginning when we start moving the device, position in which device is currently in will be evaluated as initial position (0,0) which we can see from the figure. Later as we move from the starting position, we can see different direction of trajectory and the position and orientation of the sensor are changing as well, which was already predicted by observing the formulas for position and velocity. It is important to track the sensor's movement because the beam needs to follow the same direction as the sensor.

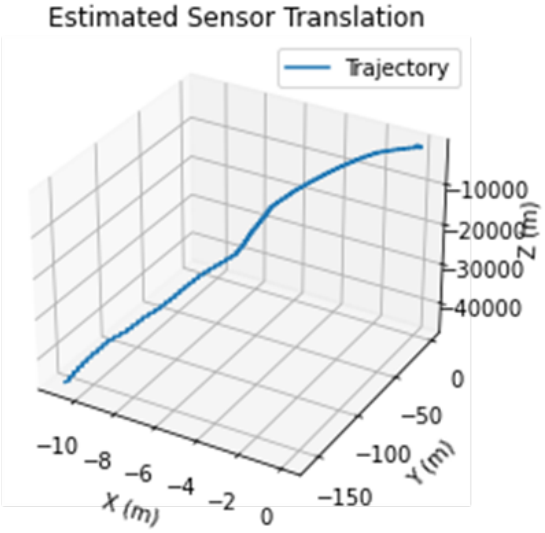


Figure 5.2: Tracking movement of the IMU sensor

Now when we have successfully tracked the sensor's movement, including its orientation and trajectory, we use a complementary filter to improve the accuracy of this tracking. The complementary filter combines data from the accelerometer and gyroscope: the accelerometer gives reliable long-term orientation, while the gyroscope provides smoother short-term changes. By blending them, the filter reduces noise and drift, giving a more stable and accurate estimate of the sensor's orientation over time. Filter has its own parameter α that affects how we estimate the sensor's orientation, which is important for calculating its movement path, or trajectory. To understand better the effect of the complementary filter, we analyzed how changing the α factor influences the estimated sensor trajectory which is illustrated in the Figure 5.3. In the previous plot, we used a fixed α value to track the movement. Now, by adjusting α, we can observe how the trajectory changes: when alpha is set closer to 1, the trajectory becomes smoother and more responsive due to the dominance of the gyroscope data, but over time, it may drift away from the actual path. On the other hand, when alpha is reduced, the trajectory appears more stable and less affected by drift, as it relies more on accelerometer data, but this also introduces more noise and jitter. These differences clearly show how tuning alpha impacts the balance between smoothness and accuracy in trajectory estimation.

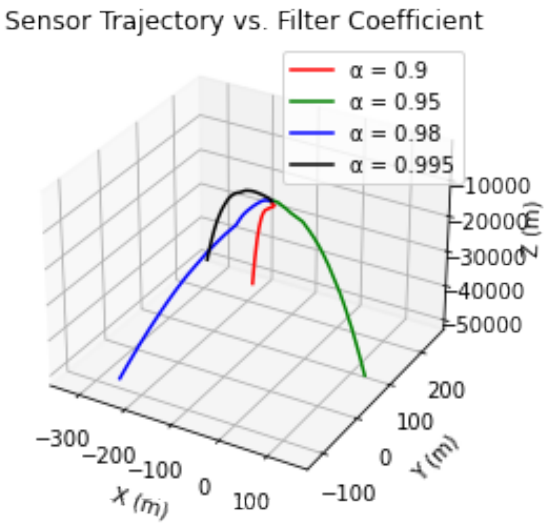


Figure 5.3: Influence of the complementary filter on the sensor trajectory

Now that the data has been processed using the complementary filter, we can estimate the sensor's orientation in terms of roll, pitch, and yaw angles. These are the three main angles used to describe the orientation of an object in 3D space. The roll and pitch angles are calculated by combining accelerometer and gyroscope data using the complementary filter, which helps reduce noise and improve stability. The yaw angle, however, is more challenging to estimate accurately, as it relies mainly on gyroscope data due to the accelerometer's limitations in detecting horizontal rotation.

In the plot shown in Figure 5.4, the roll, pitch, and yaw angles are displayed over time. From this figure, we can observe how the orientation of the sensor changes during the movement. For example, steady changes in yaw indicate rotation around the vertical axis, while fluctuations in pitch and roll reveal tilting or swinging motions. The estimated roll, pitch, and yaw angles show a realistic and expected range of motion. The roll and pitch angles vary approximately between $-75°$ and $+75°$, indicating that the sensor experienced moderate tilting and rotation around the side-to-side and front-back axes. This is typical in scenarios where the device undergoes dynamic motion but remains within physically reasonable orientation limits. Although the yaw angle appears to span from $0°$ to $360°$, this range is the result of small oscillations around the $0°/360°$ boundary rather than a true full rotation. This behavior is common in orientation data when minor movements occur near the wrap-around point. The complementary filter still performs effectively, providing stable and accurate estimates without significant drift or noise.

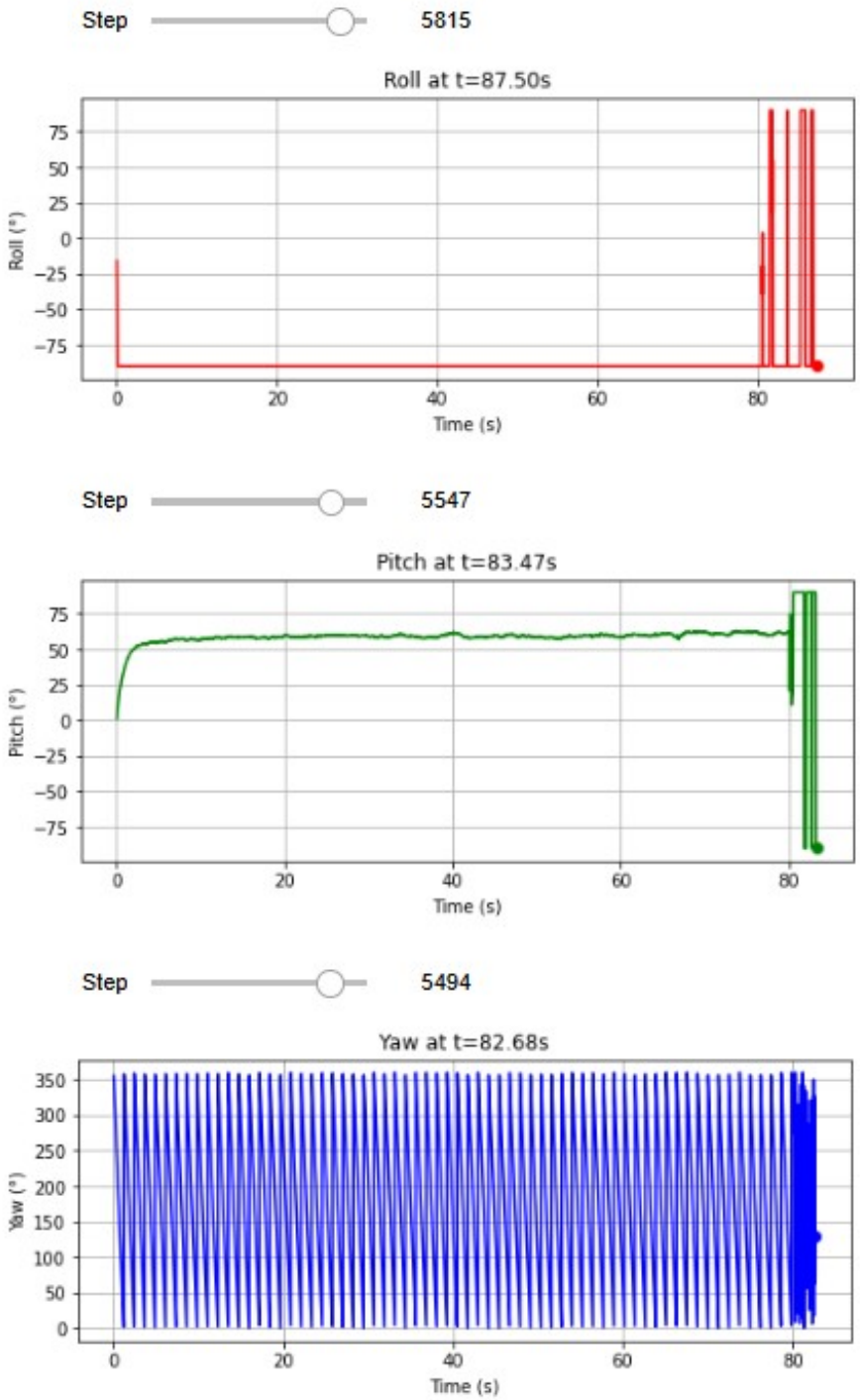


Figure 5.4: Influence of the complementary filter on the sensor trajectory

To convert these angles into more intuitive directional angles azimuth and elevation we use the rotation matrix corresponding to these Euler angles. Azimuth represents the angle between the projection of the sensor's forward direction onto the horizontal plane and a reference direction (usually true north), and the elevation denotes the angle between the sensor's forward direction and the horizontal plane. First we have to construct the rotation matrix R by applying yaw, pitch, and roll rotations sequentially around the Z, Y, and X axes, respectively. Assuming the sensor's forward direction aligns with the sensor-frame X-axis, the rotated direction vector $\mathbf{v}$ is obtained by multiplying the rotation matrix R by the unit vector along the X-axis. The components of $\mathbf{v} = (x, y, z)$ are then used to calculate the azimuth and elevation angles. Specifically, the azimuth is computed as $\arctan 2(y, x)$, representing the horizontal angle, while the elevation is obtained as $\arcsin(z)$, indicating the vertical angle. This method provides a clear geometric interpretation of the sensor's orientation.

By accurately calculating these angles, we can simulate the sensor pointing direction over time, which is essential for applications such as beam steering, target tracking, or direction-of-arrival estimation in array signal processing. In Figure 5.5, the 3D beam pattern is shown for an azimuth angle of 0° and an elevation angle of 0°. In this case the IMU sensor is not moving at all and instantly the azimuth and elevation angles are 0°, respectively. This is the first test case that demonstrates our algorithm works correctly when the sensor is stationary, the beam remains steady and does not move.

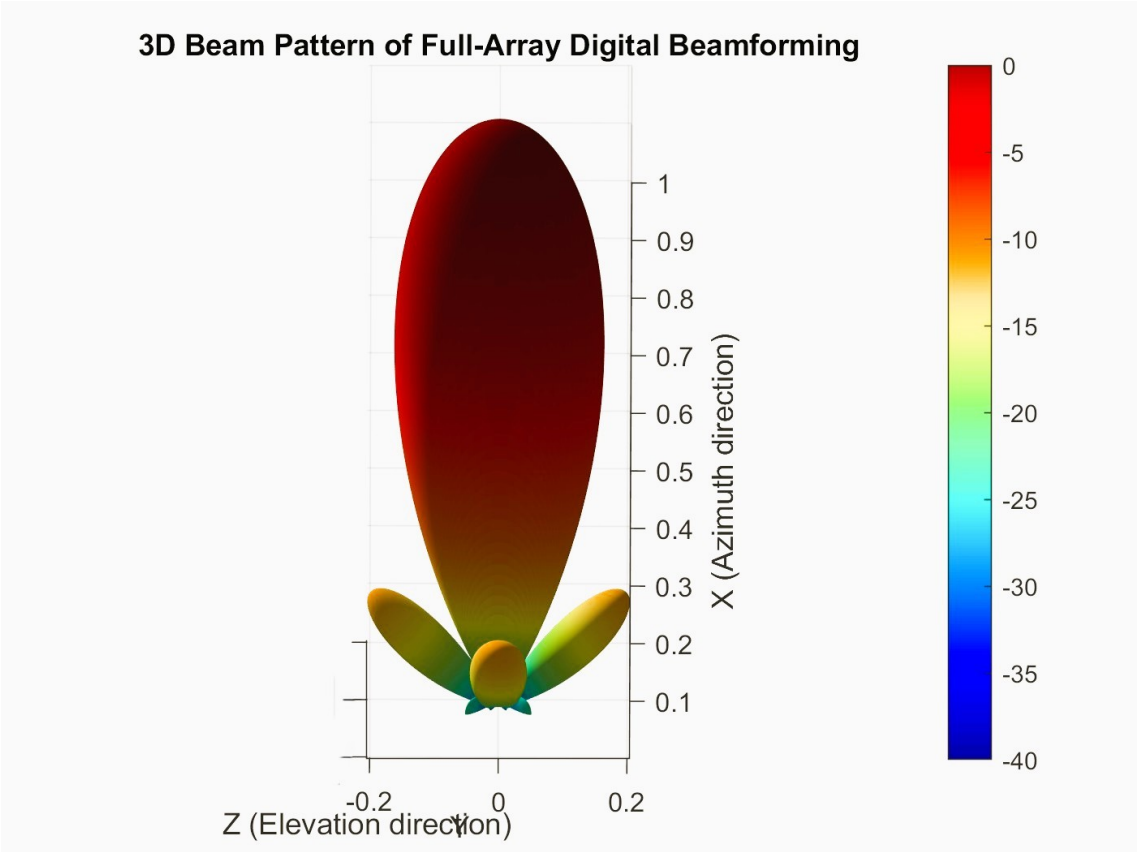


Figure 5.5: Azimuth and elevation angels at position 0°

Based on the azimuth and elevation angles of 0°, we can calculate the array factor pointing directly towards the center of the coordinate system. This condition is visually represented in Figure 5.6, where the 3D beam pattern shows a strong, centralized lobe. The central point in the plot marks the direction corresponding to these angles, and since they are both zero, it remains fixed in the center of the plot. The surrounding smaller dots around the sphere indicate the radiation pattern of the antenna in other directions. These dots show how much energy is radiated at different angles relative to the main beam. This visualization confirms that the antenna is radiating or receiving signals in the forward direction, without any tilt or rotation, and serves as a reference point for analyzing how changes in orientation would affect the beam steering and spatial response.

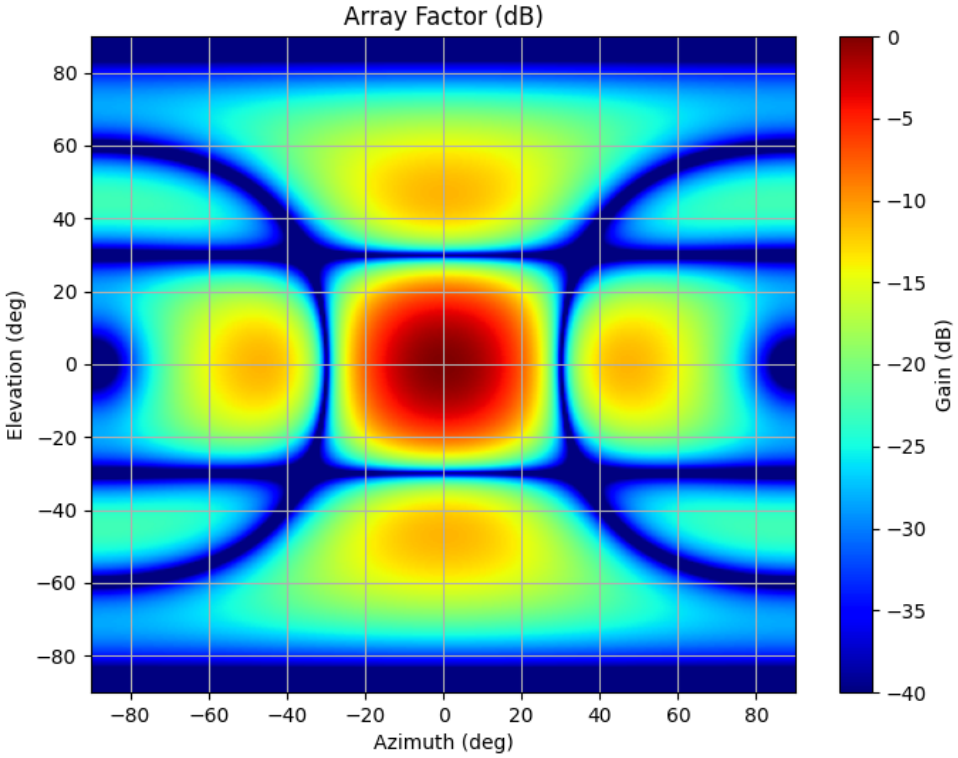


Figure 5.6: Array factor at position 0°

When the sensor is moved or rotated in space, its orientation changes relative to the original reference frame. This movement directly impacts the azimuth and elevation angles, which define the direction in which the beam is pointing. Initially, when the sensor was stationary, the azimuth and elevation angles were both at 0°, pointing straight ahead. However, after the sensor undergoes translation or rotation, these angles update to reflect the new direction of the sensor. As a result, the beam is no longer pointing in the original forward direction; instead, it is now oriented according to the updated azimuth and elevation angles. This shift means that the beam is directed toward a different region in space, aligning with the sensor's new pose. By analyzing the time series of these angles, we identify points where the azimuth and elevation approach specific values, such as −35° azimuth and 40° elevation. The antenna array thus dynamically steers its radiation pattern based on the sensor's movement, allowing it to track the new pointing direction. The azimuth and elevation angels are shown on the figure 5.7 are obtained as the median values calculated from all other measured angles, providing representative target values like −35° azimuth and 40° elevation. From the figure we can also see the beam pattern for 9 DoF (nine degrees of freedom),which reflects how the device's orientation captured through yaw, pitch, and roll affects the direction and shape of the beam in 3D space.

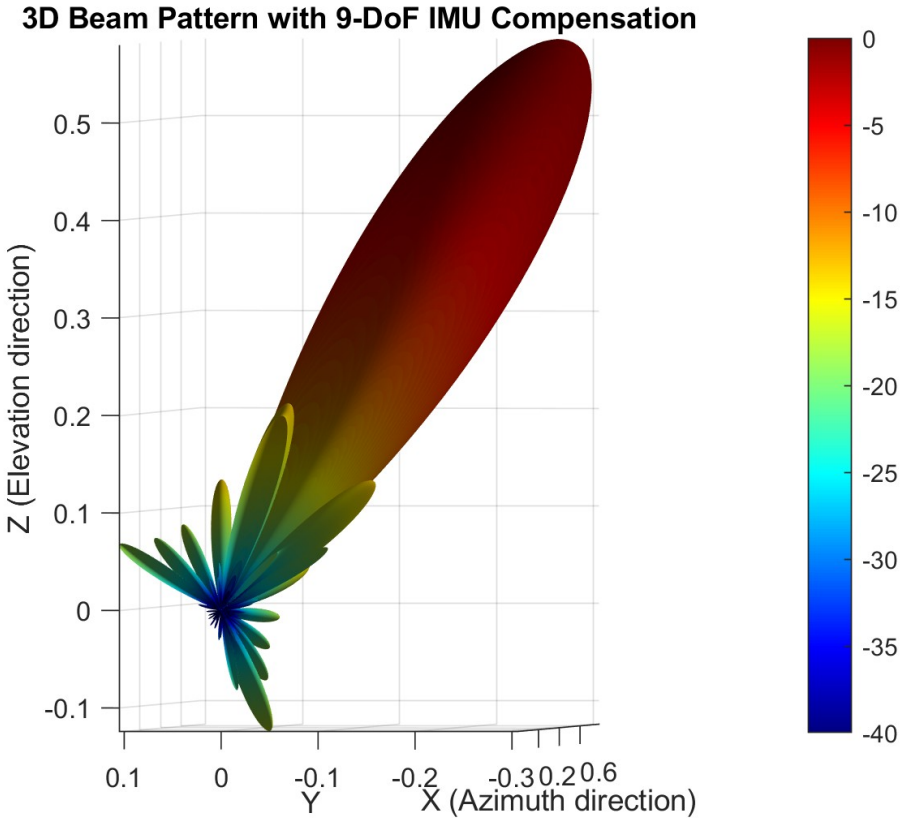


Figure 5.7: Azimuth and elevation when sensor is moved

With the relocation of the sensor to a new position and direction, the array factor of the antenna array is manipulated to maintain sync with the pose of the sensor. With the sensor's movement or rotation changed, relative phase differences among unique antenna elements vary, thereby steers the main lobe of the array factor the location of maximum reception or radiation away from the original center. This dynamic steering action causes the beam no longer to point straight ahead but rather towards the sensor's new location in space. This is visually indicated by the peak of the beam pattern shifting from center to some other point in space. This movement of the main lobe occurs because the array continuously adjusts its direction, placing its energy at the new position and orientation of the sensor. This means the beam pattern now becomes asymmetrical relative to the original frame, with the main lobe clearly pointing in the direction of the current orientation of the sensor, and side lobes and nulls adapt as well. This becomes crucial for applications that require precise beam steering and tracking, since it enables the antenna array to dynamically track the sensor's motion and maintain optimal signal quality. From the Figure 5.8 it is evident that our beam moved in the calculated direction of $-20°$ azimuth and $40°$ elevation.

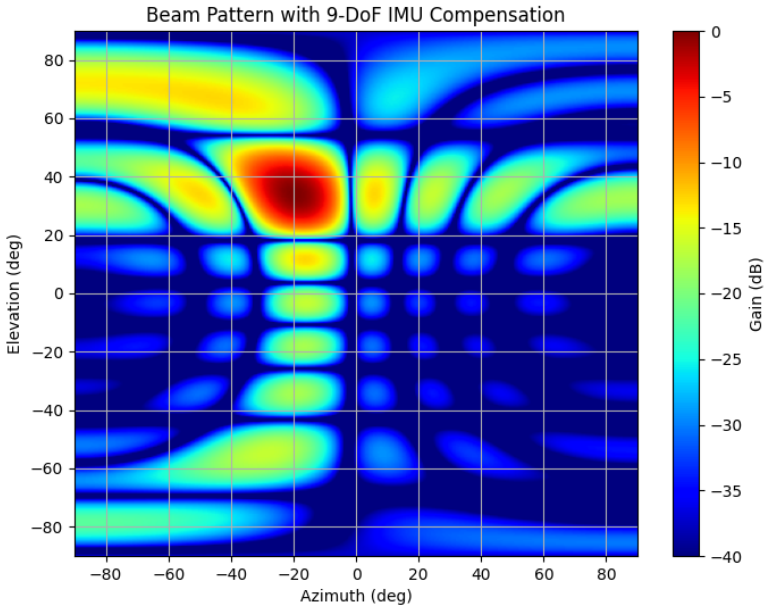


Figure 5.8: Array factor when sensor is moved

Since the array factor was very large when run in a Jupyter Notebook, it required a lot of memory space, which made the entire simulation very slow. To speed up the simulation, the new Array Factor overlay was rewritten in C and C++ and then uploaded to the PYNQ board. As shown in Table 5.2, the optimized implementation of the Array Factor overlay demonstrates significant improvements in execution time and resource efficiency. The design completes each operation in approximately 8.25 ns, well within the 10 ns target, and achieves a higher-than-expected clock speed of 120 MHz. The processing time per data item is reduced to approximately 6 microseconds, compared to over 20 microseconds in the earlier Python-based Jupiter Notebook implementation. Additionally, the design efficiently utilizes FPGA resources such as BRAM and DSP blocks, while avoiding reliance on URAM. These results confirm that the overlay is well-suited for real-time beamforming tasks.

Parameter	**Value**
Clock Speed (Target / Actual)	100 MHz / 120 MHz ± 20 MHz
Operation Time (Target / Actual)	10 ns / 8.25 ns ± 1.7 ns
Clock Cycles per Operation	593
Processing Time per Data Item	6 μs
BRAM (Block RAM) Used	64
DSP Units Used	549
Flip-Flops (FFs)	71,444
Lookup Tables (LUTs)	70,796
URAM (Ultra RAM)	Not Used

Table 5.2: Performance and Resource Utilization of Array Factor Overlay on PYNQ-Z1

Overall, the design is fairly complex but balanced, using enough resources to work efficiently. For example, if the clock speed is 100 MHz, it would take about 6 microseconds to process one data item. Array Factor is only one array stored inside PYNQ board but moving it directly to the PYNQ insead of the Jupyter Notebook was a good decision. Compared to the earlier implementation, the new C/C++ overlay achieves significantly improved performance. The Jupiter Notebook version suffered from memory overhead and slow execution time, often taking over 20 μs per data operation. In contrast, the current design processes each data item in approximately 6 μs, thanks to optimized memory access and dedicated hardware acceleration. Furthermore, the new overlay uses FPGA resources efficiently while delivering high responsiveness, which is essential for real-time beam tracking systems.

Based on the analysis of the sensor's movement and the corresponding array factor behavior, it is evident that the antenna array's beam direction dynamically responds to changes in the sensor's position and orientation. Initially, when the sensor is stationary, the beam points straight ahead, aligned with the original azimuth and elevation angles set at 0°. However, as the sensor translates or rotates, the relative phase shifts between antenna elements cause the main lobe of the array factor to steer away from the center. This steering effect leads to the beam pointing toward the sensor's updated location in space, as illustrated by the displacement of the main lobe in the radiation pattern. The beam pattern's peak, represented by the central dot in the visualizations, moves correspondingly with the sensor, demonstrating how the antenna array continuously adapts its focus to track the sensor's movement. These results confirm that the array factor effectively captures the sensor's dynamics, enabling precise beam steering that aligns with the sensor's changing pose. This capability is critical for applications requiring real-time directional tracking and adaptive beamforming.

6

Discussion

6.1 FPGA Resource Utilization

During the course of this project, the PYNQ-Z1 board was selected as the main hardware platform due to the fact that it has a Zynq SoC with an ARM processor and a programmable FPGA fabric. This powerful architecture is suitable for projects requiring parallel processing, custom hardware accelerators, and high-speed interfacing. But in this specific implementation, the FPGA resources were not utilized in depth. The reason behind this is the nature of the sensor. It is a low-power, compact sensor that provides accelerometer and gyroscope data on a shared I2C interface. The data acquisition process therefore did not include computationally demanding processes or high-bandwidth communication requiring custom logic design or parallel data processing in the FPGA fabric.

Most of the functionality, including I2C communication and basic signal processing, could be efficiently managed by the ARM processor within the processing system of the PYNQ board. Therefore, programmable logic (PL) was not used extensively, which meant that the hardware acceleration and reconfigurable logic feature of the board to its full potential was not explored in depth.

In upcoming studies, even more sophisticated sensors or real-time data processing capabilities can be incorporated in order to leverage the FPGA resources better. For instance, sensor fusion algorithms, high-speed filtering, or DMA channels for faster data transfer could be added to fully exploit the programmable nature of the PYNQ board and demonstrate its superiority over mere software-based methods.

6.2 Matlab or Jupiter

In this thesis, most of the data analysis and visualization were done using Jupyter Notebook. This platform is user-friendly and allows combining code, results, and explanations in one place, making it easy to explore data step-by-step. It is especially useful for beginners and students because it helps to understand the data quickly and document the work clearly.

However, Jupyter Notebook is not the only way to work with sensor data. For more advanced analysis, simulation, and visualization, MATLAB is widely used in both research and industry. MATLAB offers powerful tools for signal processing, beamforming, and antenna array analysis, making it easier to simulate how beams

move when sensors change position or orientation. It also has good support for connecting with hardware platforms like Arduino, which means we can combine real-time sensor data with complex simulations in one environment. This makes MATLAB very useful for testing and improving algorithms related to sensor fusion and beam steering. Arduino can handle basic data processing like filtering and calibration before sending the data to a computer for further analysis.

The focus of this research was on understanding how sensor arrays behave when the sensor moves or rotates. Using Jupyter Notebook helped in initial data exploration and visualization, but combining this with MATLAB for live sensor tracking and Arduino for detailed modeling would provide a more complete picture. This combination would not only improve the accuracy of the results but also make it easier for developers to test and implement practical solutions.

In summary, the choice of platform depends on the research goals and the user's needs. For students and developers who want an easy and flexible way to analyze sensor data, Jupyter Notebook is a great start. For deeper analysis and simulation, MATLAB provides advanced tools and integration options. Using these platforms together can create a powerful workflow that covers data collection, processing, and visualization, making it easier to understand sensor behavior and antenna array performance. Future work could focus on creating smoother integration between these tools to offer a better user experience and more efficient research process.

6.3 Planned vs. Actual Timeline

The Gantt chart in the Appendix outlines the planned schedule for the master thesis project in 2025. The initial stages included a thorough topic introduction and literature review, followed by Matlab code implementation and debugging. The project also involved investigating hardware components, exploring Linux and Vivado user interfaces, and implementing VHDL code for the sensor system. According to the plan, major milestones such as the planning report, halftime report, and final presentation were scheduled to track progress throughout the project timeline. However, during the project execution, several unforeseen challenges arose, particularly related to setting up the PYNQ board and enabling I2C communication with the sensor hardware. These technical difficulties significantly slowed down progress, resulting in less time available for implementing all the necessary calculations and equations. Consequently, some parts of the Matlab and VHDL implementation had to be scaled back or postponed, highlighting the importance of hardware integration issues in embedded system projects. Despite these setbacks, the project achieved key milestones and provided valuable insights into the complexities of sensor data acquisition and real-time processing.

7

Conclusion

This thesis explored the simulation and deployment of beamforming techniques on a mobile platform through the integration of antenna array signal processing and sensor data. The basic intent was to understand how the orientation and position of a sensor, when mounted on a dynamic platform, influence the beam direction and array factor of an antenna system.

Throughout the course of the project, we demonstrated how inertial measurements provided by an IMU sensor (MPU6050) can be used to track movement and rotation of a platform in real-time. By processing acceleration data to compute position and using gyroscopic data to compute orientation, it was possible to dynamically update the azimuth and elevation angles of the beam. These angular changes were then referenced to recompute the array factor, effectively steering the beam pattern to follow the movement of the platform. It was discovered that even slight shifts in the orientation of sensors caused significant changes in the beam direction. The central point of 0°, where the main lobe of the radiation pattern was originally situated, shifted as the platform rotated and translated, showing the new direction of transmission or arrival. This demonstrates the feasibility of using beamforming for mobile or wearable applications where directionality and precise targeting of signals are required.

Although the simulation results and theoretical context were successfully utilized, a few practical problems were encountered. Hardware setup issues, such as the configuration of I2C communication with the PYNQ board, limited the time to fully implement real-time equations and calculations in hardware. Further, the FPGA resources on the board were largely underused since the IMU sensor was lightweight and low-complexity. Yet, this project lays a solid foundation for future work on real-time, adaptive beamforming on mobile systems. Incorporating additional advanced sensor fusion techniques, using other hardware resources in the FPGA for accelerating the process, and extending to multi-sensor or multi-array systems could further enhance system responsiveness and accuracy.

In summary, beamforming on a moving platform is not merely feasible but also highly relevant to current applications in wireless communication, robotics, and the military. With further development, the system can evolve into a robust real-time adaptive array system that can achieve intelligent direction tracking and spatial filtering in complex environments.

8

Future work

In the future, several areas can be improved to make the system work faster and more efficiently. The focus will be on solving problems with communication, improving how the FPGA and processing system work together, and speeding up some processes with better hardware.

8.1 Communication Problems

Right now, the PYNQ-Z1 board sends data over WiFi/Ethernet to the laptop. This can be slow, and sometimes there is a delay because the system is using JSON (a format for sending data) and the HTTP protocol. These methods add extra steps to the communication, slowing things down. Some of the possible solutions to this problem are:

- **Compress Data:** Before sending data, we can compress it to make it smaller. Smaller data takes less time to send, which will help improve the speed.
- **Use Faster Data Formats:** Instead of using JSON, we could switch to faster formats like `msgpack` or `protobuf`. These formats are smaller and quicker to process, meaning less delay in communication.
- **Send Data in Batches:** Right now, the system sends each reading separately, which can be slow. Instead, we can send multiple readings at once (batching). This will reduce the number of times the system has to send data, speeding things up.
- **Use TCP/UDP for Faster Transmission:** Instead of using HTTP, which has some overhead, we could use `TCP` or `UDP` for faster communication. These protocols are quicker and more direct, especially for high-speed data transfer.

8.2 Improving the FPGA and Processing System Connection

Another problem is how the FPGA and processing system (PS) communicate with each other. Currently, it uses the I2C interface, which involves multiple steps and can be slow. Each time the system reads data from a sensor, it has to go through multiple processes, causing delays. Some of the possible solutions to this problem are:

- **Create a Custom IP Block:** One way to make things faster is to build a special IP block that directly handles the I2C communication. This would allow the FPGA to control the data transfer without involving the processing system, speeding up the process.
- **Use DMA (Direct Memory Access):** DMA can move data directly between memory and devices without using the processing system. This can reduce delays and make the data transfer faster.
- **Use Interrupts Instead of Polling:** Right now, the system probably checks for data continuously (polling), which uses extra CPU power. Using interrupts would allow the system to only react when new data is available, saving time and resources.

Overall, future improvements will focus on helping the system perform better and be more responsive, especially in real-time applications.

Bibliography

[1] X. Xie, B. Rong, and M. Kadoch, *6G Wireless Communications and Mobile Networking*. Wiley, 2025, see Chapter 3 for detailed discussion on 6G advancements, pages 45-67. [Online]. Available: https://benthambooks.com/book/9781681087962/

[2] L. Simić, N. Perpinias, and M. Petrova, "60 ghz outdoor urban measurement study of the feasibility of multi-gbps mm-wave cellular networks," *arXiv*, 2016. [Online]. Available: https://arxiv.org/abs/1603.02584

[3] Tescaglobal. (2023) Inertial measurement unit (IMU) | an introduction. Accessed: January 2025. [Online]. Available: https://www.tescaglobal.com/blog/inertial-measurement-unit-imu-an-introduction/

[4] J. Li, Y. Sun, L. Xiao, S. Zhou, and A. Sabharwal, "How to mobilize mmwave: A joint beam and channel tracking approach," *arXiv*, 2018. [Online]. Available: https://arxiv.org/abs/1802.02125

[5] R. M. Dreifuerst and R. W. H. Jr., "Massive mimo in 5g: How beamforming, codebooks, and feedback enable larger arrays," *arXiv*, 2023. [Online]. Available: https://arxiv.org/pdf/2301.13390

[6] K. M. O'Hara, "mmwave beamforming in dynamic, urban environments," 2021. [Online]. Available: https://resources.remcom.com/hubfs/Squarespace%20Resources/Remcom_MWJournal_mmWave_Beamforming_in_Dynamic_Urban_Environments_Nov202.pdf

[7] Z. Du, F. Liu, W. Yuan, C. Masouros, Z. Zhang, and G. Caire, "Integrated sensing and communications for v2i networks: Dynamic predictive beamforming for extended vehicle targets," *arXiv preprint arXiv:2111.10152*, 2021.

[8] Y. Lyazidi, J. Equi, R. Shreevastav, I. Siomina, and G. Fodor, "Isac architecture and sensing topology switching in 6g cellular networks," *Ericsson Research*, 2025, authors: Yazid Lyazidi (Ericsson Research, UK), Julia Equi (Ericsson Research, Finland), Ritesh Shreevastav (Ericsson Research, Sweden), Iana Siomina (Ericsson Research, Sweden), Gábor Fodor (KTH Royal Institute of Technology, Sweden). Email contacts provided in the original paper. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/white-papers?

[9] M. F. Keskin, M. M. Mojahedian, J. O. Lacruz, C. Marcus, O. Eriksson, A. Giorgetti, J. Widmer, and H. Wymeersch, "Fundamental trade-offs in monostatic isac: A holistic investigation towards 6g," *arXiv e-prints*, 2024, accessed: 2025-04-25. [Online]. Available: https://arxiv.org/abs/2401.18011

[10] C. Luo, A. Tang, F. Gao, J. Liu, and X. Wang, "Channel modeling framework for bistatic isac under 3gpp standard," 2024, accessed: 2025-04-25. [Online]. Available: https://ieeevtc.org/vtc2024spring/DATA/PID2024001865.pdf

[11] Z. Han, L. Han, X. Zhang, Y. Wang, L. Ma, M. Lou, J. Jin, and G. Liu, "Multistatic integrated sensing and communication system in cellular networks," *arXiv e-prints*, 2023, accessed: 2025-04-25. [Online]. Available: https://arxiv.org/abs/2305.12994

[12] X. Guo, Q. Shi, S. Zhang, C. Xing, and L. Liu, "User equipment assisted localization for 6g integrated sensing and communication," *arXiv e-prints*, 2023, accessed: 2025-04-25. [Online]. Available: https://arxiv.org/abs/2312.13013

[13] Z. Han, L. Han, X. Zhang, Y. Wang, L. Ma, M. Lou, J. Jin, and G. Liu, "Multistatic integrated sensing and communication system based on macro–micro cooperation," *Sensors*, vol. 24, no. 8, p. 2498, 2024, accessed: 2025-04-25. [Online]. Available: https://www.mdpi.com/1424-8220/24/8/2498

[14] N. González-Prelcic, D. Tagliaferri, M. F. Keskin, H. Wymeersch, and L. Song, "Six integration avenues for isac in 6g and beyond," *IEEE Vehicular Technology Magazine*, 2025, accessed: 2025-04-25. [Online]. Available: https://vtmagazine.ieee.org/2025/03/07/six-integration-avenues-for-isac-in-6g-and-beyond/

[15] A. S. K. Gupta and A. Sharma, "Compensating doppler frequency shift of high-speed rail communications," *International Journal of Wireless and Mobile Networks*, vol. 13, no. 17, pp. 52–60, 2021. [Online]. Available: https://www.ripublication.com/ijaer18/ijaerv13n17_52.pdf?

[16] P. K. Das and R. K. Gupta, "An overview of doppler effect deviation in high mobility communication systems," *AIP Conference Proceedings*, vol. 3232, no. 1, p. 020002, 2021. [Online]. Available: https://pubs.aip.org/aip/acp/article/3232/1/020002/3316583/An-overview-of-Doppler-effect-deviation-in-high

[17] N. K. Prabhu, F. D. P. Smolka, and N. P. P. Pinto, "Impact of mobility on integrated sensing and communication systems," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2495–2507, 2021. [Online]. Available: https://doi.org/10.1109/TWC.2021.3072575

[18] H. Wymeersch, *Wireless Communication Systems: Design and Analysis*. Gothenburg, Sweden: Chalmers University of Technology, 2020, sSY135 Lecture Notes.

[19] W.-M. Lo and J. H. Lee, "Cyclic prefix and its impact on ofdm performance in 5g systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 576–588, 2018. [Online]. Available: https://doi.org/10.1109/JSAC.2018.2787034

[20] M. Ivanskiha, *Phased Array Antennas*. Gothenburg, Sweden: Chalmers University of Technology, 2025, specific pages: pp. 45-88.

[21] Avnet-Silica. (2021) Beamforming: Fundamentals to implementation. Accessed: January 2025. [Online]. Available: https://www.avnet.com/wps/portal/avnet/solutions/5g

[22] Q. Chaudhari, "What is the difference between analog, digital and hybrid beamforming?" 2025, accessed: 2025-04-25. [Online]. Available: https://wirelesspi.com/what-is-the-difference-between-analog-digital-and-hybrid-beamforming/

[23] A. Theory, "Antenna theory - arrays," 2025, accessed: 2025-04-25. [Online]. Available: https://www.antenna-theory.com/arrays/main.php

[24] J. Grant, "Modelling an antenna array," *Medium*, n.d., accessed: January 2025. [Online]. Available: https://medium.com/@john_grant/antenna-arrays-and-python-plotting-with-pyplot-22c0c4adad67

[25] J. Wang, R. Dizaji, and A. M. Ponsford, "An analysis of phase array radar system on a moving platform," *Raytheon Canada Technical Articles*, n.d., accessed: January 2025.

[26] Wikipedia contributors, "Homogeneous coordinates," https://en.wikipedia.org/wiki/Homogeneous_coordinates, 2025, accessed: 2025-04-25.

[27] Computer Science Department, Columbia University, "Homogeneous transformation matrices," https://www.cs.columbia.edu/~allen/F17/NOTES/geometry.pdf, 2023, accessed June 2025.

[28] MathWorks, Inc., "Convert Rotation Matrix to Homogeneous Transformation," https://www.mathworks.com/help/robotics/ref/rotm2tform.html, accessed June 2025.

[29] D. Patel and D. B. Patel, "Low cost and robust solar tracking system based on data of daily and seasonal variation in sun position with respect to specific location on earth," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 3, no. 9, 2014, accessed: June 20 2025. [Online]. Available: http://www.ijirset.com/

[30] Wikipedia contributors, "Kalman filter," https://en.wikipedia.org/wiki/Kalman_filter, accessed June 2025.

[31] VectorNav, "Complementary Filter for IMU Sensor Fusion," https://www.vectornav.com/resources/inertial-navigation-primer/math-fundamentals/math-filtering, accessed June 2025.

[32] W. C. Michels, "Phase Shifts and the Doppler Effect," *American Journal of Physics*, vol. 24, no. 2, pp. 51–53, 1956, Accessed via AIP Publishing.

[33] MathWorks, Inc., "Beamforming and Direction-of-Arrival Estimation," https://www.mathworks.com/help/phased/beamforming-and-direction-finding.html, accessed June 2025.

[34] M. Hughes, "How to interpret IMU sensor data for dead-reckoning: Rotation matrix creation," *Technical Articles*, n.d., accessed: January 2025. [Online]. Available: https://www.xilinx.com/support/documentation-navigation/embedded-processing.html

[35] Xilinx, "PYNQ Overlays Documentation," https://pynq.readthedocs.io/en/latest/pynq_overlays.html, 2025, accessed: 2025-06-10.

[36] PYNQ Project, "PYNQ Libraries — PYNQ Documentation," https://pynq.readthedocs.io/en/latest/pynq_libraries.html, 2025, accessed June 10, 2025.

A

Appendix 1

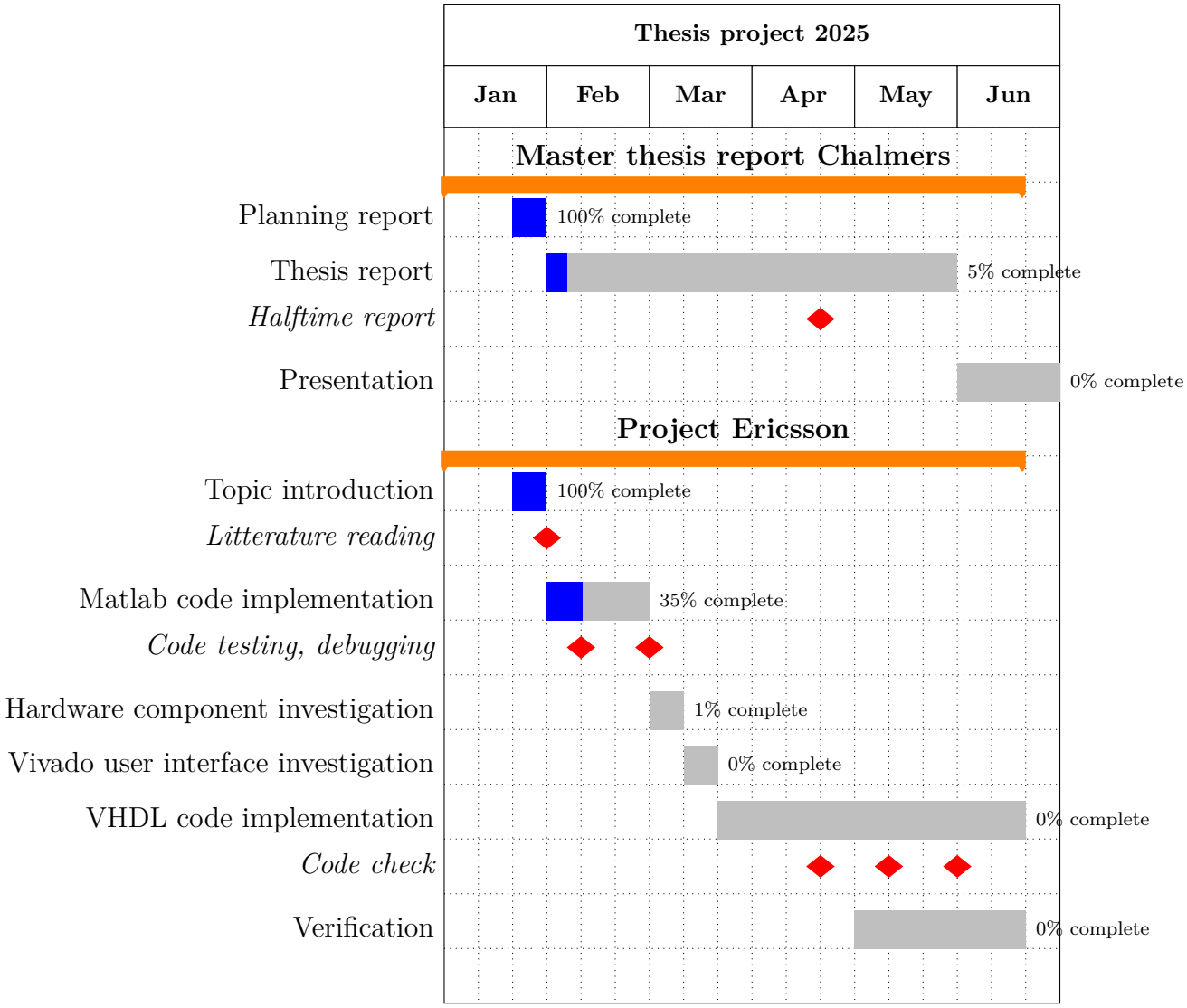


Figure A.1: Gantt diagram for master thesis project 2025.