Experimental Study of Communication Range Enhancement of Drones

CP301: Development Engineering Project

Submitted By:

Aaryan Yashvardhan Sharma	2021EEB1259
Kirtdeep Kaur	2021EEB1291
Rudra Kumar Chourasia	2021EEB1208
Yash Agarwal	2021EEB1225

Submitted To: Prof. Ashwani Sharma
Supervised By: Sundeep Kumar

2nd Semester A.Y. 2023-24
Submitted On: 10/05/2024

Department of Electrical Engineering
Indian Institute of Technology, Ropar

ABSTRACT

In radiofrequency systems, antennas play a vital role as essential components. It is crucial to optimize their quality to ensure optimal functionality. In this study, we actively investigated the performance of a leaky wave antenna designed for multiple tones. We conducted experiments to examine the beam patterns generated by the antenna at two different frequencies. We assessed the antenna's suitability for drone integration to establish secure communication links and determined which orientation of the transmitting antenna and receiver performed better over different frequency ranges.

We conducted our experiments using the ADALM Pluto SDR, which we controlled via GNU Radio using a Raspberry Pi computer. We performed various calibrations to eliminate or normalize errors across all frequencies. We employ spectrum analysers to identify and correct any offsets in the RF front end. Additionally, we used QPSK modulation and demodulation schemes for communicating purposes and conducted a qualitative analysis of the antennae's performance at different frequencies.

Under the guidance of our teaching assistant, Sandeep sir, we computed and verified the overall performance of a leaky wave antenna concerning practical data, whether the communicating antenna performs similarly or not.

ACKNOWLEDGEMENT

We are highly grateful to Prof. Rajeev Ahuja, Director, IIT Ropar, Rupnagar, for providing this opportunity to carry out the major project in the college.

The constant guidance and encouragement received from Prof. Ashwani Sharma, EE Department, IIT Ropar, has been of great help in carrying out the project work and is acknowledged with reverential thanks.

We want to express a deep sense of gratitude and profuse thanks to Sundeep Kumar. Completing the project this way was possible with his wise counsel and guidance. Throughout the semester, he has helped us with significant clarifications and suggested solutions and alternatives.

We thank research scholars, our faculty advisor, Prof. Pardeep Duhan and UG guide, and Prof. Abhishek Sharma of the electrical engineering department of IIT Ropar for their intellectual support throughout this work.

We are indebted to all who have contributed to this report.

Aaryan Yashvardhan Sharma
Kirtdeep Kaur
Rudra Kumar Chourasia
Yash Agarwal

CONTENTS

INTRODUCTION

We will begin by discussing the basics and underlying principles of our project. Main aim of our minor project is to examine antenna quality that will be mounted on the drone. For testing purposes, we mounted antennas on physical stand and later to be extended on drones.

Unmanned aerial vehicles (UAVs), commonly referred to as drones, have revolutionised various fields, including surveillance, agriculture, and emergency response. One critical aspect that influences the effectiveness of drones in these applications is their communication range. Extending the communication range of drones enables them to operate more effectively over larger areas and in challenging environments. In this project, we embark on an experimental study focused on enhancing the communication range of drones.

The ability to establish and maintain reliable communication links with drones is essential for real-time data transmission, control, and coordination of their activities. However, factors such as signal interference, atmospheric conditions, and hardware limitations can impose constraints on the communication range of drones. Therefore, it becomes imperative to explore strategies and technologies to extend this range while ensuring robust and secure communication.

Our project aims to address this challenge through a systematic experimental approach. We will investigate various techniques and technologies for communication range enhancement, ranging from antenna optimization to signal processing algorithms. By conducting experiments in controlled environments and real-world scenarios, we seek to gain insights into the factors influencing communication range and evaluate the effectiveness of different enhancement strategies.

Key aspects of our experimental study include:

- Signal Processing Algorithms: We will investigate signal processing techniques aimed at improving the reliability and robustness of communication links under adverse conditions. This may involve error correction coding, adaptive modulation, and interference mitigation algorithms.
- Hardware Integration: We will integrate the developed communication enhancement techniques into drone platforms, ensuring compatibility, reliability, and efficiency in real-world deployments.
- Performance Evaluation: Through comprehensive testing and analysis, we will assess the performance of the enhanced communication systems in terms of range extension, signal quality, data throughput, and overall reliability.

By advancing our understanding of communication range enhancement for drones through empirical experimentation, we aim to contribute to the development of more capable and reliable UAV systems for various applications. Our findings have the potential to inform future research directions and practical implementations in fields such as aerial surveillance, disaster response, and precision agriculture.

Hardware Used: ADALMPluto, Raspberry Pi, Spectrum Analyser and Leaky Wave Antennae.

Software Used: GNU Radio and PlutoSDR.

HARDWARE AND SOFTWARE USED

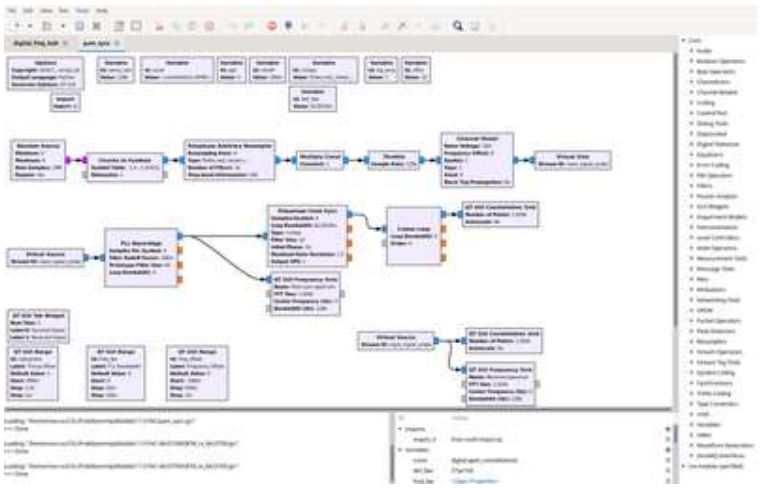
GNU Radio:

We extensively used GNU Radio; an open-source software toolkit widely utilized for software-defined radio (SDR) systems. It offers a flexible framework for designing, simulating, and deploying radio communication systems on general-purpose computing platforms. Initially developed by Eric Blossom in the early 2000s, GNU Radio was officially launched in 2001 under the GNU General Public License (GPL). Since then, it has evolved into a powerful tool for prototyping and implementing various radio communication systems.

GNU Radio plays a crucial role in the project focused on enhancing the communication range of drones. It facilitates implementing and testing different communication techniques and algorithms, enabling researchers to develop and evaluate signal processing algorithms, modulation schemes, and protocol implementations in a software environment. Researchers can rapidly prototype and iterate on different ideas by leveraging GNU Radio without needing specialized hardware.

GNU Radio offers robust support for interfacing with hardware devices, enabling users to connect with various radio front ends, data converters, and peripherals. Features include compatibility with diverse Software-Defined Radio (SDR) platforms such as the Universal Software Radio Peripheral (USRP) and PLUTO modules. Moreover, it facilitates interfacing with network devices, allowing data streaming over local or remote networks, enabling distributed radio systems and networked communication. Overall, GNU Radio is a potent tool for developing and deploying radio communication systems, offering flexibility, extensibility, and hardware interfacing capabilities that are indispensable for researchers and developers in software-defined radio and wireless communications.

GNU Radio has several blocks which are more than enough for implementing any digital RF frontend. Some of the common blocks that we use are variable, time and frequency sink, data converter i.e. int to string, int to char, etc.



Generic GNU Radio Interface

PlutoSDR:

PlutoSDR is a versatile software-defined radio platform renowned for its adaptability and extensive feature set. One notable aspect of its flexibility lies in its configurable software, allowing users to customize and modify its behaviour to suit their needs. One standard customization involves altering the configuration file to extend the local oscillator value, thereby expanding the frequency range accessible to the device. By adjusting this parameter, users can unlock additional frequency bands and explore a broader spectrum of radio signals, opening new possibilities for experimentation and research.

In addition to extending the local oscillator value, users can leverage PlutoSDR's configurable software to configure its IP address, enabling seamless connectivity with multiple Pluto modules in a network formulation. This capability is handy for collaborative research projects or large-scale deployments where multiple PlutoSDR devices must be interconnected for data sharing or synchronization. By assigning unique IP addresses to each Pluto module and configuring the network settings accordingly, users can establish a robust communication infrastructure that facilitates efficient data exchange and collaboration across multiple devices.

Furthermore, the ability to modify the PlutoSDR's configuration file and customize its software extends beyond basic adjustments to oscillator values and network settings. Advanced users and developers can delve deeper into the device's software stack, exploring opportunities for firmware modifications, driver enhancements, and even the development of custom applications tailored to specific use cases. This level of customization empowers users to push the boundaries of what is possible with the PlutoSDR, unleashing its full potential as a powerful tool for innovation and experimentation in software-defined radio.

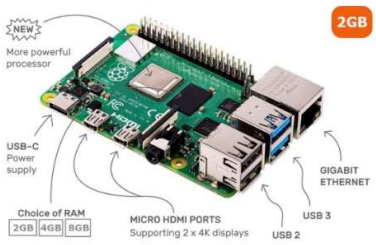
ADALM PLUTO:

In our project, we chose the ADALM Pluto as the Software-Defined Radio (SDR) platform to run GNU Radio for several compelling reasons. Firstly, the ADALM Pluto offers a cost-effective solution without compromising on performance, making it accessible to researchers with limited budgets. Its wide frequency RF range RF coverage (325 MHz to 3.8 GHz) allows for versatile experimentation across various radio bands. Additionally, the ADALM Pluto is compact and portable, making it convenient for field experiments and demonstrations.
Its compatibility with GNU Radio simplifies integration into our software development environment, enabling seamless implementation and testing of communication algorithms which were extensively used in the project such as phase shift keying modulations (PSK) and packet transmissions. Furthermore, the support of the ADALM Pluto for multiple input/output channels extends to Windows, OSX, and Linux systems. Its support was essential as we conducted the study in multiple stages, extensively utilizing both Windows and Linux operating systems.

ADALM PLUTO

Raspberry Pi:

We opted for Raspberry Pi as the controlling computer for the Pluto SDR and GNU Radio due to its compact size, low power consumption, and affordability. These qualities make it an ideal choice for embedded systems and field deployments, such as drone applications, where space and power constraints are significant factors. The Raspberry Pi provides convenient interface for connecting peripherals and sensors, enhancing its versatility for controlling external hardware like the Pluto SDR. Moreover, Raspberry Pi's robust community support and vast ecosystem of software and libraries simplify development and troubleshooting tasks. In our project there were stages when we sought help from the original developer of GNU Radio which was of immense help in the flow of our studies. Utilising Raspberry Pi ensures a cost-effective and efficient solution for running GNU Radio, enabling flexible experimentation and deployment scenarios while maintaining portability and ease of use. It has two USB 2 and 3 ports, USB C power supply, 64-bit 2.4 GHz quad-core ARM Cortex A76 processor, ethernet and 4 GB of RAM and 2 micro-HDMI ports supporting 4K displays.

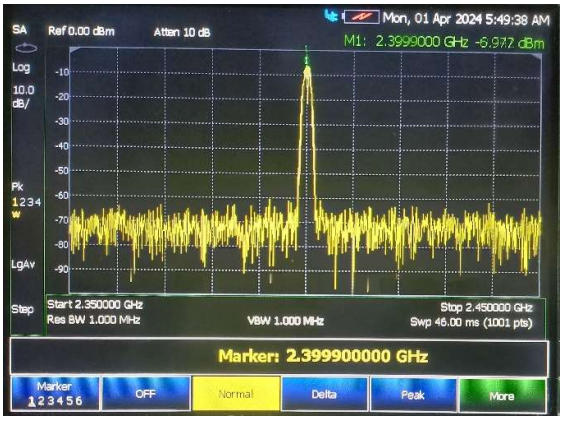


Raspberry Pi Components and Features

Spectrum Analyser:

The FieldFox handheld spectrum analyser from Keysight is a portable and versatile tool specifically crafted for on-site RF system testing and troubleshooting. With its high performance and compact design, it's ideal for a wide range of field applications, boasting a bandwidth of up to 44 GHz.

To detect the frequency offset in the local oscillator of the ADALM Pluto, we utilized the analyser. In this case, we were expecting a local oscillator frequency of 2.4 GHz but by connecting it to the ADALM Pluto, we observed it at 2.3999 GHz instead with 100 kHz offset.



100 kHz Offset

Using the spectrum analyser, we could accurately measure the frequency offset and assess its impact on the system's performance. This information is crucial for troubleshooting and fine-tuning RF systems to ensure they operate within specified parameters.

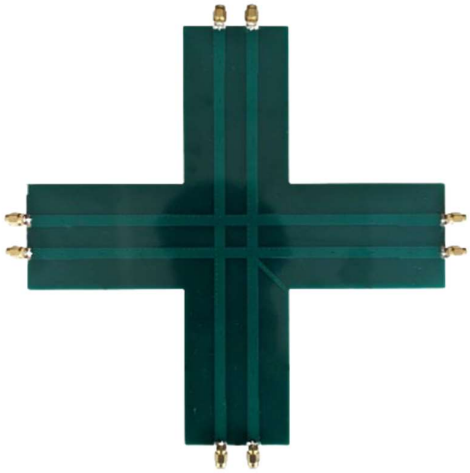
To compensate the offset, we adjusted the frequency or used sideband modulation to suppress the oscillator at 2.3999 GHz, achieving precise transmission tuning at 2.4 GHz. Failure to compensate for the offset could lead to significant consequences, such as degraded signal quality, decreased system performance, or even interference with neighbouring frequencies. This underscores the critical importance of meticulous calibration in ensuring optimal functionality and reliability of RF systems.

Leaky Wave Antennae:

A leaky wave antenna is a specialized type of antenna that operates by intentionally allowing electromagnetic waves to leak or radiate along its structure, forming a directional beam pattern. Unlike traditional antennas that emit radiation primarily perpendicular to their structure, leaky wave antennas produce radiation at an angle relative to their axis, providing unique capabilities and applications.

A critical aspect of leaky wave antennas is their beam width, which refers to the angular spread of the radiation pattern. This characteristic determines the coverage area of antenna and is often adjustable to suit specific requirements. Also, leaky wave antennas exhibit a controllable inclination angle, allowing for precise radiation targeting in a desired direction.

The Half Power Beam Width (HPBW) is another crucial parameter of leaky wave antennas, representing the angular width of the beam at half the maximum power level. This metric provides insight into the antenna's directional characteristics and is essential for determining its suitability for applications.

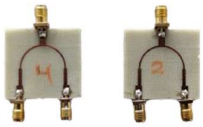


Leakey Wave Antenna

Frequency of operation is a fundamental consideration when designing and utilizing leaky wave antennas. These antennas operate within specific frequency bands, supporting various ranges depending on their construction and configuration.

In terms of construction, leaky wave antennas often consist of a dielectric substrate with periodic or non-periodic structures, such as slots or strips, etched onto its surface. These structures interact with the propagating electromagnetic waves, causing leakage radiation and the formation of the desired beam pattern.

The working principle of leaky wave antennas involves the propagation of electromagnetic waves along the antenna structure, coupled with the leakage mechanism induced by the periodic or non-periodic elements. As the waves propagate, they interact with these elements, leading to the emission of radiation at controlled angles, thus forming the directional beam pattern characteristic of leaky wave antennas.



Network Feeders

Overall, leaky wave antennas offer a versatile and efficient means of achieving directional radiation patterns with controllable beam widths, inclination angles, and frequency characteristics. Their unique construction and working principle make them valuable assets in various applications, including communication systems, radar systems, and wireless networks, where precise targeting and efficient spectrum utilization are paramount.

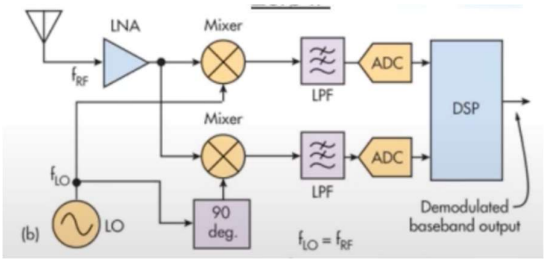
A significant thing to notice is that the beam formation occurs opposite to the excitation port.

Architecture, Layout, Features and Working of ADALM PLUTO:

Throughout the project, we have extensively and solely used ADALM PLUTO. It must not be sufficient just to mention it without discussing its vitality and functionality in depth which will later be needed to understand the integration process of software and hardware.
Software-defined radio (SDR) is a radio communication system where components that have been traditionally implemented in hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer or embedded system. While the concept of SDR is not new, the rapidly evolving capabilities of digital electronics render practical many processes which were once only theoretically possible.

Traditional RF evaluation platforms were bulky and hard to replicate. It had external oscillator, power supplies, SMA cables and tuning filters with no inbuilt antennae for field deployments. Smaller form factor was industrially desired which led to the birth of ADALM PLUTO. The ADALM-PLUTO SDR's architecture includes a transceiver chip, memory, flash, USB, and an FPGA, providing a compact and efficient design.

It is based on homodyne or synchrodyne radio architecture which can transmit signal at baseband without intervention of any local oscillator or intermediate frequency, hence the name Zero-IF receiver.
A homodyne transceiver combines incoming RF signal with a local oscillator signal to down convert it to a lower intermediate frequency (IF). The mixer generates both sum and difference frequencies, with the difference frequency representing the baseband signal containing the original information. A low-pass filter isolates the desired IF signal, which is then amplified and processed to extract the information. This method simplifies signal processing by directly converting the RF signal to a lower frequency, facilitating easier handling and extraction of useful data.



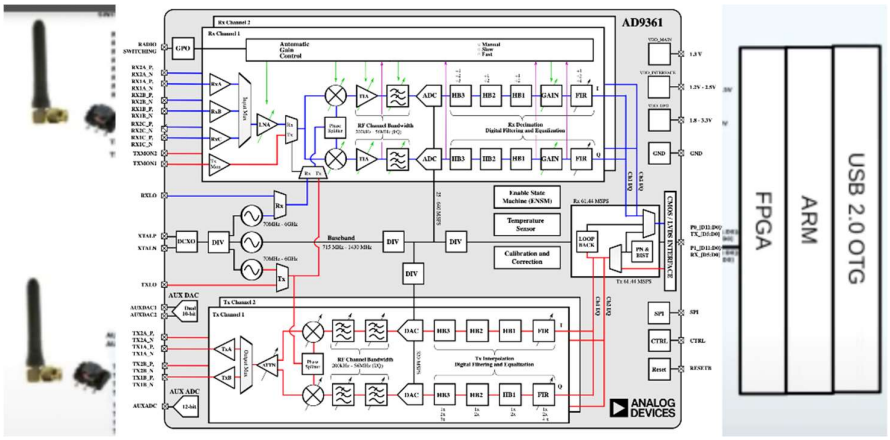
Homodyne Receiver Architecture

We prefer PLUTO over conventional method because of its versatility ranging from automatic gain control, analogue baseband filtering, DC correction, external LNA control, digital programmable baseband filter, multichip sync to RX IQ tracking (in-phase and quadrature).

PLUTO has AD9363 chip for transceiver manufactured by Analog Devices. The AD9363 chip stands out for its wide frequency range, versatile signal processing, and high integration, reducing external component needs. Its low power consumption suits battery-operated devices, while its excellent linearity and sensitivity ensure reliable signal transmission.

PCB Layout

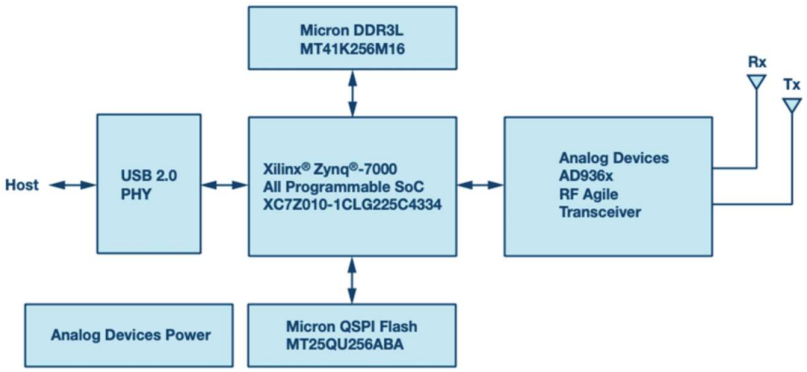
Configurable and programmable, it adapts to specific needs, enhancing performance and versatility. Built to stringent quality standards, it offers robustness for diverse applications. Advanced digital signal processing capabilities enable features like adaptive filtering and interference cancellation, further improving system performance. These technical advantages collectively position the AD9363 chip as a top choice for engineers and designers seeking efficient, high-performance transceiver solutions. It has tuning range of 325 MHz to 3.8 GHz with bandwidth of 200 kHz to 20 MHz with average power consumption of 1.3 W making it most ideal chip for development of digital RF front end over conventional setups consuming watts of energy. Capable of sampling I/Q samples at rate of 65.1 kSps to 61.44 MSps. Tuning range of 70 MHz to 6 GHz can be achieved but performance is undocumented.



FPGA Interface and Architecture

Configurable and programmable, it adapts to specific needs, enhancing performance and versatility. Built to stringent quality standards, it offers robustness for diverse applications. Advanced digital signal processing capabilities enable features like adaptive filtering and interference cancellation, further improving system performance. These technical advantages collectively position the AD9363 chip as a top choice for engineers and designers seeking efficient, high-performance transceiver solutions. It has tuning range of 325 MHz to 3.8 GHz with bandwidth of 200 kHz to 20 MHz with average power consumption of 1.3 W making it most ideal chip for development of digital RF front end over conventional setups consuming watts of energy. Capable of sampling I/Q samples at rate of 65.1 kSps to 61.44 MSps. Tuning range of 70 MHz to 6 GHz can be achieved but performance is undocumented.

Xilinx Zynq is a System on Chip (SoC) combining FPGA fabric with ARM processor cores. It's used in Pluto SDR for its versatility and performance. The FPGA fabric enables real-time signal processing and customisation, crucial for software-defined radio (SDR) applications like Pluto. ARM cores provide a familiar processing environment for running software stacks and applications. This integration optimises Pluto's capabilities, allowing it to handle complex radio tasks efficiently while offering flexibility for future enhancements and customisations. Zynq's balance of FPGA flexibility and ARM processing power makes it an ideal choice for Pluto SDR, meeting demands for high-performance, adaptable radio systems.



Block Diagram

PROCEDURE

Link budget:

It evolves from the Friis link equation, which relates the transmitter's gain to the receiver's gain. It considers gains and losses from the transmitter to the receiver in a communication system. Link budgets describe one direction of the wireless link. The outcome will indicate approximately the level of SNR we should anticipate at our receiver. Further analysis would be required to verify if that SNR is sufficiently high for our application. In signal power budgeting, we typically have the transmitted power for a system. The objective is to determine the received power within the system.

We have four parameters to determine the received power:

Pt: Transmission power
G_t: Gain of transmit antenna
G_r: Gain of receive antenna
L_p: Distance between Tx and Rx (i.e., how much wireless path loss)

Transmission power: Transmission power is straightforward; it will be a value in watts, dBW, or dBm. Every transmitter has one or more amplifiers, and the transmit power is primarily a function of those amplifiers.

Antenna gains: It indicates the directivity of the antenna. Antennas are either omnidirectional or directional. Omnidirectional antennas radiate power in all directions, and their gain generally lies between 0-3 dB. Directional antennas radiate power in a specific direction and will have a higher gain, usually 5 dB or higher, and anywhere up to 60 dB when converging towards the central beam or lobe.

In a link budget, we must assume that any directional antenna, whether transmitting or receiving, is pointed in the right direction. If checked, our link budget will be accurate, and communication can recover. Our link budgets generally assume ideal circumstances while adding miscellaneous losses to account for practical factors.

Channel coding:

We need channel coding because wireless channels are noisy, and our received signal will need fixing. Just as CRCs detect errors at the receiving end, channel coding detects and corrects errors at the receiver.

Example: If we allow some room for error, we can transmit at a higher-order modulation scheme without experiencing a broken link. For instance, consider the following constellations showing QPSK (left) and 16QAM (right) under the same amount of noise. QPSK provides 2 bits per symbol, while 16QAM offers twice the data rate at 4 bits per symbol. However, note how, in the QPSK constellation, the symbols tend not to cross the symbol decision boundary or the x-axis and y-axis, implying that the symbols will be received correctly. Meanwhile, in the 16QAM plot, there is an overlap in the clusters, resulting in many incorrectly received symbols. Source coding compresses the transmitted data as much as possible so as not to exceed the channel bandwidth requirements.

At a low SNR, we need a low-order modulation scheme (e.g., QPSK) to deal with the noise, and at a high SNR, you can use modulation like 256QAM to achieve more bits per second. Channel coding follows the same principle; we want lower code rates at low SNRs and can use a code rate of almost one at high SNRs. Modern communication systems have a set of combined modulation and coding schemes called MCS. Each MCS specifies a modulation scheme and a coding scheme for specific SNR levels.

Shannon limit: The Shannon limit is a theory that tells us how many bits per second of error-free information we can send: $C = B \log_2\left(1 + \frac{S}{N}\right)$
C: Channel capacity (bits/sec)
B: Bandwidth of the channel (Hz)
S: Average received signal power (W)
N: Average noise power (W)

Filters:

In DSP, where the input and output are signals, a filter has one input signal and one output signal. If we want to give two inputs to a filter, we must first add them or perform the necessary operations to combine them into one signal. Filters facilitate separating combined or mixed signals (e.g., extracting the desired signal), removing excess noise after receiving a signal and restoring distorted signals (e.g., an audio equaliser is a filter).

There are four basic types of filters: low-pass, high-pass, band-pass, and band-stop (or commonly referred to band-reject). Each filter has different frequency ranges that they allow to pass and which they block. The range of frequencies a filter lets through is known as the "passband", and "stopband" refers to what is blocked. In the case of the low-pass filter, it passes low frequencies and stops high frequencies, so 0 Hz will always be in the passband. 0 Hz will always be in the stopband for a high-pass and band-pass filter.

An example of this is the low-pass filter. In addition to the cutoff frequency, the other main parameter of our low-pass filter is "transition width". Transition width, also measured in Hz, instructs the filter on how quickly it should transition between the passband and stopband, as an instant transition is impossible. We aim for a smaller transition bandwidth, but we do not simply make it minimum or zero because doing so requires more taps in our filter, leading to increased computation time.

In filters, "taps" refer to the individual coefficients or weights assigned to different points in the filter's impulse response. Each tap corresponds to a specific delay element in the filter's architecture. By adjusting the values of these taps, the filter can modify the characteristics of the input signal, such as its frequency response or phase. Taps are crucial for determining the filter's behaviour and effectiveness in processing signals.

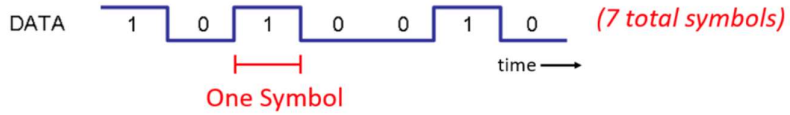
A 50-tap filter can run daily using only 1% of the CPU on a Raspberry Pi. Meanwhile, a 50,000-tap filter will overload our CPU! Typically, we use a filter design tool to determine the number of taps it outputs, and if it exceeds a certain threshold (e.g., more than 100), we increase the transition width. This decision depends on the application and hardware running the filter. A filter also has its time

domain representation, known as the filter's impulse response. The impulse response is simply the taps for a FIR type filter.

Taps can be real or complex and are suitable for natural or complex input. Real taps yield symmetrical frequency responses around 0 Hz, while complex taps enable asymmetric responses. FIR filters are more straightforward and achieved through taps, while IIR filters, though more complex and potentially unstable, are more efficient, requiring less CPU and memory.

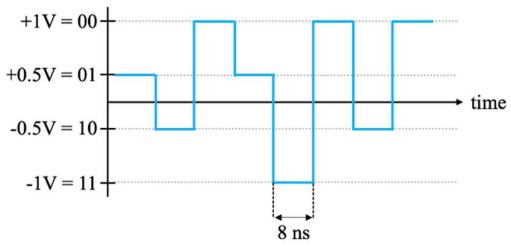
Digital modulation:

Modulation aims to optimise spectral efficiency by maximising data in a minimal spectrum, considering speed, bandwidth, and creative techniques for transmission. It aims to optimise spectral efficiency by maximising data within the limited spectrum. We seek to maximise "spectral efficiency" (bits sec-1 Hz-1) by transmitting faster, recognising Fourier properties. We will explore alternative techniques beyond speed, balancing trade-offs and fostering creativity in modulation strategies.



Symbol transmission is fundamental in conveying data. In modulation schemes, a symbol represents a unit of data encoded into a single transmission unit. Each symbol represents bits sent consecutively in a signal corresponding to a specific phase, amplitude, or frequency modulation state. For example, in Quadrature Phase Shift Keying (QPSK), each symbol represents a unique phase shift of the carrier signal. QPSK maps two data bits to each symbol, allowing for twice the data rate compared to Binary Phase Shift Keying (BPSK), where each symbol represents one bit. Thus, QPSK can transmit four different phase states (00, 01, 10, 11), each corresponding to a different symbol on the constellation diagram.

For instance, Ethernet employs 4-level amplitude modulation (2 bits/symbol) with 8 ns symbols.



Wireless transmission differs due to antenna limitations and bandwidth issues. Carriers, like FM radio frequencies, undergo modulation, altering amplitude, phase, or frequency to carry data which will be discussed later in the report.

In various wireless and wired communication protocols, such as those using PSK or QAM, differential coding is a crucial step before modulation (or after demodulation). Take, for instance, receiving a BPSK signal. The signal might encounter random delays during transmission, causing the constellation to rotate randomly. When the receiver synchronises and aligns the BPSK to the real axis, it cannot discern a 180-degree phase shift due to symmetry. One approach includes known symbols, pilot

symbols, within the transmitted information. These aid the receiver in deciphering clusters as 1s or 0s. However, sending pilot symbols reduces the data rate. An alternative is differential coding.

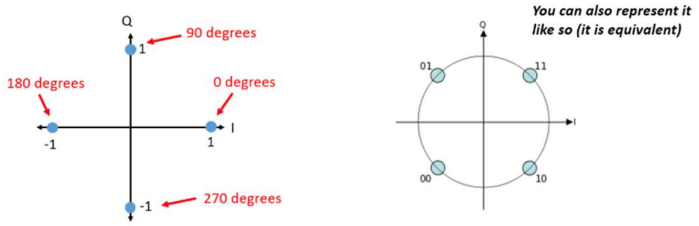
In its simplest form with BPSK, each symbol represents one bit. Instead of directly encoding 1s and 0s, differential BPSK coding involves transmitting a 0 if the input bit matches the encoding of the preceding bit and a 1 if it differs. This method maintains the bit count while eliminating the 180-degree phase ambiguity.

The encoding is represented by the equation: $y_i = y_{i-1} \oplus x_i$
Starting the output sequence requires an initial arbitrary bit, as the output depends on the previous step's output. However, the choice between 1 or 0 for the starter symbol is inconsequential for decoding. Hence, differential encoding resembles a diagram with a delay-by-1 operation.

QPSK scheme:

Quadrature Phase Shift Keying (QPSK) digital modulation scheme conveys data by changing the carrier signal phase, doubling the data rate of traditional Binary Phase Shift Keying (BPSK) by encoding two bits per symbol. Two carriers, one in-phase (I) and one quadrature-phase (Q), are generated to generate QPSK. The input binary data stream has two streams, one controlling the phase of the I carrier and the other controlling the phase of the Q carrier.

Modulation combines the two carrier signals with the modulated data streams, creating a constellation diagram where each point represents a unique phase combination of the I and Q signals. Demodulation involves separating the I and Q components of the received signal and determining the phase difference between them. Comparing this phase difference to the reference constellation points recovers the original data.



Constellation Diagram of QPSK

Offset correction is crucial in QPSK to eliminate phase, frequency, and timing errors. We corrected phase offset by estimating the phase difference between the received signal and the local oscillator. Adjust frequency offset by adjusting the frequency of the local oscillator—correct timing offset by synchronizing the receiver's clock with the received signal.
We mitigate noise in QPSK through error correction coding and adaptive equalization techniques. Error correction coding adds redundancy to the transmitted data, enabling the receiver to detect and correct errors. Adaptive equalization adjusts the receiver's filters to compensate for channel distortion caused by noise.

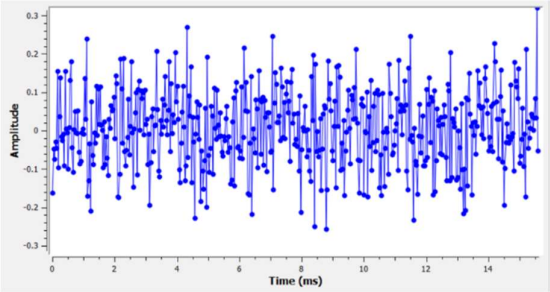
In GNU Radio, QPSK modulation and demodulation are implemented using various blocks such as Constellation Modulator, Constellation Decoder, Costas Loop, and Equalizer. These blocks enable the creation of QPSK transceivers to test antenna quality and communication performance. A QPSK transceiver measures parameters like bit error rate (BER) and SNR under different antenna conditions

to test antenna quality. In the end we analyse the received constellation diagram and BER/SNR metrics to evaluate and optimize antenna performance.

Modelling Noise:

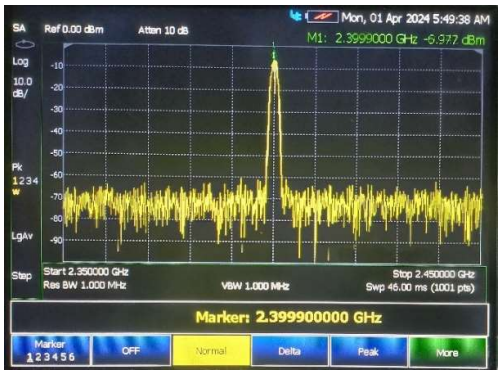
Noise in RF systems refer to unwanted random fluctuations in the signal, originating from various sources such as thermal effects, electronic components, and environmental factors. It degrades signal quality, impacting communication and detection accuracy. Managing noise is crucial for optimizing system performance and ensuring reliable transmission and reception of signals.

Note how the average value is zero in the time domain graph. If the average value was not zero, we could subtract it, call it a bias, and leave us with an average of zero. Also note that the individual points in the graph are not "uniformly random", i.e., larger values are a rarity, and most points are closer to zero.



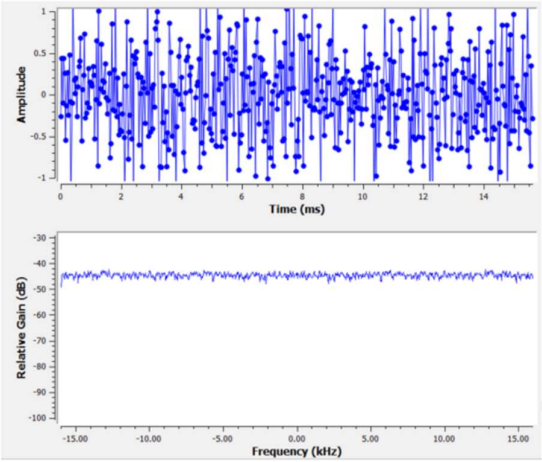
Temporal Noise Profile

We call this type of noise "Gaussian noise". It is a good model for noise that comes from many natural sources, such as thermal vibrations of atoms in the silicon of our receiver's RF components. It can also arise from noise floor. The noise floor in RF systems is the background noise level present in the system when no signal is transmitted or received. It represents the minimum detectable signal level and is measured in decibels relative to a reference level, such as dBm. The typical noise floor value depends on factors like system design, frequency band, and environmental conditions, but it often ranges from around -100 dBm to -150 dBm. In our electromagnetics lab, the noise floor was slightly elevated at around -70 dBm.



Noise Floor Centred Around –70 dBm

The Gaussian distribution has two parameters: mean and variance. We already discussed how the mean can be considered zero because we can permanently remove the mean or bias if it is not zero. The variance changes how "strong" the noise is. A higher variance will result in more significant numbers. It is for this reason that variance defines the noise power.
The following graphs show some simulated noise in the time domain (top) and a plot of the Power Spectral Density (PSD) of that noise (below). These plots were taken from GNU Radio.

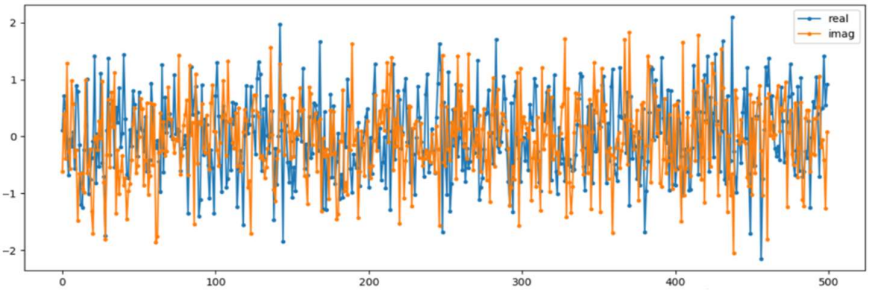


PSD Content of Different Frequencies

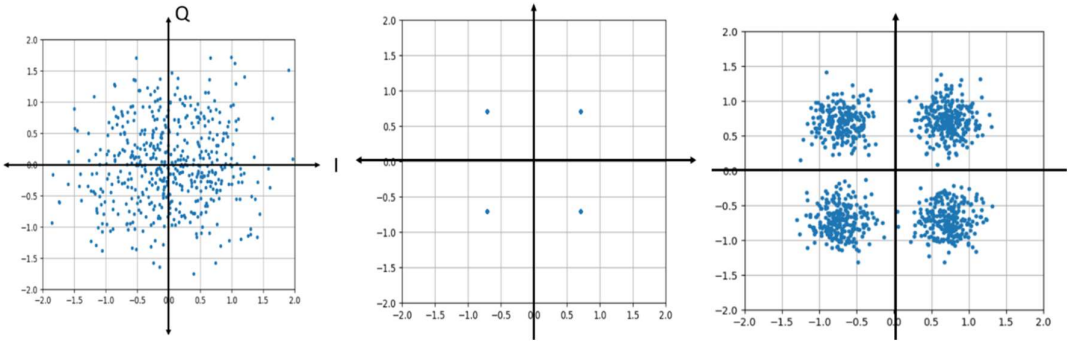
Additive White Gaussian Noise (AWGN) is an abbreviation you will often hear in the DSP and SDR world. The GN, Gaussian Noise, we already discussed. Additive means the noise is being added to our received signal. In the frequency domain, white means the spectrum is flat across our entire observation band. It will almost always be white in practice or approximately white. We will use AWGN as the only form of noise when dealing with communications links, link budgets and other communication aspects.

AWGN in the time domain is also Gaussian noise in the frequency domain, although it looks like a flat line when we take the magnitude and perform averaging
It looks roughly the same across all frequencies and is relatively flat. It turns out that Gaussian noise in the time domain is also Gaussian noise in the frequency domain. However, they are similar because the frequency domain contains magnitude from FFT with a mixture of logarithmic representations.

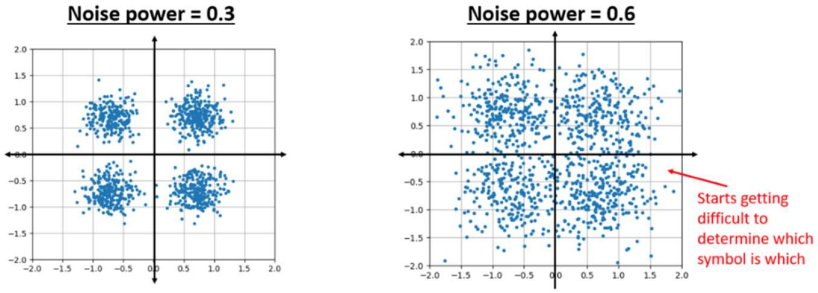
"Complex Gaussian" noise is experienced when we have a signal at baseband; the noise power is split between the real and imaginary portions equally. Most importantly, the real and imaginary parts are independent of each other.



Complex Noise Profile



AWGN on I-Q Plot *QPSK* *QPSK + AWGN*



Effect of Noise on Constellation Diagram

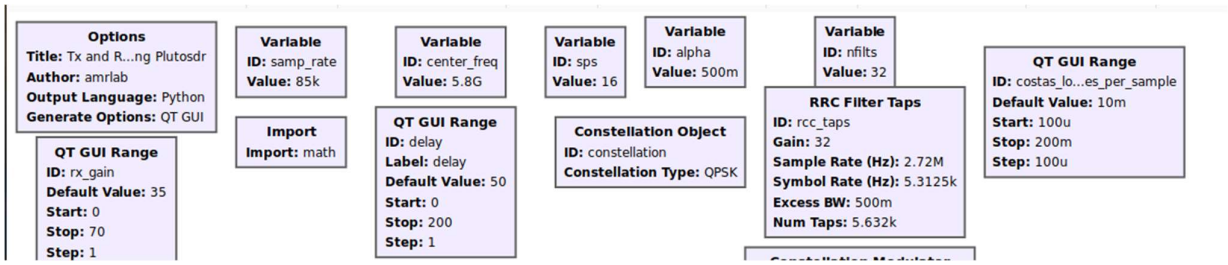
As anticipated, it resembles a random cluster centred at 0 + 0j, the origin. Increasing noise level scatters constellations further around from their loci.

We are starting to understand why transmitting data wirelessly isn't that simple. We want to send as many bits per symbol as possible, but if the noise is too high, we will get erroneous bits on the receiving end.

TEST SETUP

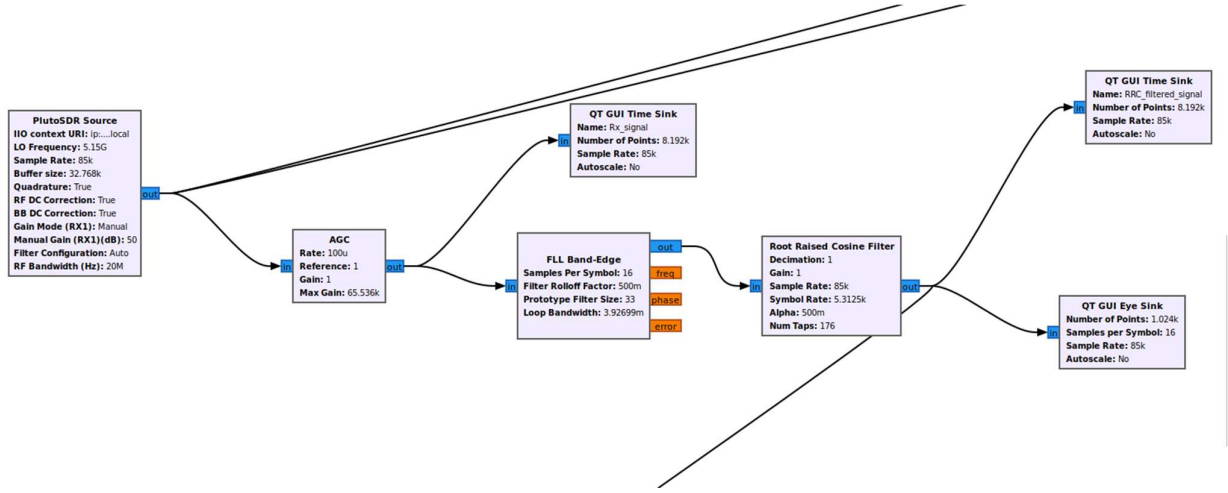
SOFTWARE

Variables:



1. `samp_rate`: Sampling rate, frequency at which a signal is sampled.
2. `center_freq`: Central frequency, midpoint frequency of a signal.
3. `sample per sec`: Number of samples taken per second.
4. `Costas loop`: Type of phase-locked loop used in communication systems for coherent demodulation.
5. `tx_attenuation`: Transmission attenuation, reduction in signal strength during transmission. `rx_gain`: Receiver gain, amplification applied to incoming signals in a receiver.
6. `constellation`: Representation of signal states in complex plane for modulation schemes like QAM. `offset`: Deviation from a reference point or value in a signal or system.

Receiver and Frequency Locked Loop (FLL):



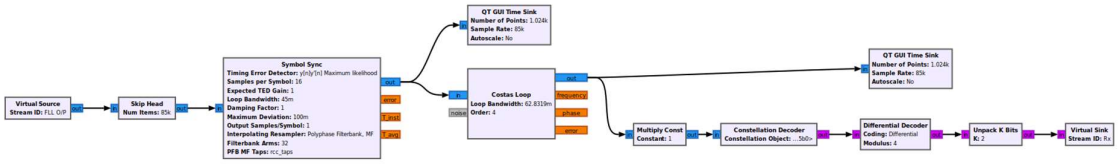
In GNU Radio, Automatic Gain Control (AGC) is a system that automatically adjusts a signal's gain to maintain a constant output power level, ensuring optimal signal strength without distortion or saturation. This feature is crucial in dynamic environments where signal levels vary widely, such as in wireless communications or software-defined radio applications.

FLL Band Edge, or Frequency-Locked Loop Band Edge, is a component used to track and adjust the frequency of a signal precisely at the edges of a defined frequency band. This functionality is essential

in radio frequency applications where accurate frequency synchronization is required to ensure efficient signal reception and transmission.

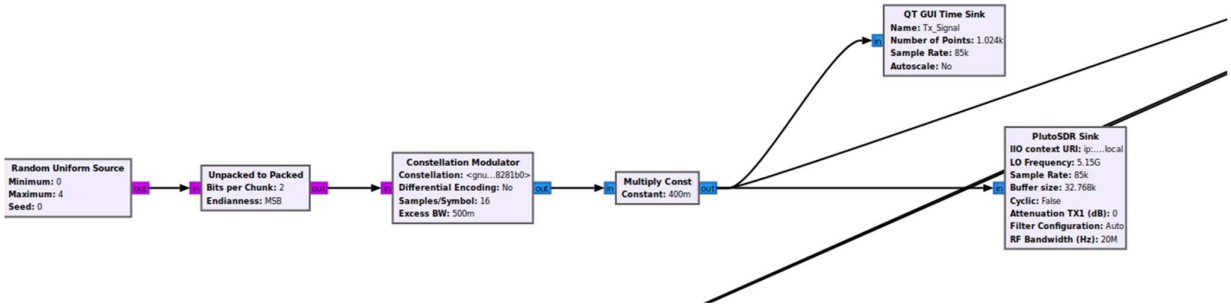
A root-raised cosine (RRC) filter is a digital filter commonly used in communication systems to shape the transmitted signal's pulse to minimize inter-symbol interference (ISI) and spectral regrowth. This filter has a square root response to ensure a constant envelope in the transmitted signal, making it suitable for applications such as digital modulation schemes like QPSK and QAM. RRC Filter Taps define the coefficients of an RRC filter, determining its frequency response and shaping characteristics. These taps specify the filter's impulse response, which is crucial for pulse shaping in digital communication systems to mitigate inter-symbol interference and meet spectral mask requirements.

Symbol synchronisation and decoding:



Symbol Sync in GNU Radio synchronizes received symbols with their expected positions in the signal, which is crucial for proper demodulation. Maximum Likelihood synchronization optimally aligns symbols by minimizing errors and enhancing signal recovery accuracy. Costas Loop is a carrier recovery mechanism used in communication systems to synchronize the receiver's local oscillator with the carrier signal, ensuring accurate demodulation of phase-shift keying (PSK) signals. Constellation Decoder decodes received symbols into their corresponding digital data points, facilitating signal demodulation and data recovery in digital communication systems. Differential Decoder reconstructs the original data from differentially encoded symbols, which is essential for demodulating signals encoded using differential modulation techniques like DPSK (Differential Phase Shift Keying). Unpack k Bits separates a stream of bits into groups of k bits, typically used after demodulation to organize received data into manageable chunks for further processing or analysis.

Transmitter sink:



Flow Diagram for Transmitter sink

Pack k Bits in GNU Radio combines a sequence of binary bits into groups of k bits, facilitating data transmission or storage by organizing data into packets of a specified size for further processing or

modulation. Random Uniform Source generates a stream of pseudo-random numbers uniformly distributed within a specified range, often used in simulations or testing scenarios to emulate stochastic processes or provide input for randomized algorithms. Constellation Modulator maps digital data symbols onto a constellation diagram, representing the amplitude and phase of each symbol in a modulation scheme. It encodes input bits into corresponding symbols for transmission over a communication channel, essential for modulating digital signals in various communication systems like QPSK or QAM.

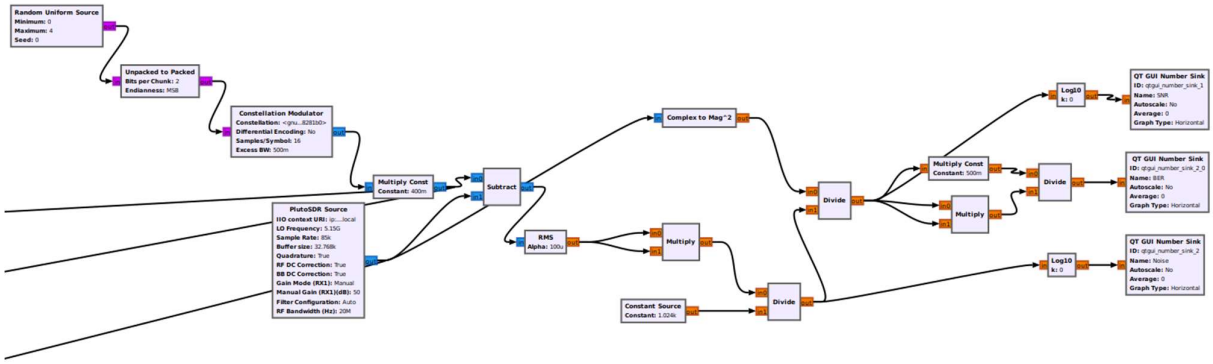
SNR and BER:

Random Uniform Source: This block is used to generate several samples of random numbers of [min, max) meaning the max value won't be included. Great for creating bytes of information for a modulator. Parameters:

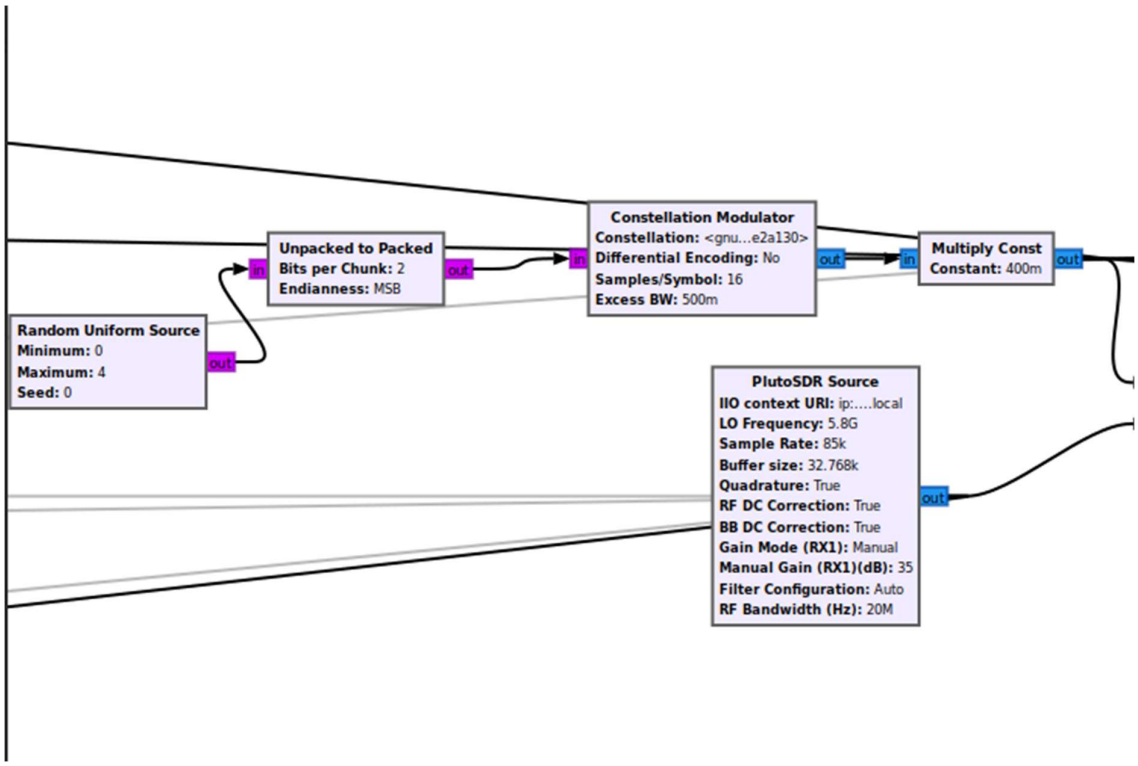
1. Minimum: Defines the minimal integer value output.
2. Maximum: Output values are below this value.
3. Seed: This is used for Pseudo Random Number Generator. Defaults to 0.

Random Uniform Source: This block is used to generate several samples of random numbers of [min, max) meaning the max value won't be included. Great for creating bytes of information for a modulator. Parameters:

4. Minimum: Defines the minimal integer value output.
5. Maximum: Output values are below this value.
6. Seed: This is used for Pseudo Random Number Generator. Defaults to 0.



Flow Diagram for Calculating SNR from BER



Unpacked to Packed: This block converts a stream a unpacked bytes or shorts into a stream of packed bytes or shorts. The low bits are extracted from each input byte. These bits are then packed densely into the output bytes. Parameters:

1. Bits per Chunk: Number of bits to pack into each group
2. Endianness: Most or Least significant Bit first
3. Num Ports: Number of input streams to operate on

Constellation Modulator: In this this block the input is a byte stream and the output is the complex modulated signal at the baseband. Parameters:

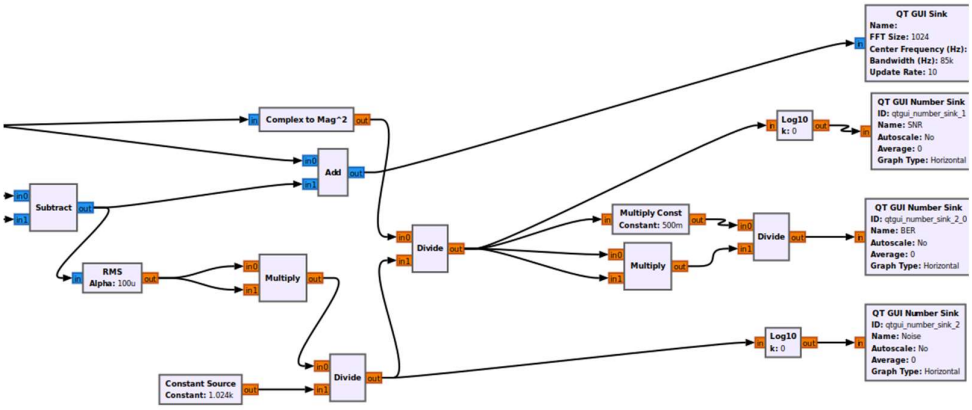
1. Constellation: Determines the modulation type, provide a constellation object here.
2. Samples per Symbol: Samples per baud >=2 (int)
3. Differential Encoding: Whether to use differential encoding (Boolean)
4. Excess BW: Root-raised cosine (RRC) filter excess bandwidth (float)
5. Verbose: Print information about modulator (Boolean)
6. Log: Log the modulation data to files (Boolean)

PlutoSDR Source: The PlutoSDR is a low-cost SDR made by Analog Devices. It can operate from 70 MHz to 6 GHz using simple “hack” and has a max sample rate of 56 MHz.

Parameters:

1. LO Frequency: Selects the RX local oscillator frequency.
2. IIO Context URI: IP address of the unit (to be written in quotes)
3. Sample Rate: Sample rate is samples per second, this will define how much bandwidth your SDR receives at once. The limit is >= 520833 and <= 61440000

4. RF bandwidth: Configures RX analogue filters: RX TIA LPF and RX BB LPF. Limits: >=200000 and <= 52000000
5. Buffer Size: Size of the internal buffer in samples. The IIO blocks will only input/output one buffer of a samples at a time. To get a highest continuous sample rate, try using a number in the millions.
6. Quadrature: True/False
7. RF DC Correction: True/False
8. BB DC Correction: True/False
9. Gain Mode (Rx1): Selects one of the available modes: manual, slow attack, hybrid and fast attack. For most spectrum sensing type applications, use a manual gain, so that you know when a signal is present or not, and its relative power.
10. Manual Gain (Rx1)(dB): Gain value, max of 71dB, or 62dB when over 4 GHz centre frequency.
11. Filter: Allows a FIR filter configuration to be loaded from a file.
12. Filter Auto: When enabled loads a default filter and thereby enables lower sampling/ baseband rates.



Constant Source: Produces a constant source of type complex, float, int, or short. Will continuously output this constant value. Parameters:

1. Constant (R): Output value
2. QT GUI Sink

This is essentially just a combination of the following four GUI's:

1. QT GUI Frequency Sink
2. QT GUI Time Sink
3. QT GUI Constellation Sink
4. QT GUI Number Sink

Hacking PlutoSDR:

It typically refers to modifying or customizing its firmware, software, or hardware beyond its intended use. Hacking it could involve tasks such as:

1. Custom Firmware: Developing firmware to add new features or improve performance.

2. Software Modifications: Tweaking Software applications that run on the PlutoSDR for specific purposes on functionalities such as output power, modulation schemes, or frequency range. Using advanced SDR software, we may be able to optimize settings to maximize range.
3. Hardware Modifications: Physically altering the hardware components of the PlutoSDR for specialized applications. It can be achieved by many ways like upgrading the antenna can improve both range and the quality of the signal, adding RF amplifiers can boost the signal power which may increase the transmission range or modifying the hardware to operate at higher frequencies.

Hacking the PlutoSDR to increase the RF range can be a valuable learning experience. In our project we attempted to increase the RF range of the PlutoSDR through the terminal commands involving the modification of its firmware or software parameters. We followed the following steps which are provided by Analog Devices:

1. Open the terminal (either host or VM)

   ```
   ssh root@192.168.2.1
   ```

   The default password is 'analog'

2. We should see the PlutoSDR welcome screen. We have now SSHed into the ARM CPU on the pluto itself.
   If the Pluto have firmware version 0.31 or lower, then use the following commands:

   ```
   fw_setenv attr_name compatible
   fw_setenv attr_val ad9364
   reboot
   ```

   For 0.32 and higher versions use:

   ```
   fw_setenv compatible ad9364
   reboot
   ```

3. Now we will be able to tune up to 6 GHz and down to 70 MHz and we should use a sample rate up to 56 MHz.

HARDWARE

Duplex Mode of PlutoSDR (Transceiver):

The duplex mode of PlutoSDR refers to its ability to transmit and receive signals, enabling full-duplex communication simultaneously. This mode offers several advantages in various applications:

1. It allows for efficiently utilising the available spectrum by enabling concurrent transmission and reception, maximising throughput and minimising latency.
2. Full-duplex operation enhances system flexibility, enabling dynamic adjustment of transmission parameters based on real-time feedback from received signals. This capability is advantageous in rapidly adapting to changing environmental conditions or network dynamics.
3. Full-duplex communication enhances spectral efficiency, enabling more efficient use of limited bandwidth resources, which is crucial in congested or spectrum-limited environments.

However, the full-duplex mode of PlutoSDR also presents certain drawbacks and limitations. One major limitation is the potential for self-interference, where transmitted signals interfere with simultaneously received signals, leading to degradation in signal quality and throughput. Controlling self-interference requires sophisticated signal processing techniques and hardware design considerations, which may increase system complexity and cost. Furthermore, full-duplex operation may impose additional requirements on the hardware, such as increased power consumption and thermal management, which can impact the overall system performance and reliability.

In addition to these general drawbacks, PlutoSDR's full-duplex mode may also be subject to specific limitations related to synchronisation errors and hardware constraints. Synchronisation errors between the transmitter and receiver can introduce signal processing and decoding inaccuracies, reducing communication reliability and throughput. Moreover, hardware limitations, such as the available processing power and memory resources, may impose constraints on the achievable performance of full-duplex communication systems implemented using PlutoSDR.

However, we will avoid duplex mode for the noise it introduces in the system, hampering the quality testing of the antenna procedure.

Multitone Transmission and Reception:

The PlutoSDR can transmit signals at dual frequencies by configuring its local oscillator. Nonetheless, this feat can only be achieved partially due to inherent hardware constraints, chiefly the availability of a single local oscillator. Overcoming this limitation necessitates the implementation of a function that sequentially transmits signals at multiple frequencies, each separated by a specified time delay, which can be externally adjusted. Regrettably, this approach introduces the risk of heightened noise levels and potential interference between symbols. Moreover, mitigating these undesirable effects poses a formidable challenge, as noise and interference removal is intricate, labour-intensive, and often yields unsatisfactory results, particularly in the context of synchronisation requirements.

One strategy to circumvent these issues involves individually capturing readings for each frequency and subsequently merging the acquired data into unified plots. This approach minimises the adverse impact of noise and interference, as the signals are processed independently before being combined. It not only simplifies the synchronisation process but also enhances the clarity and accuracy of the transmitted data. However, it is imperative to acknowledge that while this method offers a practical solution to the challenges posed by the PlutoSDR's hardware limitations, it may entail trade-offs in transmission efficiency and overall system complexity. Therefore, careful consideration must be given to balancing performance optimisation and operational feasibility when designing transmission protocols for dual-frequency applications using the PlutoSDR.

However, we will not use the multitone capability of the PlutoSDR due to the limitations discussed above.

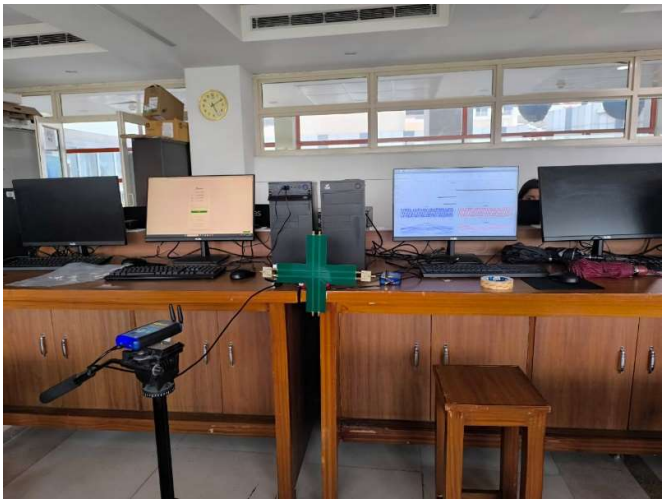
Antennae and Test Bench:

Initially, we designed a GNU Radio program to transmit data through QPSK modulation. We fed the generated signal to the antenna via a cable connected to the PlutoSDR. We utilised tripods, tapes, and USB cables to extend the connections to the antenna wherever necessary. We employed two PlutoSDRs: one as a source to antennae and the other as a receiver mounted on the tripod.

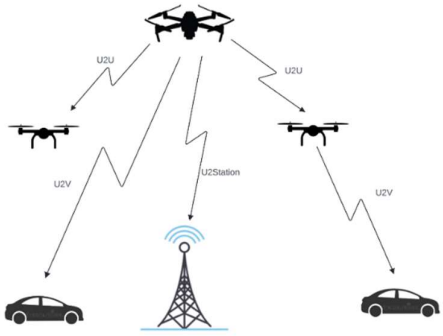
We adjusted the orientation of the PlutoSDRs back and forth to align with the antennas' main lobe and inclination angles for different frequencies. We also used feeders for connecting excitation source (PlutoSDR) to the antennae through SMA pins.

Leaky Wave Antenna (LWA)

Initially, Sandeep sir designed the antenna and documented the simulation and experimental results. We utilised these results to identify the lobes, including their inclinations relative to the vertical axis. We conducted tests at two different frequencies and temporarily adjusted the inclinations to match the simulated and experimental results he documented, obtaining a quite appreciable and decent constellation diagram indicating minimal data loss, high SNR and low BER. Tolerance was not more than ±5° in either direction.



Mock Test Setup in EM Lab



Intended Application of Drone Antenna

In the actual application scenario, the antenna mounted on the UAV serves as a pivotal communication link, facilitating interaction with other UAVs, ground vehicles, or the central base station. Within this framework, one UAV equipped with the antenna is an access point or master, communicating with other UAVs or enslaved people. Achieving synchronisation among these nodes is essential to eliminate interference noise and synchronisation errors, which could significantly degrade the BER and SNR, thereby impairing signal transmission efficiency. It is essential to note that the positioning of both UAVs and vehicles must fall within the spatial coverage of the antenna's main lobe.

Typically, moving vehicles remain within the beam's range for extended periods due to the wide beam widths characteristic of UAV antennas. Nonetheless, their susceptibility to signal degradation varies depending on the separation distance and the communication frequency.

To optimise spatial coverage and ensure seamless communication, a distributed network of UAVs maintains continuous connectivity. When a vehicle transitions out of the communication range of one UAV, it seamlessly transitions into the coverage area of another UAV, ensuring uninterrupted communication. It is crucial to recognise that vehicles also possess antennas with specific beam patterns, necessitating alignment between the transmitter and receiver beams for successful communication. Typically, linearly polarised antennas in GSM (Global System for Mobile Communication) facilitate robust and reliable communication links in such scenarios.

Spatial Coverage Through Multiple UAVs

EXPERIMENTAL SETUP

In the final setup, we fixed the antenna on the wall above. We operated the antenna using Raspberry Pi mounted on the legs and PlutoSDR mounted on top of the tripod. Necessary connections have been made, and a monitor and laptop were connected to the view transmitter and receiver side data effectively.

We used PlutoSDR as the receiver and transmitter. The receiver was mounted on a stool and placed directly below the antenna. The spatial orientation of beam and power delivery performance was evaluated using BER and noise power as performance measures.

Averaging functions were used on the BER and noise to obtain stable values, and then the SNR was computed. The vertical distance from the roof is 1.9 m, and the horizontal span varies from 0 to 4.5m at a step of 10 cm. Readings at these locations were needed.

Demonstration and Experimentation Test Setup in EM Lab

RESULTS

Vertical distance btw tran and rcv = 190 cm		
freq=5.15GHz		
distance in cm	ber(out of 100)	snr(dB)
0	0.0002	4.4
10	0.0251	2.3
20	0.0204	2.39
30	0.0245	2.31
40	0.1256	1.6
50	0	5.86
60	3.2432	0.188
70	0.1285	1.59
80	2.3387	0.33
90	0.8072	0.792
100	0.9527	0.72
110	1068.981045	-2.33
120	4774.96293	-2.98
130	90985.04293	-4.26
140	3154786.722	-5.8
150	6.59128E-06	5.88
160	7567806.242	-6.18
170	39716411.74	-6.9
180	477496293	-7.98
190	1.2852E+14	-13.41
200	7.74408E+12	-12.19
210	40641525808	-9.91
220	32282711.45	-6.81
230	4774962930	-8.98
240	81090504868	-10.21
250	2.8772E+17	-16.76
260	1.04465E+12	-11.32
270	40641525808	-9.91
280	3303467240	-8.82
290	4.56005E+11	-10.96
300	1.09388E+13	-12.34
310	9.5273E+15	-15.28
320	1.09388E+18	-17.34
330	6.59128E+18	-18.12
340	3.22827E+19	-18.81
350	8.49122E+19	-19.23
360	3.62218E+18	-17.86
370	3.0128E+17	-16.78
380	4.77496E+15	-14.98
390	6.59128E+15	-15.12
400	4.35482E+17	-16.94
410	6.59128E+16	-16.12
420	8.49122E+15	-15.23
430	4.56005E+15	-14.96
440	3.45915E+15	-14.84
450	7.92447E+16	-16.2

SNR profile for 5.15 GHZ

freq=5.35GHz		
distance in cm	ber	snr
0	4.0642	0.09
10	2.2854	0.34
20	1.618	0.49
30	0.3972	1.1
40	0.0032	3.2
50	0.0063	2.9
60	0.0008	3.8
70	0.0001	4.9
80	0.3972	1.1
90	7.924466	-0.2
100	39.716412	-0.9
110	644.12478	-2.11
120	868.90041	-2.24
130	7744.0831	-3.19
140	48861.861	-3.99
150	13151.34	-3.42
160	10208.69	-3.31
170	40641.526	-3.91
180	165565.56	-4.52
190	477496.29	-4.98
200	644124.78	-5.11
210	1172114.4	-5.37
220	3012797.9	-5.78
230	5116465	-6.01
240	6744814.4	-6.13
250	810905.05	-5.21
260	77440.831	-4.19
270	4774962.9	-5.98
280	26851590	-6.73
290	140919147	-7.45
300	477496293	-7.98
310	8109050.5	-6.21
320	1409191.5	-5.45
330	102086897	-7.31
340	477496293	-7.98
350	810905049	-8.21
360	1.409E+09	-8.45
370	1.227E+10	-9.39
380	6.011E+10	-10.08
390	1.476E+10	-9.47
400	4.886E+11	-10.99
410	8.491E+10	-10.23
420	6.441E+11	-11.11
430	1.069E+12	-11.33
440	1.227E+12	-11.39
450	1.51E+13	-12.48

SNR profile for 5.35 GHZ

Here, we observed that whenever we stood directly below the antenna, or the horizontal span was zero, we saw that the BER decreased, and SNR improved drastically. As the horizontal span increased, we observed a dip in SNR, but then there was a spike after a certain distance, which denoted the presence of the beam's main lobe in that direction towards the point we were measuring. Finding the angle of inclination, θ from $\tan\theta = \frac{horizontal\ length}{vertical\ height}$ gave us following results:

Frequency (GHz)	5.15	5.35	5.725	5.825
Peak detected at (cm)	50	70	330	350
Actual (θ in degrees)	14.74	20.22	60.07	61.5
Theoretical (θ in degrees)	22	32	51	52

Experimental Results of LWA (Directly Used)

freq=5.725GHz		
distance in cm	ber	snr
0	244889.4097	-4.69
10	147560.4613	-4.47
20	114543.3826	-4.36
30	388123.5583	-4.89
40	644124.7758	-5.11
50	587448.7775	-5.07
60	810905.0487	-5.21
70	477496.293	-4.98
80	511646.4961	-5.01
90	659128.3693	-5.12
100	1255943.216	-5.4
110	6591283.693	-6.12
120	33804148.77	-6.83
130	12559432.16	-6.4
140	338041487.7	-7.83
150	3881235.583	-5.89
160	6441247.758	-6.11
170	2393150.462	-5.68
180	104464.8065	-4.32
190	30127.9793	-3.78
200	6591.283693	-3.12
210	2338.675706	-2.67
220	435.481795	-1.94
230	106.8981045	-1.33
240	81.09050487	-1.21
250	23.38675706	-0.67
260	3.792887875	0.12
270	0.064412478	1.89
280	0.003082975	3.21
290	0.001377114	3.56
300	0.000308298	4.21
310	0.000379289	4.12
320	0.000523564	3.98
330	0.00020369	4.39
340	0.000388124	4.11
350	0.00053576	3.97
360	0.008297935	2.78
370	0.161796828	1.49
380	0.615134385	0.91
390	1.19941646	0.62
400	5.116464961	-0.01
410	128.5197891	-1.41
420	21328.97594	-3.63
430	477496.293	-4.98
440	15099758.6	-6.48
450	177406694.6	-7.55

SNR profile for 5.15 GHZ

freq=5.825GHz		
distance in cm	ber	snr
0	52356.43	-4.02
10	65912.84	-4.12
20	810905	-5.21
30	15451.48	-3.49
40	33804.15	-3.83
50	169.4221	-1.53
60	2233.418	-2.65
70	6591.284	-3.12
80	14756.05	-3.47
90	47749.63	-3.98
100	810905	-5.21
110	338041.5	-4.83
120	739.5542	-2.17
130	4158.819	-2.92
140	477.4963	-1.98
150	14756.05	-3.47
160	2084347	-5.62
170	868.9004	-2.24
180	477.4963	-1.98
190	185.7676	-1.57
200	122.7354	-1.39
210	84.91218	-1.23
220	64.41248	-1.11
230	43.54818	-0.94
240	19.00947	-0.58
250	14.09191	-0.45
260	6.744814	-0.13
270	3.459155	0.16
280	0.95273	0.72
290	0.053576	1.97
300	0.08689	1.76
310	0.037929	2.12
320	0.007568	2.82
330	0.003881	3.11
340	0.000587	3.93
350	0.000371	4.13
360	0.000674	3.87
370	0.000477	4.02
380	0.000489	4.01
390	0.000615	3.91
400	0.037929	2.12
410	0.134577	1.57
420	0.345915	1.16
430	0.233868	1.33
440	0.011194	2.65
450	3.792888	0.12

SNR profile for 5.35 GHZ

Vertical Distance between Transmitter and Receiver = 190 cm				
Freq in GHz	5.15	5.35	5.725	5.825
Beam Width in m	70	80	100	130
Beam Width in degree	20.23512	22.84524	27.77262	34.39778

The following output was compiled and presented as the result of the entire developmental engineering project, which revolved around analysing the antenna and finding its quality and beamwidth for its use in RF applications for UAV communication.

The provided beamwidth measurements at different frequencies offer crucial information about the directional characteristics of the antenna. At 5.15 GHz, the beamwidth was 70 meters, corresponding to an angular beamwidth of approximately 20.24 degrees. Similarly, the beamwidth increased with frequency, reaching 130 meters at 5.825 GHz, with an angular beamwidth of approximately 34.40 degrees.

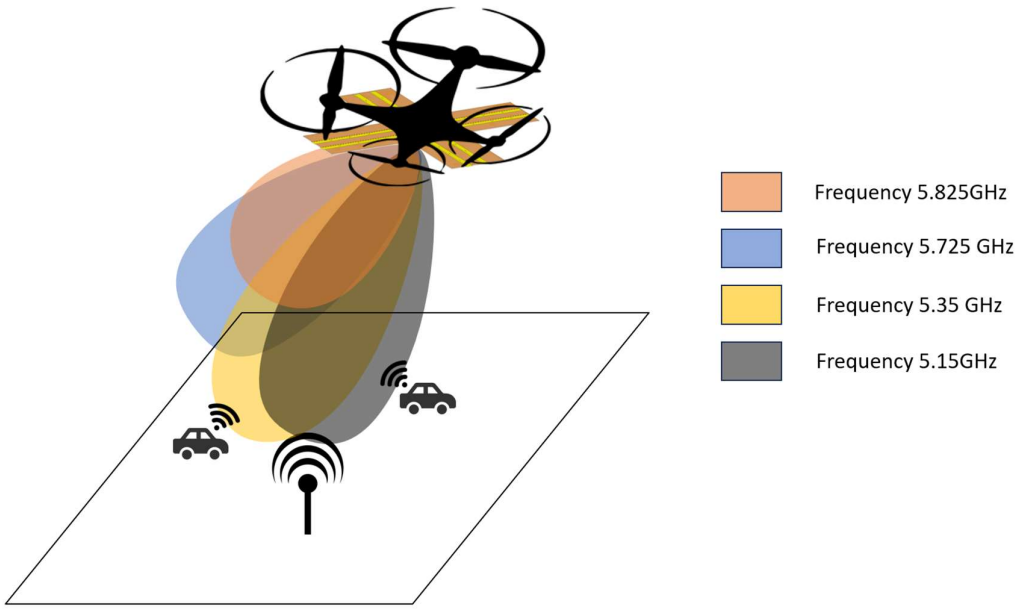
These results highlight the antenna's performance in terms of coverage area and directionality across different frequency bands. The wider beamwidth observed at higher frequencies indicates a broader coverage area but potentially reduced directional accuracy. Overall, the combination of experimental findings and analytical insights obtained throughout this project contributes to a deeper understanding of drone communication range enhancement. These insights can inform future research directions and practical implementations, paving the way for more capable and reliable UAV systems in various applications.

CONCLUSION

Throughout this project, we embarked on a systematic exploration of communication range enhancement for drones. We began by thoroughly understanding the underlying principles of communication systems and delving into modulation, transmission, demodulation, and synchronisation. In our experimental setup, we employed various tools and techniques to analyse and enhance the communication range of our specified antenna. Leveraging the capabilities of ADALM Pluto SDR and GNU Radio, we conducted a series of tests and optimisations to improve signal quality, reliability, and range.

Our approach involved modulating the transmitted signals using QPSK modulation, transmitting them through the specified antenna, and demodulating them at the receiver. We also implemented synchronisation mechanisms to ensure proper alignment of the received signals with the symbol timing and carrier phase.

The results of our experimentation provided valuable insights into the performance of the communication system. We could assess the effectiveness of our enhancements and optimisations by analysing metrics such as signal-to-noise ratio (SNR), bit error rate (BER), and beamwidth**.**



Analysis of the Final Objective: Beam Patterns and Quality Over Different Frequencies

REFERENCES

PYSDR: https://pysdr.org/

SDR Architecture: https://www.youtube.com/watch?v=2cMVKM5KwYk

GNU Radio:

1. https://wiki.gnuradio.org/index.php?title=What_Is_GNU_Radiohttps://wiki.gnuradio.org/index.php?title=Signal_Data_Types
2. https://wiki.gnuradio.org/index.php?title=Converting_Data_Types
3. https://wiki.gnuradio.org/index.php?title=QPSK_Mod_and_Demod
4. https://wiki.gnuradio.org/index.php?title=Variables_in_Flowgraphs
5. https://wiki.gnuradio.org/index.php?title=What_Is_GNU_Radiohttps://wiki.gnuradio.org/index.php?title=Signal_Data_Types