TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

B-11-BAS-2018/23

Development of an Unmanned Aerial System for Radio Frequency Source Localization

By:

Keshav Poudel (075AER018)

Prabhav Regmi (075AER026)

Sandesh Parajuli (075AER037)

A PROJECT TO THE DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN AEROSPACE ENGINEERING

DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING
LALITPUR, NEPAL

MARCH, 2023

COPYRIGHT

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a project report entitled **"Development of an Unmanned Aerial System for Radio Frequency Source Localization"** submitted by **Keshav Poudel, Prabhav Regmi and Sandesh Parajuli** in partial fulfillment of the requirements for the degree of Bachelor of Aerospace Engineering.

Supervisor: **Arun Bikram Thapa**, Assistant Professor
Department of Mechanical and Aerospace Engineering
Institute of Engineering, Pulchowk Campus

External Examiner: **Vishal Paudel**, Senior Operations Engineer, OEP Division Operations Department
Nepal Airlines Cooperation

Committee Chairperson: **Dr. Surya Prasad Adhikari**, Associate Professor
Department of Mechanical and Aerospace Engineering
Institute of Engineering, Pulchowk Campus

DATE:

ABSTRACT

Radio-based direction finding is an old concept for manned aircraft where a pilot manually flies towards the directed bearing. However, it is a relatively new field of study for unmanned aerial systems like quad-rotors especially motivated by the desire of making the homing process autonomous. In this study, the received signal strength method to localize the radio source has been applied and tested. The proposed method employs a UAS equipped with a radio frequency (RF) receiver to get RSSI readings from the RF source. The obtained RSSI measurements at three receiving locations are converted to approximate distances from the transmitter using a log-distance path loss model, and the localization algorithm based on trilateration is run onboard the microcomputer to localize the radio transmitter. Through simulations and experiments, the performance of the suggested method is assessed. The verification mission employing the suggested technique localized the RF source with a distance error of 15.8 meters and an angle error of 8.62 degrees, at best, whereas the actual UAS-based flight localized it to the distance accuracy of 25 meters, and the angle accuracy of 8.32 degrees at a test domain of 100 meters radius. As the measured signal strength values are highly noisy, with the standard deviation of the Gaussian distributed random noise variable reaching as high as 5, Kalman filter is designed for the off-board application to reduce the effect of such noises during the calibration phase. It is observed to provide smoother filtering characteristics than Simple Moving Average and Exponential Moving Average filters which seemed to overfit the noises. Finally, two approaches: range of distance, and moving towards/away, are presented to consider uncertainties in distance approximations, and it is recommended to build a system that works reliably within the distance confidence interval below 86 $\%$ to achieve the localization accuracy of at least 30 $\%$.

Keywords: *Unmanned Aerial System, Radio Frequency Localization, RSSI, Trilateration, Gaussian Noise, Kalman Filter*

ACKNOWLEDGEMENT

We would like to express our deepest gratitude towards Asst. Prof. Arun Bikram Thapa, our project supervisor, for his unwavering guidance, support, and encouragement during the course of this year-long project. His indispensable presence has played a pivotal role in ensuring the seamless progression and success of our journey.

We are grateful to the Department of Mechanical and Aerospace Engineering, Institute of Engineering, Pulchowk Campus for granting us the valuable platform to embark on a collaborative endeavor. This undertaking has allowed us to implement the knowledge accumulated over the years as a major project for our fourth year. It has not only expanded our understanding but also provided us with a profound experience of teamwork, greatly enriching our educational journey.

Additionally, we extend our gratitude to Lucky Babu Jayswal for his assistance in the manual operation of the drone. Likewise, special thanks to Mikesh Paudel for dedicating his valuable time and service during the outdoor tests. Furthermore, we would like to express our appreciation to all our professors, mentors, friends, and family members who have offered direct and indirect support throughout this project. Their contributions have been instrumental in its successful completion.

Any kind of suggestion or criticism will be highly appreciated and acknowledged.

Authors:

Keshav Poudel (075AER018)

Prabhav Regmi (075AER026)

Sandesh Parajuli (075AER037)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACCRONYMS AND ABBREVIATIONS

AoA	Angle of Arrival
AWGN	Additive White Gaussian Noise
CAAN	Civil Aviation Authority of Nepal
CAD	Computer Aided Design
DHT	Humidity and Temperature
DoA	Direction of Arrival
EMA	Exponential Moving Average
ESC	Electronic Speed Controller
GPS	Global Positioning System
GUI	Graphical User Interface
IDE	Integrated Development Environment
Li-Po	Lithium-Polymer
LoRa	Long Range
NCAR	Nepalese Civil Airworthiness Requirements
RF	Radio Frequency
RPM	Revolution Per Minute
RSSI	Received Signal Strength Indicator
SD	Secure Digital
SMA	Simple Moving Average
SPI	Serial Peripheral Interface
TDoA	Time Difference of Arrival
ToA	Time of Arrival
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UHF	Ultra-High Frequency
VLOS	Visual Line Of Sight
WSN	Wireless Sensor Network

CHAPTER ONE: INTRODUCTION

1.1 Background

The effectiveness of Unmanned Aerial Systems has led to their applications in diverse fields. Notably, they have gained significant traction in search and rescue operations due to their ability to efficiently access remote and hard-to-reach locations that would otherwise be inaccessible or unidentified to immediate human responders. The identification of a location represents a crucial prerequisite for conducting search and rescue missions. GPS is the industry standard when it comes to location findings and it is widely being used in the 21^{st} century. GPS systems, however, are highly power-intrusive systems, and in case of a power outage, access to GPS signals will be cut off. Mobile phones are the most prevalent means of obtaining GPS location information; however, their battery life is insufficient for extended usage. Radio-based localization proves advantageous in this regard since the radio signal source can operate for several months with minimal power consumption. Consequently, the work presented here focuses on localizing the target by utilizing Radio Frequency (RF) localization technique, under the condition that the subject in search is equipped with an RF-transmitting device. The technique involves equipping the subject with an RF source, the strength of which is then utilized by the localization algorithm to determine the location of the subject. Localization can be achieved through various methods, including range-based and range-free schemes [1]. The approach taken in this study employs a range-based technique that utilizes Received Signal Strength Indication (RSSI) measurements.

The process of localization relies on filtered RSSI values owing to its acceptable degree of accuracy and relatively lower complexity compared to other techniques. Furthermore, the RSSI-based method is regarded as the most cost-effective means of localization in outdoor settings. However, in indoor conditions, the performance of this technique is somewhat limited [2]. Nonetheless, the use of RSSI measurements remains a popular and effective means of achieving localization in various settings. The practice of target localization has been extensively studied and applied using fixed reference nodes [2]. However, the current research seeks to extend the same methodology

to Unmanned Aerial Systems (UAS) for enhanced target-tracking capabilities.

1.2 Problem statement

GPS, being power intrusive, can be extremely unreliable in cases when electric power sources are scarce. This can result in total loss of connection, which poses a significant risk if the subject is a beneficiary of a search and rescue mission. Therefore, a technology to track the subject with an uninterrupted connection is desirable.

1.3 Objectives

1.3.1 Main objective

The main objective of the project is to develop a quadrotor-based Unmanned Aerial System that will detect the radio frequency signals emitted by the transmitter, and travel to the transmitter location by using a target localization algorithm to a minimum accuracy of 30 meters with respect to the transmitter and 20 degrees with respect to the origin (mission start point) within the test domain of 100 meters radius.

1.3.2 Specific objectives

- To fabricate a drone and incorporate various sensors required for target localization.
- To develop custom transmitter and receiver modules.
- To develop a MATLAB program to simulate the target localization algorithm.
- To study and apply different techniques to filter out the noisy RSSI signals.
- To present methods to quantify uncertainty considerations in distance estimations.

1.4 Applications

Some potential applications of RF source localization using UAS include:

- Search and rescue operations: By detecting the signal from a communication device of the missing person, drones with RF sensors may rapidly and effectively look for them in disaster zones or other hazardous situations.
- Emergency response for hazardous incidents: Drones equipped with RF sensors can be used to quickly find and identify the source of a fire, gas leak, or other hazardous situation. Emergency responders may be better able to prioritize and coordinate their efforts with the aid of this information.
- Security and surveillance: Drones with RF sensors can be used to monitor and track suspicious activity, as well as detect and locate illicit Wi-Fi transmissions in off-limits locations.
- Environmental monitoring and research: RF sensors aboard drones can be used for environmental monitoring and research, including mapping and measuring RF signals in far-off places that are difficult to access. This data can be crucial for determining how wireless communication affects the environment.
- Wildlife tracking and monitoring: By detecting the signal from tracking devices connected to animals, RF sensors on drones can be utilized for wildlife tracking and monitoring. The movements and behavior of wildlife can be better understood with the help of this knowledge.

1.5 Features

RF source localization using drones refers to the process of determining the location of a RF-emitting source from a drone-borne device. Some of the key features of RF source localization using drones include:

- Real-time data collection: By using RF sensors on drones, researchers may quickly and precisely gather information on the position and strength of radio waves.
- Increased mobility: Compared to conventional ground-based techniques, drones may fly to difficult-to-reach places, giving them more access for RF source localization.
- Remote monitoring: Remote monitoring of RF sources is made possible by localizing RF sources utilizing drones, which enables real-time data gathering and processing from a secure distance.
- Cost-effectiveness: Compared to conventional approaches that demand specialized equipment and personnel, drones offer a cost-effective solution for RF source localization.
- Greater accuracy and dependability: Drones equipped with high-precision sensors and cutting-edge algorithms enable RF source localization to be more accurate and reliable.
- Ease of deployment and use: Drones are simple to deploy and operate, and they require little training to be used effectively for RF source localization.

1.6 Feasibility analysis

1.6.1 Economic Feasibility

The project incorporates assembling a quadrotor and applying various electronic components for transmitting and receiving purposes. The cost of UAS components and electronics accessories is the major chunk of the budget. However, UAS components (drone frame, microcontroller, motors, ESCs, battery, etc.) are available at the Department of Mechanical and Aerospace Engineering. Remaining expenses are met by the members themselves. Further, application for funds from institutions is also considered. The expenses of fabrication and application of this drone are not outstandingly expensive. Therefore, the project is economically feasible.

1.6.2 Technical Feasibility

The design and manufacturing requirements for the completion of the drone is met completely by the resources available in the Department of Mechanical and Aerospace Engineering. Other technical requirements are not unprecedented and have been done individually but not in combination. The theory required to build is not new and had been incorporated in the syllabus for Bachelor in Aerospace Engineering. For these reasons the project is technically feasible

1.6.3 Operational feasibility

The localization of the RF source doesn't require huge pristine space. The required space is easily met by the Campus area. The calibration of UHF source can be done easily by the space requirement. The major operational hurdle for the drone is the permission to fly. According to the rules of the air document, drones under 2kg do not require permission to fly in an unrestricted area. As the manufactured drone which localizes the RF source will be under 2 kg, the project is operationally feasible.

1.7 System Requirements

For drone-based RF source localization, the following software and hardware requirements are necessary:

1.7.1 Software Requirements

1. Drone flight control software (ArduPilot)
2. Microcontroller (Arduino IDE)
3. RF signal processing software (e.g., MATLAB or Python with relevant libraries)
4. Geolocation algorithms (e.g., trilateration) to determine the location of the RF source

5. Mapping software (e.g., Google Maps) to visualize the location of the RF source in a geographical context

6. Data storing and analyzing software (Cool term and Excel)

1.7.2 Hardware Requirements

- Drone with GPS and altitude control capabilities and stable hovering characteristics
- RF receiver (with antennas) capable of detecting signals from the RF source
- A computer or data processor to collect, process, and analyze the RF data
- Communication system (e.g., Wi-Fi or telemetry) to transmit data from the drone to the computer

1.8 Scope

- Possible targets may not have GPS access due to some reasons. In such cases the target identification is done by the locals and by helicopter which is expensive and lengthy process.
- There are many endangered wild animals in Nepal so this project could help to track their location and study their behaviors.
- It can also be used in disaster rescue operation.
- It can be useful to track lost and distressed trekkers in remote trekking locations.

Figure 1.1: Pictorial Realization of Project Objective

CHAPTER TWO: LITERATURE REVIEW

2.1 Localization modality

Radio frequency (RF) based localization has a rich history, dating back to World War II when it was used to locate soldiers in emergency situations [3]. Earlier approaches to sensing the RF signal were based on creating maps of the signal strength over the area by using mobile robots [4]. This approach was later used on an unmanned aerial vehicle for localization of radio-tagged sturgeon assuming that a stationary radio source is transmitting with constant power [5]. However, this method is highly inefficient as the vehicle has to move over the entire area to generate a signal strength map. Talking about the strength map, researchers have also dabbled with infrared (IR) based localization techniques, primarily for indoor purposes, where a badge emitting an IR signal every 10 seconds was worn by a person and sensors placed at various locations localized the target [4]. However, IR being sensitive to sunlight, has limited application in outdoor localization. RF networks, on the other hand, have greater reliability, range, scalability, and maintainability.

Different RF localization techniques have been studied including angle of arrival (AOA) [6], time of arrival (TOA) [7], time difference of arrival (TDOA) [8], and received signal strength indication (RSSI) [9, 10, 11]. The accuracy of these methods depends on the target source being under the line of sight of the localizing system [3].

AOA-based localization is based on estimating the bearing by rotating a directional antenna to measure signal strengths at different directions [12]. However, it necessitates additional hardware increasing the overall weight and complexity of the system. With the advancement of highly maneuverable UAVs, the need for an actuator mechanism to rotate the antenna can be eliminated. However, studies have reported the rotation time for a single measurement to be 40s [13] and 45s [5] thereby consuming a significant portion of battery life, and increasing the localization time. ToA and TDoA have been reported to provide more accurate location estimates at shorter time [14]. However, they are very sensitive to timing errors and their implementation requires sophisticated

hardware. RSSI is the cheapest and simplest technique for localization. While it can suffer from inaccuracy due to the reflection and attenuation of signals when impacted by the environment, the issue can be mitigated by implementing a more detailed physical model of the environment [15]. The combinations of RSSI, AOA, TOA, and TDOA are presented in different studies [16, 17]. Combining TDOA and TOA techniques has been found to offer better accuracy than RSSI, but it comes with a substantially higher cost [16]. Therefore, the RSSI-based technique is seen as a good candidate to achieve the desired localization accuracy at a minimum cost.

Recent RSSI-based works have focused on the localization of a target by using multiple UAVs simultaneously [18]. Using multiple UAVs can be expensive, so it becomes desirable to use a single UAV. For this, RSSI-based outdoor localization techniques have been developed. An RSSI technique based on the clustering method along with Singular Value Decomposition has been reported to provide an accuracy of 7m [19].

RSSI values are obtained from radio transceivers. LoRa (Long Range) technology is often used in low power wide area networks (LPWANs) to support IoT applications. It is a low-power unit with a range of up to 15 km and is inexpensive, making it highly suitable for long-range indoor and outdoor localization [20]. LoRa is being incorporated in UAS to improve the network coverage to communicate during a disaster [21]. However, its use in UAV-based target localization is yet to be seen. In the work presented ahead, an RF signal source is localized using RSSI measurements from the LoRa transceivers for an outdoor environment. The mission test flights were carried on an open-controlled environment within the visual line of sight as limited by CAAN regulation [22].

2.2 Filtering

The received signal strength values are affected by multipath fading [23], antenna orientation [24], ground effects [25], and other temporal and spatial changes in the environment [26] including weather [27], human activity [28, 29], etc. Likewise, strong sensor mobility at the measuring nodes also hampers the communication channel which results in errors while receiving the signals and increases packet delivery latency [30].

Filtering of the signals to reduce the effects of perturbations and fading in RSSI-based technique is proposed in [31]. The paper suggests that filtering improves the ranging accuracy by eliminating the noises. To reduce the noises and improve the overall accuracy of the RSSI-based localization, various filtering techniques are analyzed and compared in [32]. Low computational filters like Simple Moving Average (SMA), Exponential Moving Average (EMA), Moving Median, and Moving Mode are selected for the comparison. The results indicated that SMA and EMA provide the best overall filtering performance. In [33], the Kalman filter is also used to mitigate the noises present in the received signal. Its efficacy has been observed to be comparable to that of a simple moving average filter when signal shadowing is absent. However, in scenarios with higher levels of signal shadowing, the Kalman filter has demonstrated a notably reduced mean error. The results suggest that the Kalman filter exhibits enhanced robustness and suitability for implementation in dynamically changing environmental conditions. In a LoRaWAN protocol, the Kalman filter is reported to improve the positioning accuracy by a whooping 22 % [34]. The study reports no significant effect in accuracy while using Fast Fourier Transform (FFT) and Particle filtering methods.

CHAPTER THREE: METHODOLOGY

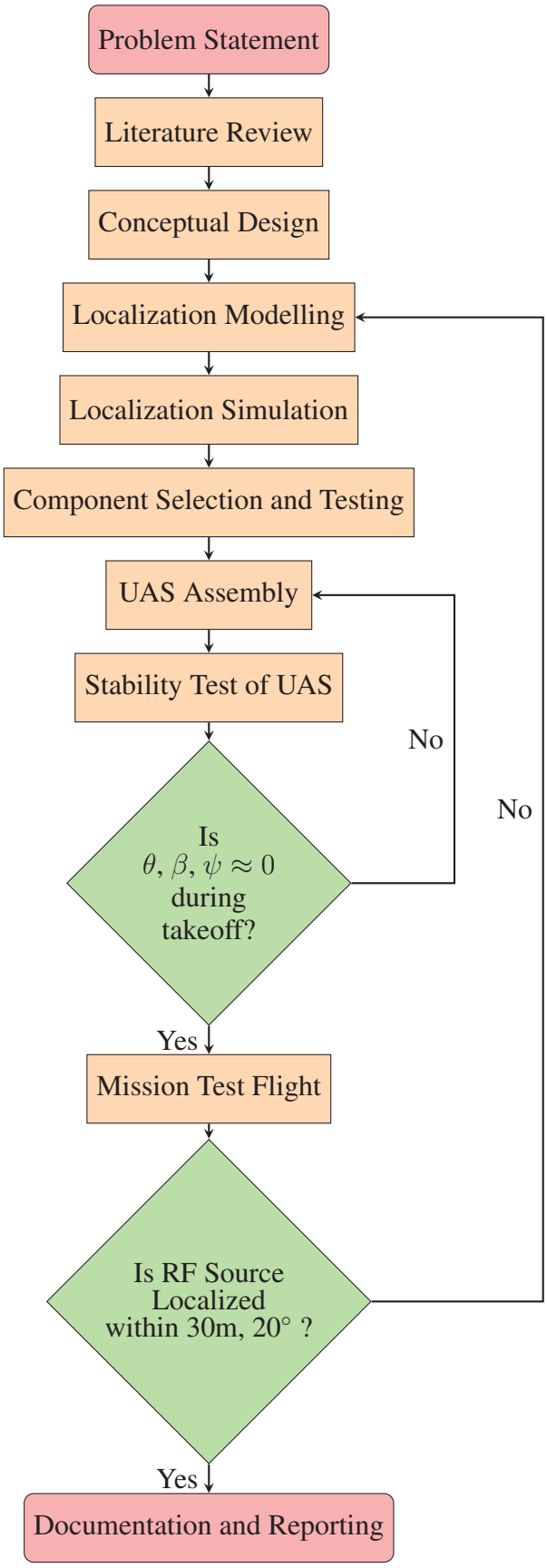

Figure 3.1: Methodology Flowchart

3.1 Localization Modeling

The objective of this work is to use a UAS to localize a stationary RF transmitter to best possible accuracy. It is aspired to attain reliable positioning accuracy with a single UAS by discarding inaccurate RSSI measurements and identifying the most accurate ones. RSSI measurements in UASs vary due to various elements such as structures, trees, wind, speed of the vehicle, etc. As a result, a localization technique based on several sample measurements is required.

As the position of the UAS in space will be known, a model is necessary to obtain the distance of the UAS from the transmitter to further proceed with localization. As we will be receiving radio frequency signals, a distance conversion model based on the Received Signal Strength Indicator (RSSI) values is selected.

3.1.1 RSSI Based Localization

RSSI values are the measure of the power of received radio signals measured in dBm. In RSSI-based localization, a receiver will measure the RSSI values of the signal transmitted by the transmitter. When RSSI values are measured at three known locations, those values can be translated into the distance values based on some signal propagation model. Then, we will have a system of three non-linear equations, which are linearized and solved to obtain the transmitter coordinates. The flowchart for RSSI based localization is shown in figure 3.2.

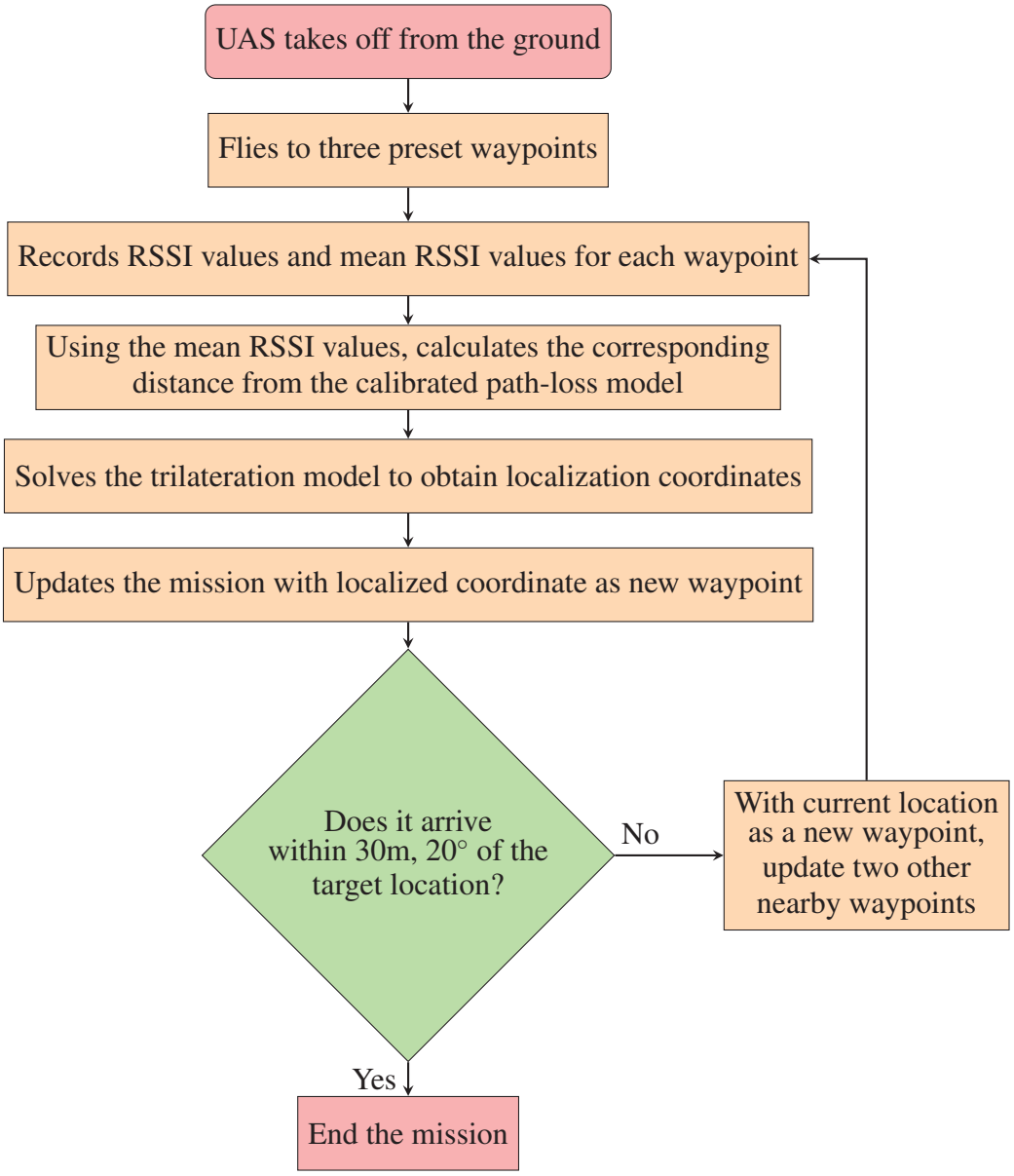

Figure 3.2: RSSI based Localization

3.1.2 Distance Estimation Model

RSSI values (measured in dBm) give the power of the received signals as the signal propagates from transmitter to receiver. As the signal power decays with distance, RSSI values can be used to infer the proximity between the receiver and transmitter by approximating the distance between them. One of the most widely used distance estimation model for radio wave propagation is the log-normal model [35] given by

equation as:

$$PL(d) = PL_{d_0} + 10nlog_{10}\frac{d}{d_0} + X_\sigma \tag{3.1}$$

Here, $PL(d)$ is the path loss at a distance d, d_0 is the reference distance, PL_{d_0} is the experimentally measured average path loss at d_0, and n is the path loss index. $X(\sigma)$ is the Gaussian distributed random variable with mean, μ = 0, and some finite value of the standard deviation, σ.

Equation 3.1 in terms of RSSI can be written as:

$$RSSI = RSSI_0 - 10nlog_{10}\frac{d}{d_0} - X_\sigma \tag{3.2}$$

The effect of $X(\sigma)$ can be removed to some extent by filtering. If we neglect $X(\sigma)$, and take the reference distance d_0 equal to 1 meters then, equation 3.2 becomes

$$RSSI = RSSI_0 - 10nlog_{10}(d) \tag{3.3}$$

$RSSI0$ can be experimentally determined and the value of n can be computed by minimizing the sum of squares.

$$f = \sum \left(RSSI - RSSI_0 + 10\mathrm{n}\log_{10}(d)\right)^2$$

Here, f is the function to minimize. Then, the distance between the receiver and the transmitter can be approximated by

$$d = 10^{\frac{\mathrm{RSSI}_0 - RSSI}{10n}} \tag{3.4}$$

3.1.3 Trilateration Model

The trilateration model aims to calculate the localized transmitter coordinates based on three known measurement coordinates and the approximated distances of those coordinates from the transmitter.

$$(x - x_1)^2 + (y - y_1)^2 = d_1^2 \tag{3.5}$$

$$(x - x_2)^2 + (y - y_2)^2 = d_2^2 \tag{3.6}$$

$$(x - x_3)^2 + (y - y_3)^2 = d_3^2 \tag{3.7}$$

$$Ax = B \tag{3.8}$$

$$x = A^{-1}B \tag{3.9}$$

where,

$$A = \begin{bmatrix} 2(x_1 - x_2) & 2(y_1 - y_2) \\ 2(x_1 - x_3) & 2(y_1 - y_3) \end{bmatrix} \tag{3.10}$$

$$B = \begin{bmatrix} r_2^2 - r_1^2 + x_1^2 - x_2^2 + y_1^2 - y_2^2 \\ r_3^2 - r_1^2 + x_1^2 - x_3^2 + y_1^2 - y_3^2 \end{bmatrix} \tag{3.11}$$

$$x = \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.12}$$

It is based on simple matrix operations, and is basically converting a system of three non-linear circle equations into a linear form as in the equation 3.8 and solving for matrix, x, which contains the co-ordinates of the localized transmitter.

3.1.4 Conversion to Geographical Coordinates

Conversion of the localized cartesian coordinates into geographical coordinates is necessary in order to command the UAS to fly to the localized GPS location. Calculation of the latitude and longitude of the localized coordinate given the latitude and longitude of the reference point and the distances east and north of the reference point is given by:

$$latitude = latitude_{ref} + \frac{y}{earth's radius} * \frac{180}{\pi} \tag{3.13}$$

$$longitude = longitude_{ref} + \frac{y}{earth's radius} * \frac{180}{\pi} * \frac{1}{cos(latitude_{ref})} \tag{3.14}$$

The formula is based on the principles of spherical trigonometry. It assumes a spherical Earth, which is an approximation that works well for small distances. It is commonly used in geodesy and navigation, and is often implemented in software libraries and programming languages. The cartesian coordinate system is fixed in such a way that the

X-axis and Y-axis are directed towards east and north respectively, such that when localized cartesian point (x,y) is obtained with respect to the reference point (0,0), values of x and y can be directly implemented in the coordinate transformation formula.

3.2 Filtering

There is a substantial presence of noise in the signal strength values, and it is highly desirable to remove those noise effects. Three different filtering techniques are studied, namely, the Kalman filter, Simple Moving Average (SMA) filter, and the Exponential Moving Average (EMA) filter. In this work, these filtering techniques are applied only in offline cases, primarily during calibration to get the smooth curve of the data points which are used to fit the log-distance path loss model. Application of these filters will remove noise from the RSSI measurements, and provide stability and robustness to the overall localization scheme.

3.2.1 Simple Moving Average Filter

The SMA is a kind of filter that averages a predetermined subset of data called a ”window” and slides this window over a time series to produce a new sequence of data points. The weight of each data point in the window is the same, thus each one contributes equally to the average. The window's length can be changed, and while expanding it produces a smoother output, it also causes the filtered data to lag more.
The SMA filter needs some data points to calculate the average and produce an output. At the beginning of the time series, there are not enough data points to do this. Therefore, the output is either undefined or set to a default value. Once enough data points are available to fill the window, the filter can calculate the average of the data subset and give an output sequence.
The SMA filter is a useful tool for removing high-frequency noise while preserving low-frequency trends in noisy data. It's easy to use and doesn't require a lot of processing power, making it ideal for real-time applications. As each data point in RSSI filtering is given equal weight, it is possible to apply a SMA filter with a window size

of 'n'. Nevertheless, the SMA filter's output sequence is not established at the start of the time series until there are adequate numbers of data points to compute an average. After there are enough data points, the filter may calculate the average and offer an output sequence. When a past average is known, the SMA filter's key advantage is that it is straightforward to compute for subsequent time occurrences.

Expression to calculate SMA at time t,

$$SMA_t = \frac{1}{N} \sum_{i=0}^{N-1} x_{t-i} \tag{3.15}$$

where, SMA_t is the SMA at time t, calculated as the sum of the previous N RSSI measurements (x_{t-i} where i ranges from 0 to $N-1$) divided by N. The value of N determines the number of past measurements included in the moving average calculation, with a higher value of N resulting in a smoother output with less noise.

3.2.2 Exponentially Moving Average Filter

EMA is another kind of filter in which older data points are given less weight, whereas newer data points are given greater weight. Unlike other filters, which need a certain number of data points to produce a result, this filter starts creating an output right away. A parameter known as the "smoothing parameter (α)" determines the weight given to each data item. A higher value of α gives more weight to recent measurements and less weight to older measurements, resulting in a faster response to changes in the RSSI signal. The value of α is typically a value between 0 and 1, and is chosen based on the desired filtering effect for the specific application. The filter may add noise and lag to the output if the smoothing parameter is set too high. If it is set too low, the output could be excessively smooth and insensitive to input signal variations. The EMA filter is helpful when a smooth output is sought, but if it is not properly tuned, it might cause lag in the output.

Expression to calculate EMA at time t,

$$EMA_t = \alpha x_t + (1 - \alpha) EMA_{t-1} \tag{3.16}$$

Here, EMA_t is the EMA at time t, calculated as a weighted average of the current RSSI measurement x_t and the previous EMA value EMA_{t-1}, where the weight given to x_t

is determined by the smoothing parameter α.

3.2.3 Kalman Filter

The Kalman filter is an effective way of removing the noise of the signal and obtaining a smoothed estimate of the true signal. Kalman filter combines two sets of information as it functions: predictions of the model of the system, and the measurements of the system. To filter out the RSSI values, we need to supply the Kalman filter with the mathematical model that gives the expected behavior of the RSSI values as a function of distance. Then, by combining the predicted and measured values, Kalman filter produces a smoothed estimate of the RSSI values which are less affected by noises, thereby allowing us to make more accurate estimation of the distance between the transmitter and receiver.

The working algorithm of the Kalman filter is summarized below:
i. Initialize the state vector 'x' and the covariance matrix 'P'
ii. Define the system model matrix 'A', the measurement matrix 'C', and the process noise covariance 'Q'.
iii. Define the measurement noise covariance 'R'.
iv. Initialize the filtered signal vector.
v. For each measurement in the input signal, 'noisy-signal', do the following:

a. Predict the next state using the system model: x = A * x.
b. Predict the next covariance matrix using the system model and the process noise covariance: P = A * P * A' + Q.
c. Compute the Kalman gain K using the current covariance matrix, the measurement matrix, and the measurement noise covariance: K = P * C' / (C * P * C' + R).
d. Update the state estimate using the Kalman gain and the difference between the measurement and the predicted measurement: x = x + K * (noisy-signal(i) - C * x).
e. Update the covariance matrix using the Kalman gain and the measurement matrix: P = (eye(size(P)) - K * C) * P.
f. Calculate the filtered signal using the current state estimate and the measurement

matrix: filtered-signal(i) = C * x.

vi. Return the filtered signal.

The selection of appropriate values for the covariance matrices, P, Q, and R is critical for the design of the Kalman filter. The estimation of these values can be done based on the statistical properties of the system and measurement noises, and/or trial and error method where the values are adjusted based on desired filter performance. For our purposes, we have selected R as equal to the variance of the noisy signal. P and Q are both 2 by 2 diagonal matrices with diagonal elements in P being 100 and Q being 0.0000001.

3.3 Considering Uncertainty

Although the Gaussian noise in the signal, as denoted by X_σ in equation 3.1 can be reduced by the use of filters, the signal data will still not be completely free of unexpected variations in RSSI values caused by several unknown and unmodeled factors. This will inherently bring uncertainty in distance approximation. Therefore, it will be unrealistic to state a single distance value based on the mean of RSSI measurements.
Two different approaches can be used to consider uncertainty: providing range of distance as output [36], and/or providing moving towards or away information as output.

3.3.1 Range of Distance

The standard deviation in the filtered RSSI signals can be used to output a predicted range of distances based on 68-95-99.7 rule. If X is a normally distributed random variable with mean μ and standard deviation σ, then

$$\begin{aligned}\text{Prob}(\mu - \sigma \leq X \leq \sigma + \mu) &\approx 0.6826\\ \text{Prob}(\mu - 2\sigma \leq X \leq 2\sigma + \mu) &\approx 0.9544\\ \text{Prob}(\mu - 3\sigma \leq X \leq 3\sigma + \mu) &\approx 0.9974\end{aligned}$$

If X is a random variable that denotes the filtered RSSI value, and sigma be the standard deviation of the still existing noises, then there is a 68% probability of the target being

in the range $[f(X+\sigma), f(X-\sigma)]$ where $f(x)$ is the distance function for the fitted path loss model.

$$f(x) = 10^{\frac{\text{RSSI0} - \text{x}}{10n}} \tag{3.17}$$

It is assumed that the standard deviation of the RSSI values at every distance is constant, and equal to the standard deviation of the noise values.

3.3.2 Moving Towards/Away

The idea of implementing the moving towards/away information involves continuously reading RSSI values, translating them into distances from the target at two consecutive time stamps, and computing the instantaneous velocity. The instantaneous velocity is given by equation 3.18.

$$V_{ins,i+1} = \frac{d_{i+1} - d_i}{t_{i+1} - t_i} \tag{3.18}$$

As the instantaneous velocity is calculated based on the information supplied by the filtered instantaneous RSSI measurements, the velocity information might appear noisy. So, it will be necessary to use a smoothing filter for instantaneous velocity calculations.

RSSI $\rightarrow$ Filter1 $\rightarrow$ Distance Model $\rightarrow$ Instantaneous Velocity $\rightarrow$ Smoothing Filter

Negative values of instantaneous velocity mean that the system is moving towards the transmitter, and the positive values mean that it is moving away.

3.4 UAS Development

3.4.1 Software Development

A MATLAB-based application is created to select drone components by taking input for the drone's weight, number of motors, and drone range required. The application is designed to provide the necessary propeller, motor, ESC and battery capacity for drone building. A database of motor static thrust ratings is kept, providing the necessary information on the propeller's pitch and KV rating. Using the input weight of the drone,

the thrust per motor is calculated, and then compared with the database's thrust values to select the appropriate motor and propeller. The data base is created with the motors and propeller that are available in Nepal.

3.4.2 CAD Geometry

The drone proposed to be assembled is designed in CAD software CATIA V5 to understand the overall design and architecture.

Figure 3.3: CAD model of the drone

3.4.3 UAS Hardware

An X-type drone configuration consisting of four 980kv DC brushless motors is used. To vary the speed, four electronic speed controllers are present. Pixhawk 4 and Arduino Uno are used as the flight controller and the flight computer respectively. Other peripheral components and sensors include PM07, SX1278 LoRa transceiver, etc. To power all the components a 3-cell Li-Po battery is used. 10*4.5 propellers are used during the mission. GPS module and telemetry are connected to Pixhawk to provide the necessary data to control and perform the mission.

Figure 3.4: Developed UAS (Quad)

To perform the localization mission, antenna and sensors are attached to the drone. After the assembly of the drone, control, stability as well as performance of the drone are tested and the necessary modifications are done accordingly.

The hardware that was used for the drone and localization are:

1. **Motor**:

   The motor is the fundamental component of a drone, required to rotate the propellers to produce the necessary thrust. The number of anticlockwise rotating motors must match with the clockwise rotating motor in order to equalize the force produced by the propeller. The KV rating of the motor determines the size and thrust it can produce. KV rating is the RPM constant of a motor, indicating the number of revolutions a motor makes after the application of 1 volt without any load attached. When a high KV motor is paired with a large propeller, it will spin the propeller with high rpm which requires more torque, causing heating and premature failure.

   Four 980kv brushless motors are used in the drone.

2. **Propellers**:

   Propellers are vital components of the drone responsible for the generation of the required lift force. The performance of a propeller depends on its size and pitch.

The pitch of a propeller is defined as the traveling distance per single revolution of the propeller. Lower pitch values account for more torque, while higher pitch propellers account for greater lift force. Smaller propellers with a high pitch are better suited for quick and high-maneuvering drones, while larger propellers with lower pitches are suited for carrying heavier loads.

A 10*4.5 (inch) propeller is attached to each motor. The diagonally opposite propellers rotate in a similar direction, with two rotating in a clockwise direction and the other two in an anticlockwise direction.

3. **ESC**:

   Electronic Speed Controllers (ESCs) are components of drones capable of adjusting the speed of electric motors. A signal from the flight controller causes the ESC to change the voltage provided to the motor, which in turn changes the RPM value and allows the drone to maneuver. ESCs for drones are typically rated for the maximum current drawn by the motor and have a refresh rate in hertz, indicating how many times per second the ESC can change the motor speed. For quadcopters, high refresh rate ESCs are used. An ESC with a 10A rating will draw a maximum continuous current of 10A, and the ESC current rating is usually chosen higher than the maximum current to prevent burnout. ESCs also have a burst rating, which is the maximum amount of current an ESC can handle for a short period without damage.

   30 A ESC is incorporated in the drone hardware to control the speed of the drone motors.

4. **Battery**:

   The battery is crucial for powering all the electrical and electronic components of the drone. Typically, Li-Po batteries are used, with a nominal voltage of 3.7V. Cells are kept together in series to increase the voltage. The number of cells in the battery is known via the number followed by the letter 'S'; for example, 3S signifies a 3-cell battery. The main parameters for battery selection are battery capacity and C rating. Battery capacity is the measured current in milli Ampere-hours (mAh) that it can supply for a unit period of time. For example, a 5000mAh battery would be able to supply 5A current for an hour (C-rating:1). For a higher

value of C rating, say 5, the same battery can provide 25 A current for 12 minutes. Therefore, the C rating of the battery increases the maximum safe current drawn by the battery, allowing the same battery to be used to lift heavier payloads.

$$\text{Maximum Safe Current draw } (mA) = \text{ Battery capacity } * C - \text{ rating}$$

5. **Pixhawk**:

   Pixhawk 4 is the primary flight controller used in the quadrotor to control it during the localization mission. The quadrotor setup uses Pixhawk connected to the M8N GPS module, telemetry, and Arduino Uno.

6. **Arduino Uno**:

   Arduino Uno is a flight computer that is used to receive RSSI values, process them, carry out the necessary computations, and send the output to the ground station. Various sensors are incorporated with the Arduino to measure the RSSI from the transmitter, and to measure the humidity and temperature of the environment. The data from various sensors are stored in the SD card via the SD card module of Arduino. Pixhawk is used to power the Arduino via the telemetry port. The sensors that are used with the Arduino are:

   - **DHT11 sensor:**

     The DHT11 sensor is a digital temperature and humidity sensor that is commonly used with Arduino. It measures the temperature and humidity of the surrounding air. The DHT11 sensor has a single-wire digital interface, making it easy to connect with Arduino boards.

     The DHT11 sensor is widely used in various applications such as weather stations, air quality monitoring systems, etc. It is a cost-effective solution for measuring temperature and humidity and is suitable for both indoor and outdoor applications. The use of the DHT11 sensor with RSSI measurement can help in observing the impact of temperature and humidity on radio signal strength.

   - **Micro SD card module:**

     The SD card module is a device used for storing data observed by the Arduino. It is a compact and easy-to-use device that, based on the size of the

SD card, can store large amounts of data. The SD card module is commonly used in data logging applications, where data is collected over a period and stored on the SD card for post-analysis.

The SD card module works by interfacing with the SPI interface of the Arduino board. It has a slot for inserting the SD card, and the data is stored on the SD card using the FAT file system. The SD card module can be used in various applications such as weather monitoring systems, GPS tracking systems, and data logging systems.

7. **Telemetry radio:**

   Telemetry radio is a wireless communication device used for transmitting flight data from the drone to the ground station. It is commonly used in drone applications for real-time monitoring of the drone's flight status. The telemetry radio is connected to the Pixhawk flight controller using a 915 MHz RF module.

   The telemetry radio provides real-time data such as altitude, speed, GPS position, and battery status to the ground station. It also allows for sending commands to the drone, such as changing the flight mode or setting the waypoint. The telemetry radio is a critical component of drone applications, as it enables safe and efficient operation of the drone.

8. **M8N GPS module:**

   The M8N GPS module is a high-performance GPS receiver that is commonly used in drone applications. It provides accurate GPS positioning data to the drone, which is essential for accurate flight control and navigation. The M8N GPS module has a built-in compass, which aids in localization algorithms.

   The M8N GPS module works by receiving signals from multiple GPS satellites and calculating the position of the drone using the received data. The GPS data is transmitted to the drone's flight controller, where it is used for navigation and control. The M8N GPS module is a critical component of drone applications, as it enables a reliable and safe operation of the drone.

3.4.4 Hardware for 433MHz band

3.4.4.1 LoRa-ESP 8266 Transmitter:

A prototype transmitting device was made with SX1278 LoRa module, using ESP 8266 as microcontroller. When the switch is turned on, the device starts transmitting “Help! Help!!” packets on the frequency band of 433 MHz, which is a license-free RF band for Asia.

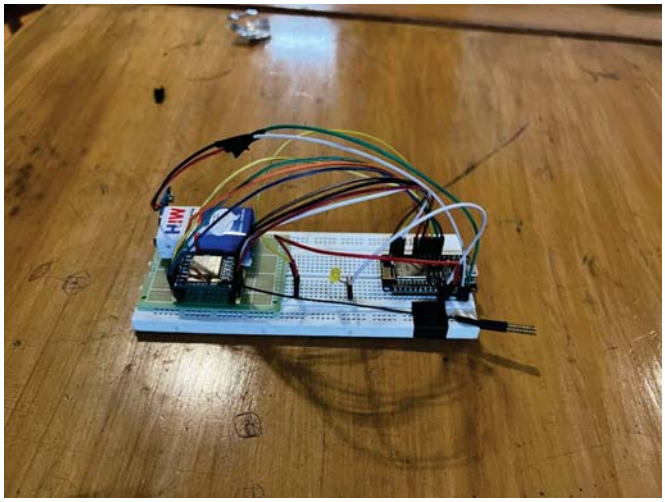

Figure 3.5: LoRa transmitter

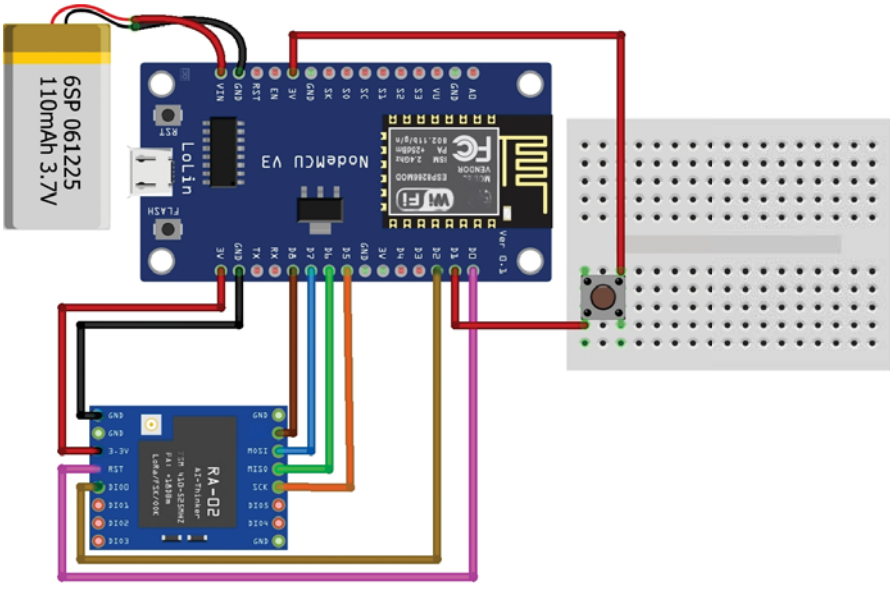

Figure 3.6: Circuit diagram for transmitter

The circuit diagram shown in the figure 3.6 depicts the connection between a SX1278 LoRa module, an ESP8266 module, and a battery.

The LoRa module is connected to the ESP8266 module via serial communication, and the ESP8266 module is powered by a battery. The LoRa module is used for long-range wireless communication and operates at low power. The ESP8266 module is a Wi-Fi enabled microcontroller with built-in Wi-Fi capabilities. The ESP8266 is connected to the battery via a switch, which is used to turn the device on and off. The battery provides power to the ESP8266 module, allowing it to operate independently. This makes the system portable and suitable for use in remote locations.

3.4.4.2 LoRa-Arduino Receiver:

It is a prototype receiving device with Arduino as microcontroller. It is programmed to continuously receive the transmitted signals along with their RSSI, after syncing with the transmitter. Other peripheral sensors and devices like DHT 11 Temperature and Humidity sensor, and micro SD card module are also interfaced with Arduino.

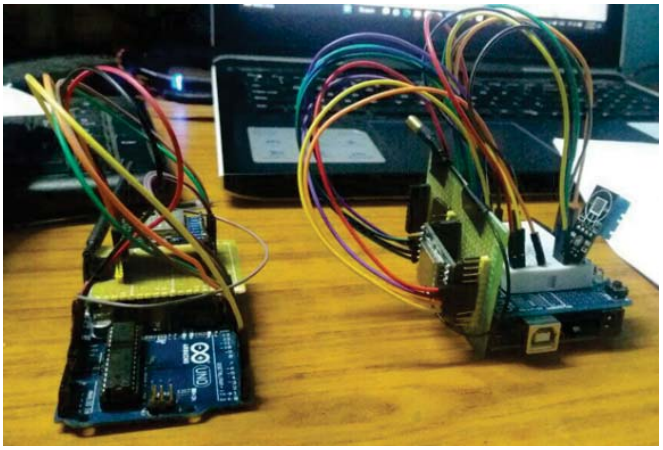

Figure 3.7: LoRa receivers

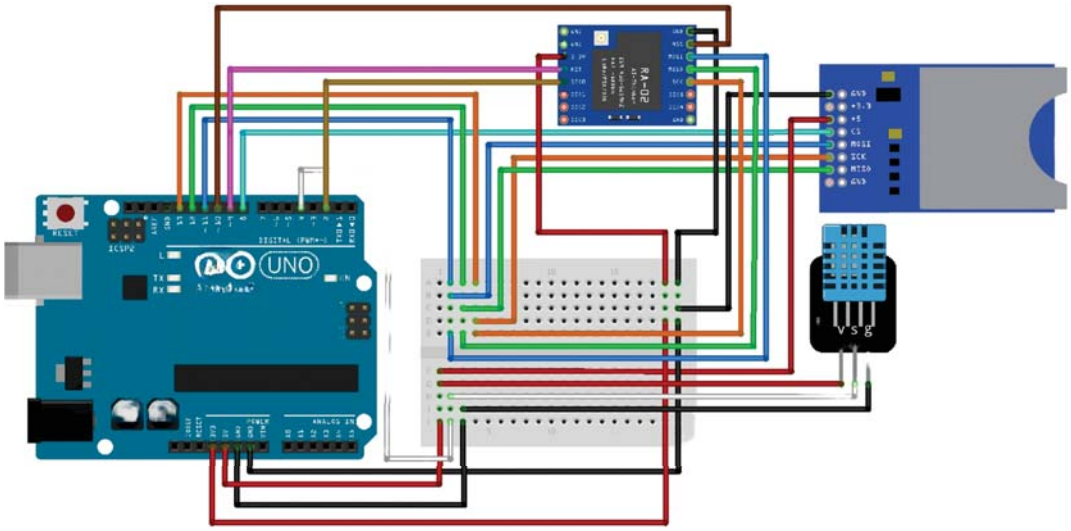

Figure 3.8: Circuit diagram for Arduino-based LoRa receiver

The circuit diagram shown in figure 3.8 depicts the connection between an Arduino, a LoRa module, a DHT11 temperature and humidity sensor, and an SD card module.

The LoRa module is connected to the Arduino through serial communication. The DHT11 sensor is connected to one of the digital pins of the Arduino for data transfer. The SD card module is connected to the Arduino through the SPI interface for data transfer.

Since we are using two modules (LoRa and SD card module) that communicate with Arduino using Serial Peripheral Interface (SPI), they utilize the same MOSI, MISO, and Serial Clock pins. However, different chip-select pins should be used.

The LoRa module is used to receive the transmitter-transmitted signals. The DHT11 sensor is used to read the temperature and humidity data. The SD card module is used to store the temperature and humidity data along with the RSSI of the received signals for later analysis. The Arduino writes the data to the SD card in a text file format, which can be easily read and analyzed using a computer.

3.4.5 Hardware for 2.4 GHz Band

3.4.5.1 Wi-Fi router:

A Wi-Fi router was used as the transmitter for high-frequency low range and indoor testing purposes. It transmits radio signals at 2.4 GHz

Figure 3.9: Wi-Fi router

3.4.5.2 ESP 8266-Arduino Receiver:

The ESP 8266 comes with a built-in Wi-Fi module. It is programmed to connect with the router network, and read the RSSI values. Those RSSI values are then serially communicated to the Arduino. The RSSI data are stored in a separate SD card interfaced to Arduino through a micro-SD card adapter module.

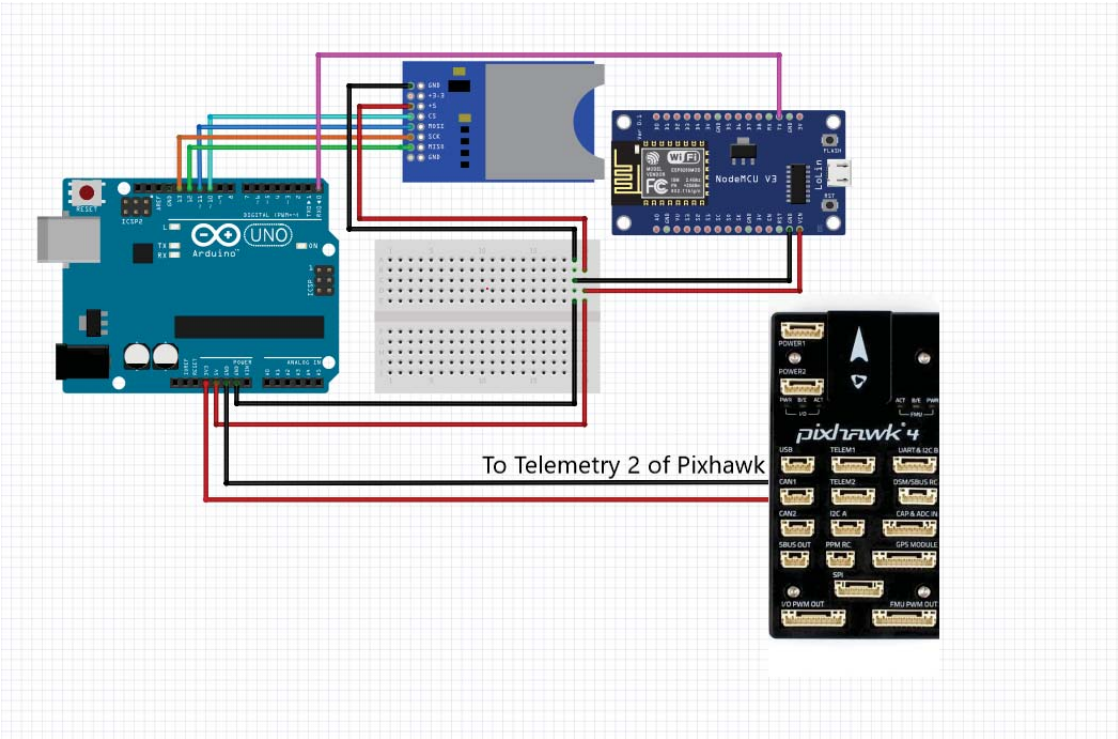

Figure 3.10: Circuit diagram for ESP-Arduino receiver

The circuit diagram in figure 3.10 shows the connection between an Arduino, a Pixhawk flight controller, an SD card module, and an ESP8266 Wi-Fi module.

In this circuit, the SD card module is connected to the Arduino through a SPI interface (MOSI, MISO, SCK, and CS pins). The ESP8266 is connected to the Arduino through a serial communication interface (Tx and Rx pins). The Arduino reads the Wi-Fi RSSI data from ESP8266 module using the serial port and stores them on the SD card.

3.5 Sensing modality

Measurement of RSSI values as the drone reaches a defined waypoint is the backbone of our localization model. The sensing modality consists of two different setups for the different frequency bands.

3.5.1 Sensing modality for 433Mhz

The sensing modality for 433Mhz radio signals consists of two modules:
1. First module is a SX1278 LoRa transceiver which is connected with the antenna and can measure RSSI of RF source. It communicates with the microcontroller module with Serial Peripheral Interface (SPI).
2. Second module is the microcontroller module, which is the Arduino. It reads the values of RSSI from RF module, processes them, and runs the localization calculation, and sends the localized coordinates to the ground station. The ground station then gives the coordinates to the Pixhawk through Ground station controlling software. The proposed sensing modality is as shown in figure 3.11.

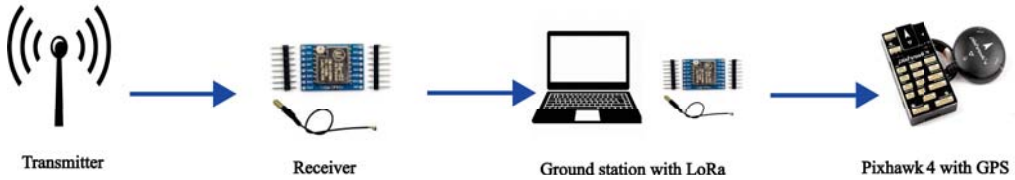

Figure 3.11: Sensing modality for 433Mhz

3.5.2 Sensing modality for 2.4GHz

The 2.4 GHz sensing module consists of an ESP 8266 Wi-Fi microchip capable of measuring RSSI values from the transmitter, processing them, and performing the localization calculation. It sends the localized coordinates to a server that can be remotely accessed from the ground station. The coordinates are then relayed to the Pixhawk via the ground station control software. The proposed sensing modality is as shown in figure 3.12. However, this modality is not actually implemented in outdoor flight test. Simple wireless sensor network-based localization was performed for 2.4 GHz frequency bandwidth.

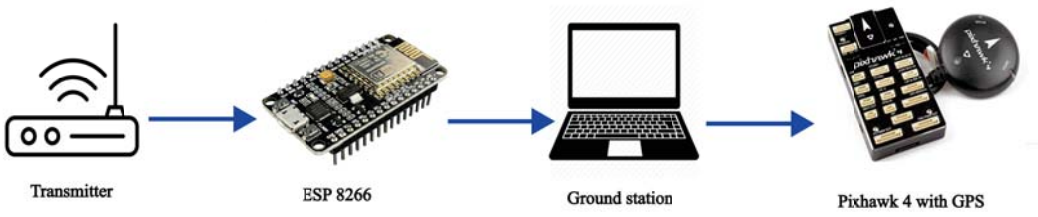

Figure 3.12: Sensing modality for 2.4GHz

3.6 Localization architecture

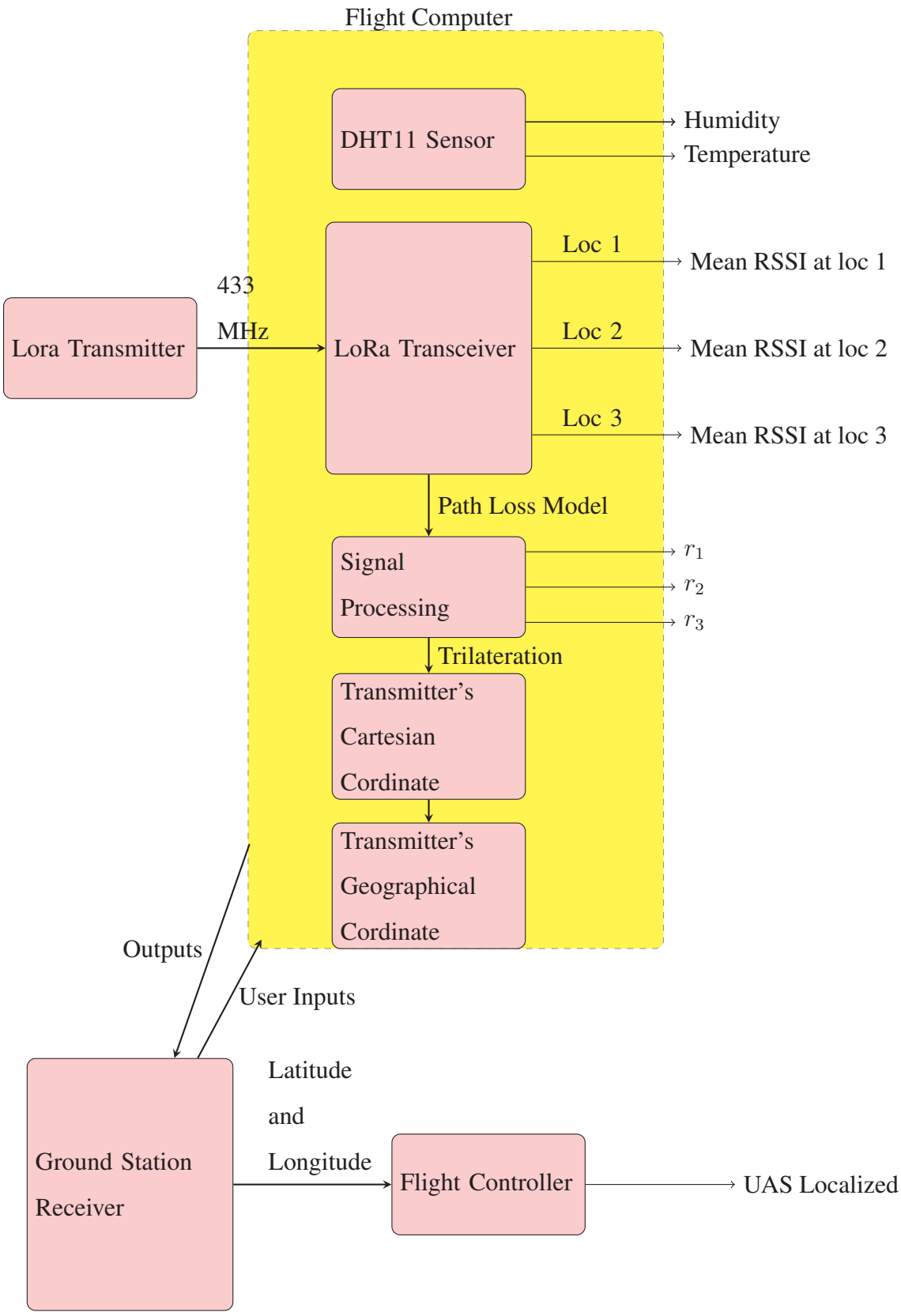

Figure 3.13: Localization Modelling

3.7 Localization Mission

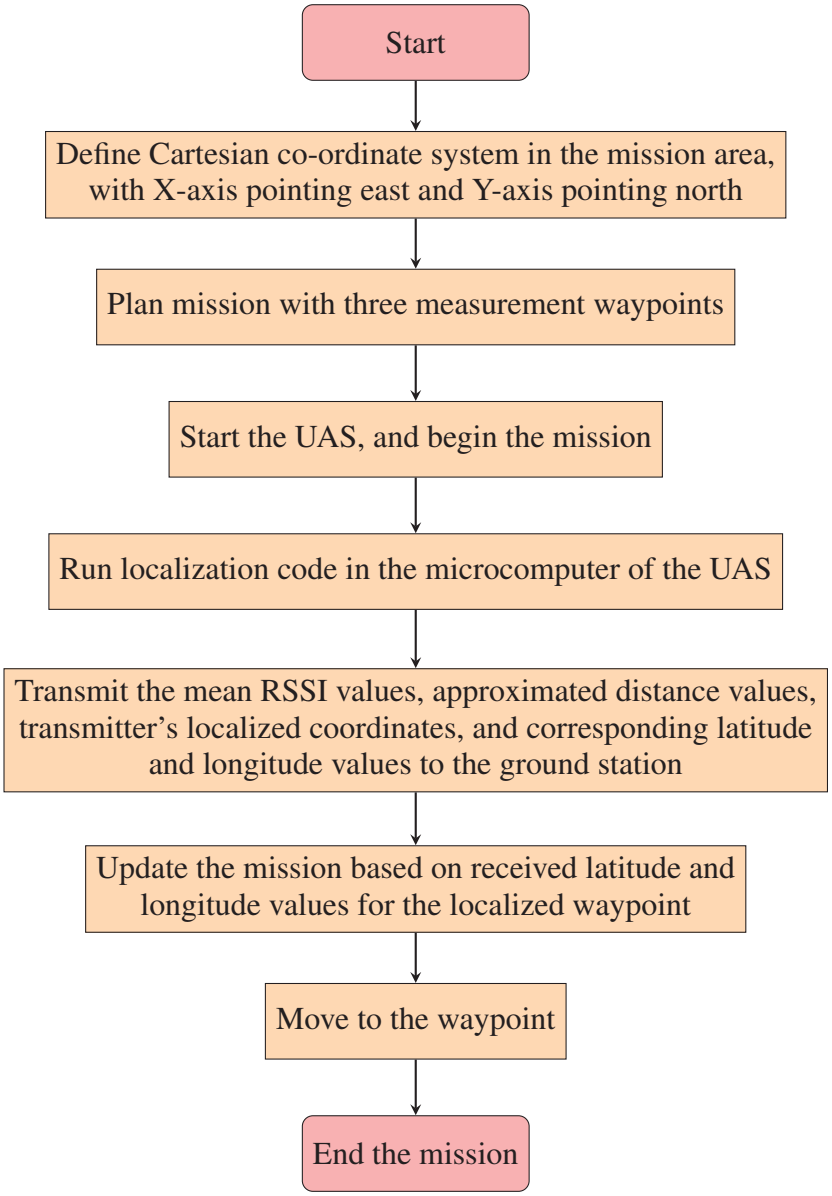

Figure 3.14: Flowchart for Localization Mission

CHAPTER FOUR: RESULTS AND DISCUSSION

4.1 MATLAB Simulation

The localization algorithm was first simulated in MATLAB before its experimental verification tests, and main mission implementation. Two different types of simulations were performed. First, a generalized case simulation was performed to understand the effects of Gaussian noise X_σ on localization accuracy. This was performed before conducting any experiments. Second, a simulation pertaining to a specific experimental localization case was performed. Here, the simulation model is fed with parameters (n, RSSI0 and σ) obtained from the characterization experiment. The specific case simulation is discussed in section 4.7.3.

Equation 3.2, rewritten below, describes the simulation model.

$$RSSI = RSSI0 - 10n\log_{10}\frac{d}{d_0} - X(\sigma)$$

The path-loss exponent, n, is an important parameter of the propagation model given by 3.2. It basically determines how fast the signal strength decreases with distance.

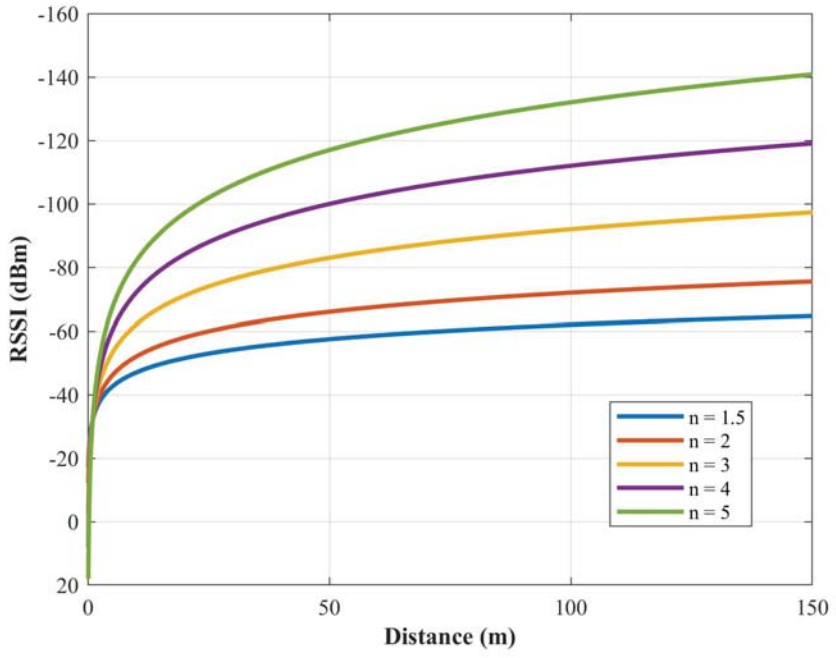

Figure 4.1: Variation of RSSI with distance

Figure 4.1 gives the plot of RSSI vs distance for different values of n (neglecting X_σ). As n increases, the RSSI vs distance curve becomes steeper. This means that the signal strength decreases more rapidly with distance, and the signal cannot travel as far as it would for a lower value of n. This has implications in RSSI-based ranging. If we want signal strength to vary noticeably for proper distancing, we are always limited by its range. We will see in the later sections that the value of n was obtained nearly equal to 5 for the Wi-Fi signal for the indoor case, and around 2 for the LoRa module in the outdoor case. Further, the test performed at Pepsicola area, as presented later, shows that even though the LoRa module can give the signal range in kilometers, RSSI to distance conversion at higher distances is quite unreliable and heavily affected by the environment.

The implication of X_σ in equation 3.2 is captured by a noise model known as Additive White Gaussian Noise (AWGN). It is based on Gaussian distributed random variable, and is used to mimic the random noises that occur in nature.

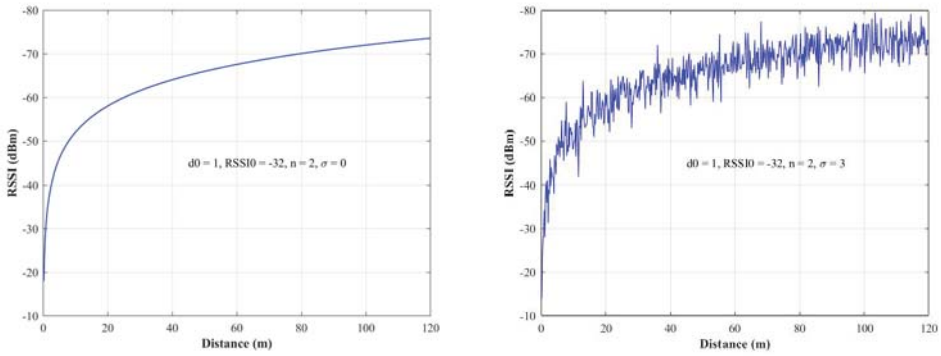

Figure 4.2: Simulation without AWGN on the left and with AWGN σ = 3 on the right

Figure 4.2 shows that the injection of AWGN model gives a more realistic picture of signal strength read at different distances (Compare with figure 4.32, which shows the actual measured RSSI values).

4.1.1 General Case Simulation

An omnidirectional transmitter located at (250,250) is modeled in a 2-dimensional space of 500 m by 500 m. Standard values are selected for simulation parameters, i.e., n = 2, RSSI0 = -32 dBm.

Capabilities of the simulation model:

- Computes the RSSI values at each grid point based on the path loss model, and associated parameters.
- Allows the user to input three different positions to measure corresponding RSSI values.
- Reads 50 RSSI values at each location, computes their mean, and translates into corresponding distance values.
- Solves the system of three non-linear equations to obtain the coordinates of the transmitting device.

The RSSI distribution over the space for varying values of the standard deviation of the Gaussian noise, and the corresponding localization result obtained through simulation is presented.

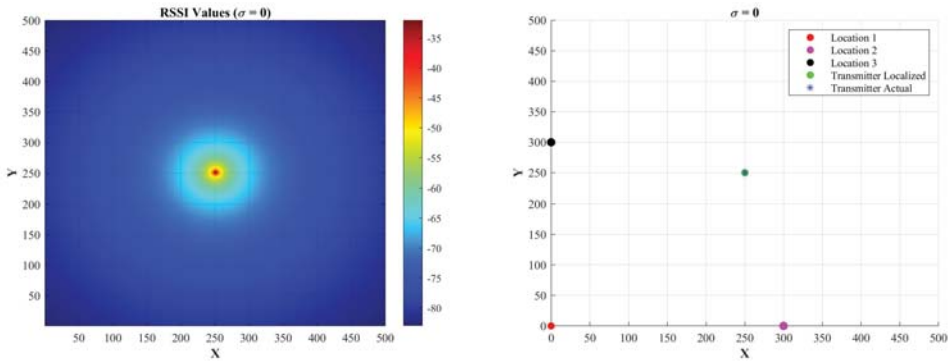

Figure 4.3: σ = 0 variation with RSSI on the left and Localization at σ = 0 on the right

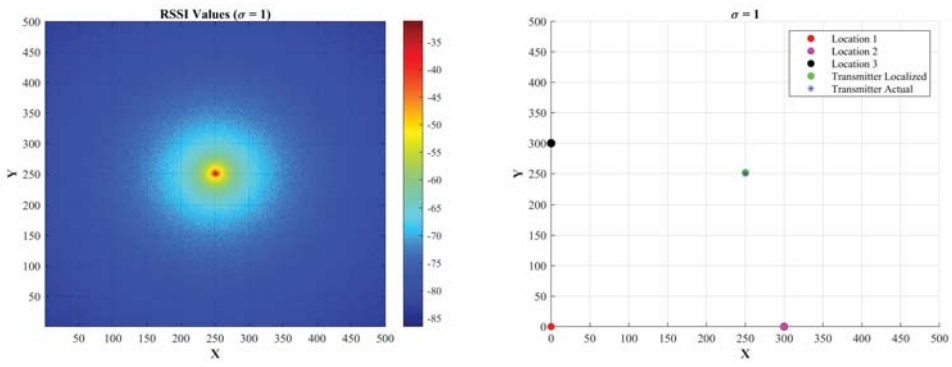

Figure 4.4: $\sigma = 1$ variation with RSSI on the left and Localization at $\sigma = 1$ on the right

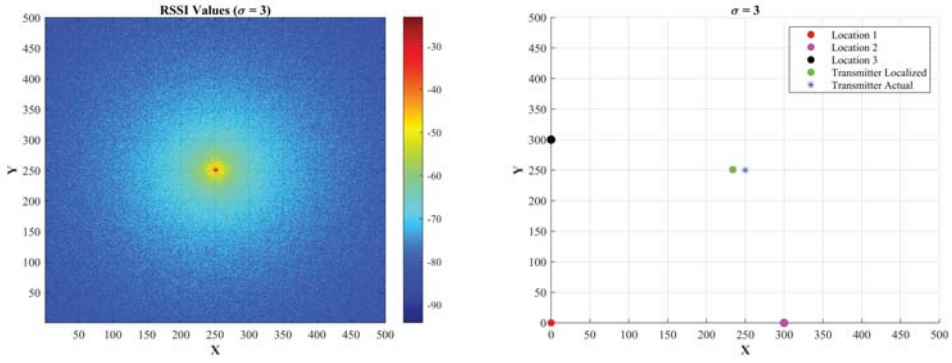

Figure 4.5: $\sigma = 3$ variation with RSSI on the left and Localization at $\sigma = 3$ on the right

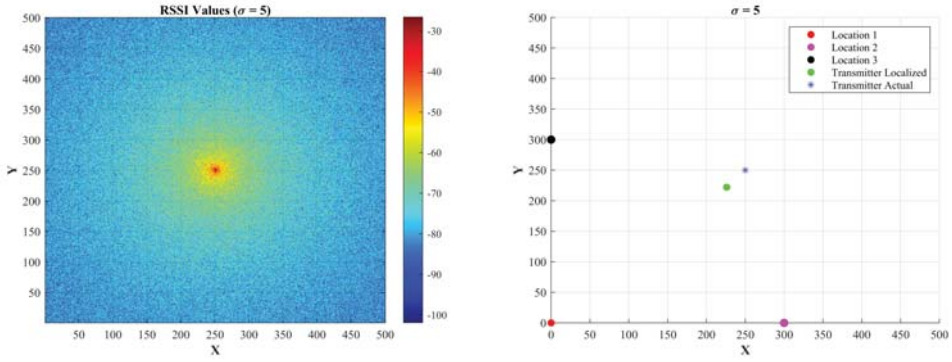

Figure 4.6: $\sigma = 5$ variation with RSSI on the left and Localization at $\sigma = 5$ on the right

Table 4.1: Variation of σ with distance error

σ	**Distance Error**
0	0
1	1.85 m
3	16.15 m
5	36.79 m

The arguably observed trend is that the localization error increases with the increase in σ. However, this is not strictly true given the randomness in AWGN values at all three locations. Moreover, the distance error for the same σ value can vary for different iterations. However, the higher value of σ brings a larger variation in signal values, possibly making the localization scheme unstable. Therefore, the engineering challenge is not just to perfectly calibrate for the environment, but also to reduce highly varying random noises. The problem with experimental implementation is that neither the environmental factors are perfectly calibrated, nor the random noises are avoided.

4.2 Localization setup

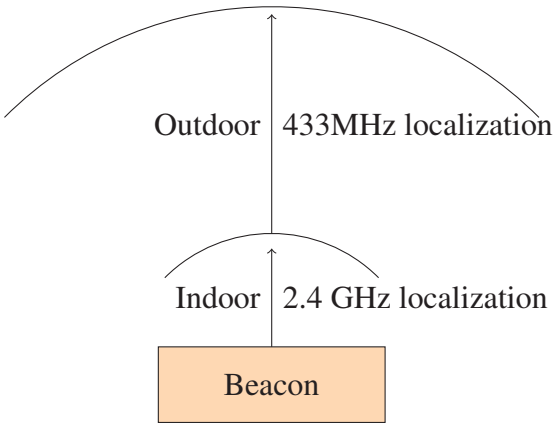

Figure 4.7: Localization setup

A 2.4 GHz WiFi transmitter is used for indoor localization, and a 433 MHz LoRa transmitter is used for outdoor purposes. A transmitting beacon is kept at a known location,

which is then compared with the localized position to determine the accuracy of the localization scheme. Before performing the actual flight-based tests, Wireless Sensor Network (WSN)-based tests were performed, where RSSI values are read at three different locations by the static sensors positioned there. As it was not practical to fly a quadrotor in the indoor environment, flight-based tests were only performed for the outdoor environment.

4.2.1 and 4.2.2 discuss the testing approaches and results for the WSN-based tests in indoor and outdoor environments respectively. Mobility-based tests are discussed in 4.6 and 4.7.

4.2.1 WSN-based indoor localization of 2.4 GHz RF transmitter

4.2.1.1 Coordinate system

The indoor localization was performed in an unobstructed room with a width of approximately 5 m, specifically at the basement of the Centre of Energy Studies at Pulchowk Campus. A 2.4 GHz WiFi router was used as the radio source. After calibrating the setup environment, measurements were taken from three pre-defined positions, and the mean RSSI values are calculated for each position. The coordinate system used for indoor localization is shown in figure 4.9.

Figure 4.8: Indoor localization space

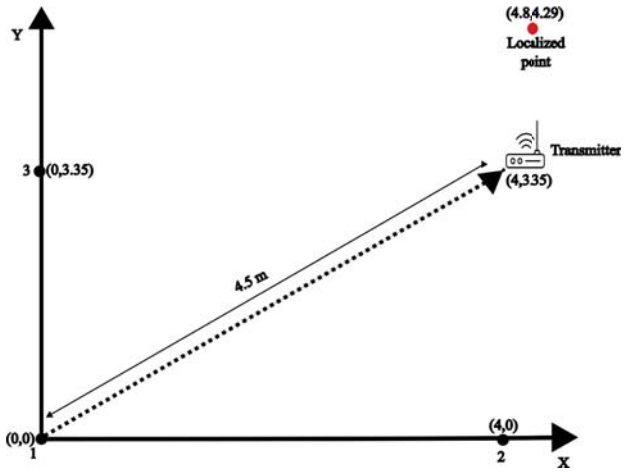

Figure 4.9: Coordinate system for indoor localization

4.2.1.2 Calibration

For calibration, signal strength values were measured at five different locations, namely at 1 m, 2 m, 3 m, 4 m and 4.5 m, from the transmitter. The corresponding mean RSSI values at these locations are tabulated in table 4.2.

Table 4.2: RSSI value at different distances for calibration

Distance(m)	**Mean RSSI (dBm)**
1	-38.22
2	-50.93
3	-60.08
4	-66.07
4.5	-71.8

With these data points, the log-distance path loss model is fitted. The value for the path-loss exponent, n, is obtained to be 4.7821.

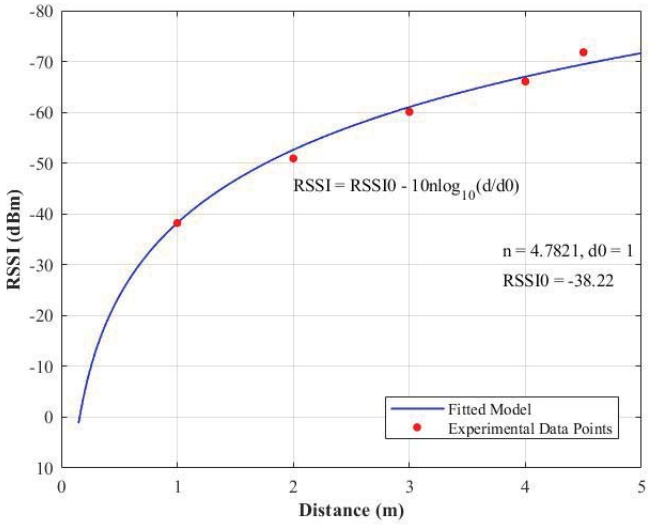

Figure 4.10: Indoor Calibration

4.2.1.3 Test and Results

The mean RSSI values at the three different pre-defined positions were recorded, and the distance associated with each RSSI value was evaluated from the calibrated model. Using the distance and known Cartesian coordinates, the coordinates of the location are obtained (4.8, 4.29). This brings us the distance error of 1.29 m, and the angular error of 1.8 degrees (as measured with origin as vertex).

Table 4.3: Measured RSSI at the coordinates

Coordinates	**Mean RSSI (dBm)**
(0,0)	-71.7
(4,0)	-48.6
(0,3.35)	-59.3

Table 4.4: The Localized Data of Indoor Localization

Test	**Localized Cartesian**		**Distance Error (m)**	**Angle Error (°)**
	X	Y		
1	4.8	4.29	1.2	1.8

Given the smaller testing area, the transmitter was initially expected to be localized with a distance error of less than 1 m. As seen in figure 4.25, the RSSI values at a particular location are slightly unstable. Therefore, it is believed that the other Wi-Fi signals of similar frequency (2.4 GHz) interfered with the signal in consideration, and caused error in distance estimation.

4.2.2 WSN-based outdoor localization of 433MHz RF transmitter

A 433MHz transmitter module made using ESP 8266 microcontroller and LoRa module is the object to localize. The receiver setup contains a LoRa module with Arduino Uno. The setup also incorporates a DHT 11 temperature and humidity sensor to measure the temperature and humidity at the location. Different tests were conducted at Pulchowk Campus cricket ground and Pepsicola region.

4.2.2.1 Localization in campus area:

A transmitter was placed in a fixed position of the Campus cricket ground and the receivers were scattered in multiple known locations. Using CoolTerm, the data obtained from the LoRa module were stored and processed. From the processed RSSI values, the unique path loss model was calibrated. After successful calibration, RSSI values were measured at three known locations, and the trilateration was carried out to localize the radio source.

Figure 4.11: LoRa test location in the campus ground

The results of calibration and trilateration are discussed below:

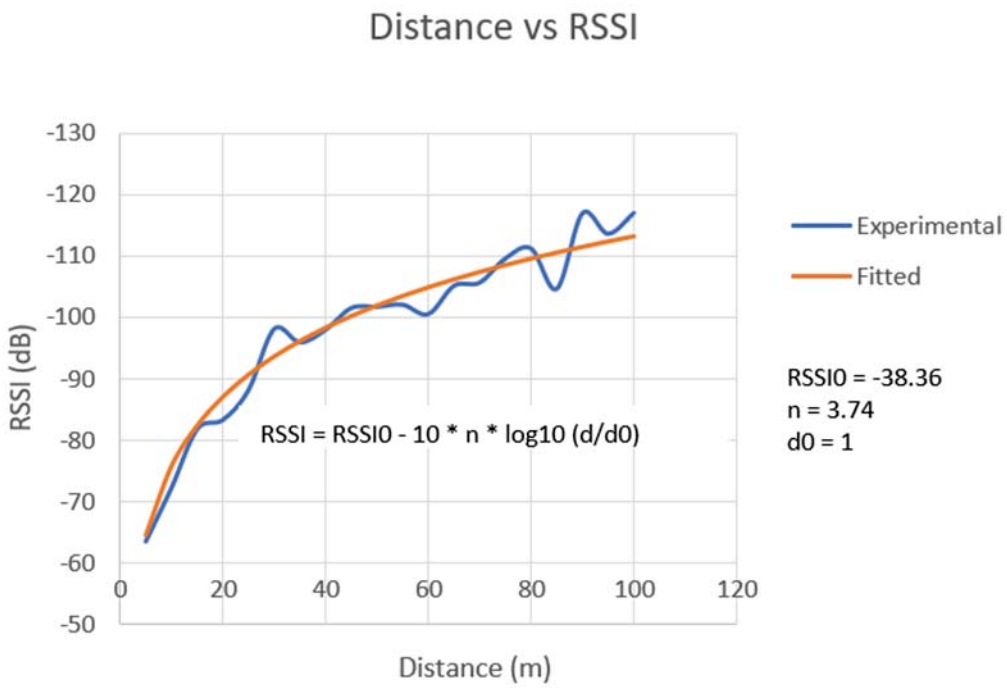

Figure 4.12: Graph of Distance vs RSSI

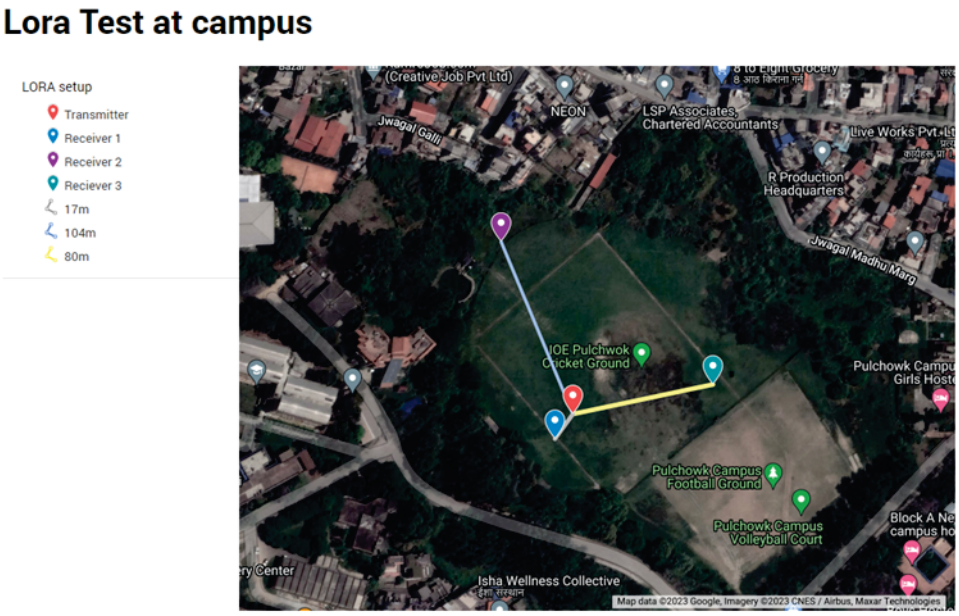

Figure 4.13: LoRa test result at Campus

Table 4.5: Transmitter GPS coordinates for LoRa test at campus

Transmitter	GPS coordinates	
	Latitude	Longitude
	27.68294	85.32195

Table 4.6: LoRa test result with RSSI and Distance

Receiver Node	GPS coordinates		Measured average RSSI value	Actual Distance	Calculated Distance
	Latitude	Longitude			
1	27.68282	85.32185	-88.42 dB	17 m	21.8 m
2	27.68381	85.32154	-115.52 dB	104 m	115.64 m
3	27.68309	85.32275	-106.45 dB	80 m	66.16 m

If we consider node 1 to be origin, and consider Y-axis to pass through the transmitter, then the coordinates of the nodes are:

N1(0,0) = (X1, Y1)

N2(-102,30) = (X2, Y2)

N3(41.5,68) = (X3, Y3)

(0, 17) = Actual transmitter coordinates

To localize the transmitter using trilateration, we iteratively solved the system of three non-linear equations using Newton-Raphson method, and obtained transmitter co-ordinates as: T (11.825, 9.245), which is around 14 meters offset from the actual location, (0, 17).

4.2.2.2 RSSI measurements in Pepsicola:

As the LoRa module has a range of 10 km, RSSI tests were conducted for larger distances to see if the proposed localization model will be suitable for greater distances. A

total of two tests were done in the region of Pepsicola where the transmitter was kept on the roof of Suncity apartment. The receivers were kept at separate distances from the transmitter provided that VLOS was maintained. The value of RSSI was recorded for 15 minutes at each location and then the transmitter was moved to another location for the next test.

In the first test, we wanted to observe the effects of high altitude above ground level with the RSSI value so one of the receivers was kept at Jagdol hill top while other two were kept near the apartment (within 1km).

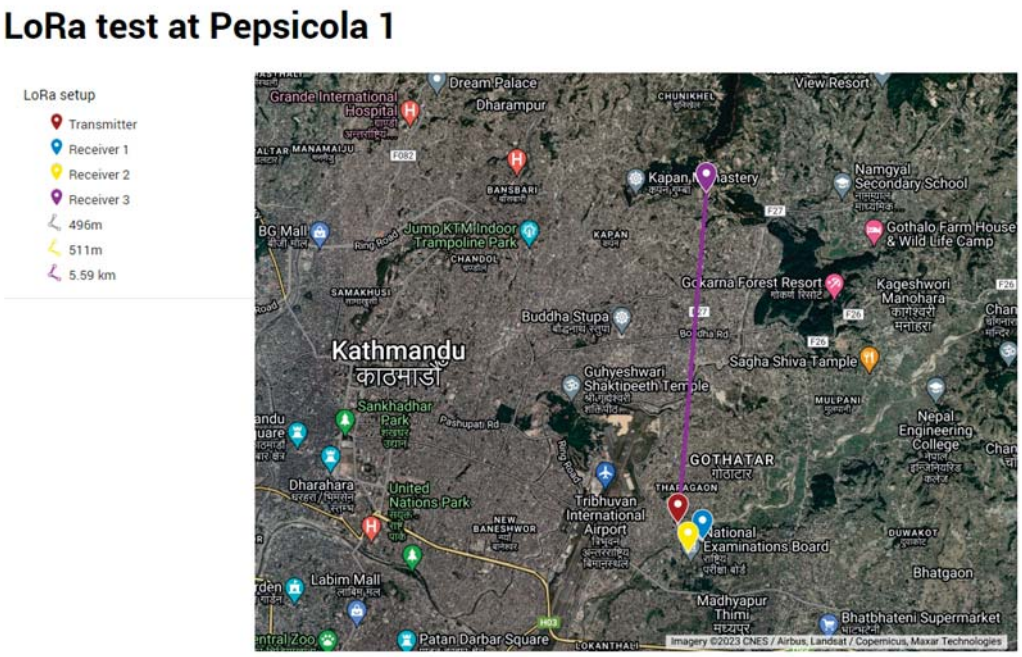

Figure 4.14: LoRa test one at Pepsicola

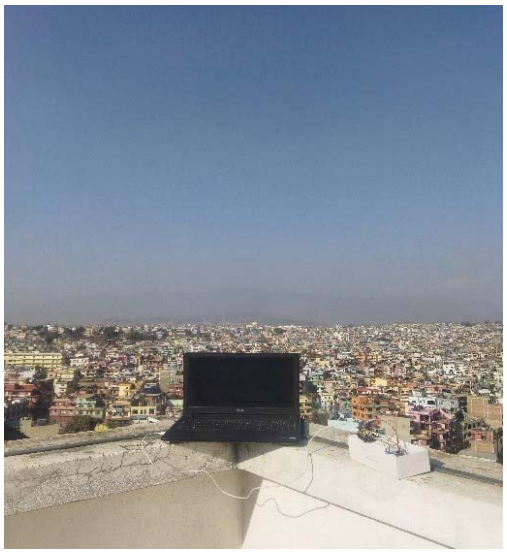

Figure 4.15: LoRa transmitter

Figure 4.16: LoRa receiver

Table 4.7: Transmitter GPS coordinates for test one of LoRa

Transmitter	GPS coordinates	
	Latitude	Longitude
	27.69292	85.37155

Table 4.8: Test results of LoRa test one

Receiver Node	GPS coordinates		Actual Distance	Mean RSSI value
	Latitude	Longitude		
1	27.69048	85.37577	496 m	-101.196
2	27.68862	85.37335	511 m	-96.34
3	27.74293	85.37642	5.59 km	-109.73

In test two, the transmitter was kept at another position still on the top of Suncity apartment. A total of four receivers were placed at four different positions with varying distances from the transmitter. The data was stored using Cool term software, and the duration of data capture was 15 minutes.

The RSSI values received at Jagdol hill (receiver node 3 at table 4.8) offered an interesting result, and the signal strength received there was higher than expected (compare, for example, with receiver node 1 at table 4.10. This is because the Jagdol hill was located at a high altitude and at clear line of sight from Suncity apartment, reducing the effects of multipath. This observation is important, and gives us an idea that flying UAS at higher altitude from the ground level will offer better signal readings.

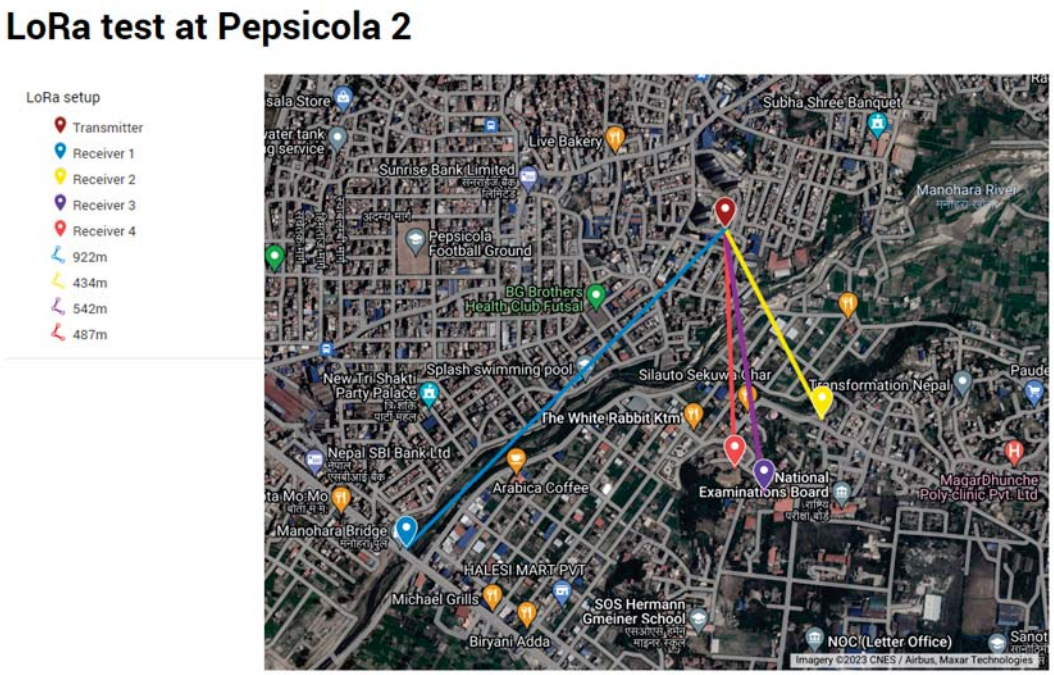

Figure 4.17: LoRa test two at Pepsicola

Figure 4.18: LoRa transmitter

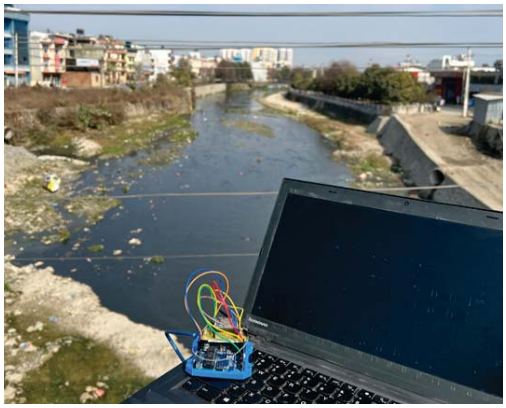

Figure 4.19: LoRa receiver

Table 4.9: Transmitter GPS coordinates for test two of LoRa

Transmitter	GPS coordinates	
	Latitude	Longitude
	27.69209	85.37135

Table 4.10: Test results for test two of LoRa

Receiver Node	GPS coordinates		Actual Distance	Mean RSSI value
	Latitude	Longitude		
1	27.68623	85.36473	922 m	-112.606
2	27.68862	85.37335	434 m	-96.34
3	27.68728	85.37215	542 m	-86.66
4	27.68772	85.37154	487 m	-87.93

The results from both tests do not match the variations resulting from the path-loss model, and the distance conversion is slightly unreliable. This is because at larger distances when environmental factors are varying and unstable at different measurement directions, even slight variations in RSSI values can result in significant distance errors.

As calibration in the outdoor Pepsicola region is difficult without actually flying the UAS, which requires special permission from the Civil Aviation Authority of Nepal, the further localization applications were focused only on the Campus area.

4.3 Temperature, relative humidity and RSSI

Temperature and relative humidity data of the air were measured at the receiver's end along with RSSI while receiving the signals to infer whether the signal strength is affected by variations in temperature and humidity, as mentioned extensively in the literature [37, 38]. The results drawn from the data are shown in figures 4.20 and 4.21.

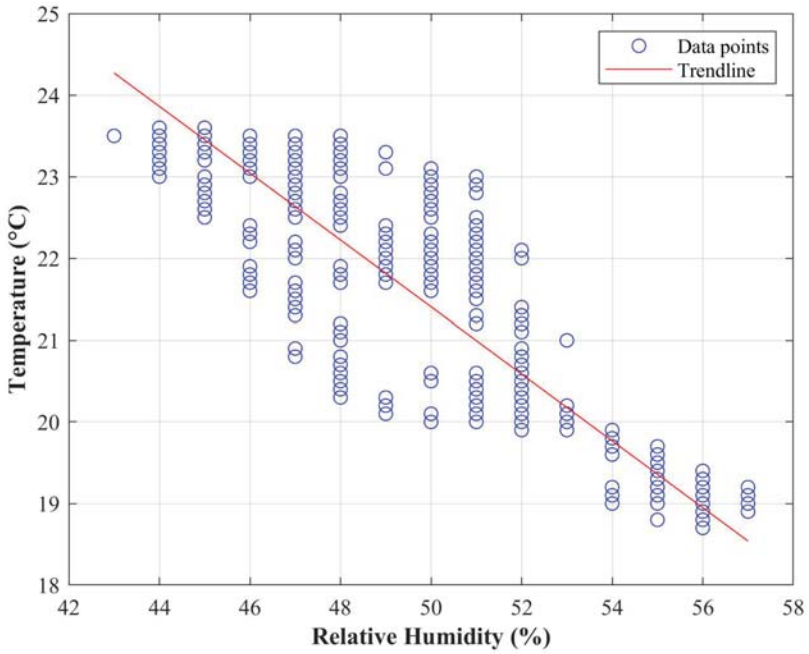

Figure 4.20: Temperature vs Relative Humidity

The obtained data of temperature and relative humidity when plotted with each other provide an important result. A strong negative correlation was observed between temperature and relative humidity, i.e. as relative humidity increased, the temperature decreased, and vice versa. The Karl-Pearson correlation coefficient was found to be -0.9214. This is physically explainable. As the temperature increases the water-storing capacity of air increases thus decreasing the relative humidity.

Effects of temperature and relative humidity on RSSI values could not be independently performed as different additional infrastructure would be necessary. In a combined form, there is a strong correlation between relative humidity and temperature. Therefore, it is acceptable to use only one variable to study the effect in RSSI measurements. As relative humidity offered a greater range of data points, it was selected.

The RSSI vs Relative Humidity graph in figure 4.21 shows that there is a small negative correlation between the variables. The higher the relative humidity, the lower the signal strength. This result goes with the result from [39], which states that the signal strength is inversely proportional to the relative humidity at constant pressure. Here, the distance associated with the RSSI value increases when relative humidity increases

and the distance decreases as relative humidity decreases.

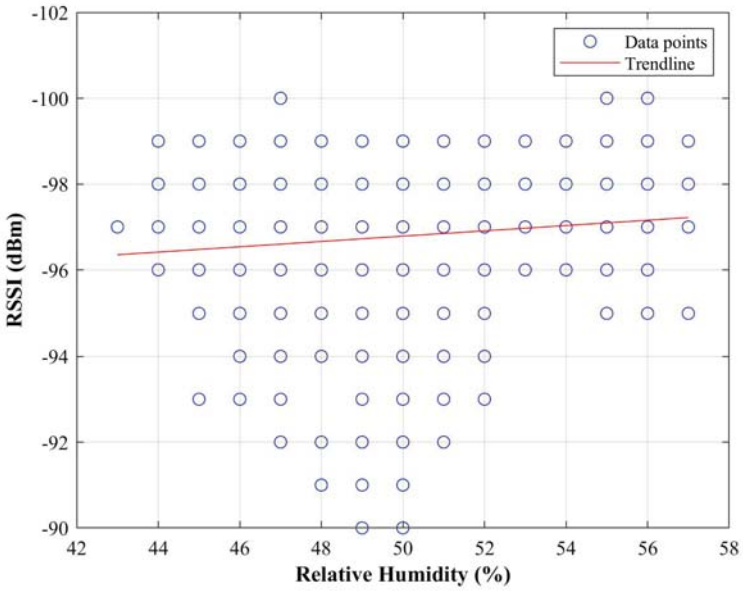

Figure 4.21: Relative Humidity vs RSSI

4.4 RSSI variation with day time

Three different tests were performed at three different times of the day and found that RSSI values vary even for the same locations at different times. Interestingly, the curves shifted upwards, i.e. the signal strength decreased. This can be attributed to the increase in relative humidity as the time progresses from 11 am to 5 pm.

Figure 4.22: The known five nodes for RSSI calculation

The measured RSSI at the known distance is plotted in the table below:

Table 4.11: RSSI value at different distances and time

Distance(m)	RSSI value at 11 am	RSSI value at 4pm	RSSI value at 5pm
28	-63.56	-67	-76.76
23	-61.03	-65	-73.6
18	-58.02	-64	-70.25
13	-56.15	-62	-67.21
8	-51.32	-56	-65.42

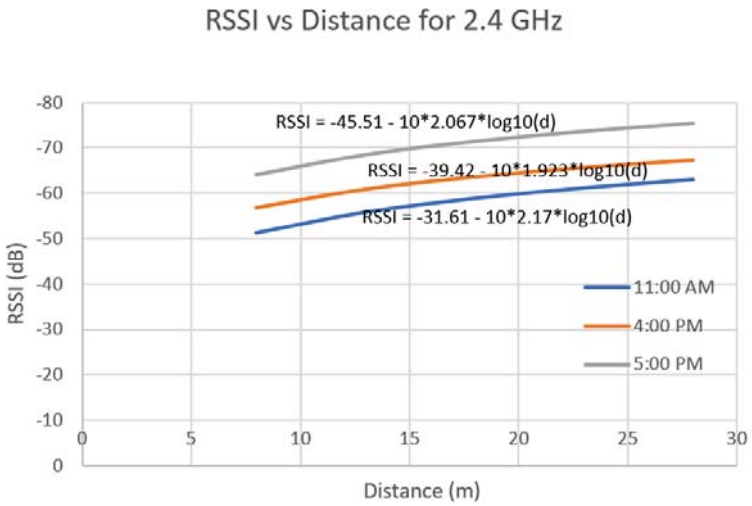

Figure 4.23: RSSI vs distance at different times of the day.

4.5 Filtering of noisy signals

The results from these tests show that the RSSI values are greatly affected by factors like obstacles, interference, antenna orientation, multipath, weather, and various other environmental conditions. This results in multiple measurements even at the same distance, and can dramatically affect the localization accuracy. The noise effect on the RSSI values for indoor and outdoor environment are visualized in figure 4.24 and figure 4.25 respectively. It is important to use the most reliable RSSI values for calibration and mission implementation. Therefore, an approach to filter out the noisy signals is desired.

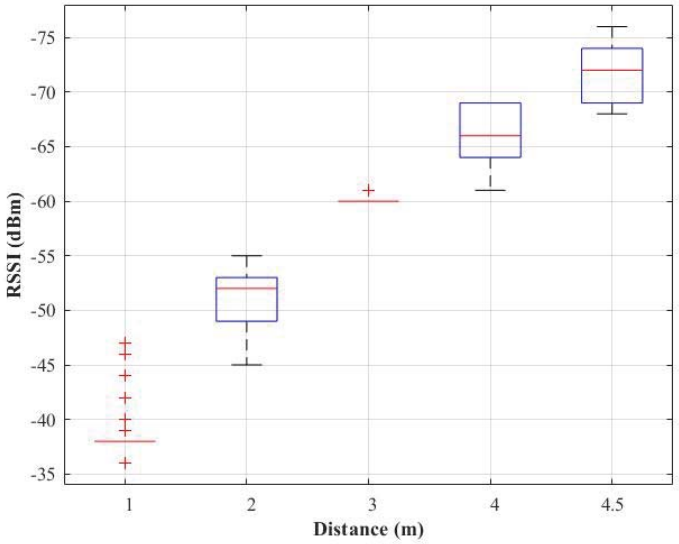

Figure 4.24: Noise effect on the RSSI measurements at different distances (indoor environment).

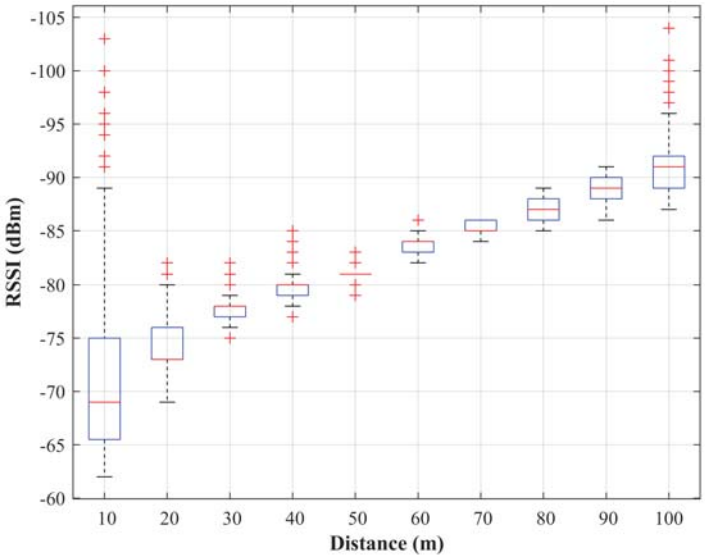

Figure 4.25: Noise effect on the RSSI measurements at different distances (outdoor environment).

In figures 4.26, 4.27, and 4.28, Simple Moving Average filter (15 window samples), Exponential Moving Average filter (smoothing parameter of 0.1), and Kalman filter are used over continuous RSSI measurements from 0 m to 120 m distance respectively. It is seen that the Kalman filter has very low responsiveness to changes and gives an al-

most smooth fit to the noisy signal data. Therefore, the Kalman filter can be particularly helpful to generate the filtered data that will be used to fit the path loss model during the calibration phase.

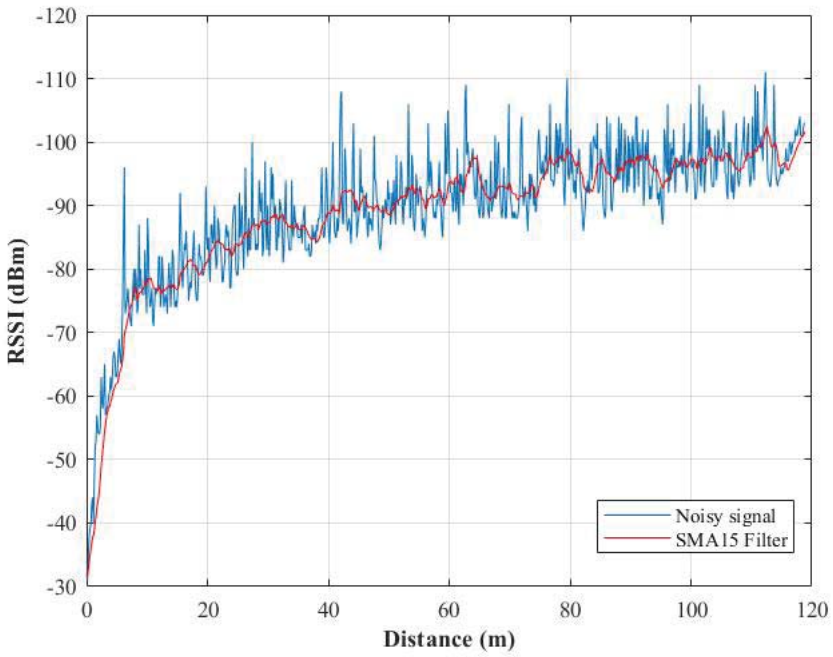

Figure 4.26: SMA

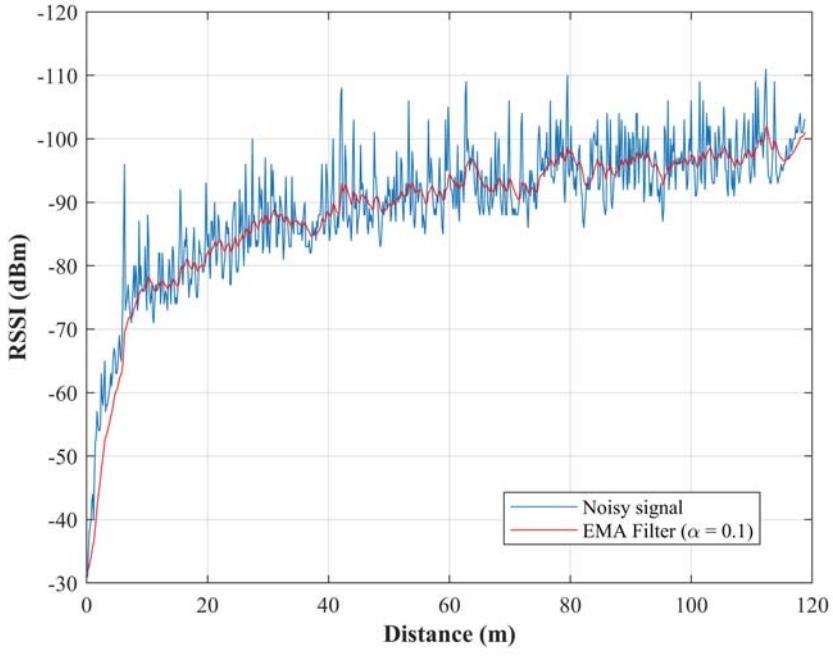

Figure 4.27: EMA

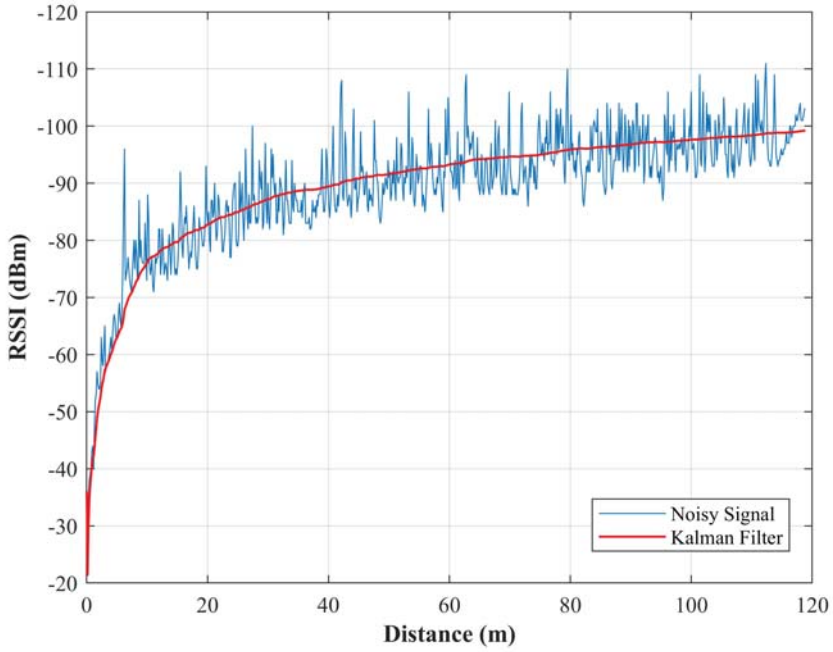

Figure 4.28: Kalman Filter

SMA and EMA filters are much simpler than the Kalman filter and offer better responsiveness to signal changes. It is observed that SMA and EMA offer very similar fitting to the signal data. However, EMA uses just two samples at a time, whereas SMA is using 15 samples. Therefore, EMA has very low computational cost than SMA. These moving average filters are not just simply good for several applications, they are also optimal for a problem involving the reduction of random white noise while keeping the sharpest step response [40]. In this work, however, filters are not expected to demonstrate a response to the noisy signals. Therefore, the Kalman filter is selected for most of filtering-related applications.

4.6 Algorithm verification

It is important to verify the developed localization code (Appendix II) before actually implementing it on a UAS. Therefore, three different localization tests, to be discussed in 4.6.3, were carried out. Before the tests, a coordinate system was set up, and the parameters of the path loss model were obtained from the calibration experiment.

4.6.1 Coordinate System

The coordinate system developed at the Pulchowk Campus Cricket Ground with three measurement points (blue location markers) is shown in figure 4.29. One of the points, the origin, located at the geographical coordinates (27.68292, 85.32213), served as the starting point for RSSI measurement. The other two points are located to the North and East, with coordinates (27.68293, 85.32255) and (27.68355, 85.32219), respectively. The North and East points have the Cartesian coordinates of (0,70) and (40,0) respectively.

Figure 4.29: Coordinate system

4.6.2 Calibration

It is necessary to carry out a pre-mission calibration test to determine the path loss exponent, n, and RSSI0, the signal strength value at 1m. RSSI values are read at different discrete points. The mean values for the corresponding distances from the transmitter are plotted. Then, the log-distance path loss model is fitted by minimizing the sum of squares. Values of n and RSSI0 are obtained to be 2.2377 and -45 dBm respectively.

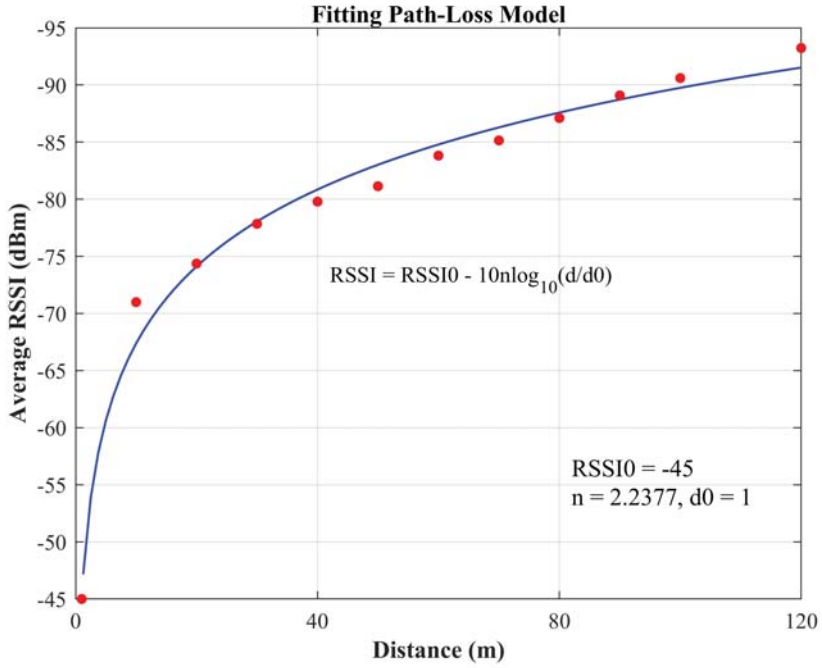

Figure 4.30: Fitted log-distance path loss model for the characterization experiment.

4.6.3 Testing and Results

The test involved placing three different markers at the measurement locations defined at the developed coordinated system, with one marker positioned at the origin and the other two markers at Y-axis and X-axis i.e. the North and the East position respectively. These markers helped to identify the location for RSSI data acquisition. A scooter was used to move the receiving system to the next location after measuring RSSI values at one. The results from the three different tests are presented in table 4.12 and 4.13.

Table 4.12: Mean RSSI for various tests performed

Coordinates	Test 1 Mean RSSI (dBm)	Test 2 Mean RSSI (dBm)	Test 3 Mean RSSI (dBm)
(0,0)	-86.92	-86.92	-84.99
(40,0)	-75.97	-77.67	-76.32
(0,70)	-90.03	-90.61	-89.04

Table 4.13: Algorithm Verification

Test	Localized Cartesian		Localized Geographical (°)		Distance Error (m)	Angle Error(°)
	X	Y	Latitude	Longitude		
1	82.43	-0.74	27.682913	85.322966	20.5	9.26
2	79.36	-10.32	27.682827	85.5322935	28.4	16
3	59.02	0.13	27.682921	85.322729	15..8	8.62

From the Cartesian coordinates of the markers and the mean RSSI value measured at each marker position, the target's localized Cartesian coordinates are obtained. Mean RSSI values are obtained at the markers using the hardware of the 433MHz band. Using mean RSSI values in the calibrated path loss model, the distances from the markers to the transmitter are computed. Knowing the distances from the three markers, the trilateration algorithm was run in real-time on the Arduino Uno setup which provided the necessary localized geographic coordinates.

The algorithm validation involved conducting three tests. The first two tests yielded values with distance errors of 20.5m and 28.4m respectively. These errors could have been due to disturbances in the ground and other environmental factors. However, in the final test, the localization algorithm only provided an error of 15.8m. This result is particularly noteworthy because it compares favorably to the distance error of about 20 m that can be expected with GPS with the low fix. Therefore, from the third test, the algorithm can be proven valid and tests on the UAS could be performed.

4.7 Main Mission

After the validation of the localization algorithm, a mission was conducted using a drone equipped with the localization algorithm hardware at Pulchowk Cricket ground. The mission involved creating axes for target localization.

4.7.1 Coordinate System

The coordinate system used for the mission was different than that of the algorithm verification. To find the coordinates of the three positions, they were marked. The origin was placed at the geographic coordinates of (27.6831534, 85.3221424) which was also the starting point of the mission flight path. The other two positions were kept at the north and east axis with coordinates of (27.6836279, 85.3221545) and (27.683138, 85.3224484) respectively. The geographical coordinate system was converted to Cartesian Coordinate system. where the north point and the east point had the coordinates of (0,50) and (30,0) respectively.

Figure 4.31: Coordinate system

4.7.2 Calibration

To accurately fit the log-distance path loss model for the testing environment, calibration flight was carried out by continuously measuring RSSI values at different distances from the transmitter. The measured values were extremely noisy ($\sigma \approx 5$), so the noisy signals were filtered using Kalman filter.

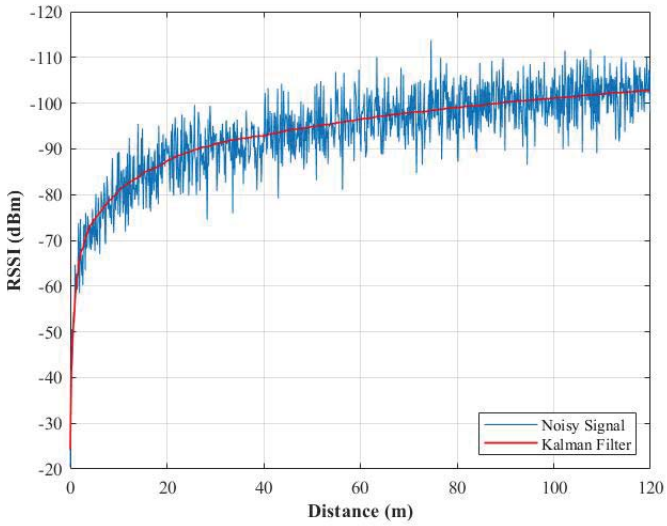

Figure 4.32: Noisy signal filtered out for calibration

The log-distance path loss model is then fitted to the filtered values. Figure 4.33 shows that the model almost perfectly fits the filtered values, with RSSI0 = -59.1 and n = 2.1.

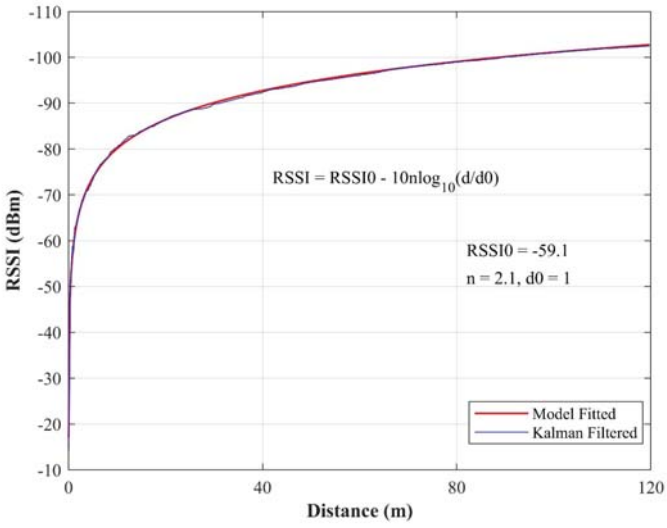

Figure 4.33: Fitted log-distance path loss model

4.7.3 Simulation

The simulation of the localization algorithm for the main mission localization case was carried out in MATLAB. The simulation, however, is not completely independent of

experimental results. The simulation needs to be fed with the values of path loss exponent, n, and mean RSSI value at reference distance (d_0), RSSI0, which are obtained after a characterization experiment. The simulation model also incorporates the effects of Gaussian noises. Therefore, it is also necessary to estimate the standard deviation of the unfiltered Gaussian noise for the environment beforehand.

Rewriting equation 3.2 in terms of the Gaussian noise, we have

$$X_\sigma = RSSI0 - 10nlog_{10}\frac{d}{d_0} - RSSI \tag{4.19}$$

From the noisy RSSI values obtained from the characterization experiment (figure 4.32) and the fitted parameters of the log-distance model(figure 4.33), X_σ is obtained. Here,
Mean (X_σ) = - 0.0552 (≈ 0)
Standard Deviation = 4.6, i.e. σ = 4.6
So, the parameters to feed to the simulation model are shown in table 4.14

Table 4.14: Values of parameters to feed to the simulation model

Parameters	Values
n	2.1
RSSI0	-59.1
σ	4.6

A perfectly omnidirectional transmitter located at (64.77,-7.5) is modeled in a 2-dimensional space of 250 m by 250 m. The noisy signal is added to the RSSI value at a location by the command sigma*randn. The simulated RSSI distribution over the test domain is shown in figure 4.34.

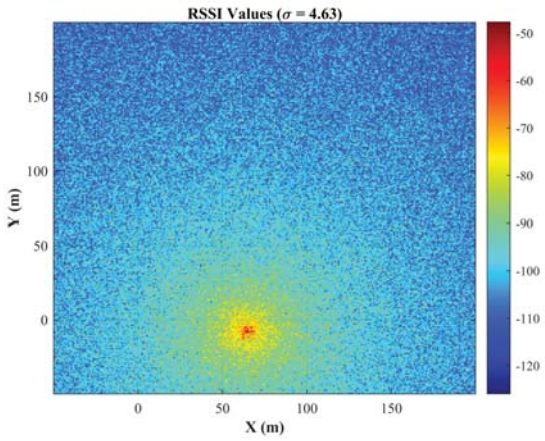

Figure 4.34: Simulated RSSI distribution over the test domain

The localization simulation was carried out for two different cases: the receiver reading a single RSSI value at each measurement point, and the receiver reading 100 RSSI values and using the mean RSSI value for each measurement point.

4.7.3.1 Reading single RSSI sample

The simulation model is such that a single value of RSSI is read at each of the three locations, and those three RSSI values are used to compute the corresponding distances. Figure 4.35 shows the localized transmitter position with respect to the actual position. Clearly, there is a significant error in localization.

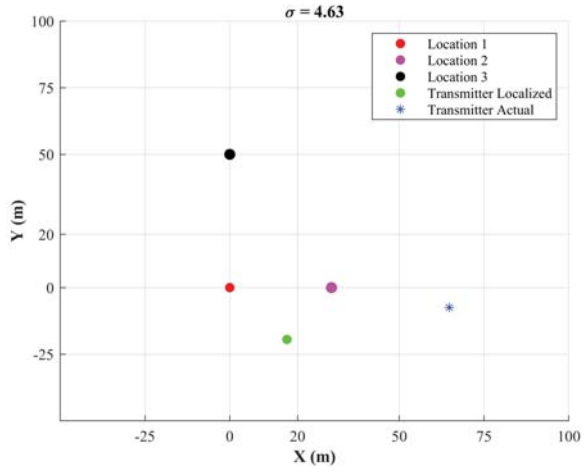

Figure 4.35: Localization based on single RSSI sample at each measurement point

As the value of Gaussian noise changes over time (with the mean value, μ being zero, and the standard deviation, σ being 4.6), 100 similar iterations for localization were performed, and the corresponding distance and angle errors are observed for each localization mission.

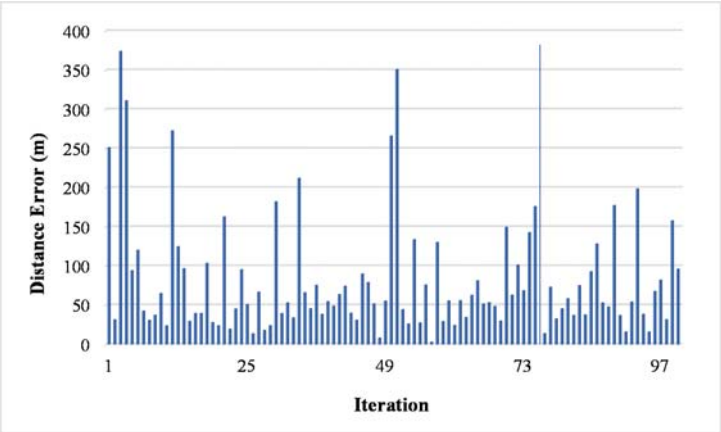

Figure 4.36: Distance errors for 100 similar missions (1 RSSI sample at each measurement point.

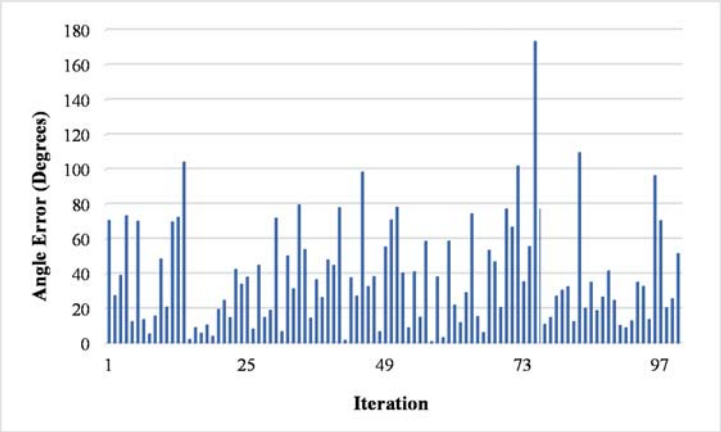

Figure 4.37: Angle errors for 100 similar missions (1 RSSI sample at each measurement point.

Figures 4.36 and 4.37 not only show significant errors in the distance and angle of the localized transmitter with respect to the origin (0,0) respectively but also highlight the highly unstable nature of the localization scheme. Sometimes, the transmitter is localized to a distance error as low as 3 meters, whereas sometimes the error shoots as high as 382 meters. Therefore, reading just a single RSSI value at each point is prone to inaccuracy, and gives highly unstable localization results even for the similar missions.

4.7.3.2 Reading multiple RSSI samples

One of the approaches to improve the accuracy and stability of the localization scheme is to measure multiple RSSI values at each point and use the mean value for that point for further computation. Reading multiple RSSI values at a point sums up the Gaussian noise variables, and their cumulative effect approaches toward their mean value, i.e. 0, thereby bringing accuracy and stability to the scheme.

Figure 4.38 shows the localization result. Compare it to figure 4.35, and it is clearly observed that the localization accuracy has significantly increased.

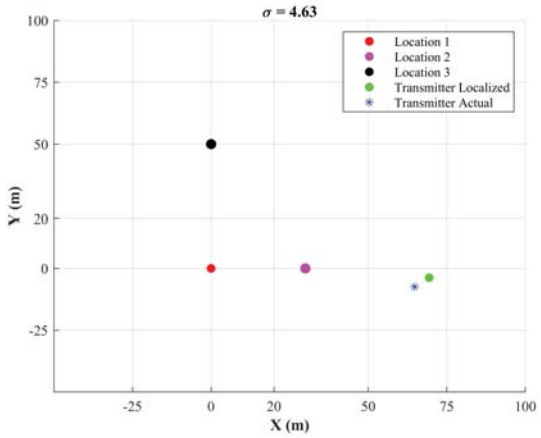

Figure 4.38: Localization based on the mean value of 100 RSSI samples at each measurement point.

To study the stability of the localization scheme, 100 similar missions are simulated, and their distance and angle error are presented in figure 4.39 and figure 4.40 respectively. In every case, the distance and angle errors are less than 20 meters and 20° respectively. In 83 % of cases, the transmitter is localized to a maximum distance error of 10 meters, and in 84 % of cases, it is localized to the maximum angle error of 8°.

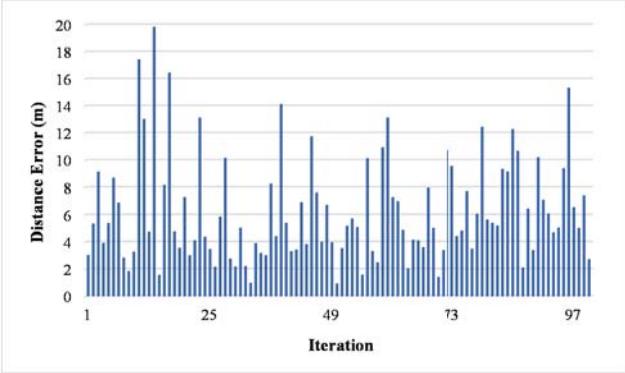

Figure 4.39: Distance errors for 100 similar missions (100 RSSI samples for each measurement point.

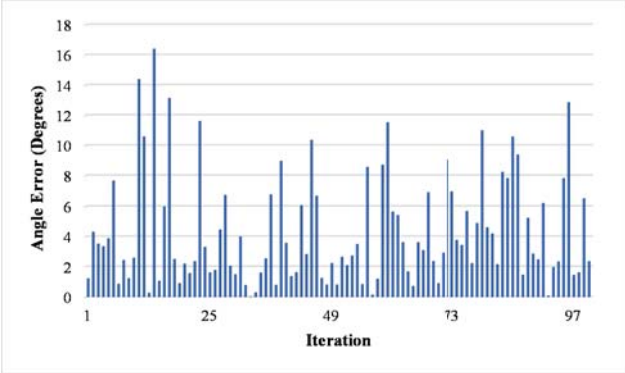

Figure 4.40: Angle errors for 100 similar missions (100 RSSI samples for each measurement point.

Although, it was desired to increase the number of RSSI samples to read at a measurement point, it would increase the localization time, and the battery resources UAS would require. Therefore, the number of packets to read at a point was set to 75 for the mission flight.

4.7.4 Mission Flight

The mission flight was planned in the QGroundControl application using the three positions as waypoints in the coordinate system. The flight altitude was set to 3 m from the ground, and the flight speed was selected as 5 m/s. At each waypoint, the drone hovered for 25 seconds to acquire data on RSSI values. After obtaining the mean RSSI value at

the third position, the drone landed there, and the localized coordinates were provided to the ground station. Another mission was then planned to the localized position using the obtained geographical coordinates.

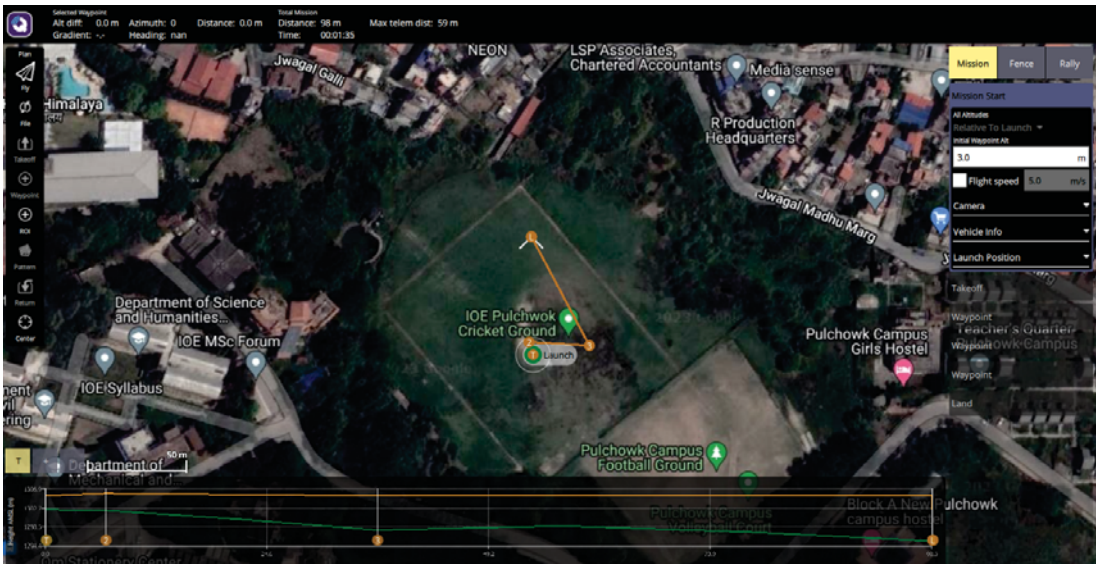

Figure 4.41: Main mission flight plan

Figure 4.42: Main mission actual flight path

4.7.5 Mission Result

The mean RSSI values received at the three waypoints are shown in table 4.15. Using the calibration of RSSI values with distance, the corresponding distance for each RSSI value was calculated by the onboard computer, which then performed the trilateration calculations, and provided the localized latitude and longitude to be (27.682944, 85.323031). This resulted in a distance error of 25.6 m from the source and an angular error of 8.32 degrees with respect to the origin.

Table 4.15: Mean RSSI received at different co-ordinates

Coordinate	RSSI (dBm)
(0,0)	-97.93
(30,0)	-88.51
(0,50)	-101.02

Table 4.16: Main Mission Localization

Test	Localized Cartesian		Localized Geographical (°)		Distance Error (m)	Angle Error (°)
	X	Y	Latitude	Longitude		
1	87.62	-23.36	27.682944	85.323031	25.6	8.32

The results from the main mission show that the transmitter is localized within the desired minimum distance error of 30 meters with respect to its actual position and 20° with respect to the origin. The results could have been even better if not for the GPS offset at measurement locations, which was showing errors as high as 5 meters.

With this mission test, the main objective of the project is successfully met.

Figure 4.43: Localization flight plan

Figure 4.44: Localization actual flight path

4.8 Uncertainty Considerations

New sets of experiments were carried out for the presentation of the uncertainty consideration concept. First, a characterization experiment was carried out to fit a path loss model where the RSSI values are measured continuously over 120 m distance. The obtained noisy RSSI values were filtered using the Kalman filter, and the filtered data

points were used to obtain the path loss model given by equation 4.20.

$$RSSI = -49.6732–10 * 2.4381 * log_{10}d \tag{4.20}$$

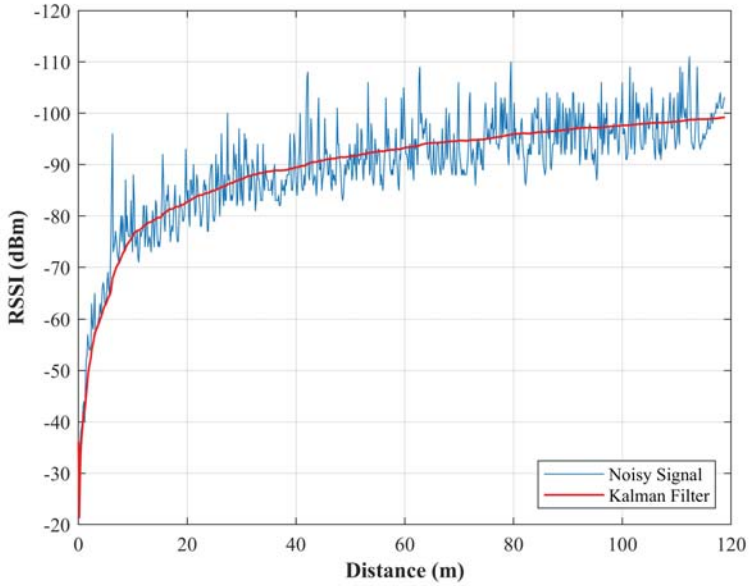

Figure 4.45: Kalman Filtered

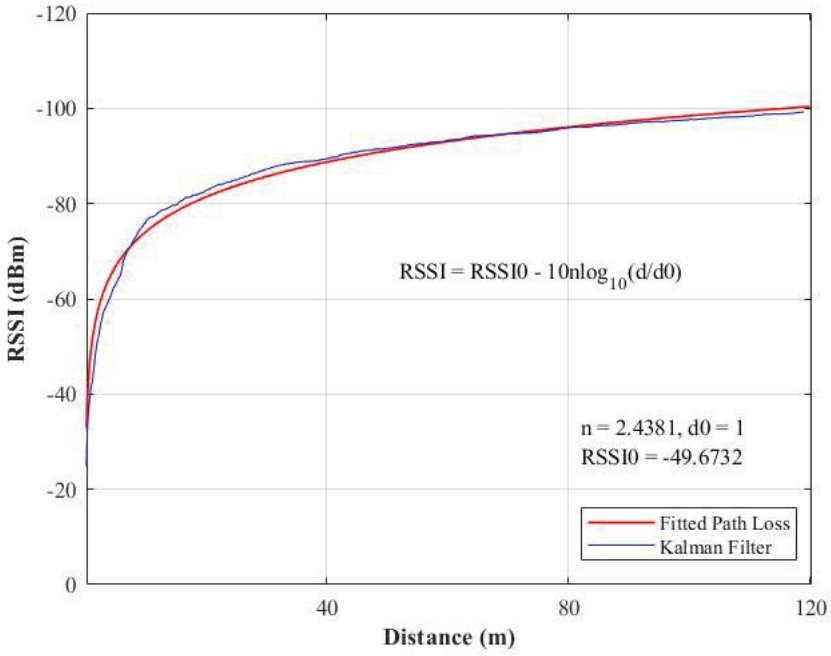

Figure 4.46: Fitted Path Loss

Then a validation experiment was performed where the UAS flew towards the transmitter on a straight path from 120 meters away. The RSSI values were continuously measured throughout the flight and stored on the SD card.

The stored RSSI values are retrieved and filtered using Kalman filter.

4.8.1 Range of Distance

The existing noises in the filtered RSSI values are given by an equation 4.21

$$X_{noise} = RSSI0 - 10nlog_{10}d - X \tag{4.21}$$

where, X and X_{noise} are the array of filtered signals and existing noise signals respectively. Each element in X_{noise} denotes the RSSI noise at the particular distance.

Table 4.17: Mean and standard deviation of X_{noise}

Mean (μ)	-0.3695 dBm
Standard Deviation (σ)	1.985 dBm

As preferred, the mean value of the existing noise is close to zero. It is then assumed that the standard deviation of RSSI values at every distance is constant and equal to the standard deviation of the noisy signal, i.e. $\sigma = 1.985$

Then, for every value of X, $[f(X + m.\sigma), f(X - m.\sigma)]$ are evaluated to obtain the range of probable distances, where f(x) is given by equation 3.17, and m = 1, 1.5, 2, and 3. Range of distances for the entire flight for 68.26% (m=1), 86.64% (m=1.5), 95.44% (m=2) and 99.74% (m=3) confidence interval are shown in figures below:

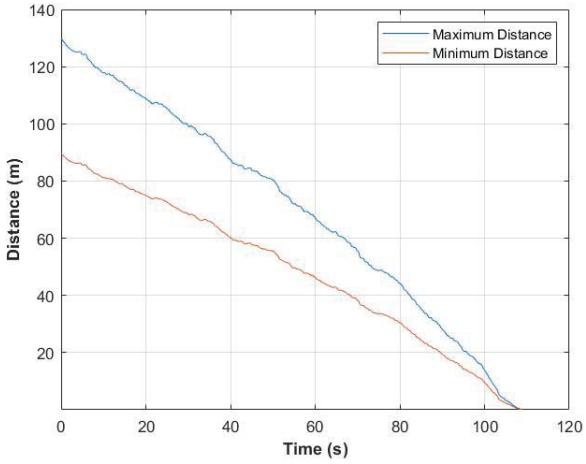

Figure 4.47: Range of distances for 68% confidence interval

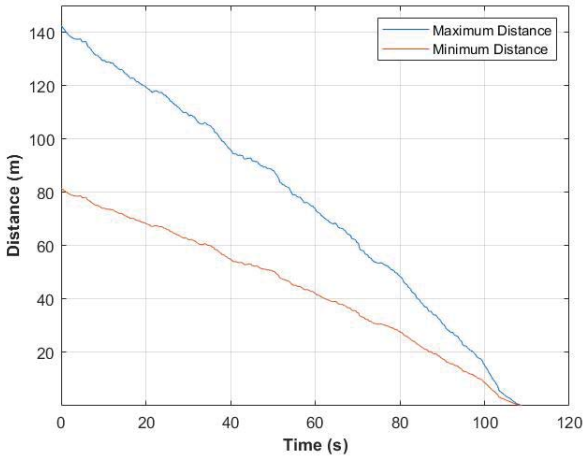

Figure 4.48: Range of distances for 86% confidence interval

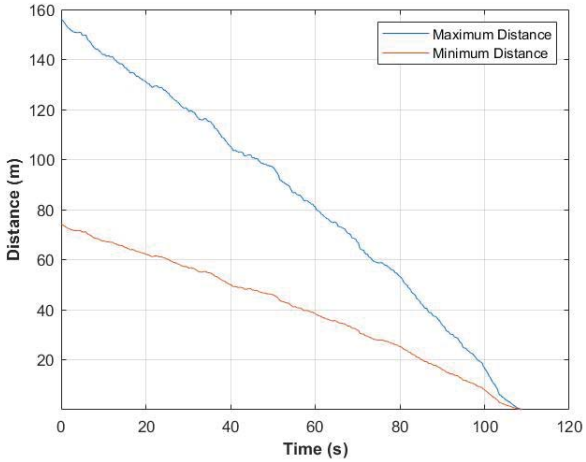

Figure 4.49: Range of distances for 95% confidence interval

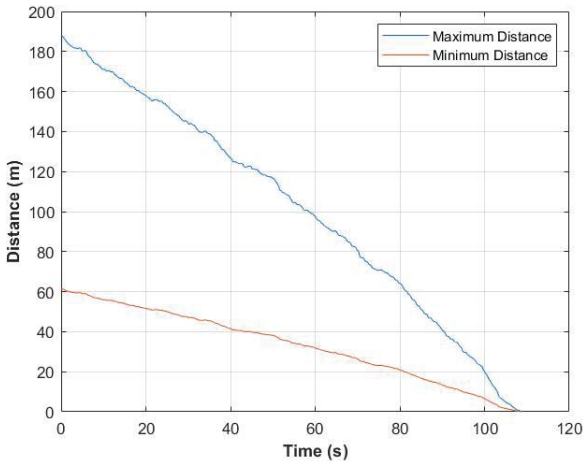

Figure 4.50: Range of distances for 99% confidence interval

As we increase the probability, the predicted distance range at a particular time increased as expected. It can be observed that the range of distances is gradually decreasing for all the cases as we proceed forward in time, and the predicted minimum and maximum distances have finally converged as the vehicle reaches the transmitter location. This is because at larger distances, we have higher values of negative RSSI, and even a slight RSSI variation will result in significant distance error owing to the logarithmic nature of the path loss model. To put this into perspective, for the model given by 4.20, a 1 dBm variation of RSSI values from -59 dBm to -60 dBm will bring the distance error of 0.25 m, whereas, the same variation from -99 dBm to -100 dBm will give the distance error of 10.5 m.

With the increase in the confidence interval, we are introducing more uncertainty in our distance predictions. Selection of best confidence interval needs extensive validation tests, which is beyond the scope of this work, and depends upon the requirements and capabilities of the particular system (for example, does UAS has resources to carry out multiple localization iterations? If yes, a system with greater distance confidence interval might also be acceptable) . Considering the maximum distance estimation error of around 30% to be acceptable for the farthest distance, confidence interval below 86.6%is recommended. Note that, the decrease in the distance confidence interval, however, demands a system with increased robustness, reliability and stability.

4.8.2 Moving towards/away

Another approach to consider uncertainty in distance measurement based on RSSI values is to provide the information on whether the UAS is moving towards or away from the transmitter source. The filtered RSSI values of the validation experiment are converted to the distance values based upon the model given by equation 2. The instantaneous velocity is then computed for each time step. The graph of instantaneous velocity looks heavily noise. So, the smoothing filter (EMA with α = 0.02) is used as shown in figures 4.51 and 4.52.

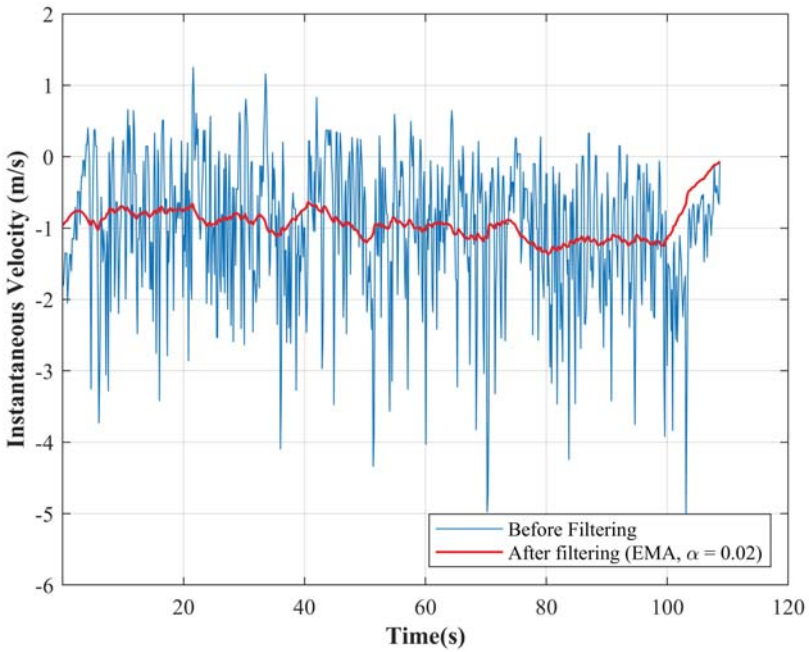

Figure 4.51: Unfiltered and filtered velocity

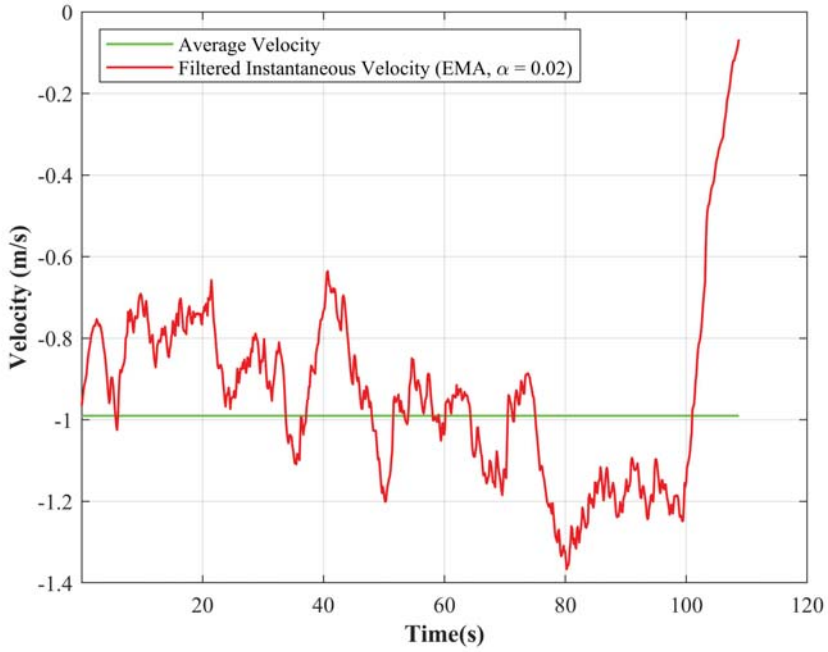

Figure 4.52: Filtered instantaneous and average velocity

The filtered signal shows that the instantaneous velocity is negative at every instance which is exactly what was expected, as the UAS was continuously moving towards the transmitter, making $d_{t+1} < d_t$. Therefore, it is accurately predicted that the UAS was moving towards the transmitter at every instant of time.

4.8.2.1 Velocity Estimation

It was also possible to estimate the average velocity based upon the filtered RSSI values. The measured distance between the transmitter and the starting location was 120 meters, and the mission time was 110 seconds. As we expect the UAS to take 10 seconds for take off and landing, the average mission speed is 1.2 m/s, which is also the value set while uploading the mission. Completely based upon the RSSI measurements, and the corresponding distance estimates for starting and ending mission points, the average velocity of the UAS was predicted to be -1 m/s (- sign meaning that the UAS is moving towards transmitter). This, compared to the measured value of -1.2 m/s, accounts for an error of 16.6 $\%$.

Table 4.18: Velocity Estimation

Average Velocity (Measured)	-1.2 m/s
Average Velocity (Predicted)	-1 m/s
Error	16.6 %

This prediction is, however, based upon single distance estimates for starting and end points. If we consider the probabilistic approach of mentioning the distance ranges, the predicted average velocity intervals are shown in table 4.19.

Table 4.19: Distance Confidence Interval & Predicted Average Velocity (Absolute) Interval

Distance Confidence Interval	Predicted Average Velocity (Absolute) Interval (m/s)
68.26 %	0.84 m/s to 1.16 m/s
86.64 %	0.78 m/s to 1.25 m/s
95.44 %	0.72 m/s to 1.36 m/s
99.74 %	0.6 m/s to 1.6 m/s

4.9 Limitations

Considering the primitive level of the system developed with its research and development at Pulchowk Campus being at an early phase, the acceptable error tolerance was set as 30 meters within an angular accuracy of 15 degrees for the test domain of 100 meters radius. Given this, the developed system localized the transmitter to an acceptable level of accuracy. However, the system has some important limitations offering an scope for future enhancement. The limitations of the developed system are listed below:

- It will only be capable of locating a stationary transmitter source that emits radio signals with constant power.
- The proposed localization modality is affected by multi-path and signal fading.

- It is based on the assumption that the transmitter is perfectly omnidirectional. Therefore, it does not model the possible variation in RSSI values measured at equal distances at different directions as shown by the results of the tri-line test shown in figure 4.53 and 4.54.

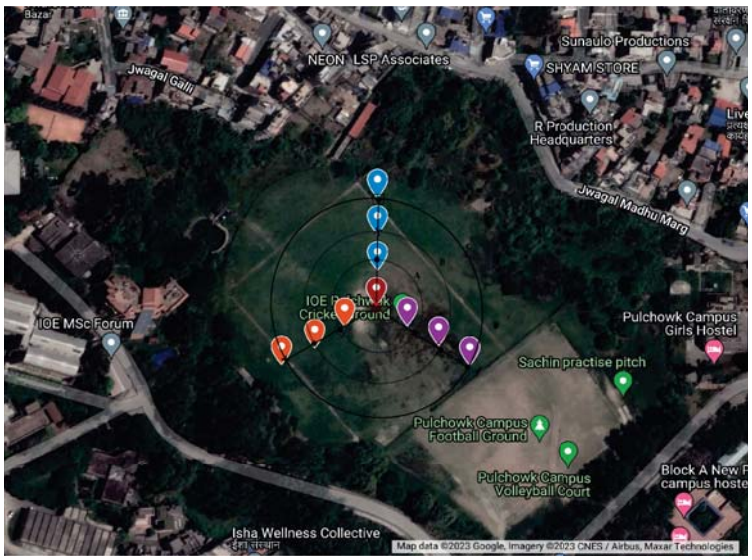

Figure 4.53: Tri-line test region (geographic perspective).

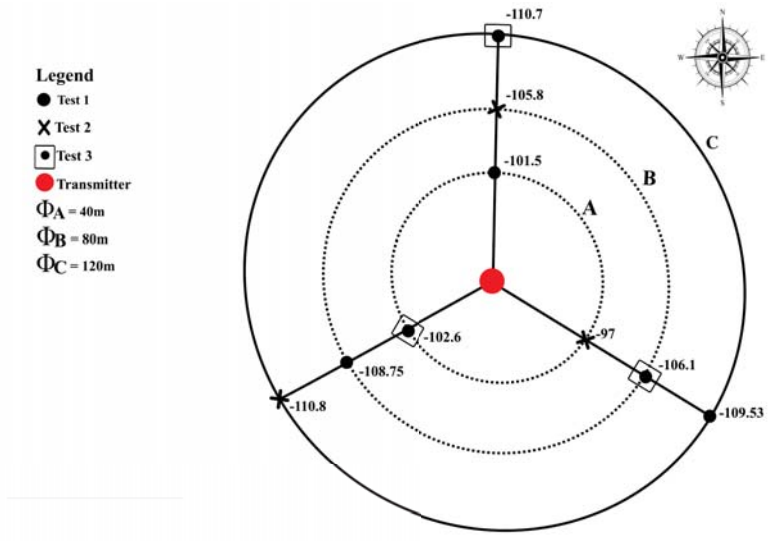

Figure 4.54: Schematic of tri-line test region with average RSSI values at test points

CHAPTER FIVE: CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

A UAS system, with underlying tools, techniques, and capabilities for the localization of radio frequency sources for indoor and outdoor environments, is presented. For indoor localization, a 2.4 GHz Wi-Fi router is used as the radio transmitter, and the ESP8266-Arduino combination has been prototyped as the receiver. Since it was not possible to fly the fabricated UAS in the indoor environment, only the WSN-based localization was performed, and the transmitter was localized with a distance error of 1.2m. Most of the work afterward involved localization for an outdoor environment. 433 MHz LoRa-based custom prototype transmitters and receivers were developed. A series of tests: UAS-based and Wireless Sensor Network (WSN) based, have been performed at the premises of the Pulchowk Campus and Pepsicola area. LoRa-used WSN-based test at the Pulchowk Campus localized the transmitter with a distance error of 14 m. A localization algorithm is developed for UAS implementation, and during its verification test, the transmitter was localized at best with a distance error of 15.8 m and an angle error of 8.6°. The main mission was simulated in MATLAB simultaneously with practical implementation. The simulation results showed that under ideal conditions, except for the presence of Gaussian noises, the transmitter can be localized within the accuracy of 5 to 10 meters by increasing the number of RSSI samples read at each measurement point. Using mean RSSI values, computed from 100 sample values at each measurement, the UAS, in its final test mission, localized the transmitter with a distance error of 25 meters, and an angle error of 8 degrees. As the acceptable level of accuracy was set at 30 meters within the angular accuracy of 15 meters, the results are accepted.

Graphical correlations between temperature, humidity and RSSI have been established. It is clearly understood that RSSI-based localization, despite being simple, is highly affected by weather conditions and environmental dynamics. Therefore, it is prone to inaccuracy and requires extensive testing for calibration. Therefore, it is recommended

to measure the reference values before each flight. Simulation results show that it is desirable to reduce the effect of random noises, and simply taking large number of RSSI measurements at a measurement point and using the mean value for distance calculation brings significant improvement in the accuracy and stability of the localization scheme. However, this will be achieved at the expense of increased localization time.

Different filtering techniques like Simple Moving Average, Exponential Moving Average, and Kalman filter are studied and compared. Kalman filter is designed for the off-board application and is observed to provide better filtering characteristics than SMA and EMA. Finally, two approaches for considering uncertainties in distance approximations at each time step are presented: range of distances and moving towards/away. For the maximum distance estimation error of 30 $\%$ to be acceptable, a distance confidence interval below 86 $\%$ is recommended.

5.2 Recommendations

The project offers several rooms for enhancements in the future. One of the immediate actions would be to incorporate the directional antenna into the system and develop a corresponding localization algorithm based on RSSI and Angle of Arrival data. This will increase the accuracy, as well as reduce the localization time. If accuracy becomes the major concern, and not the resources, then employing multiple UAS systems that communicate with each other in real-time, instead of just one, will help develop a high-fidelity localization scheme. The project, currently, employs the deterministic model for localization purposes, i.e you measure the RSSI, translate to distance, use trilateration and locate the source. In the bigger picture, it can be enhanced into the probabilistic model in the near future, which, contrary to what the name suggests, is expected to be more accurate, as it will be based on real-time iterative path planning.

References

[1] N. A. Alrajeh, M. Bashir, and B. Shams, "Localization techniques in wireless sensor networks," *International journal of distributed sensor networks*, vol. 9, no. 6, p. 304628, 2013.

[2] C. Alippi and G. Vanini, "A rssi-based and calibrated centralized localization technique for wireless sensor networks," in *Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)*, pp. 5–pp, IEEE, 2006.

[3] M. Stella, M. Russo, and D. Begušić, "Rf localization in indoor environment.," *Radioengineering*, vol. 21, no. 2, 2012.

[4] P. Bahl and V. N. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *Proceedings IEEE INFOCOM 2000. Conference on computer communications. Nineteenth annual joint conference of the IEEE computer and communications societies (Cat. No. 00CH37064)*, vol. 2, pp. 775–784, Ieee, 2000.

[5] O. M. Cliff, R. Fitch, S. Sukkarieh, D. L. Saunders, and R. Heinsohn, "Online localization of radio-tagged wildlife with an autonomous aerial robot system," in *Robotics: Science and Systems*, 2015.

[6] Y. Hou, X. Yang, and Q. H. Abbasi, "Efficient aoa-based wireless indoor localization for hospital outpatients using mobile devices," *Sensors*, vol. 18, no. 11, p. 3698, 2018.

[7] J. He, Y. Geng, and K. Pahlavan, "Toward accurate human tracking: Modeling time-of-arrival for wireless wearable sensors in multipath environment," *IEEE Sensors Journal*, vol. 14, no. 11, pp. 3996–4006, 2014.

[8] F. Quitin, C. Cheng, M. Leng, and W. Tay, "A wireless tdoa estimation architecture using software-defined radios,"

[9] M. I. M. Ismail, R. A. Dzyauddin, S. Samsul, N. A. Azmi, Y. Yamada, M. F. M. Yakub, and N. A. B. A. Salleh, "An rssi-based wireless sensor node localisation

using trilateration and multilateration methods for outdoor environment," *arXiv preprint arXiv:1912.07801*, 2019.

[10] K. Heurtefeux and F. Valois, "Is rssi a good choice for localization in wireless sensor network?," in *2012 IEEE 26th international conference on advanced information networking and applications*, pp. 732–739, IEEE, 2012.

[11] S. Jondhale, R. Deshpande, S. Walke, and A. Jondhale, "Issues and challenges in rssi based target localization and tracking in wireless sensor networks," in *2016 international conference on automatic control and dynamic optimization techniques (ICACDOT)*, pp. 594–598, IEEE, 2016.

[12] J. Graefenstein, A. Albert, P. Biber, and A. Schilling, "Wireless node localization based on rssi using a rotating antenna on a mobile robot," in *2009 6Th Workshop on positioning, navigation and communication*, pp. 253–259, IEEE, 2009.

[13] K. VonEhr, S. Hilaski, B. E. Dunne, and J. Ward, "Software defined radio for direction-finding in uav wildlife tracking," in *2016 IEEE International Conference on Electro Information Technology (EIT)*, pp. 0464–0469, IEEE, 2016.

[14] I. Jami, M. Ali, and R. Ormondroyd, "Comparison of methods of locating and tracking cellular mobiles," 1999.

[15] K. Whitehouse, C. Karlof, and D. Culler, "A practical evaluation of radio signal strength for ranging-based localization," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 1, pp. 41–52, 2007.

[16] M. Aernouts, N. BniLam, R. Berkvens, and M. Weyn, "Tdaoa: A combination of tdoa and aoa localization with lorawan," *Internet of Things*, vol. 11, p. 100236, 2020.

[17] S. Yang and H. Cha, "An empirical study of antenna characteristics toward rf-based localization for ieee 802.15. 4 sensor nodes," in *Wireless Sensor Networks: 4th European Conference, EWSN 2007, Delft, The Netherlands, January 29-31, 2007. Proceedings 4*, pp. 309–324, Springer, 2007.

[18] D. Oh and J. Han, "Smart search system of autonomous flight uavs for disaster rescue," *Sensors*, vol. 21, no. 20, p. 6810, 2021.

[19] S. Yucer, F. Tektas, M. V. Kilinc, I. Kandemir, H. Celebi, Y. Genc, and Y. S. Akgul, "Rssi-based outdoor localization with single unmanned aerial vehicle," *arXiv preprint arXiv:2004.10083*, 2020.

[20] K.-H. Lam, C.-C. Cheung, and W.-C. Lee, "Rssi-based lora localization systems for large-scale indoor and outdoor environments," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 11778–11791, 2019.

[21] O. A. Saraereh, A. Alsaraira, I. Khan, and P. Uthansakul, "Performance evaluation of uav-enabled lora networks for disaster management applications," *Sensors*, vol. 20, no. 8, p. 2396, 2020.

[22] Civil Aviation Authority of Nepal, "Uas requirement," September 2022.

[23] A. Bensky, *Wireless positioning technologies and applications*. Artech House, 2016.

[24] M. Wadhwa, M. Song, V. Rali, and S. Shetty, "The impact of antenna orientation on wireless sensor network performance," in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pp. 143–147, IEEE, 2009.

[25] S. H. Ahmed, S. H. Bouk, N. Javaid, and I. Sasase, "Combined human, antenna orientation in elevation direction and ground effect on rssi in wireless sensor networks," in *2012 10th International Conference on Frontiers of Information Technology*, pp. 46–49, IEEE, 2012.

[26] A. E. Kosba, A. Abdelkader, and M. Youssef, "Analysis of a device-free passive tracking system in typical wireless environments," in *2009 3rd international conference on new technologies, mobility and security*, pp. 1–5, IEEE, 2009.

[27] A. Guidara, G. Fersi, F. Derbel, and M. B. Jemaa, "Impacts of temperature and humidity variations on rssi in indoor wireless sensor networks," *Procedia Computer Science*, vol. 126, pp. 1072–1081, 2018.

[28] E. B. Hamida and G. Chelius, "Investigating the impact of human activity on the performance of wireless networks—an experimental approach," in *2010 IEEE*

International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM), pp. 1–8, IEEE, 2010.

[29] A. Booranawong, K. Sengchuai, and N. Jindapetch, "Implementation and test of an rssi-based indoor target localization system: Human movement effects on the accuracy," *Measurement*, vol. 133, pp. 370–382, 2019.

[30] Q. Dong, F. Zhu, Y. Cai, L. Fang, and M. Lu, "Analysis of rssi feasibility for sensor positioning in exterior environment," in *2021 Wireless Telecommunications Symposium (WTS)*, pp. 1–7, IEEE, 2021.

[31] K. Zhang, Y. Zhang, and S. Wan, "Research of rssi indoor ranging algorithm based on gaussian-kalman linear filtering," in *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pp. 1628–1632, IEEE, 2016.

[32] M. A. Koledoye, D. De Martini, S. Rigoni, and T. Facchinetti, "A comparison of rssi filtering techniques for range-based localization," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 761–767, IEEE, 2018.

[33] B.-C. Seet, Q. Zhang, C. H. Foh, and A. C. Fong, "Hybrid rf mapping and kalman filtered spring relaxation for sensor network localization," *IEEE Sensors Journal*, vol. 12, no. 5, pp. 1427–1435, 2011.

[34] K. Ibwe and S. Pande, "Filtering effect on rssi-based indoor localization methods," *Tanzania Journal of Engineering and Technology*, vol. 41, no. 4, 2023.

[35] T. S. Rappaport, "Wireless communications–principles and practice, (the book end)," *Microwave Journal*, vol. 45, no. 12, pp. 128–129, 2002.

[36] J. Smith, S. Jones, and J. Brown, "Implementing a distance estimator for a wildlife tracking system based on 802.15.4," *Journal of Wildlife Technology*, vol. 10, no. 2, pp. 45–57, 2020.

[37] K. Bannister, G. Giorgetti, and S. Gupta, "Wireless sensor networking for hot applications: Effects of temperature on signal strength, data collection and lo-

calization," in *Proceedings of the 5th workshop on embedded networked sensors (HotEmNets' 08)*, pp. 1–5, 2008.

[38] B. Capsuto and J. Frolik, "A system to monitor signal fade due to weather phenomena for outdoor sensor systems," in *Fifth International Conference on Information Processing in Sensor Networks (IPSN 2006)*, Citeseer, 2006.

[39] R. Mat, N. H. Sabri, R. Umar, S. Ahmad, S. N. A. S. Zafar, A. Omar, and W. A. Mustafa, "Effect of humidity on tropospheric received signal strength (rss) in ultra-high frequency (uhf) band," in *Journal of Physics: Conference Series*, vol. 1529, p. 042048, IOP Publishing, 2020.

[40] S. W. Smith *et al.*, "The scientist and engineer's guide to digital signal processing," 1997.

APPENDICES

Appendix I: LoRa-ESP8266 transmitter code for Arduino IDE

```
#include <SPI.h>
#include <LoRa.h>

#define ss 15
#define rst 16
#define dio0 4

int counter = 0;
int button_pin = 5;
boolean button_state = LOW;

void setup()
{
  Serial.begin(115200);
   pinMode(button_pin, INPUT);
   while (!Serial);
  Serial.println("LoRa Sender");
  LoRa.setPins(ss, rst, dio0);
    if (!LoRa.begin(433E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
 LoRa.setSyncWord(0xFF);
 Serial.println("LoRa Initializing OK!");
}
void loop() {
button_state = digitalRead(button_pin);
if(button_state == HIGH)
{
  //Serial.print("Sending packet: ");
  Serial.print(counter);
  Serial.println("Help! Help!!");
  // send packet
```

```
  LoRa.beginPacket();
  LoRa.print("Help! Help!!");
  LoRa.endPacket();
counter++;
delay(100);
}
else{
    Serial.println("...");

    delay(1000);
}
}
```

Appendix II: Localization Code running on-board on the flight computer (Arduino Uno)

```
#include <SPI.h>
#include <LoRa.h>

#define ss 10
#define rst 9
#define dio0 3
#define NUM_PACKETS 75
int rssi_values[NUM_PACKETS];
int packet_count = 0;
int mean_count = 0;

float r1;
float r2;
float r3;
float mean_rssi;
float mean_rssi_1;
float mean_rssi_2;
float mean_rssi_3;
float mean_rssi_4;
float mean_rssi_5;

// Defining x, y coordinates of the three measuring locations
float x1 = 0.0;
float y1 = 0.0;
float x2 = 30.0;
float y2 = 0.0;
float x3 = 0.0;
float y3 = 50.0;

// Calibrated Parameters
float RSSI0 = -59.1;
float n = 2.1;
float d0 = 1;
```

```
 // Defining latitude and longitude of the reference point
float ref_lat = 27.6831534 ; //latitude of reference point in degrees
float ref_lon = 85.3221424; // longitude of reference point in
    degrees

// Defining constants for latitude and longitude conversion
float deg2rad = PI/180.0; // conversion factor from degrees to
    radians
float earth_radius = 6378137.0; // radius of the earth in meters

 void setup() {
    Serial.begin(115200);
        while (!Serial);
        Serial.println("LoRa Receiver");
LoRa.setPins(ss, rst, dio0);
if (!LoRa.begin(433E6)) { // or 915E6
Serial.println("Starting LoRa failed!");
while (1);
 }
LoRa.setSyncWord(0xFF);
delay(2000);
Serial.println("LoRa Initializing OK!");
delay(15000);

// register the receive callback
LoRa.onReceive(onReceive);
// put the radio into receive mode
LoRa.receive();
}
void loop() {
// do nothing
}

void onReceive(int packetSize) {
// check if the packet is valid
if (packetSize == 0) {
 // invalid packet, skip it
LoRa.receive();
return;
```

```
 }
// received a valid packet
Serial.print("Received: '");
// read packet
for (int i = 0; i < packetSize; i++) {
Serial.print((char)LoRa.read());
}
Serial.print("' RSSI: ");
Serial.println(LoRa.packetRssi());
// store the RSSI value
rssi_values[packet_count] = LoRa.packetRssi();
packet_count++;
if (packet_count == NUM_PACKETS) {
// reset the packet count and compute the average RSSI value
packet_count = 0;
int sum = 0;
for (int i = 0; i < NUM_PACKETS; i++) {
sum += rssi_values[i];
}
mean_rssi = (float)sum / NUM_PACKETS;
Serial.print("Mean RSSI: ");
Serial.println(mean_rssi);
 mean_count++;
if (mean_count == 1) {
 mean_rssi_1 = mean_rssi;
}
else if (mean_count == 2) {
mean_rssi_2 = mean_rssi;
}
else if (mean_count == 3) {
mean_rssi_3 = mean_rssi;
}
else if (mean_count == 4) {
mean_rssi_4 = mean_rssi;
}
else if (mean_count == 5) {
 mean_rssi_5 = mean_rssi;
Serial.print("Mean RSSI at location 1: ");
Serial.println(mean_rssi_1);
```

```
Serial.print("Mean RSSI while moving from 1 to 2: ");
Serial.println(mean_rssi_2);
Serial.print("Mean RSSI at location 2: ");
Serial.println(mean_rssi_3);
Serial.print("Mean RSSI while moving from 2 to 3: ");
Serial.println(mean_rssi_4);
Serial.print("Mean RSSI 3 at location 3: ");
Serial.println(mean_rssi_5);

 // Calculating distances (example)
r1 = d0 * pow(10, (RSSI0 - mean_rssi_1)/(10*n));
r2 = d0 * pow(10, (RSSI0 - mean_rssi_3)/(10*n));
r3 = d0 * pow(10, (RSSI0 - mean_rssi_5)/(10*n));

Serial.print("D1: ");
Serial.println(r1);
Serial.print("D2: ");
Serial.println(r2);
Serial.print("D3: ");
Serial.println(r3);

 // Calculating transmitter co-ordinates
// define the matrix of coefficients
float A[2][2] = {
{2.0*(x1-x2), 2.0*(y1-y2)},
{2.0*(x1-x3), 2.0*(y1-y3)}
};

// define the vector of constants
float b[2] = {
r2*r2 - r1*r1 + x1*x1 - x2*x2 + y1*y1 - y2*y2,
r3*r3 - r1*r1 + x1*x1 - x3*x3 + y1*y1 - y3*y3
};

// compute the inverse of A
float det = A[0][0]*A[1][1] - A[0][1]*A[1][0];
float invA[2][2] = {
{A[1][1]/det, -A[0][1]/det},
{-A[1][0]/det, A[0][0]/det}
```

```
};
// compute the solution vector x
float x[2] = {
invA[0][0]*b[0] + invA[0][1]*b[1],
invA[1][0]*b[0] + invA[1][1]*b[1]
};
// print the solution
Serial.print("x = ");
Serial.print(x[0]);
Serial.print(", y = ");
Serial.println(x[1]);

// calculate the latitude and longitude of the point
float lat = ref_lat + (x[1]/earth_radius) * (1.0/deg2rad);
float lon = ref_lon + (x[0]/earth_radius) * (1.0/deg2rad) / cos(
    ref_lat*deg2rad);

// Displaying latitude and longitude values
Serial.print("Latitude: ");
Serial.println(lat, 6); // print latitude with 6 decimal places
Serial.print("Longitude: ");
Serial.println(lon, 6); // print longitude with 6 decimal places

// Setting LoRa to transmitting mode
LoRa.beginPacket();
LoRa.print("Mean RSSI at Location 1:");
LoRa.println(mean_rssi_1);
LoRa.print("Mean RSSI while moving from 1 to 2: ");
LoRa.println(mean_rssi_2);
LoRa.print("Mean RSSI at location 2: ");
LoRa.println(mean_rssi_3);
LoRa.print("Mean RSSI while moving from 2 to 3: ");
LoRa.println(mean_rssi_4);
LoRa.print("Mean RSSI at location 3: ");
LoRa.println(mean_rssi_5);
LoRa.endPacket();
LoRa.beginPacket();
LoRa.print("D1: ");
LoRa.println(r1);
```

```
LoRa.print("D2: ");
LoRa.println(r2);
LoRa.print("D3: ");
LoRa.println(r3);
LoRa.endPacket();
LoRa.beginPacket();
LoRa.println("Cartesian");
LoRa.print("x = ");
LoRa.print(x[0]);
LoRa.print(", y = ");
LoRa.println(x[1]);
LoRa.endPacket();
LoRa.beginPacket();
LoRa.print("1Lat: ");
LoRa.print(lat, 6);
LoRa.print(" Long: ");
LoRa.println(lon, 6);
LoRa.endPacket();
LoRa.end();
return;
}
}
// put the radio back into receive mode
LoRa.receive();
}
```

Appendix III: MATLAB Simulation Code

```
clc;
cear all;

% Define grid size and grid points
grid_size = 500;
step_size = 1;
[X,Y] = meshgrid(0:step_size:grid_size-step_size, 0:step_size:
    grid_size-step_size);

% Define transmitter location
tx_x = 250;
tx_y = 250;

% Define reference distance and reference RSSI
d0 = 1;
RSSI0 = -32;

% Define path loss exponent
n = 2;

% Define variance of the Gaussian noise
sigma = 2;

% Calculate distance of each grid point from the transmitter
D = sqrt((X-tx_x).^2 + (Y-tx_y).^2);
% Calculate path loss
path_loss = 10*n*log10(D/d0);

AWGN_mean = zeros(size(X));
for j = 1:1:50
Random = randn(size(X));
AWGN_mean = AWGN_mean + Random;
end
AWGN_mean = sigma * AWGN_mean/50;

% Calculate the RSSI
```

```
RSSI = RSSI0 - path_loss - AWGN_mean;

% Plot the RSSI values
figure();
imagesc(RSSI);
colormap('jet');
colorbar;
xlabel('X','FontWeight', 'bold');
ylabel('Y','FontWeight', 'bold');
set(gca,'YDir','normal');
set(gca,'fontname','times');
title('RSSI Values (\sigma = 5)')
grid on;

% Input three measurement locations
location1 = input('Enter coordinates for location 1 in format [X1 Y1
    ]): ');
location2 = input('Enter coordinates for location 2 in format [X2 Y2
    ]): ');
location3 = input('Enter coordinates for location 3 in format [X3 Y3
    ]): ');

% Read the RSSI values at three measurement locations
RSSI1 = interp2(X, Y, RSSI, location1(1), location1(2));
RSSI2 = interp2(X, Y, RSSI, location2(1), location2(2));
RSSI3 = interp2(X, Y, RSSI, location3(1), location3(2));

if isnan(RSSI1) || isnan(RSSI2) || isnan(RSSI3)
disp('Invalid location entered');
else
 % Convert RSSI to distance using the path loss model
d1 = d0 * 10^((RSSI0 - (RSSI1))/(10*n));
d2 = d0 * 10^((RSSI0 - (RSSI2))/(10*n));
d3 = d0 * 10^((RSSI0 - (RSSI3))/(10*n));
end

% Solving system of linear equations
x1 = location1(1);
y1 = location1(2);
```

```
x2 = location2(1);
y2 = location2(2);
x3 = location3(1);
y3 = location3(2);

% Define the system of equations
f = @(x) [((x(1)-x1)^2 + (x(2)-y1)^2 - d1^2);
((x(1)-x2)^2 + (x(2)-y2)^2 - d2^2);
((x(1)-x3)^2 + (x(2)-y3)^2 - d3^2)];

% Define the Jacobian matrix
J = @(x) [2*(x(1)-x1), 2*(x(2)-y1);
2*(x(1)-x2), 2*(x(2)-y2);
2*(x(1)-x3), 2*(x(2)-y3)];

% Initial guess for the solution
x0 = [100; 100];

% Solve the system of equations using fsolve
options = optimoptions(@fsolve, 'Display', 'off');
x = fsolve(f, x0, options);

% Values in x are the solution
sol_x = x(1);
sol_y = x(2);

% Plot the result
figure();
scatter(location1(1), location1(2),'r', 'filled');
hold on;
scatter(location2(1), location2(2),50, 'm', 'filled');
scatter(location3(1), location3(2),50, 'k', 'filled');
scatter(x(1), x(2), 'g', 'filled');
scatter(250, 250,'*','b');
legend('Location 1', 'Location 2', 'Location 3', 'Transmitter
    Localized', 'Transmitter Actual');
xlim([0 grid_size]);
ylim([0 grid_size]);
set(gca,'YDir','normal');
```

```
set(gca,'fontname','times');
xlabel('X','FontWeight', 'bold');
ylabel('Y','FontWeight', 'bold');
title('\sigma = 5');
grid on;
```

Appendix IV: Kalman, SMA, and EMA Filter

```
clc;
clear all;
%%Read the data from the Excel file,      ToFilter     , in this case
[num,txt,raw] = xlsread('ToFilter.xlsx');

% Extract the signal strength and distance columns from the data
signal_column = 1; % assuming the signal strength column is the first
    column
dist_column = 2; % assuming the distance column is the second column
noisy_signal = num(:, signal_column);
dist = num(:, dist_column);

% Define the parameters of the log path loss model (standard
   parameters are used)
n = 2; % Path loss exponent
RSSI0 = -32; % Reference RSSI value at reference distance d0
d0 = 1; % Reference distance

% Calculate the measurement matrix based on the log path loss model
H = -10*n*log10(dist/d0) + RSSI0;

% Reshape H to be a column vector with the same number of elements as
    dist
H = reshape(H, [], 1);

% Define the measurement matrix C for the Kalman filter
C = [H'; ones(1, length(dist))];

% Calculate the measurement noise covariance R based on the variance
   of the unfiltered signal
R = var(noisy_signal);
% Initialize the state vector
x = [RSSI0; 0];
% Initialize the covariance matrix
P = diag([100, 100]);
% Define the process noise covariance matrix Q
```

```
q = 0.0000001; % Process noise intensity
Q = [q, 0; 0, q];
% Define the state transition matrix A for the Kalman filter
A = eye(2);

% Run the Kalman filter over the signal
filtered_signal_kalman = zeros(size(noisy_signal));
for i = 1:length(noisy_signal)
% Predict the next state
x = A * x;
P = A * P * A' + Q;
% Update the state estimate
K = P * C(:,i) / (C(:,i)' * P * C(:,i) + R);
x = x + K * (noisy_signal(i) - C(:,i)' * x);
P = (eye(2) - K * C(:,i)') * P;
% Calculate the filtered signal
filtered_signal_kalman(i) = H(i) * x(1) + x(2);
end

% Run the moving average filter over the signal
window_size = 20;
filtered_signal_ma = movmean(noisy_signal, window_size);

% Run the exponential moving average filter over the signal
alpha = 0.1;
filtered_signal_ema = filter(alpha, [1 alpha-1], noisy_signal);
% Plot the noisy signal and the filtered signals
plot(dist, noisy_signal, dist, filtered_signal_kalman, dist,
    filtered_signal_ma, dist, filtered_signal_ema);
xlabel("Distance");
ylabel("RSSI");
legend("Noisy Signal", "Kalman Filter", "Moving Average Filter", "
    Exponential Moving Average Filter", 'Location', 'southeast');

% Reverse the y-axis to display increasing negative values
set(gca, 'YDir', 'reverse');
set(gca,'fontname','times');
```

Appendix V: Fitting Path Loss Model

```
clc;
clear all;

% Read the data from the Excel file
[num,txt,raw] = xlsread('Filtered.xlsx');
% Extract the signal strength and distance columns from the data
signal_column = 1; % assuming the signal strength column is the first
    column
dist_column = 2; % assuming the distance column is the second column
RSSI = num(:, signal_column);
d = num(:, dist_column);

n = 2; % starting value for the path loss exponent
d0 = 1; % reference distance

%% When RSSI0 is known
% RSSI0 = -58.5;
% % Define the function to minimize
% fun = @(x) sum((RSSI - RSSI0 + 10 * x(1) * log10(d / d0)).^2);
% % Minimize the function using the fminsearch function
% x = fminsearch(fun, n);
%% Get the path loss exponent and reference RSSI
% n = x(1);

%% When RSSI0 is not known
% Define the function to minimize
fun = @(x) sum((RSSI - x(1) + 10 * x(2) * log10(d / d0)).^2);
% Minimize the function using the fminsearch function
x = fminsearch(fun, [RSSI(1), n]);
% Get the path loss exponent and reference RSSI
RSSI0 = x(1);
n = x(2);
```

Appendix VI: Quadcopter Performance Calculator

```
  %% drone performance calculator
clc;
clear all;
%%input parameters
framewt=1; % the total frame weight of the drone in kg
payload= 0.623; %the payload of the drone in kg
TWR=2; % thrust to weight ratio typically 2
n=4; %no of motor
range=300; %maximum range required
%% total weight calculation
totalwt = framewt + payload;
%% total thrust required
maxthrust=totalwt*TWR;
motorthrustr = maxthrust/n*1000; %the maximum motor thrust per motor
%% finding the diametr and pitch required
% tther file name in which the data of drone is stored
file_name = 'thrustcalc.xlsx';
sheet_name = 'Sheet1';
% the value of thrust calculated for each motor
thrust_to_find = motorthrustr;
% Reading the excel file
data = xlsread(file_name, sheet_name);
% Extract the thrust column
thrust_column = data(:, 9); % assuming thrust column is the second
    column in the Excel file
greater_indices = find(thrust_column > thrust_to_find);%finds the
    value of thrust that is greater than the calculated data
if isempty(greater_indices)
        error('Thrust value not found in the Excel file');
end
% Find the minimum thrust value that is greater than the known thrust
     value
min_thrust = min(thrust_column(greater_indices));
row_index = find(thrust_column == min_thrust, 1);
if isempty(row_index)
        error('Thrust value not found in the Excel file');
```

```
end
% Extract data from another column corresponding to the thrust value
diameter_find = data(row_index, 3); % the column of diameter is 5
diameter = diameter_find *100/2.54 ;
pitch_find =data(row_index, 4); % the column of pitch is 6
pitch = pitch_find ;
fprintf('The diameter of propeller required inches =');
disp(diameter);
fprintf('The pitch of propeller required in inches =');
disp(pitch_find *100/2.54);
fprintf('The current of esc to be used in amps =');
I=data(row_index, 5);%current at maximum throttle
display(round(data(row_index, 5)*1.5,-1)+10);
voltage_find = data(row_index, 2);
fprintf('The no of cells of battery required =');
ncell=round(voltage_find/3.7);%no of cells of battery
display(round(voltage_find/3.7));
fprintf('The KV rating of motor =');
KV=data(row_index, 1);
display(data(row_index, 1));
I0=data(row_index, 6);
V=ncell*3.7;
%% the range and flight time of the drone
thr1 = 0.8 ; % maximum throttle settings of motor
thr2 = 0.4 ; % minimum throttle settings of motor
I1=thr1*I; % the throttle at maximum throttle settings
I2=thr2*I; %the throttle at minimum throttle settings
bd= 0.8 ; %the battery discharge rating set
EscI= I1 ; % the maximum current drawn by the motor is the esc
    current
Esc2= I2 ;
Itotal = EscI*n; % the maximum current drawn by the motor which is
    also the battery current drawn
Itotal2 = Esc2*n;
Iw = Itotal/totalwt ; % the maximum current in amps required to lift
    one kg of drone
Iw2 = Itotal2/totalwt ;
rangem=range*0.000621371;
fltime = (rangem*63120)/(KV*V*60*pitch); % the flight time in minutes
```

```
mah = fltime*Iw*1000;
fprintf('The mah of battery required =');
display(mah);
```

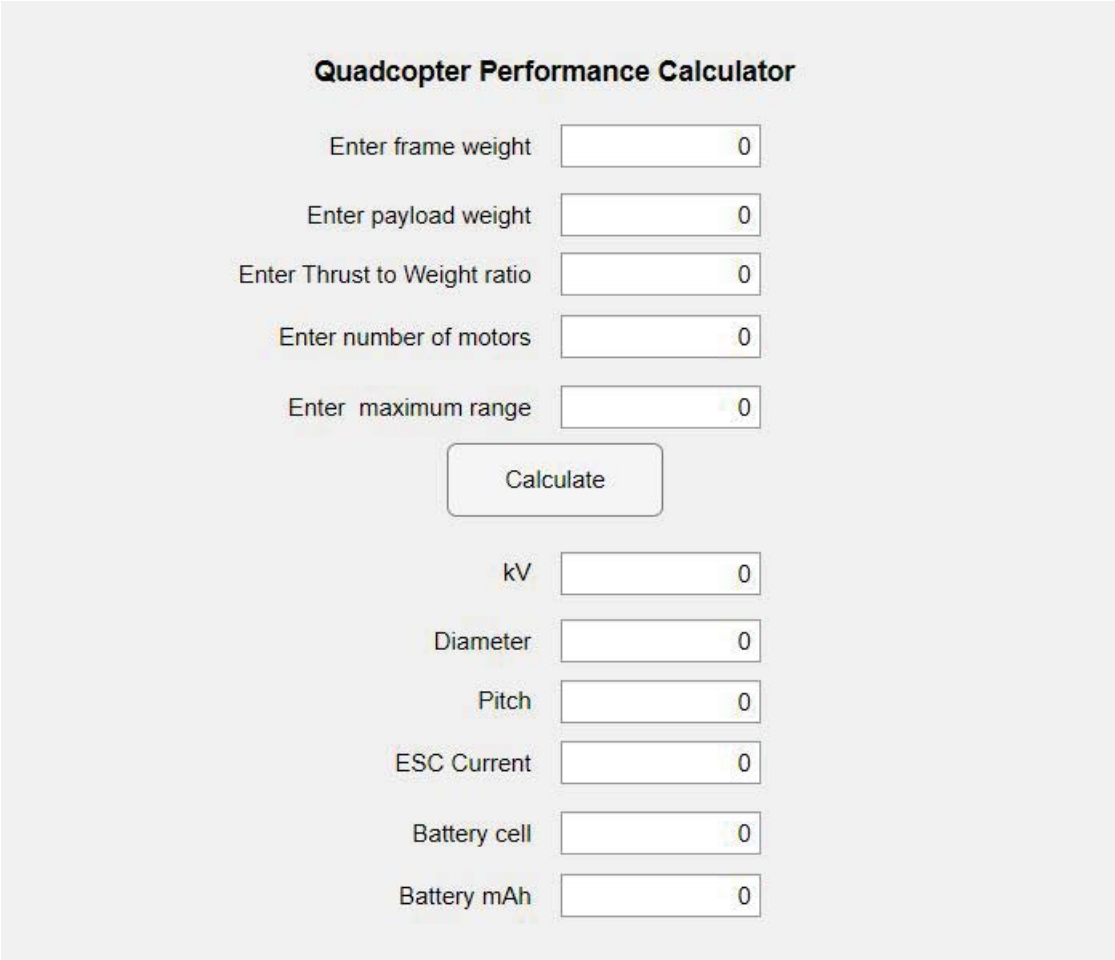

Figure 7.1: Quadcopter performance calculator GUI

KV	V	D	pitch	Current	No load current		Brand	Thrust
980	11	0.254	0.1143	15	0.3		D2212 Motor	860
980	7.4	0.254	0.1143	15	0.3		D2212 Motor	470
1000	11.7	0.254	0.1143	45	0.8		Turnigy Aerodrive SK3	1450
1000	14.8	0.254	0.1143	45	0.8		Turnigy Aerodrive SK3	2196
487	22.42	0.3302	0.1651	50	0.7		Thrust 50 motor	3234
487	22.63	0.3556	0.1778	50	0.7		Thrust 50 motor	4236
380	22.2	0.3556	0.12192	14	0.4		MN3508 KV380	1723
380	22.2	0.381	0.127	14	0.4		MN3508 KV380	1875
2300	16	0.127	0.1016	40	0.3		Emax RS2205S	1257
980	11	0.2286	0.11938	15	0.3		D2212 Motor	660
2300	12	0.127	0.1143	40	0.3		Emax RS2205S	774
2600	16	0.1016	0.1143	40	0.3		Emax RS2205S	995
1450	22.2	0.1778	0.127	50	0.3		Turnigy 3648	2300
1450	11.7	0.1778	0.127	50	0.3		Turnigy 3648	1450
2450	11.7	0.1524	0.1143	40	1.6		SunnySky X2212-2450kv	1100

Figure 7.2: Database