Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Gard Steinsvik

Radio-Tracking of Sheep: Developing a MAVLink-Enabled Ground Control Station with Location Estimation Based on Range-Only Measurements

Master's thesis in Computer Science
Supervisor: Svein-Olaf Hvasshovd
June 2021

NTNU
Norwegian University of
Science and Technology

Gard Steinsvik

Radio-Tracking of Sheep: Developing a MAVLink-Enabled Ground Control Station with Location Estimation Based on Range-Only Measurements

Master's thesis in Computer Science
Supervisor: Svein-Olaf Hvasshovd
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Acknowledgements

I want to thank Grzegorz Swiderski and Trygve Nerland for their cooperation in this project. Grzegorz's knowledge of the nRF chips and radio technology and Trygve's drone construction proficiency have been invaluable for this project's success.

I wish to thank Nordic Semiconductor for providing us with the nRF52833 development kits used to construct our prototype. I am also very grateful for the help of Erlend Klokkervold from the Norwegian Mapping Authority, providing me with assistance and early access to brand-new elevation data resources.

Furthermore, I want to thank the student organizations Abakus, TIHLDE, and Online for the free coffee that fueled the writing of this thesis.

Finally, I wish to thank my supervisor Svein-Olaf Hvasshovd for the guidance, encouragement, and feedback throughout the entire project.

Abstract

Locating free-ranging animals in large geographic areas is a challenging task many sheep farmers struggle with every fall when the sheep are to be gathered. The time required to gather the animals can be significantly reduced by using technology to aid the sheep-locating process. Existing solutions for technology-based animal tracking often utilize bulky and expensive GPS collars.

This thesis is one of three reports covering the research and development of a proof-of-concept prototype locating sheep using small and lightweight radio chips. The radio chip is tiny enough to fit in a sheep's ear ID tag and can be detected by other tags located over a kilometer away. The system uses an autonomous drone to traverse large areas in search of sheep. The drone is equipped with an equivalent chip able to detect the sheep ear tags. When the drone detects sheep, it stores its GPS location and distance measurement to the sheep. These measurements are used to estimate each sheep's position in the terrain. A ground control station software on the operator's PC manages the flight planning and sheep data processing.

In this document, the ground control station software and the location estimation are described and evaluated. A review of state-of-the-art ground control stations has been conducted to provide a baseline for understanding and to refine our system's requirements. This thesis presents the system architecture and implementation of the ground control station and the location estimation algorithms.

The project group has performed numerous tests and field experiments to develop and evaluate our prototype. The results from the most significant tests are summarized and discussed.

The system prototype we have created is showing great potential. The system has few external dependencies, relying only on map and elevation data from outside sources. We have demonstrated that our system can autonomously survey a large area and locate sheep within it with outstanding accuracy. On average, our experiments show capabilities of estimating sheep within 19 meters of their actual location. This result is more than acceptable in a real-world scenario, as the sheep would eventually move by the time the farmer can travel to the reported location.

In summary, we have successfully created a proof-of-concept prototype able to locate sheep using a ground control station, an autonomous drone, and a set of communicating radio chips. The system is cheaper than its competitors, and it causes no extra inconvenience for the animals. Our test results suggest that this concept is a feasible solution for locating sheep on a large scale.

Sammendrag

Å spore opp frittgående dyr over store geografiske områder er en utfordrende oppgave for mange sauebønder når sauene skal sankes inn på høsten. Ved å bruke teknologi som hjelpemiddel i letearbeidet kan tiden som kreves for å sanke sauene reduseres betraktelig. Eksisterende løsninger for teknologibaserte sporingsløsninger for dyr baser seg som oftest på dyre og tunge GPS-halsbånd.

Denne rapporten er én av tre masteroppgaver som dekker undersøkelser og utvikling av et konseptbevis for et system som sporer sau ved hjelp av kompakte og lette radiobrikker. Radiobrikkene er små nok til å plasseres inne i sauenes øremerker, og den kan oppdages av andre radiobrikker på over en kilometers avstand. Systemet benytter en en tilsvarende radiobrikke festet til en autonom drone som skal traversere store områder på leting etter sau. Når dronen oppdager sau vil den lagre sin GPS-plassering sammen med en avstandsmåling til sauen. Disse målingene kan brukes for å estimere sauenes posisjon i terrenget. Bakkestasjonsprogramvare som kjøres på operatørens datamaskin planlegger flyruter og prosesserer sauedata.

Dette dokumentet beskriver og evaluerer bakkestasjonen og metodene for posisjonsestimering. For å legge grunnlaget for utviklingen av bakkestasjonen i dette prosjektet har det blitt gått gjennom et sett moderne bakkestasjoner for å forstå og utvikle systemets krav. Denne avhandlingen presenterer systemarkitektur og implementasjon av bakkestasjonen og algoritmene for posisjonsestimering.

Prosjektgruppa har gjennomført en rekke tester og eksperimenter i felt for å utvikle og evaluere systemet. Resultatene fra de mest signifikante testene er oppsummert og diskutert.

Prototypen av systemet som vi har utviklet viser et stort potensiale. Systemet har få eksterne avhengigheter, da det kun er kart- og høydedata som hentes fra kilder utenfor systemet. Vi har demonstrert at vårt system kan autonomt gjennomsøke et stort geografisk område og spore opp sau med fremragende nøyaktighet. Eksperimentene våre viser at systemet vårt kan spore sau med en gjennomsnittlig nøyaktighet på 19 meter. Dette resultatet er mer enn godt nok for å finne en sau i virkeligheten, da sauen uansett vil bevege seg noe innen bonden rekker å dra til den oppgitte posisjonen.

Oppsummert har vi laget et konseptbevis som kan gjenfinne sau ved bruk av en bakkestasjon, en autonom drone og et sett med kommuniserende radiobrikker. Sammenliknet med konkurrerende systemer er vårt billigere og mindre til hinder for dyrene. Testresultatene våre antyder at dette er et gjennomførbart konsept for sporing av sau i stor skala.

Contents

List of Figures

List of Tables

Abbreviations

API ..... Application Programming Interface

BLE ..... Bluetooth Low Energy

BVLoS ..... Beyond Visual Line of Sight

CPP ..... Coverage Path Planning

GDoP ..... Geometric Dilution of Precision

GCS ..... Ground Control Station

GPS ..... Global Positioning System

GUI ..... Graphical User Interface

IoT ..... Internet of Things

MAVLink .... Micro Air Vehicle Link

MSL ..... Mean Sea Level

ROI ..... Region of Interest

RTL ..... Return to Launch

RTT ..... Round-Trip Time

SITL ..... Software in the Loop

SoC ..... System on Chip

ToA ..... Time of Arrival

UART ..... Universal Asynchronous Receiver-Transmitter

UAV ..... Unmanned Aerial Vehicle

UI ..... User Interface

UDP ..... User Datagram Protocol

VLOS ..... Visual Line og Sight

Chapter 1

Introduction

Sheep farmers leaving their sheep grazing in outfield pastures throughout the summer meets a significant challenge of gathering them all again in the autumn. Localizing the herd is a time-consuming exercise that goes through multiple search stages, spanning several weeks. In the beginning, most of the sheep are gathered from the main grazing areas, but after this, the farmer might not have an idea of where the rest of the herd is located. The farmer will have to spend a long time searching large geographic areas, only relying on luck and occasional observations from people in the immediate area.

This project aims to develop a system that automates the search process, allowing the sheep farmer to localize his sheep without searching the outfield pasture himself. This system will contain three components: An autonomous drone, a set of computer chips communicating with radio technology, and a ground control station to operate the drone and present the location of every sheep found by the system. Every sheep shall be equipped with a small Bluetooth radio chip in their ear ID tag, powerful enough to communicate with an equivalent chip attached to the autonomous drone, up to hundreds of meters away from each other. Through the ground control station, the operator can choose a large area the perform a search, whereby the ground control station will generate a flight mission for the drone to cover the selected region. When the operator transfers the mission to the drone, it will follow its designated flight path and pick up signals from nearby sheep. An estimation of the distance between them can be determined based on the round-trip time of the packets sent between the sheep and drone Bluetooth chips. The drone will be equipped with GPS, allowing it to persist the location of every distance estimation sample. When the drone returns, it will offload the findings to the ground control station. Every sheep's location can be estimated with a sufficient margin of error by using range-only tracking techniques. This system will save the sheep farmer a large amount of time-consuming work localizing his herd, allowing him to fetch every sheep directly.

1.1 Ground Control Station

This thesis will address the development of *Radio Sheep GCS*, the ground control station (GCS) component of this system of radio-tracked sheep. A GCS refers to a software or combination of hardware and software responsible for facilitating human control of unmanned aerial vehicles (UAVs). With the rise of UAVs on the consumer market, a rise of open-source and commercially available GCSs has followed. The

most common features of a GCS are a map screen allowing for pre-planning a UAV flight and visualizations of the UAV's flight telemetry such as location and velocity.

Other existing GCS alternatives' lack of modifiability and extensibility motivated the development of a completely new GCS for this project. In addition to the standard GCS features, this project requires the software to receive custom sheep radio measurements, estimate the sheep locations, and visualize where the sheep have been found. Closed-source alternatives cannot be modified, while the open-source variants have large and relatively old code-bases, making it less time-consuming to develop the GCS from scratch.

This document analyzes requirements and evaluates different state-of-the-art ground control stations to develop a GCS for this project. To find and validate the requirements, the project group has conducted a series of both simulation-based and real-world tests, which this thesis presents.

1.2 Range-Only Tracking

Tracking with range-only information is a concept where an unknown location is estimated based on distance measurements from base stations with known positions. The drone in our project acts as a mobile base station that moves to several locations performing distance measurements to unknown sheep locations.

The accuracy of the estimated locations needed is relatively low, as the sheep might move from the found places before the drone returns from its search mission. An error distance of a couple of hundred meters might be sufficient for the farmer to locate the sheep.

This thesis describes the implementation of two different location estimation algorithms based on range-only measurements. One method uses a particle filter and statistical calculations to estimate positions, and the second algorithm uses geometrical multilateration to determine possible sheep locations. The algorithms are tested and evaluated with simulation and real-world experiments.

1.3 Aim of this Thesis

Existing solutions of technology-based animal tracking often contain a bulky and expensive GPS collar, which must have a wireless connection to external networks. The problem with this approach is that you cannot track every animal, and the animals have to be in the range of a radio tower. For instance, lambs cannot carry a collar since they will be growing too much in the outfield pasture, but they still are equipped with an ear tag. The other issue is that sheep often move out of range of radio towers, making a GPS collar useless. This thesis aims to provide the research and development of the ground control station in a proof-of-concept system that localizes sheep using a GCS, an autonomous drone, and a set of radio chips small enough to fit in a sheep's ear tag. A brief evaluation of the complete system prototype will also be presented.

1.4 Outline of this Document

This master thesis structures the chapters in the following way:

Chapter 2 - Background Theory contains relevant background knowledge used to develop the ground control station and location estimation algorithms.

Chapter 3 - State of the Art presents previous work done within the field of ground control stations and sheep tracking.

Chapter 4 - System Architecture provides a full overview of the complete system, as well as the architecture of the ground control station and its sheep localization methods. This chapter will also present the tools and software libraries used to develop the system.

Chapter 5 - Results and Discussion presents the conducted tests used to develop and evaluate the system prototype.

Chapter 6 - Conclusion and Future Work summarizes the thesis with a review of the conclusions extracted from this project, as well as sketching some notes on the further development of this system.

Chapter 2

Background Theory

This chapter contains relevant background theory evaluated and applied to develop the ground control station and the location estimation algorithms.

2.1 MAVLink

MAVLink is an XML-based message protocol designed for communication between a drone and a ground control station and between the onboard components of a drone [1]. The Swiss Lorenzt Meier created the MAVLink protocol in 2009 and has since 2010 been an open-source project with an active community with over 100 contributors [2].

2.1.1 Messages

The MAVLink protocol exchanges messages with a hybrid of the publish-subscribe and point-to-point design pattern [1]. The general exchange of information messages (known as *data streams*) uses the publish-subscribe design pattern, allowing the GCS to subscribe to selected topics of relevant information messages from the drone. More advanced message exchanges that may require two-way communication use the point-to-point design pattern with retransmission. Retransmission allows an onboard drone component to directly communicate with the GCS without a direct connection, as it can rely on the other MAVLink-connected devices in the network to retransmit the message to the correct addressee. For instance, a message can be addressed to the GCS by an onboard drone component, sent to the drone autopilot, and retransmitted to the GCS.

MAVLink is agnostic to which medium it is transferred through, allowing both TCP and UDP over cable or wireless networks.

MAVLink deploys in two major versions, v1 and v2, where the communicating components negotiate the version to use. Version 1 was widely adopted in 2013, and numerous legacy peripherals still use it [3].

The MAVLink version 1 packet format, as seen in figure 2.1, has a small footprint with only 8 bytes of packet headers. The small packets are better suited for low-bandwidth systems, making MAVLink v1 a viable option to use in new projects.

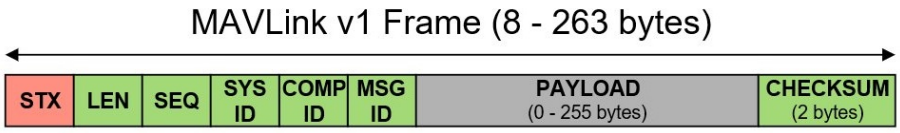


Figure 2.1: MAVLink 1 packet format.
[4]

MAVLink 2 is a backward-compatible update to the MAVLink protocol that is more extensible and feature-rich than its predecessor. Version 2 has a larger packet frame of 12 bytes, as presented in figure 2.2, and is to support more unique message IDs and extension fields. MAVLink 2 can optionally support packet signing that allows authentication of messages sent by trusted systems [4]. This feature introduces 13 more header bytes, making the packet header's total size 25 bytes.

Figure 2.2: MAVLink 2 packet format.
[4]

The *STX* field, known as the *magic byte*, is used by a MAVLink system to identify the beginning of a MAVLink packet and to differentiate a v1 and v2 message. MAVLink v1 would always use 0xFE as the STX value, and MAVLink v2 always uses 0xFD.

2.1.2 MAVLink Networks

A MAVLink network consists of systems such as ground control stations and UAVs, which may be composed of one or several components [5]. Each MAVLink network can manage 255 systems where each system can contain 255 components. Systems and components are assigned a unique ID value between 1 and 255. A MAVLink component can send messages to all systems, specific systems, all components in a system, or specific components within a system. ID 0 is reserved for broadcast messages, i.e., messages intended for all systems or components within a system.

When assigning system IDs, it is expected to use low numbers for autopilots and high numbers for ground control stations. Component IDs have reserved IDs for standard sensors such as cameras and GPS receivers and have a range of unassigned IDs for arbitrary use.

MAVLink networks have a routing system where components are expected to forward or resend messages intended for other components that haven't seen the message yet. This mechanism ensures that all components in the network can communicate directly, even without a direct communication link. E.g., a GCS can communicate with a UAV's battery, even if the battery does not have a radio transmitter, by sending messages to the flight controller that will forward the message to the battery.

2.1.3 Microservices

The MAVLink microservices are a set of higher-level protocols the MAVLink-enabled devices can implement to secure interoperability between systems.

Heartbeat/Connection Protocol

A system on the MAVLink network uses MAVLink's heartbeat protocol for advertising its presence [6]. A component's heartbeat allows other actors on the network to discover it and infer if it is connected or not. The heartbeat message should contain information about the sender, such as component type (e.g., GCS or flight controller) and MAVLink version. Table 2.1 presents a complete overview of the contents of a heartbeat message. To infer if a component is connected, a heartbeat must be sent at a given interval of 1 Hz. A component is considered disconnected if five subsequent messages are not received.

Table 2.1: MAVLink Message: HEARTBEAT.

Field Name	Description
type	Vehicle or component type. For a flight controller component the vehicle type (quadrotor, helicopter, etc.). For other components the component type (e.g. camera, gimbal, etc.). This should be used in preference to component id for identifying the component type.
autopilot	Autopilot type / class. Use MAV_AUTOPILOT_INVALID for components that are not flight controllers.
base_mode	System mode bitmap.
custom_mode	A bitfield for use for autopilot-specific flags.
system_status	System status flag.
mavink_version	MAVLink version, not writable by user, gets added by protocol because of magic data type: uint8_t_mavlink_version.

[7]

A GCS can utilize a drone's heartbeat to determine its MAVLink version and autopilot type. The autopilot flag in the heartbeat message allows the GCS to adjust its communication protocols according to the drone type it communicates with.

Mission Protocol

MAVLink's mission protocol is used to exchange a flight plan (mission) between a GCS and a drone [8]. The mission protocol describes operations to upload and download a mission, clear mission, and assigning the current mission item number.

A MISSION_ITEM_INT message specifies commands for the drone to execute. The message structure is shown i table 2.2. Every action a drone can execute is formatted in this structure, specified by the *command* field. Depending on the given command, seven parameters can be set with arbitrary data, whereby the x, y, and z fields are typically reserved for positional data.

Table 2.2: MAVLink Message: MISSION_ITEM_INT.

Field Name	**Description**
target_system	System ID of the addressee
target_component	Component ID of the addressee
seq	Waypoint ID (sequence number). Starts at zero. Increases monotonically for each waypoint, no gaps in the sequence (0,1,2,3,4).
frame	The coordinate system of the waypoint.
command	The scheduled action for the waypoint.
current	If this mission item is the current one. false:0, true:1.
autocontinue	Autocontinue to the next wayppoint.
param1	PARAM1, depends on the command field.
param2	PARAM2, depends on the command field.
param3	PARAM3, depends on the command field.
param4	PARAM4, depends on the command field.
x	PARAM5 / x position: local: x position in meters * 10^4, global: latitude in degrees * 10^7
y	PARAM6 / y position: local: y position in meters * 10^4, global: longitude in degrees * 10^7
z	PARAM7 / z position: global: altitude in meters (relative or absolute, depending on frame.
mission_type	Mission type.

[7]

For a mission to be uploaded, a sequence of messages must be sent between the GCS and the drone as specified in figure 2.3. The GCS must first initiate the exchange with a MISSION_COUNT message specifying the number of MISSION_ITEM_INTS to be transmitted. Upon receiving this message, the drone shall request the specified amount of MISSION_ITEM_INTS sequentially with a MISSION_REQUEST_INT message. The same MISSION_ITEM_INT will be requested multiple times if not received, allowing reliable transmission of missions over a potentially lossy link. Any MISSION_ITEM_INTS received out of order will be dropped by the drone. When the drone has received all the requested MISSION_ITEM_INTS, it sends a MISSION_ACK message to the GCS to signal a complete transmission.

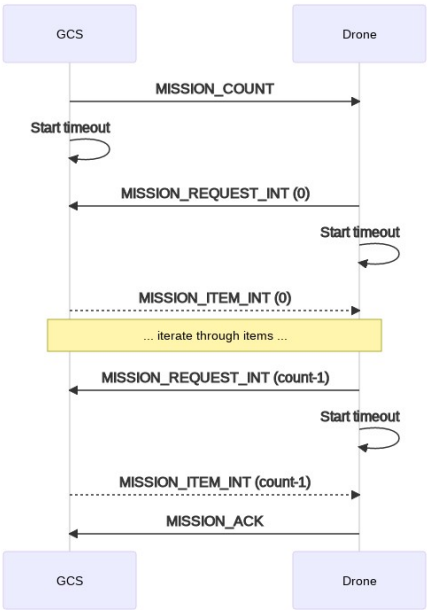


Figure 2.3: MAVLink mission protocol upload sequence.

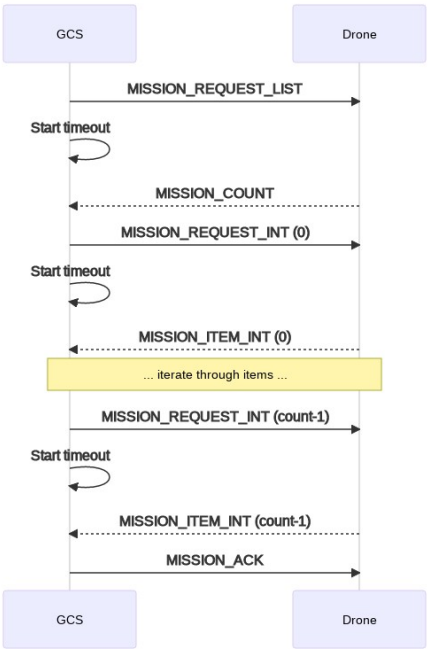


Figure 2.4: MAVLink mission protocol download sequence.

The GCS can download the drone's onboard mission in a similar pattern, as described in figure 2.4. To signal a download transmission, the GCS sends a MISSION_REQUEST_LIST to the drone, which answers with a MISSION_COUNT message containing the number of MISSION_ITEM_INTS it currently possess. Now, the GCS can sequentially request every MISSION_ITEM_INT in the same way the drone would do for the upload sequence. When the GCS has successfully received all the mission items, a MISSION_ACK must be sent to the drone.

To clear the drone's onboard mission, the GCS can send a MISSION_CLEAR_ALL message. When the drone receives this message, it should remove all of its mission items and return a MISSION_ACK message to the GCS, as shown in figure 2.5

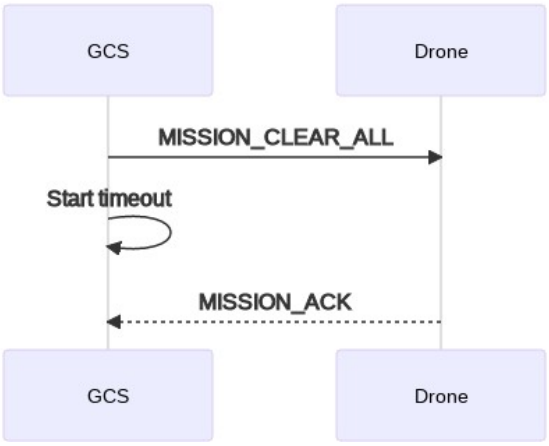


Figure 2.5: Mission clear sequence.
[8]

Command Protocol

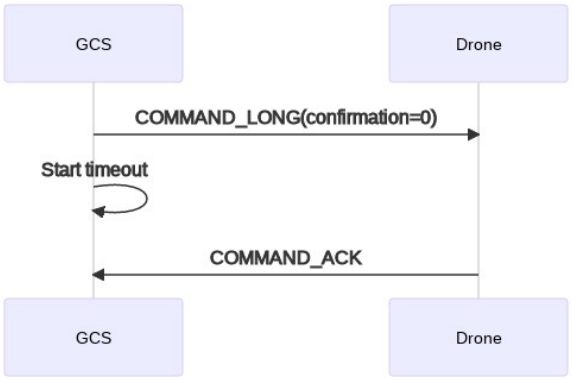
MAVLink's command protocol provides guaranteed delivery of commands from a GCS to a MAVLink-enabled component [9]. A GCS sends commands with a COMMAND_LONG message (table 2.3) encoding a command from a set of predefined commands. The *command* field specified which command to execute, and param 1 to 7 fields are used to relay necessary data for each specific command, e.g., the desired speed for a MAV_CMD_DO_CHANGE_SPEED command.

Table 2.3: MAVLink Message: COMMAND_LONG.

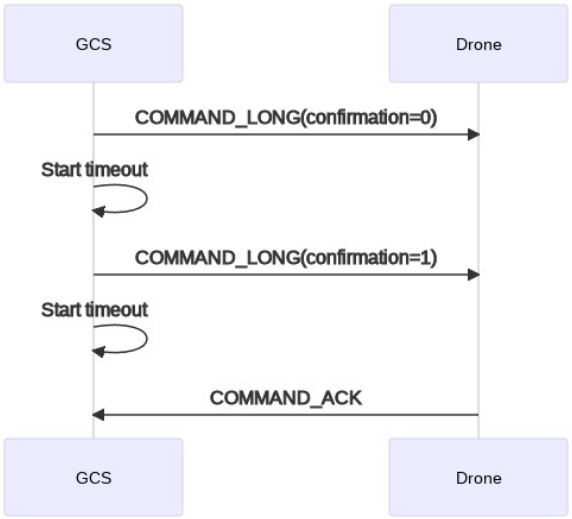
Field Name	Description
target_system	System which should execute the command.
target_component	Component which should execute the command, 0 for all components.
command	Command ID (of command to send).
confirmation	0: First transmission of this command. 1-255: Confirmation transmissions.
param1	Parameter 1 (for the specific command).
param2	Parameter 2 (for the specific command).
param3	Parameter 3 (for the specific command).
param4	Parameter 4 (for the specific command).
param5	Parameter 5 (for the specific command).
param6	Parameter 6 (for the specific command).
param7	Parameter 7 (for the specific command).

[7]

The receiving component should reply with a COMMAND_ACK message to signal that it has accepted the command. Figure 2.6 shows a command message exchange between the GCS and the drone.



(a) Successfully sending a command.



(b) Acknowledgement message failed to transmit.

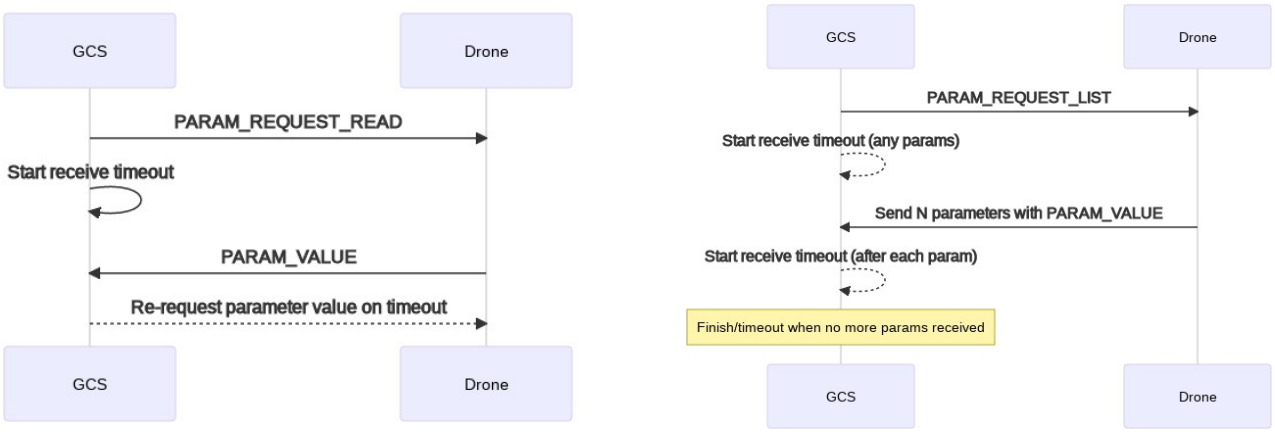
Figure 2.6: MAVLink command protocol sequence.
[9]

In the event of a transmission failure of the acknowledgment (fig 2.6b), the GCS must not retransmit the same COMMAND_LONG again, as the target might have already executed the command. The GCS must encode a new command with the confirmation flag incremented. An incremented confirmation flag signals that the drone must not execute the command again but that the acknowledgment must be retransmitted. This mechanism ensures that commands are reliably sent to the drone without the risk of the same command being executed multiple times.

Parameter Protocol

The parameter protocol is required to manage a MAVLink-enabled component's configuration settings [10]. Each parameter is stored as a key/value pair with a human-readable key name. A parameter can contain arbitrary data, enabling values unknown to the GCS to be read and set. A GCS typically keeps a local cache of all of a MAVLink device's parameters.

Figure 2.7 shows how a GCS retrieves parameters from a component over the MAVLink network.



(a) Requesting a single parameter.

(b) Requesting all parameters.

Figure 2.7: MAVLink parameter protocol request sequence. [10]

When requesting a single parameter (fig. 2.7a), the GCS specifies the name of the parameter and the target MAVLink component. When requesting all parameters (fig. 2.7b), the GCS only has to specify the target, and the target will send back all of its parameters. Note that the GCS sends no message to acknowledge a received parameter, which means that the GCS cannot identify missing parameters. If a GCS knows that a specific parameter is missing, it should be re-requested according to figure 2.7a.

For the GCS to set a parameter, it follows the sequence in figure 2.8.

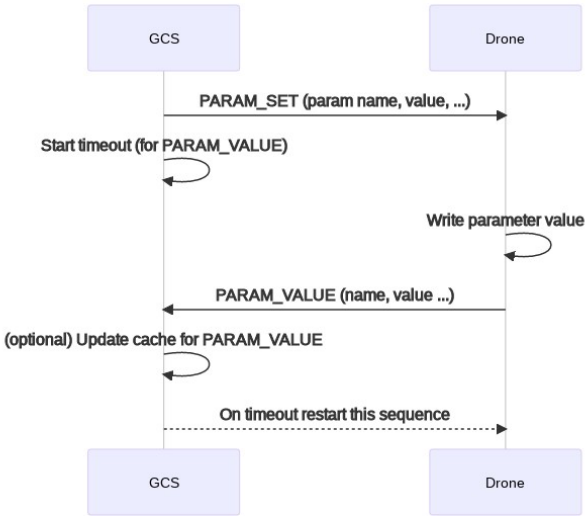


Figure 2.8: MAVLink parameter protocol write sequence.
[10]

The target component acknowledges a PARAM_SET message with a corresponding PARAM_VALUE message. The GCS should use the acknowledgment to update its cache to ensure that the target component accepted the parameter update.

Timeout and Retries

Every message in the MAVLink protocol that requires a response should set a timeout to trigger retransmission of message or abortion of the message exchange [8]. All MAVLink-enabled devices normally operates by the same timeout and retry rules, which are listed below:

- Timeout (default): 1500 ms.
- Timeout (mission items): 250 ms.
- Retries (max): 5.

If the sender receives no response after the maximum amount of retries, it must cancel the operation and return to its idle state.

2.2 Regulation of Drone Use in Norway

The Civil Aviation Authority of Norway (CAA Norway, Luftfartstilsynet) [11] regulates drones used in Norway. As of January 2021, CAA Norway has adopted new EU regulations [12]. These new regulations divide drones into three categories: Open, Specific, and Certified.

Open category is the most nonrestrictive category. This category includes operations with low risk and requires no authorization of the operator. Drones in the open category must comply with the following common rules [13]:

- The drone must weigh less than 25 kg
- The pilot needs to maintain a visual line of sight (VLOS) with the drone.
- The drone must always fly within 120 meters from the closest point of the earth.
- The drone cannot carry dangerous goods.
- The drone cannot drop items.
- The drone must be marked with the operator's registration number.

The open category is further divided into three subcategories A1, A2, and A3. A1 defines light drones (<900g) with few distance limitations to uninvolved people. Flying over crowds is not allowed. A2 defines drones weighing less than 4 kg and enforces a horizontal distance limit of at least 50 meters away from uninvolved people. A3 defines the heavier drones up to 25 kg and requires the drone to fly 150 meters away from residential, commercial, industrial, or recreational areas. Flying over uninvolved people is not allowed.

Specific category applies to more complex drone operations, such as beyond visual line of sight (BVLoS) missions. Most operations not covered by the open category fall under the specific category. All flights within this category require authorization by CAA Norway, which can demand supervision of the flight.

Certified category covers drone operations with a risk profile comparable to crewed aerial vehicles. This category applies to drone operations carrying people or dangerous cargo. Both the drone and the operator must be certified by CAA Norway to allow a flight in this category.

2.3 Location Estimation Techniques

This section presents various methods of localization of an unknown position.

2.3.1 Monte Carlo Localization

Monte Carlo Localization is a sampling-based technique where the density of a set of N random samples or *particles* are analyzed [14]. Variants of this method are generally known as *partilce filters*.

Samples can be weighted to fit the problem domain better. The density of the generated particles represents the probability of the target's state space. A set of samples approximates a probability distribution, which can determine the most probable location.

2.3.2 Multilateration

Multilateration is a localization technique using geometric shapes to determine a position of an object. By performing distance measurements from several known locations to the object, the position of the object can be estimated.

Figure 2.9 shows a 2D multilateration problem with three known locations (known as *trilateration*).

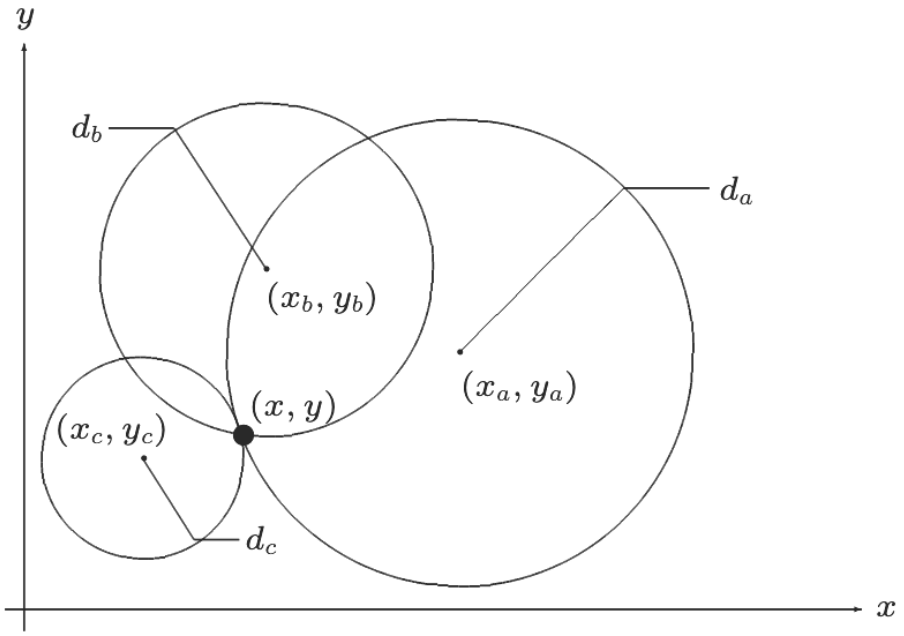


Figure 2.9: Two-dimensional multilateration problem.
[15]

Calculating the intersection point (x, y) can be done by the linearization in equation 2.1 [15].

$$\begin{aligned} d_a^2 &= (x - x_a)^2 + (y - y_a)^2, \\ d_b^2 &= (x - x_b)^2 + (y - y_b)^2, \\ d_c^2 &= (x - x_c)^2 + (y - y_c)^2 \end{aligned} \tag{2.1}$$

This linear system can be simplified to equation 2.2 [15].

$$\begin{bmatrix} (x_c - x_a)(y_c - y_a) \\ (x_c - x_b)(y_c - y_b) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{(d_a^2 - d_c^2) + (x_c^2 - x_a^2) + (y_c^2 - y_a^2)}{2} \\ \frac{(d_b^2 - d_c^2) + (x_c^2 - x_b^2) + (y_c^2 - y_b^2)}{2} \end{bmatrix} \tag{2.2}$$

Given a system with uncertainty in the distance measurements, geometric dilution of precision (GDoP) will occur. Figure 2.10 illustrates how the position of the known locations affects GDoP.

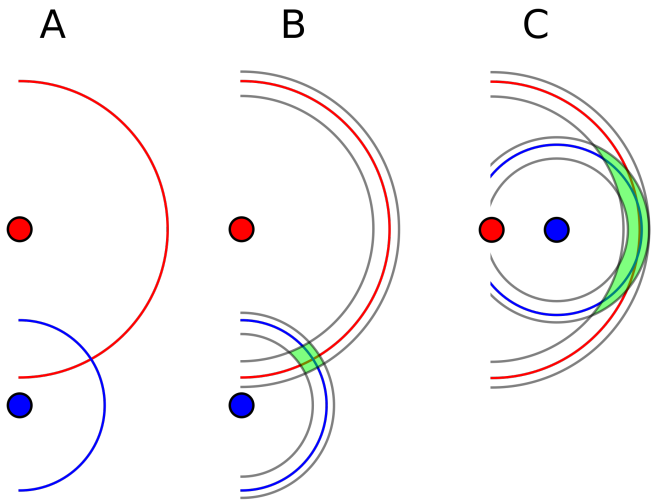


Figure 2.10: Geometric delution of precision in multilateration. [16]

In situation A, no uncertainty is present, and the intersection point is unambiguous. Case B introduces uncertainty to the distance measurements, and the intersection is now an area instead of a single point. Case C shows how poor placement of the known positions increases the intersection area without a higher measurement uncertainty.

Chapter 3

State of the Art

This chapter presents and evaluates previous work in the domain of ground control stations and sheep-tracking. The solutions presented are commercially available products, open-source software, or solutions introduced in literature reviews. No previous ground control station has ever combined flight planning with sheep-tracking, but both exist as separate systems. It is valuable for this project to evaluate both systems and uncover their benefits and flaws.

3.1 Ground Control Stations

This section highlights previous work in the GCS domain. The main focus points towards their implementations of route planning and their quality of user interface. This section will also highlight clever functionality and functions that differentiate the ground control stations.

3.1.1 QGroundControl

QGroundControl (QGC) is a popular cross-platform open-source GCS [17]. QGC runs on both desktop and mobile platforms, supporting Windows, macOS, Linux, Android, and iOS. The user interface is clean and intuitive for simple needs but at the same time packed with advanced functionality for professional operators. QCG supports all autopilot software that communicates with the MAVLink protocol, and it can control multiple vehicles simultaneously.

The route planning utility in QGC is comprehensive and allows for a high level of detail control of the created route. The operator can build a route using waypoints and three different complex flight patterns: *Survey*, *Structure Scan*, and *Corridor Scan*. Survey lets you choose an area on a map where the drone will traverse back and forth through the area until it is completely covered, as shown in figure 3.1. Structure Scan lets you choose an area where the drone will fly around multiple times at different heights, ideal for photographing buildings and structures. Corridor Scan lets you create a flight pattern following a poly-line, which can be used for maintenance and inspections of power lines or train tracks.

Figure 3.1: QGroundControl's Survey flight pattern.

By combining waypoints and these patterns, you can create a complex flight plan, as shown in figure 3.2. In this example, the drone will follow the waypoints 1 to 3 and do a Survey of an area, follow waypoints 44 to 47 and perform a Corridor Scan, fly through waypoints 60 and 61 and execute a Structure Scan, and lastly follow the waypoints 103 to 105 returning to the launch position.

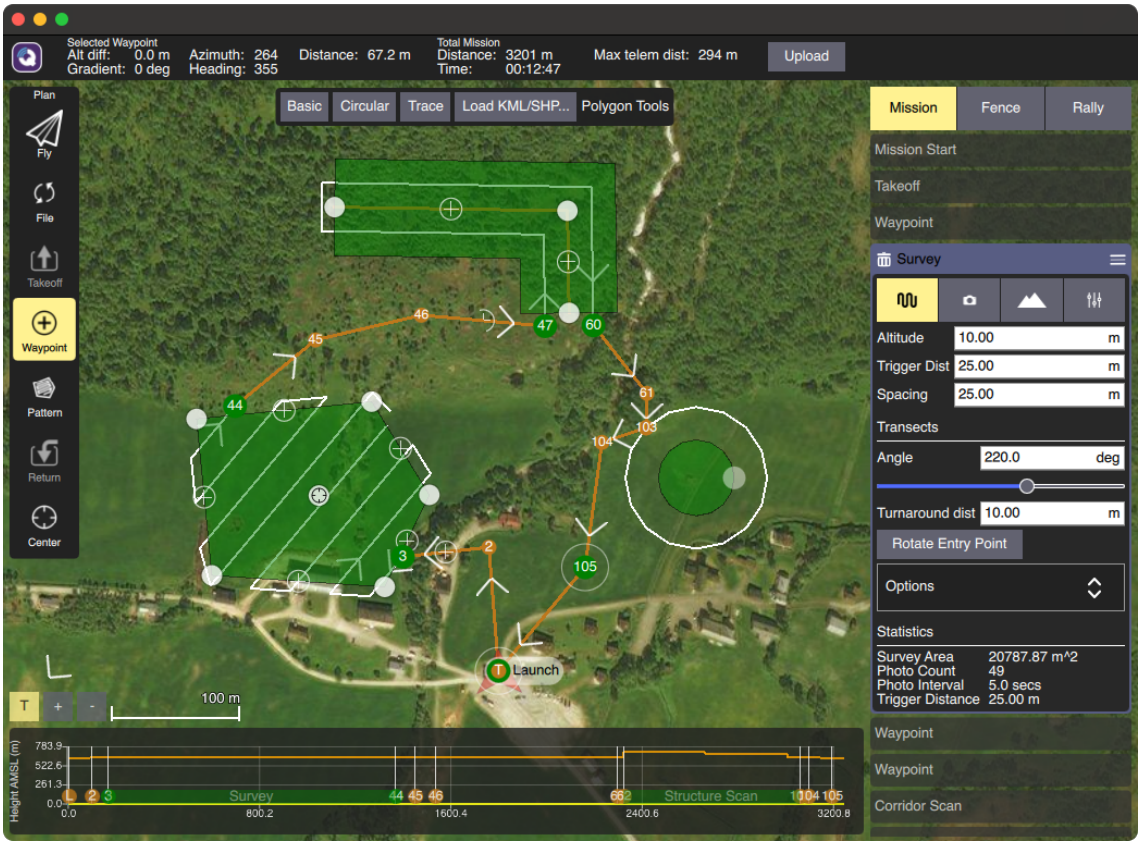


Figure 3.2: Complex flight plan in QGroundControl.

Every element in the flight plan can be individually configured with drone speed and altitude. QGC allows for easy control of the drone while it is currently conducting a mission. The mid-flight drone can quickly be stopped, configured to change speed, or re-routed to a different part of the flight plan.

QGC supports multiple map providers and can download a selected area of the map for offline use.

3.1.2 DJI Ground Station Pro

DJI Ground Station Pro (GSP) [18] is DJI's [19] proprietary ground control station for the drones manufactured by DJI. The application is only available on an Apple iPad, making it the only GCS evaluated that exclusively utilizes touch controls. GSP stands out with a highly polished and responsive user interface, a clear presentation of relevant information, and intuitive controls. DJI offers a subscription service that provides full access to a set of extra functionality and cloud storage of flight plans. For basic use, the application is free of charge.

The route planning functionality in GSP allows for creating five different types of missions, which cannot combine. *PhotoMap* lets you create a map stitched together by photos. To do this, the user draws a polygon marking the area to map. GSP then generates a flight plan for the drone that covers the area and provides commands for the drone to take photographs towards the ground at specified locations. The photos merge automatically to create the desired map of the area. *Virtual Fence* lets the operator fly the drone manually within a set area. The operator sets an area by drawing a polygon on the map, and text input fields allow setting a height and speed limit. When a virtual fence is uploaded to the drone, any attempt to breach the limits will automatically stop the drone, keeping it hovering in the air. The *3DMap Area* and *3DMap POI* missions are used to construct 3D models of areas and structures, respectively. *WayPoint Route* is the most versatile mission, where the user can set individual waypoints for the drone to follow. Every waypoint must be set manually, unlike other GCS alternatives where waypoints can be generated by drawing polygons and lines. One of the positive sides of GSP is that it clearly shows the distance between each neighboring waypoints and shows information about the total length of the mission, how long the flight time will be, and how many batteries are required to complete the route. The user can set altitude and drone speed for each waypoint individually or all waypoints at once. Figure 3.3 shows how a waypoint mission can be in GSP.

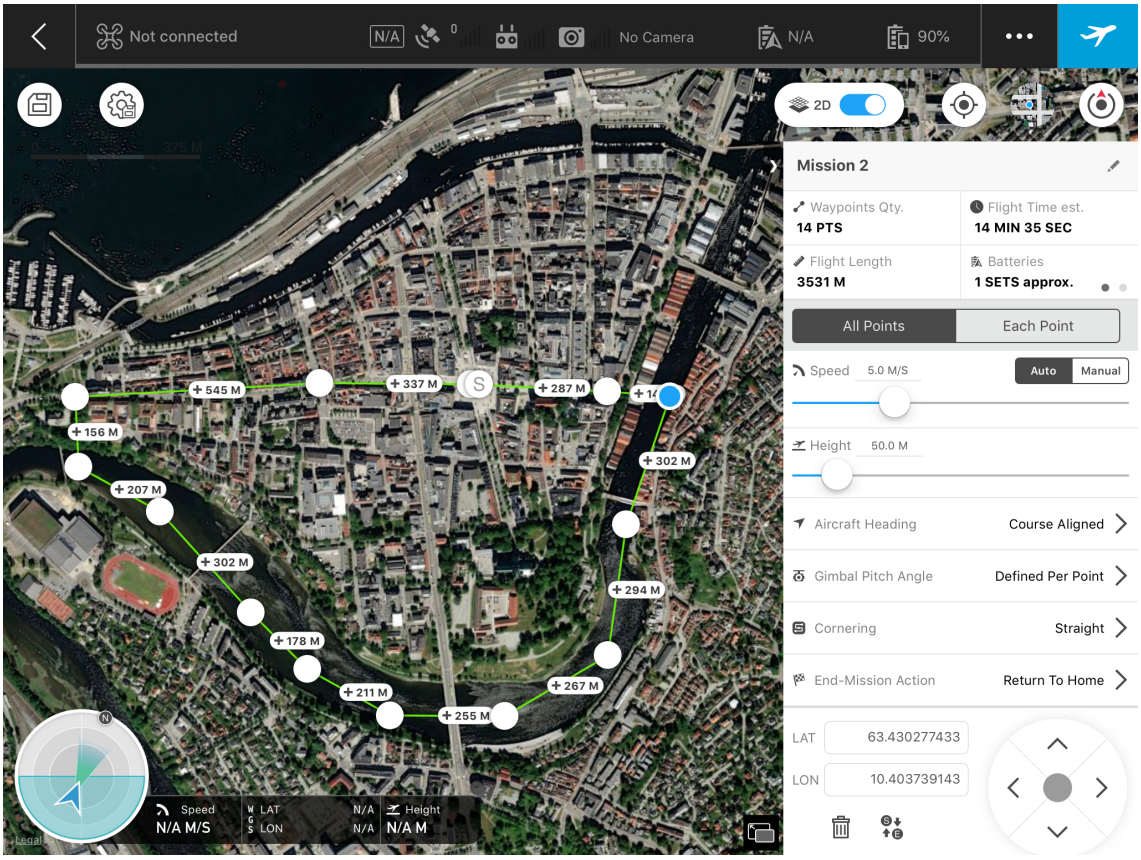


Figure 3.3: WayPoint Route in DJI Ground Station Pro.

All of the mentioned mission types can be created by selecting points and areas on a map, but GSP also allows the user to set the mission parameters using a physical drone. You would then fly the drone to set borders or record waypoints. This feature can be handy for creating missions in complex locations with many obstacles, such as urban areas and construction sites.

One key feature that separates GSP from other GCS alternatives is the digital assistant helping the drone operator to comply with legal regulations. The map clearly shows no-fly zones such as city centers with flight restrictions, military areas, prisons, and airports. It is required for the pilot to register his phone number with SMS verification to start a mission within these areas.

GSP only supports Apple Maps with a standard road view and satellite view, but it has no support for topographic maps. The application allows for downloading selected areas for offline use.

3.1.3 Mission Planner

Mission Planner (MP) [20] is a ground control station officially endorsed by ArduPilot [21] and is the most extensive and feature-rich GCS of the open-source alternatives. MP is a tool where function oversteps the system's usability, and it is only available on Microsoft's Windows operating system. While testing the system, it was prominent that MP is the gold standard for autonomous vehicle control features, despite having somewhat poor usability.

The route planning functionality in MP gives the operator complete control of most aspects of the UAV mission. MP displays a mission as a set of the MAVLink messages that will be transferred to the UAV, and it is possible to edit most data fields to fit your specific needs. MP supports two types of waypoints: Regular waypoints and spline waypoints. With regular waypoints, the drone would arrive at the point, stop, and turn towards the next one. The spline waypoints will do a spline interpolation of the path to the next waypoint, allowing the drone to keep its speed when passing through the point.

MP has excellent tools for generating waypoints to cover a region of interest (ROI). The operator draws a polygon in the software to create a mission surveying an ROI. He is then able to configure how the path will cover the area. Figure 3.4 depicts the ROI survey tool in MP. As we can see from the figure, it is possible to tailor the generated flight path to specific needs, including setting sweep line order, overlap, and overshoot.

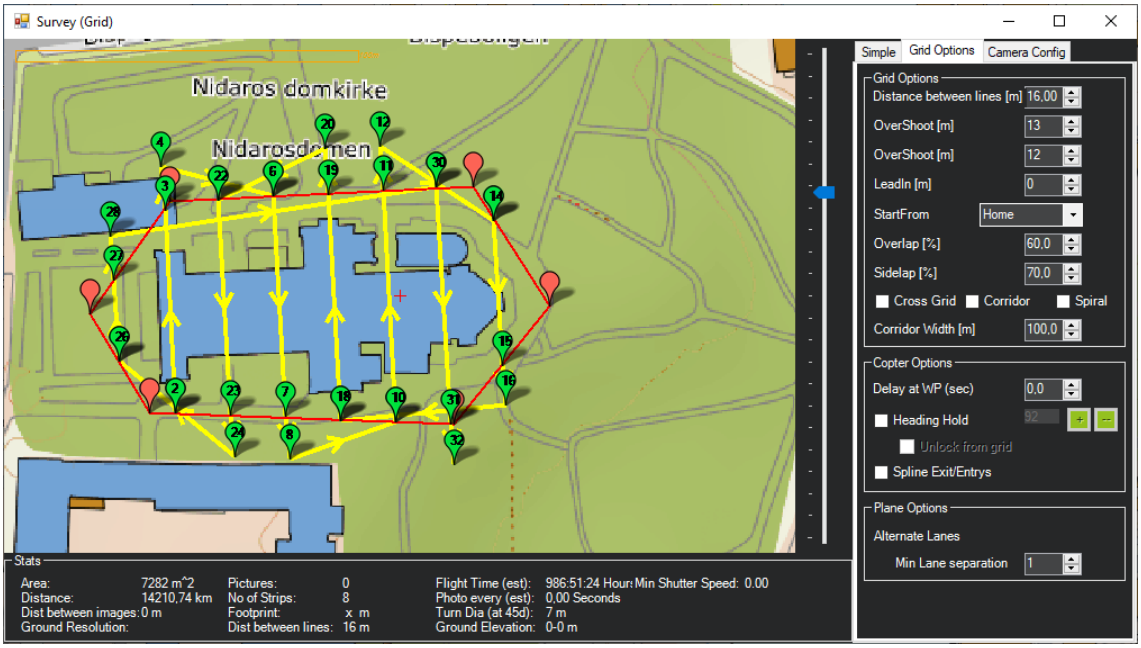


Figure 3.4: Mission Planner's tool to create a survey route over an ROI polygon.

Mission Planner structures a mission as a set of actions that can be moved and edited. This flexibility allows the drone to create missions where the drone has multiple takeoffs and landings during a single flight mission. Most other GCS alternatives have to create separate missions for each takeoff and landing. MP allows for speed and altitude control of individual waypoints. MP also allows for control

of which angle the drone will have when passing a waypoint, mainly helpful when the drone takes photographs along the flight.

By combining single waypoints and waypoints generated by the ROI survey tool, the operator can create complex missions to cover most needs. Figure 3.5 shows an example of a complex flight plan in Mission Planner.

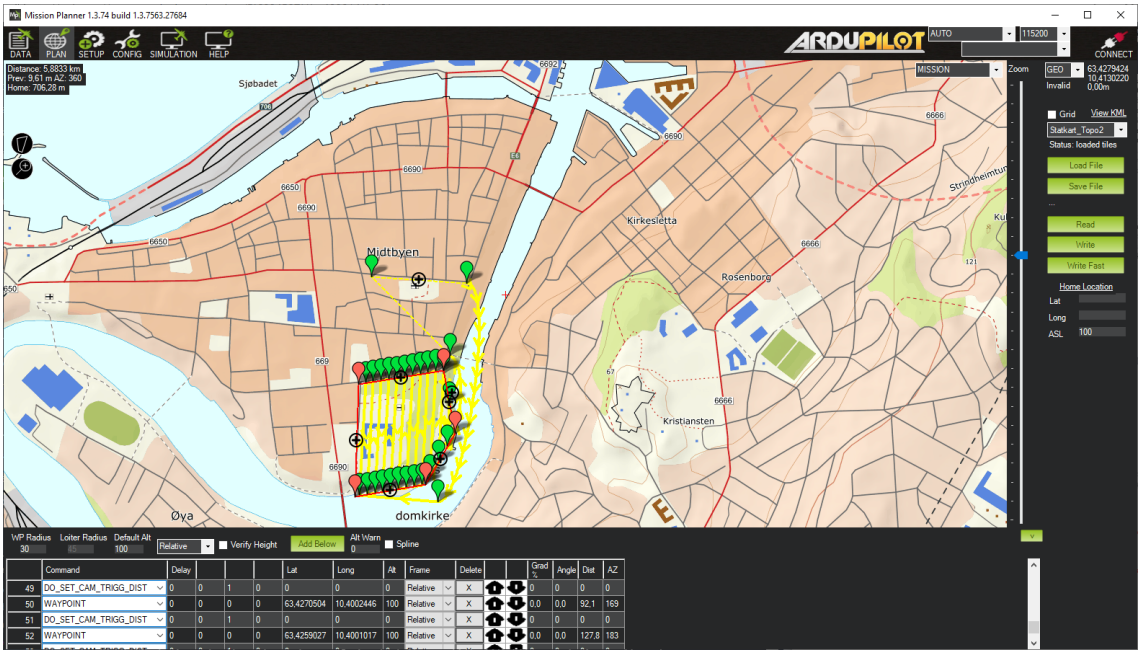


Figure 3.5: A flight plan created in Mission Planner.

As the only GCS of the reviewed alternatives, MP provides a built-in vehicle simulator. This feature is a convenient and cost-effective way to test your mission before running it on real hardware. MP supports several map sources, including area-restricted, high-quality maps such as the Norwegian Mapping Authority's terrain map of Norway. If the user possesses private map sources, MP allows for integrating these types of map sources. MP supports downloading of a user-selected area of the map source for offline use. Airports with belonging restricted areas are marked on the MP map, making it easy to avoid illegal flight areas. Overall, MP requires a higher level of technical competence than most of its competitors but allows for the most precise control of a drone's mission.

3.1.4 APM Planner 2.0

APM Planner 2.0 (APM) [22] is also a Ground Control Station officially endorsed by ArduPilot. ArduPilot addresses APM as a next-generation GCS, combining Mission Planner's user interface and QGroundControl's cross-platform capabilities. Unlike QGroundControl, APM does not support mobile platforms, only the most popular PC operating systems Windows, macOS, and Linux. Mission Planner is still far ahead of APM functionality-wise, but the user interface of APM appears as a more modern and responsive version of MP's interface.

The route planning in APM is easy to use but is too simple for complex flight plans. APM has no support for generating waypoints covering an ROI, which means that the operator must place every waypoint by hand. Like Mission Planner, APM supports regular waypoints and spline waypoints. It has the same possibility of modifying the MAVLink messages as MP, providing a high level of detail control of the mission. Figure 3.6 shows an example of a flight plan in APM.

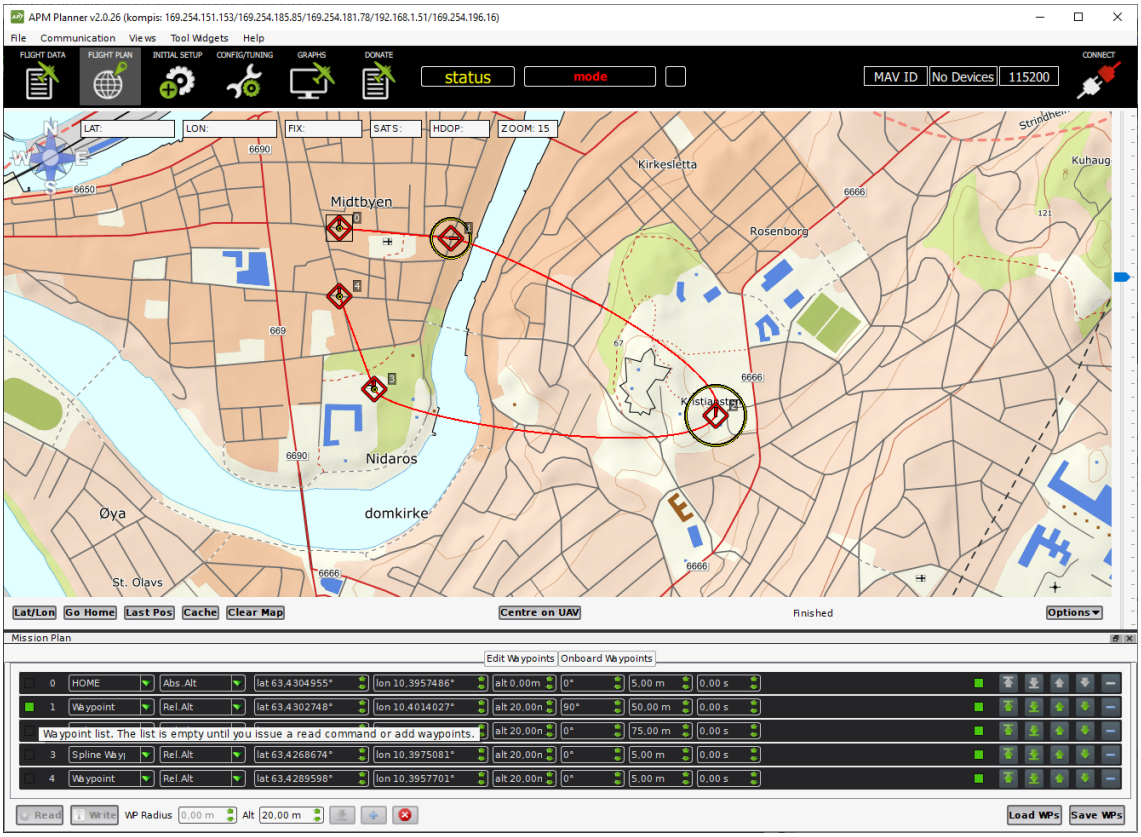


Figure 3.6: Example of flight plan in APM Planner 2.0.

APM's waypoint indicators convey more information than other GCS competitors. These diamond-shaped indicators have a small line pointing in the direction the drone will face when passing the waypoint. The indicators make it easier to see how a mission is going to be executed by a drone. Another unique feature of these indicators is the belonging acceptance radius circle, as seen in waypoints 1 and 2 in the figure. The acceptance radius shows how close the drone has to be to the waypoint before accepting it and moving to the next one.

APM can choose from 6 different map sources, including topographic maps from the Norwegian Mapping Authority. There is support for downloading maps for offline use. APM has one of the easiest methods of doing this, as all the user needs is to hold the shift button while marking the desired area with the cursor. If different zoom levels within the selected area are required, a prompt will appear where the user can choose this.

3.1.5 SmartAP GCS

SmartAP GCS (SAP) [23] is a proprietary GCS created by the UAV hardware and software company Sky-Drones. SAP runs on mobile and desktop platforms but is only compatible with autopilot flight controllers manufactured by Sky-Drones. Sky-Drone's products aim at an enterprise market, with applications such as package delivery, law enforcement, and agriculture. Some of SAP's strong points compared to other GCS alternatives are compliance with regulations, safety measures, and simple multi-UAV fleet control.

The route planning functionality in SAP is relatively basic. A route can be composed of waypoints and *grids*, SAP's ROI survey tool. A neat feature of the ROI survey tool is to specify the distance in meters between the generated sweep lines. Figure 3.7 pictures an example of how a flight plan might look. There is no option to edit the order of the waypoints, which makes the application somewhat rigid compared to other GCS alternatives.

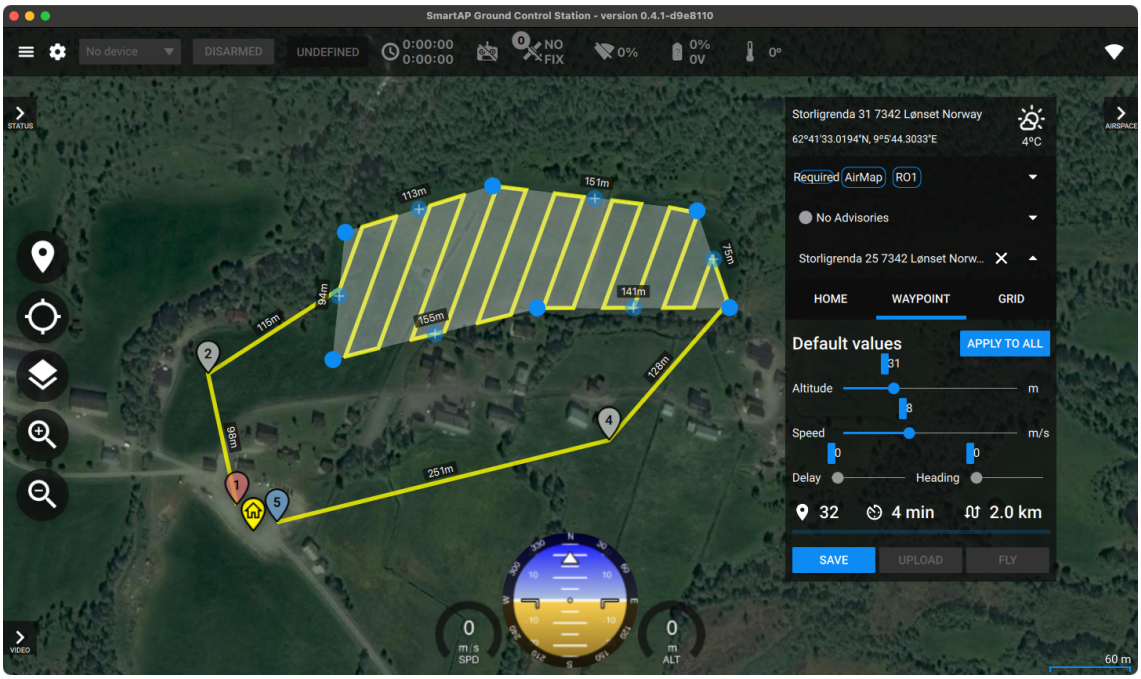


Figure 3.7: Example of flight plan in SmartAP GCS.

Using Sky-Drones' hardware allows for extensive sensor data logging and visualizations in SAP. SAP has an analytics page that allows for the visualization of arbitrary sensor data for sensors connected to the SAP flight controller. This plug-and-play feature is unique to SAP.

SAP uses Google's satellite maps, and there is no possibility to change other map types or map providers. SAP supports downloading of maps for offline use. One of the features differentiating SAP from other GCS alternatives is a digital assistant helping the operator decide if a flight mission is safe to perform. Weather data for your location and information from the airspace intelligence platform AirMap [24] powers the safety assistant. The assistant helps the operator locate restricted airspaces such as airports, prisons, and military training fields.

3.2 Sheep-Tracking

Today's existing state-of-the-art systems for tracking sheep in outfield pastures primarily include the use of GPS collars. These systems rely on relatively bulky collars to receive GPS locations and transmit data using mobile networks. The sheep usually wear the GPS bells at the same collar as traditional brass-covered steel bells, in some cases doubling the weight.

3.2.1 Findmy

Findmy is a Norwegian sheep tracking company using GPS collars [25]. The GPS collars utilize a low-orbit satellite network to transfer location data to the operator. The collar can be configured to send positional data at set intervals and send notifications to the farmer when the sheep leaves a pre-defined geofence.

The company reports an expected battery capacity of two to three seasons, varying with the sending interval. The sizes of the collars are slightly larger than traditional bells.

Findmy sells their GPS collars for 1890 NOK per unit with a mandatory subscription cost of 229 NOK per year per tracker.

3.2.2 Telespor

Telespor's *Radiobjella* (4th gen.) is one of the most lightweight sheep-tracking GPS bells on the market, weighing only 104 grams [26]. The bell's form factor is about the same size as a traditional bell. Radiobjella communicates with the operator's base station using LTE-M and Narrowband IoT networks.

The battery is reported to last one season but offers a replaceable battery. The GPS bell's unit price is 899 NOK and requires a mandatory seasonal 5-month subscription. The subscription is bought separately for 99 NOK per bell or combined with a new battery for 119 NOK.

3.2.3 Smartbells

Smartbells' *Smartbjella 2* is another GPS-tracking collar using Narrowband IoT networks [27]. It is about the same size as a traditional bell, and it weighs 170 grams.

Smartbells' system allows both viewings of sheep locations in real-time and sheep location history. The battery life capacity of the collar is reported to be ten years of standard usage and up to 17 years when sending location once per 24 hours. The battery is not replaceable.

Smartbells offers APIs to integrate their data into existing systems.

The unit price of each collar is 949 NOK with a mandatory 200 NOK per season subscription.

Chapter 4

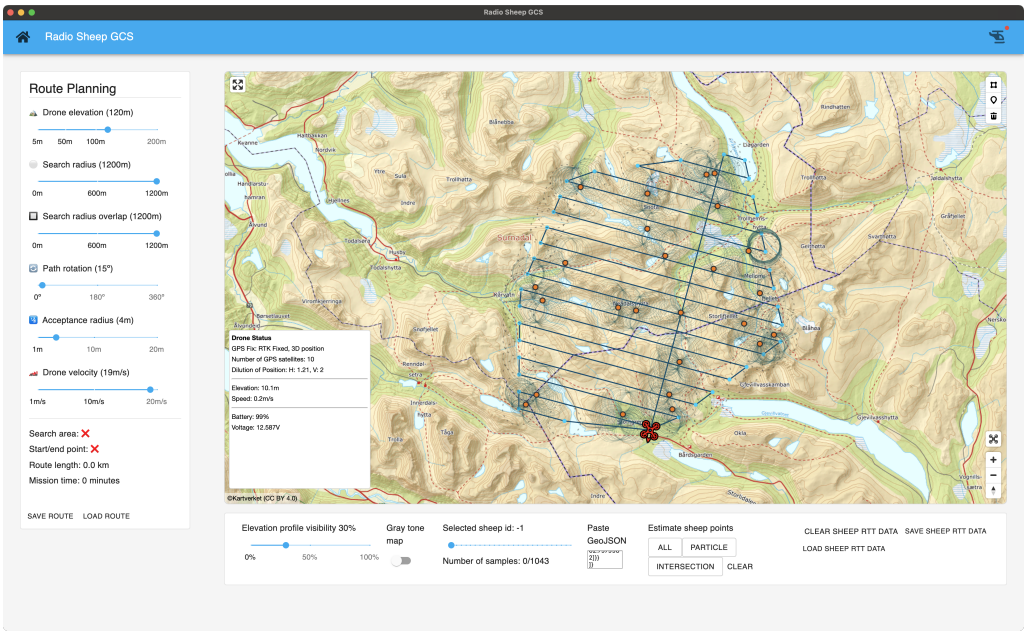
System Architecture

This chapter will present an overview of the system to understand this project's problem domain better. The chapter introduces all the parts making up the composite system, including the components out of scope for this thesis. A more detailed explanation of the ground control station and sheep localization methods, which is the main scope of this report, will also be given in this chapter.

4.1 System Overview

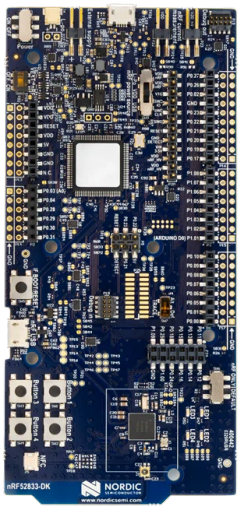
The system's ultimate goal is to locate sheep in outfield pastures, allowing for efficient herding of the animals. This system is a construct of three separate parts: A ground control station, an autonomous drone, and a set of Bluetooth-enabled radio chips.

Figure 4.1 depicts the three components developed in this project.



(a) Radio Sheep GCS, the ground control station component to manage sheep localization missions.

(b) The autonomous drone constructed for this project.



(c) The nRF52833 development kit [28] used to measure distance between the drone and the sheep.

Figure 4.1: The system's three components.

The development and construction of the drone is the scope Nerland's thesis in [29]. Swiderski presents the implementation and evaluation of the Bluetooth module in [30]. Figure 4.2 shows how the different components interoperate.

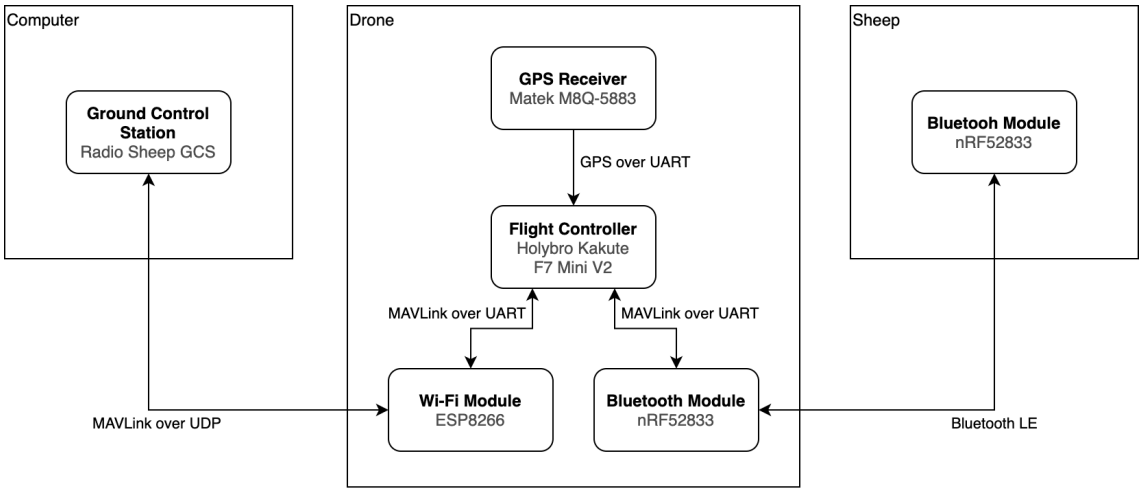


Figure 4.2: Overview of the system architecture.

The ground control station is software running on a computer and is used to plan search missions, communicate with the autonomous drone, and visualize the discovered sheep signals. As an operator, you would select an area on a digital map within the GCS, whereby the software will generate a flight plan to cover the chosen area. The GCS can open a connection to the autonomous drone so that the operator can configure the drone, such as uploading and removing a flight plan. The communication between the GCS and the drone uses the MAVLink protocol. A more detailed explanation of the GCS architecture is provided later in section 4.2.

Every sheep is to be equipped with a Bluetooth module. This module is the same system on chip (SoC) as the drone is equipped with and is namely the nRF52833 [28]. The chip itself is small and light enough to fit in the sheep ear tag, making it convenient and unnoticeable for the animals. For the drone and sheep chips to detect each other, the sheep radio will advertise its presence at regular intervals, followed by opening a small time frame where the drone radio must acknowledge if it has found the sheep. With the established connection, the drone and the sheep send a sequence of packets back and forth. The drone uses these packets to measure the round-trip time (RTT), determining the distance between them. The number of packets n sent back and forth in the sequence is still being researched. Less exchanged packets are more energy-efficient for the Bluetooth chip but are more prone to packet loss resulting in a more considerable error margin. Upon a completed transmission, or if the drone did not find the sheep, the chip would sleep for a set amount of seconds to save battery before waking up to advertise again. This process is described in figure 4.3. The RTT packets are exchanged at constant intervals, allowing both actors to know when to wake up to activate their radios. This mechanism allows for less radio usage and reduced power consumption.

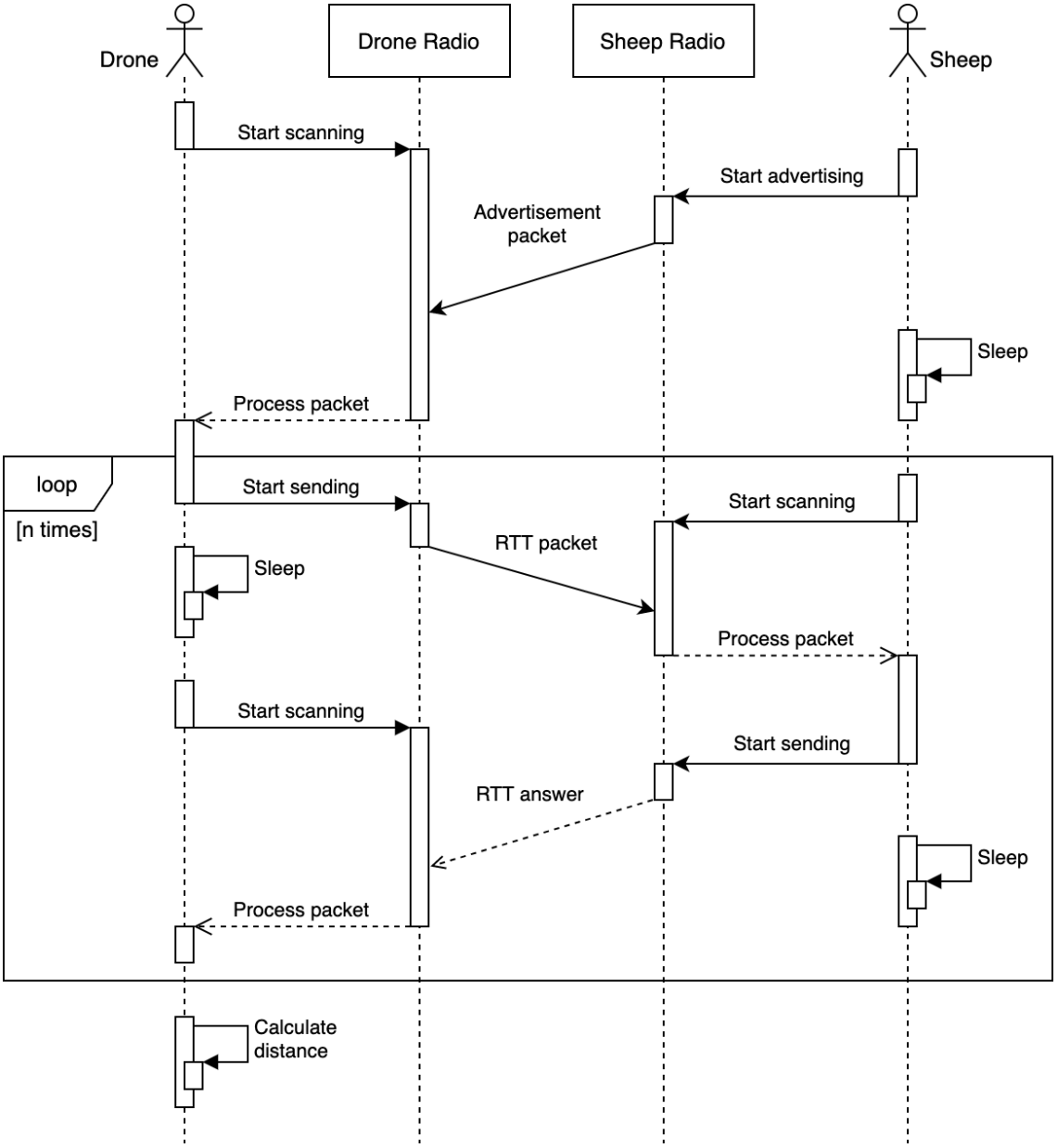


Figure 4.3: Sequence diagram of the radio communication between the drone and the sheep.

The drone used in this project is a custom-built device running the ArduPilot [21] open-source autopilot software. It navigates using instructions and waypoints provided by the GCS along with its onboard GPS receiver. When the drone is executing a search mission, it will continuously try to detect sheep in range. It utilizes the onboard Bluetooth module to set up a short communication sequence with the sheep's Bluetooth module. By measuring the RTT of the messages, the Bluetooth module estimates the distance between the drone and the sheep. The drone will then combine the RTT data with its GPS data to create an information packet, formatted as the MAVLink message in table 4.1, and send it to the GCS for processing and estimation of sheep locations.

Table 4.1: MAVLink Message: SHEEP_RTT_DATA.

Field Name	Description
seq	Sequential sample id (from power on).
lat	Drone latitude (WGS84, EGM96 ellipsoid).
lon	Drone longitude (WGS84, EGM96 ellipsoid).
alt	Drone altitude (MSL).
dis	Calculated distance based on the RTT between drone and sheep.
tid	Sheep identifcation number.
rssi	Signal strength.
timestamp	Time counter in ms since drone powered on.

The drone offloads all of its SHEEP_RTT_DATA packets to the GCS when returning from the search mission. After the exchange, the GCS has to patch together all the information and estimate every sheep's location.

The drone constructed in this project is covered by the open A1 regulation category, making our drone operations relatively unrestricted.

4.2 Radio Sheep GCS

This project's ground control station component, named *Radio Sheep GCS*, is a standalone software application designed to operate the autonomous drone and visualize discovered sheep locations. This section presents a detailed description of the structure of the GCS along with its communication procedures when interacting with external systems.

The source code of Radio Sheep GCS is available online at:
`https://github.com/GardSteinsvik/radio-sheep-gcs`

4.2.1 Application Overview

At a general level, Radio Sheep GCS contains three main components: The user interface, a communication module, and a processing module. The user interface has two primary responsibilities: route planning for the drone and sheep location visualization based on the measured sheep RTT data. The communication module is the part of the system that manages the communication between the GCS and the drone. This module includes parsing and packing MAVLink messages and managing the UDP connection the messages are sent through. All the MAVLink sub-protocols described in section 2.1 are implemented in the communication module. The processing module has the small but complex job of estimating every sheep location based on the received sheep RTT measurements and send them to the user interface. How the different parts of the GCS interact with each other is shown in figure 4.4.

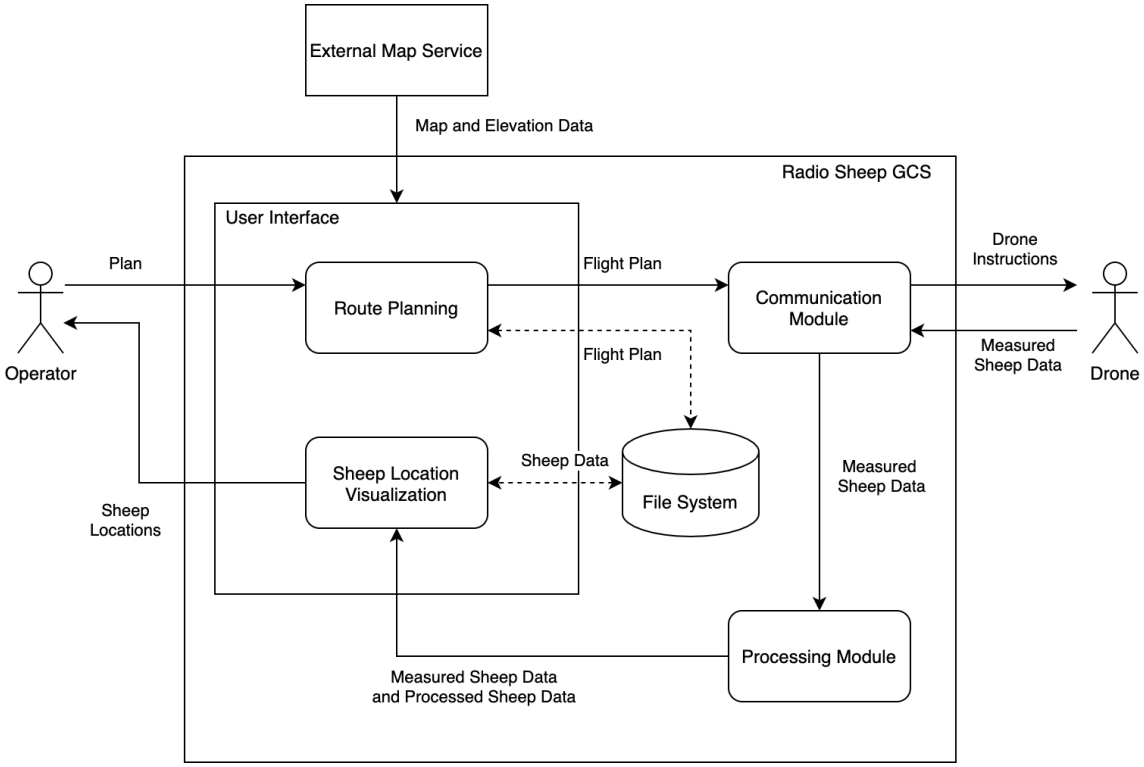


Figure 4.4: Logical view of the Radio Sheep GCS software.

An essential part of the user interface is the digital map, which helps the operator see where the drone will fly, and the found sheep locations. An external source provides the map, in this case, the Norwegian Mapping Authority [31].

The application has the possibility of saving files to persistent storage and retrieve them later. The operator can save a pre-planned search mission for the drone and load it into the system before transferring it to the drone. Radio Sheep GCS can save and load sheep measurements so that the operator can document where the drone has found sheep. All files stored by Radio Sheep GCS follow the GeoJSON [32] standard for predictable and convenient use of geodata.

4.2.2 Mission Planning

An autonomous drone requires a set of instructions to follow to fly on its own. In most cases, this includes waypoints to fly to and configuration of parameters such as speed and heading. Figure 4.5 describes the entire process from opening the planner to starting a mission. Upon opening the route planner, the system fetches the map from the map provider automatically. The map provider distributes the map data as a set of bitmap tiles that have to be stitched together by the application. Mission planning in Radio Sheep GCS consists of setting a start and stop point and selecting an ROI for the drone to survey. Upon selecting the ROI, the GCS will automatically generate a set of waypoints the drone must fly through to cover it all.

As soon as Radio Sheep GCS has generated all waypoints, it sends a request to the map provider asking for each location's elevation and terrain data. The next step is to upload the mission to the drone. The drone and Radio Sheep GCS must establish a connection to allow this. The GCS will open a UDP socket listening for incoming messages that the drone continuously broadcasts. When the first heartbeat message from the drone is received, Radio Sheep GCS will initiate a connection handshake, where they negotiate the MAVLink version to use. Radio Sheep GCS can upload a mission once it has established a running connection. The GCS encodes the flight plan as MAVLink messages and exchanges them with the drone according to MAVLink's Mission Protocol. We can recall the connection handshake and the mission protocol from section 2.1.3 on page 15. The GCS must send a command to the drone that instructs it to arm before starting the mission. The drone will prepare for flight by checking its battery and sensors, followed by starting the rotors. When the drone is armed, the GCS can send a MISSION_START MAVLink command to start the mission.

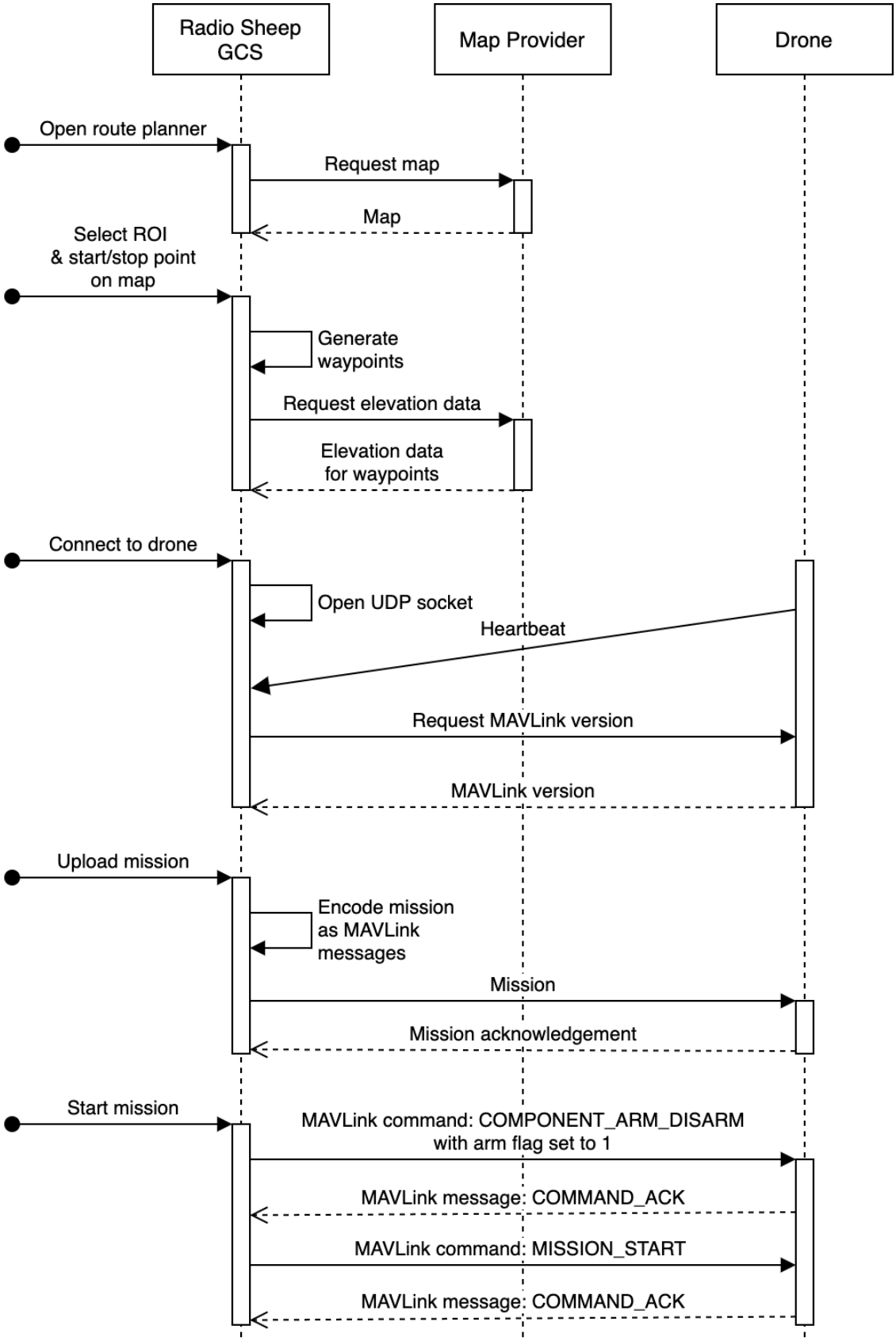


Figure 4.5: Sequence diagram of the mission planning and execution procedure in Radio Sheep GCS.

4.2.3 Flight Pattern

Creating a flight plan is the most central part of mission planning. Ideally, the user would express as much information about his plan using as few GUI operations as possible. In this project, the drone will search a large geographic area, trying to localize sheep. Placing every single waypoint for this task would be inconvenient and time-consuming for the user. Radio Sheep GCS can create a complete flight plan by simply selecting one polygon representing the ROI to search and one point for the drone to start and finish the flight.

Generating a path for a UAV with a coverage area to pass over all the points in an ROI is called Coverage Path Planning (CPP) and is thoroughly described by Coombes et al. in [33]. Covering an ROI can be achieved by generating straight sweep lines over the polygon with a distance D_x between them. Due to GDoP, the generating of sweep lines must choose a value of D_x to detect sheep from multiple sweep lines. Suppose a sheep is detected by a single sweep line only. In that case, the range-only measurements will entail two separate possible locations, as illustrated in figure 4.6.

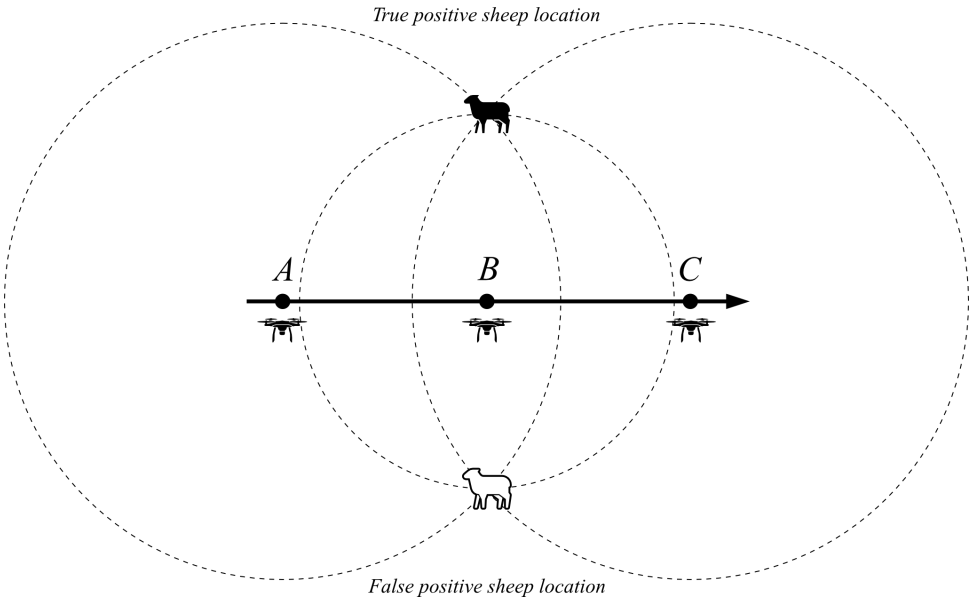
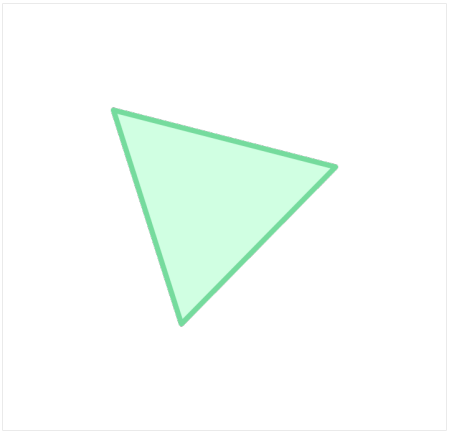


Figure 4.6: Geometric dilution of precision when a sheep is detected by a single sweep line only.

This issue resolves by ensuring that the drone detects the sheep from multiple sweep lines. By setting D_x to the UAV's maximum search radius, every sheep within the coverage area is detected by at least two sweep lines.

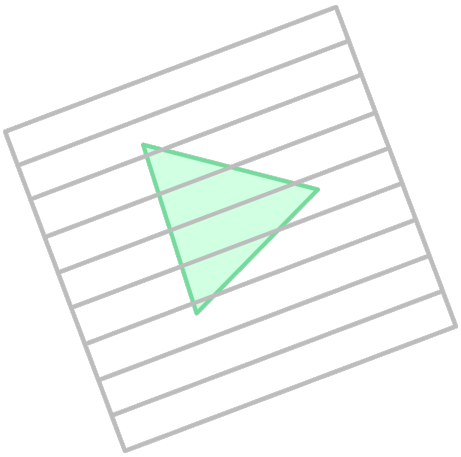
Sweep lines have a problem covering non-convex polygons, as a hole in the polygon can result in a sweep line covering areas of no interest, wasting flight time and the total range of the UAV. CPP of a non-convex ROI polygon is an NP-hard problem, as the system must decompose the polygon into n smaller convex polygons. Radio Sheep GCS has implemented CPP for convex polygons according to the procedure in figure 4.7. The system's current state has not implemented polygon decomposing as it is not strictly necessary to develop a proof-of-concept prototype.



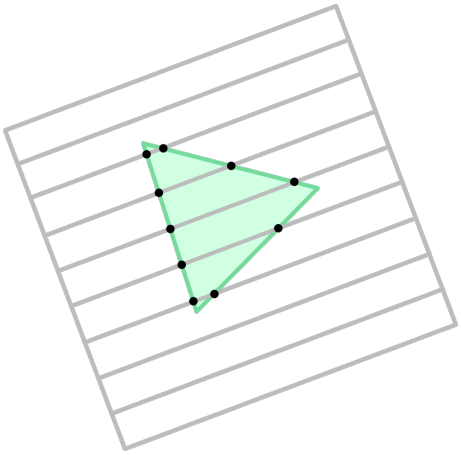
(a) The user creates a ROI polygon for the drone to survey.

(b) A squared boundary box is created around the polygon.

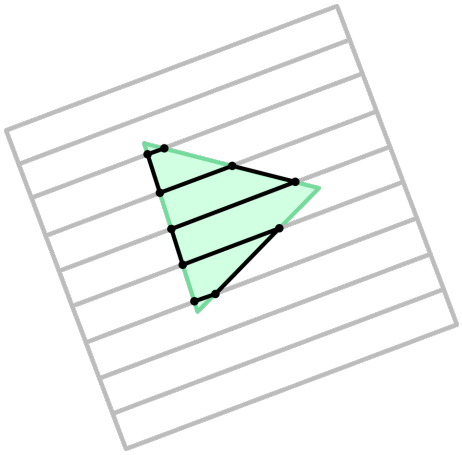
(c) Horisontal sweep lines are created to cover the boundary box.



(d) The sweep lines can be rotated by the user to better cover the polygon.



(e) Intersection points between the sweep lines and the polygon are calculated.



(f) The points are sorted to create an alternating path between them.

Figure 4.7: Coverage Path Planning of a convex polygon.

Additionally, the user can set a few parameters to customize the path generation algorithm. These parameters are listed below in table 4.2.

Table 4.2: Flight parameters used to generate the flight path.

Parameter	Description
Drone elevation	The elevation in meters above the terrain where the path should be generated.
Search radius	The the maximum distance for the drone to detect sheep.
Search radius overlap	Overlapping distance for the drone's search area when generating the flight path.
Acceptance radius	The maximum distance the drone can be from a waypoint to count it as visited.
Drone velocity	The velocity of the drone during the mission.
Path rotation	Parameter specifying the angle of rotation the sweep lines cover the ROI.

Drone elevation specifies the altitude of the flight path relative to the terrain. The system will assign each waypoint an elevation value from the Norwegian Mapping Authority added to the specified relative elevation. **Search radius** r_s represents the distance the drone can detect sheep. This in combination with **search radius overlap** r_o is the basis for the distance D_x between the sweep lines generated in figure 4.7c. The distance is calculated by the formula $D_x = 2r_s - r_o$. **Acceptance radius** is a proximity buffer for the drone to accept a waypoint as visited. The drone needs a buffer due to uncertainty in GPS data making it unable to navigate to an exact location. **Drone velocity** is the max speed the drone is allowed to fly during a mission. **Path rotation** is a parameter that directly adjusts the CPP algorithm adjusting the angle of the sweep lines, as shown in figure 4.7d.

4.2.4 Data Processing and Visualization

The second primary responsibility of Radio Sheep GCS is to process and visualize the sheep data collected by the drone. When the drone is out on a mission, it will continuously receive data from sheep and immediately attempt to send it back to the GCS. In this project, we use a custom MAVLink procedure to exchange sheep data, which figure 4.8 shows. The drone will send the same SHEEP_RTT_DATA message repeatedly until a corresponding SHEEP_RTT_ACK message is received. Upon receiving SHEEP_RTT_DATA, Radio Sheep GCS will decode the message and store the data in memory before returning a SHEEP_RTT_ACK containing the sequence number of the received message.

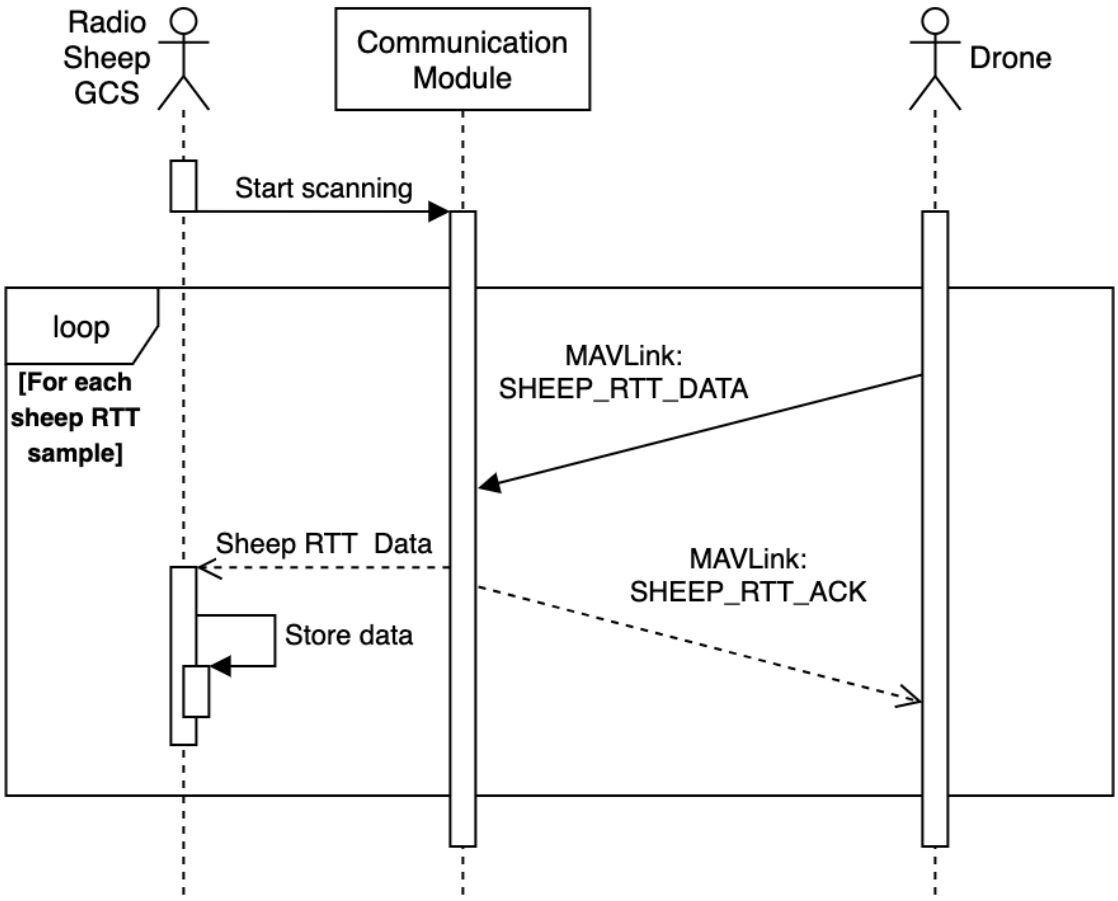


Figure 4.8: Sequence diagram of Radio Sheep GCS receiving sheep RTT data from the drone.

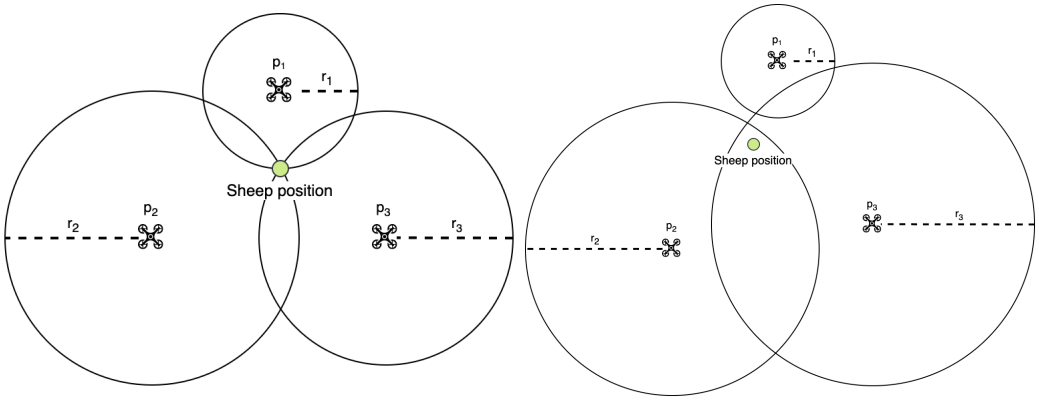
Since the sheep are not carrying GPS trackers, the GCS cannot determine their positions without pre-processing the collected data. As recalled from table 4.1 on page 40, the collected data contains information about the drone's GPS position and altitude along with the distance to the sheep. By grouping the measurements by every sheep's designated identification number, the system can solve a localization problem for each sheep individually.

When Radio Sheep GCS has finished processing the data, it will present a coordinate location with an uncertainty area per found sheep to the user.

4.3 Localization Method

After a completed drone mission, the GCS amasses a set of measurements that might reveal a vast area of possible sheep locations. Radio Sheep GCS will process the raw data and present a single point where the system believes a sheep is located.

One of the main challenges with localization is the uncertainty and noise in the measurement data. In figure 4.9, we can see how ideal measurements compare to data from a realistic scenario. The figure contains three drone locations at the point of measurement p_i and the distance from the sheep measured r_i. With data from figure 4.9a, the system can find the exact intersection between all measurements. In a realistic context, the system must solve the localization problems according to figure 4.9b.



(a) Ideal scenario.

(b) Realistic scenario.

Figure 4.9: Visual representation of sheep measurements.

To solve the localization problems, Radio Sheep GCS has implemented two different algorithms based on the solutions for RSSI-based localization suggested and evaluated by Goldoni et al. [34]. One is a statistical method using a particle filter, and one is a geometrical approach using multilateration by finding the intersection area between the measurements. The results in chapter 5 compare and evaluate the two methods.

The current implementations are simplified to work in 2D space. Since the drone and the sheep are located at different altitudes, the height differenced must be compensated for the algorithms to function correctly. Assuming that the ground is flat under the drone, the height difference is the drone's relative elevation to the ground. The measurement distance d can then be calculated with equation 4.1.

$$d = \begin{cases} \sqrt{d_m^2 - h^2}, & d_m > h \\ 0, & d_m \leq h \end{cases} \tag{4.1}$$

d_m is the raw measurement distance, and h is the drone's altitude relative to the ground. Figure 4.10 illustrates this scenario.

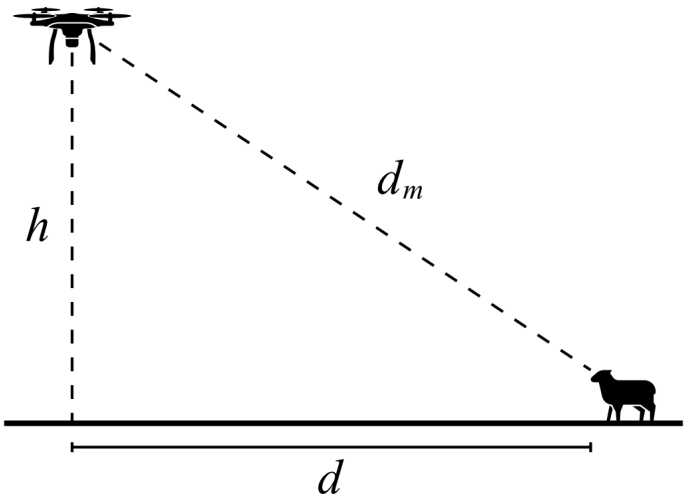


Figure 4.10: Height compensation of distance measurements.

4.3.1 Localization with Particle Filter

One of the localization methods used is a variant of a particle filter method. This method is based on statistical inference, where it converts the measurements to discrete particles representing possible solutions for the system. Conceptually, this can be viewed as illustrated in figure 4.11.

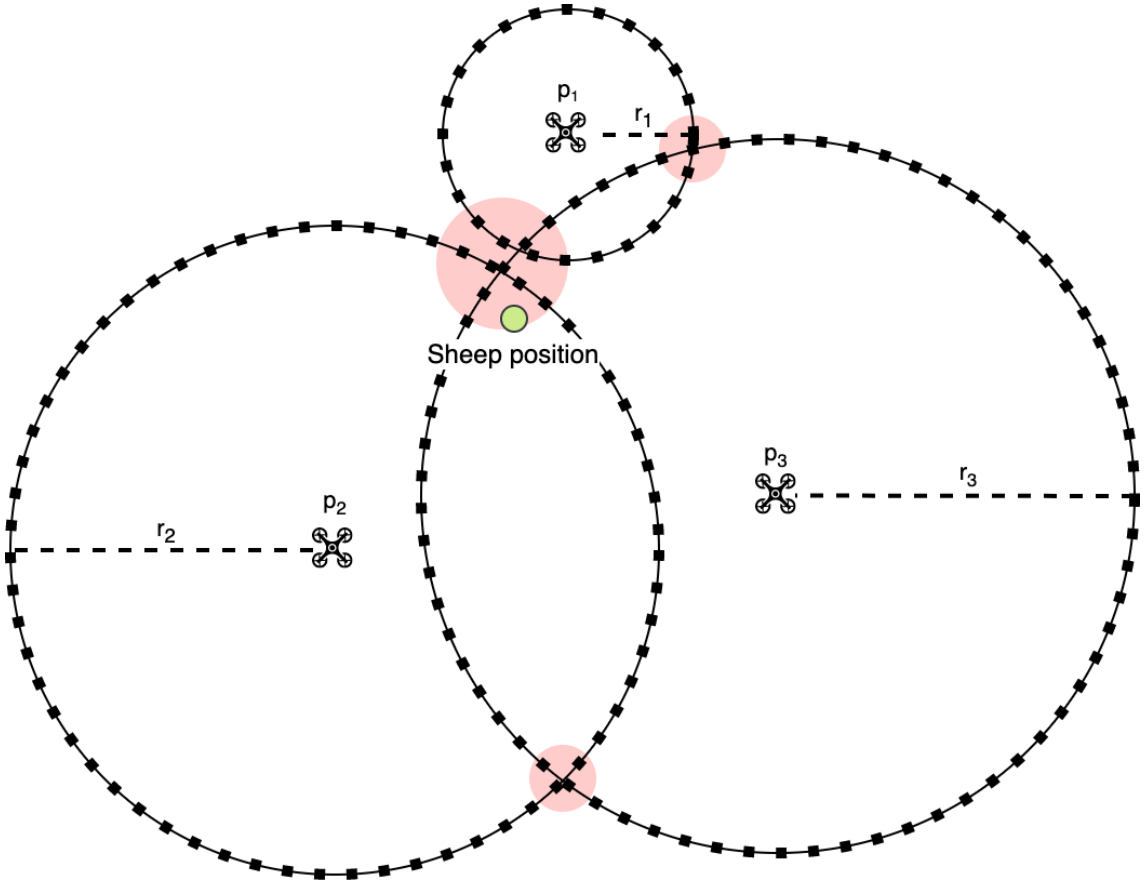


Figure 4.11: Localization with particle filter.

The particle filter method generates a fixed amount of points (particles) equally distributed on coordinates around each measurement circle. Dense clusters of particles formes in the intersections between the measurements, as we can see marked as red areas on the figure. As figure 4.12 illustrates, the algorithm analyzes the longitude and latitude components separately to estimate the sheep location coordinates. It is now easier to identify that the density of particles is higher in the red area around the actual sheep location.

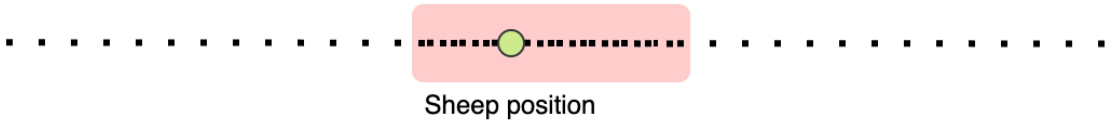


Figure 4.12: Single-axis coordinate series of longitude or latitude.

By applying the series of single-axis coordinates into a normal distribution, we can calculate the probability of every particle being the correct location. With this information, the point of highest probability can be interpolated, along with the standard deviation. In figure 4.13, we see how the example above looks when applied in a normal distribution of probability.

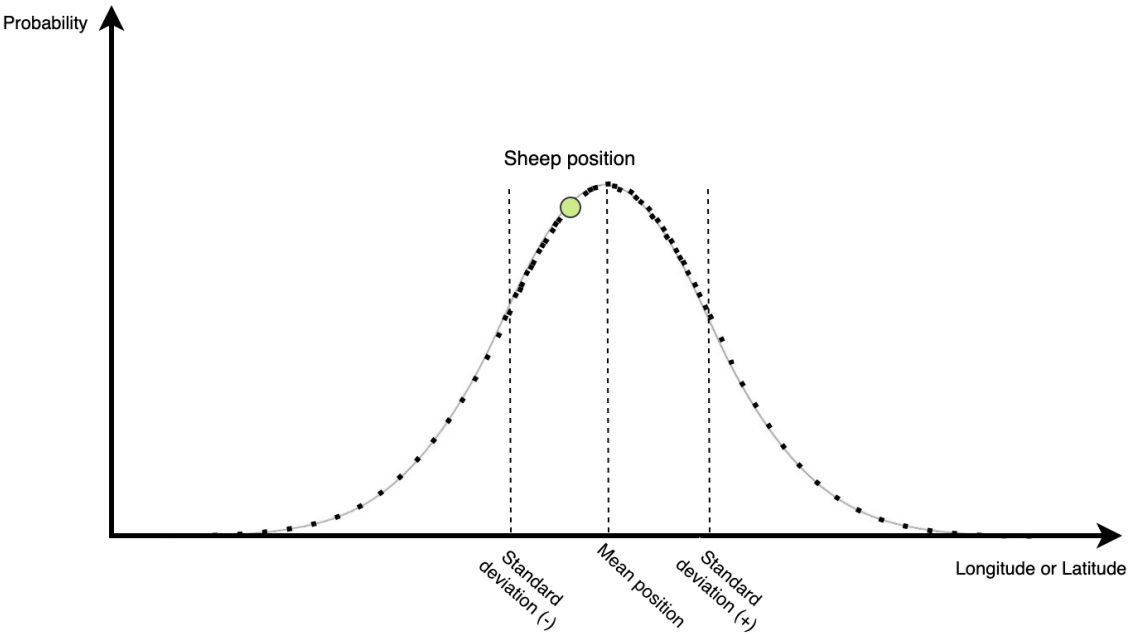


Figure 4.13: Single-axis coordinate series in a normal distribution.

The idea is that the actual sheep location is most likely within the standard deviation of the mean point. By selecting the mean point, there is a high probability of being within an acceptable range of the actual location. The standard deviation represents the resulting location's uncertainty radius.

4.3.2 Localization with Multilateration

The second method applied is a time of arrival (ToA) true-range multilateration technique. This approach tries to find the intersection point of all measurements

using geometry. Noise in the measurements makes it impossible to directly find an intersection point, introducing some uncertainty to the results.

The work of Pradhan and Hwang in [35] inspired the implemenetation of this algorithm. They suggest incrementing the error distance of the measurements, making the algorithm result in an area instead of a single point. Every time the measurements do not share an intersection area, a positive error radius e_i is added to each measure, as shown in figure 4.14. This error radius will iteratively increase while the algorithm cannot find a common intersection area. When the error radius is sufficiently long, the method will eventually calculate an intersection area, as the figure depicts as the red area.

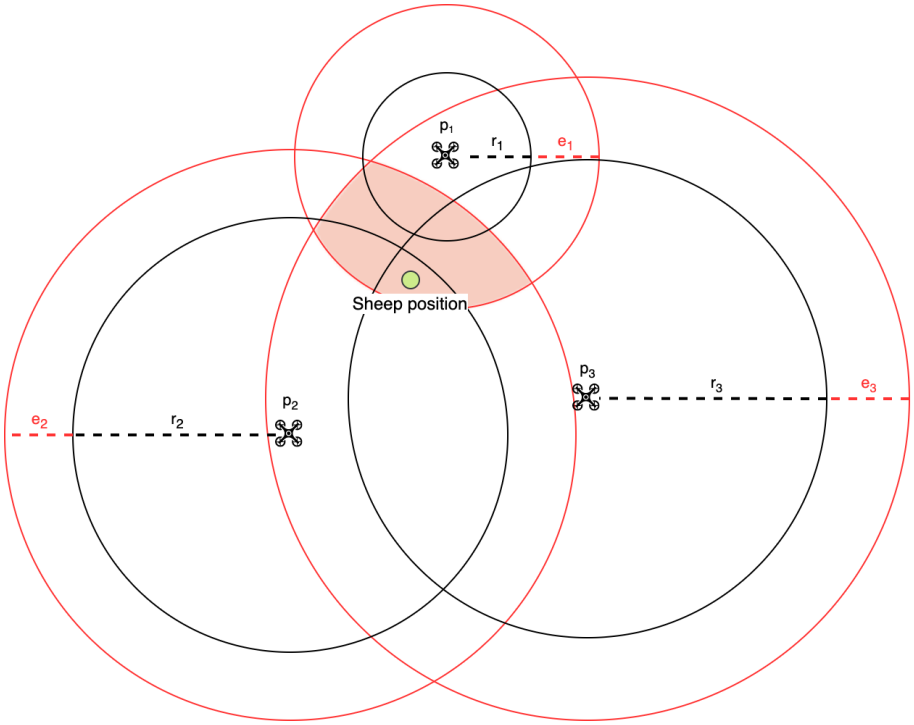


Figure 4.14: Localization with multilateration.

When the algorithm finds an intersection of all the measurements, it selects the mean point of the area. Given that the sheep location is within the intersection area, the maximum distance from the mean point to the area border determines the uncertainty radius for this method. Figure 4.15 illustrates this concept.

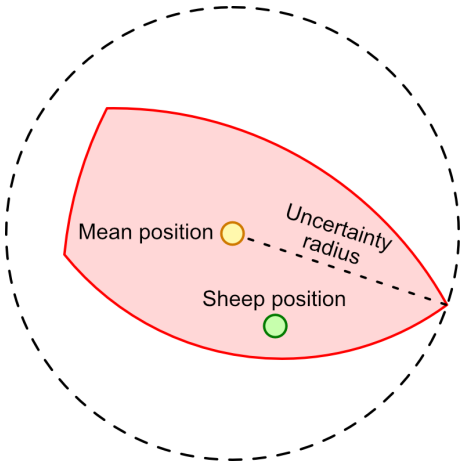


Figure 4.15: Multilateration uncertainty radius.

4.4 Software Tools and Libraries

This section briefly introduces the software tools and libraries used to develop Radio Sheep GCS.

4.4.1 Client and User Interface

To reduce development time, I chose to use the well-known web technologies HTML, CSS, and JavaScript to create the application interface. The GCS is developed as a proof of concept, making flexibility and short development time ideal. Since running the application in a web browser introduces limitations to the use of the file system and USB ports, Radio Sheep GCS was developed with the ElectronJS framework [36]. Electron is a framework for developing cross-platform GUI applications using web technologies. Electron utilizes the Node.js [37] JavaScript engine combined with the Chromium [38] rendering engine, and provides APIs to access system resources.

To assist in rapidly building the user interface, I chose to use the React [39] and Redux [40] JavaScript libraries to respectively generate HTML and manage system state. Furthermore, I used the React-based UI component library Material UI [41] to access pre-build React components such as buttons and layout structures. Material UI implements React components in Google's Material Design [42] framework.

The application is written with TypeScript [43], a programming language providing types on top of JavaScript. TypeScript is compiled into JavaScript before running, making it compatible with all the used JavaScript libraries.

4.4.2 Map and Elevation

One of the most critical parts of a GCS is the map. The operator uses it to choose the search area and to see where the system has located sheep. When choosing a map library for the application, it was essential to consider the support of the required features. The map needs support for drawing waypoints and polygons and for loading map tiles from custom sources. Mapbox [44] is a digital map provider for performant and interactive maps for the web. Radio Sheep GCS uses their JavaScript library *Mapbox GL JS*, as it is well-documented and supports all required features.

For geometrical calculations such as distance calculations between coordinates and shape generation based on coordinates, I have used the JavaScript library Turf.js [45].

An important aspect to consider when using map data to navigate a drone is terrain and elevation data quality. Low-quality terrain data makes it very hard to choose the desired ROI for the operator. Navigating with inaccurate elevation data poses a safety risk and might cause critical damage to the drone. These aspects are the reason why I chose to use both map tiles and elevation data from the Norwegian Mapping Authority, which provides open access to the most accurate map data for Norway.

4.4.3 Communication with Drone

The drone's capabilities determined the communication method used in Radio Sheep GCS, namely MAVLink messages over UDP or serial port. UDP was chosen as the desired communication protocol to secure compatibility with the drone simulator SITL and the convenience of a tetherless connection. Radio Sheep GCS uses Electron's bundled UDP library along with a modified version of the JavaScript MAVLink packing and parsing library *node-mavlink* [46]. By using the command-line-based GCS MAVProxy [47], we were able to forward serial port data over UDP without additional development of Radio Sheep GCS.

Chapter 5

Results and Discussion

5.1 Experiments and Tests

This section presents the experiments and tests we conducted during this project. The purpose of these test have been to determine capabilities of the system, and to evaluate our finished prototype.

Initially, we had to verify the basic concepts such as distance measurements with radio signals and flight testing of the drone. These preliminary experiments allowed us to uncover problems early on and tune the system for better performance. After seeing the core functionality in practice, we tested that the system components were able to work together. Ultimately, we could experiment with the complete system prototype to evaluate this project's concept.

5.1.1 nRF52833 Range Test 1

The very first experiment conducted was a range test using the nRF52833. The goal of this test was to verify the viability of the project concept by proving that the radio chips were able to measure the distance between each other. To measure the distance, two nRF52833 acting like sheep and drone respectively would perform the communication sequence we can recall from figure 4.3 on page 39.

Test Setup

We used a stationary nRF52833 acting as a drone and an nRF52833 acting like a sheep. The sheep nRF52833 was moved to five different positions where we performed a measurement. Figure 5.1 presents the locations used, where we can see that the spots are spaced evenly along a path with an unobstructed line of sight to the drone.

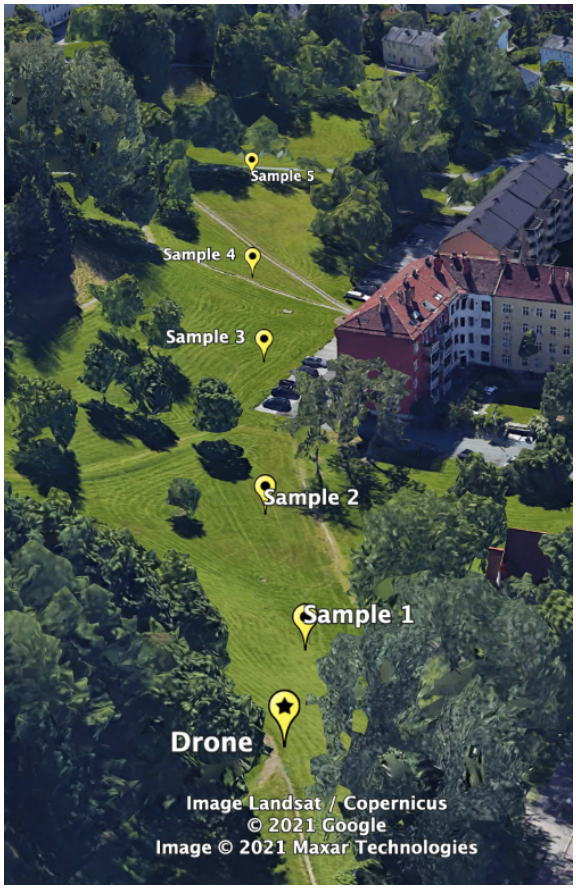


Figure 5.1: nRF52833 range test 1 locations.

Figure 5.2: Position of the nRF52833 development kit simulating the drone.

We connected the drone nRF52833 to a computer via a USB serial cable to extract data during the test, as depicted in figure 5.2. We moved the sheep nRF52833 to each sample position and logged the signal strength and distance measurements. By taking the GPS coordinates for the sample locations, we could compare the GPS distance and the measured distance.

Results

Table 5.1 presents the results of the measurements in this experiment. Overall, the results are promising as the error of measured distance is relatively low and does not seem to increase significantly with longer distances. A measured distance of over 200 meters makes it viable to develop the project further, as we could create a proof-of-concept system at a large enough scale.

Table 5.1: nRF52833 range test 1 results.

	Measured Distance (meters)			**Signal**	
Sample	**GPS**	**nRF52833**	**Difference**	**RSSI** (dBmW)	**Packet loss** (Percentage)
0	0	0	0	-17	0.0
1	32.8	35.0	+2.2	-61	0.0
2	72.7	83.0	+10.3	-69	0.6
3	133.3	136.3	+3.0	-84	68.0
4	168.8	175.5	+6.7	-69	2.4
5	221.2	227.5	+6.3	-73	0.0

An intriguing aspect of these results is that the measured distances always overshoot the actual lengths and never underestimate them. We believe that the cause of this is the reflection of radio waves, as reflected waves travel a longer distance than direct waves. In turn, this would influence the distance measurements, making them increase.

As we can see from the results of sample 2, the nRF52833-measured distance had the highest overshoot of +10.3 meters. The results can be explained by sample 2 being at the lowest point in the terrain and having a large house wall close by having a reflective angle towards the drone. Radio waves might have been reflected both by the terrain and the house wall towards the drone, increasing the measured distance. An illustration of this is portrayed in figure 5.3. The green arrow represents radio waves going straight from the sheep to the drone, and the red arrows represent the reflected radio waves interfering with the measurements.

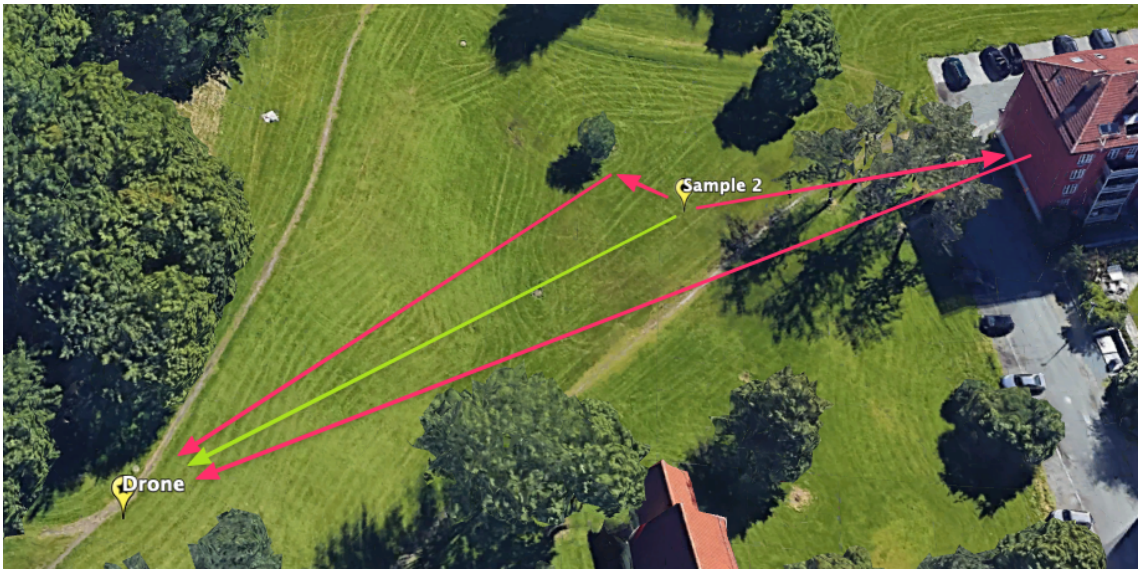


Figure 5.3: Possible reflection of radio waves in nRF52833 range test 1.

The results from sample 3 are also quite interesting. Even though the measured distance had a low error margin, the signal was much weaker than the rest of the samples, with a packet loss of 68%. We believe interference from nearby devices using 2.4 GHz radio waves might have caused this issue. The location of sample 3 was close to a residential building and a parking lot, where Wi-Fi routers and cars might send out radio waves causing destructive interference with the nRF52833 radio waves.

Key Takeaways

- A nRF52833 pair can establish a radio connection and exchange RTT packets.
- The nRF52833 can measure distances up to 230 meters with a margin of error of about 2 - 10 meters.
- Reflection of radio waves might impact the measurements by increasing the measured distance.
- External 2.4 GHz infrastructure might reduce nRF52833 radio signal strength and increase packet loss, but does not seem to affect the error margin of the measured distance.

5.1.2 nRF52833 Range Test 2

Early on in the project, we needed to know how an nRF52833 pair would behave when not directly in the line of sight of each other. Since the radio chip would be embedded in the sheep's ear tag, the sheep's head would often be blocking the direct line of sight to the drone. To test how this would affect the measurements, we wanted to experiment to compare the signal strength of the radio chips with and without an obstacle simulating a sheep head.

Test Setup

Similarly to in range test 1, we had the drone nRF52833 stationary connected to a computer. We chose two locations with a direct line of sight to the drone to place the sheep nRF52833. The locations were respectively 35 meters and 133 meters away from the drone as seen in figure 5.4.

Figure 5.4: nRF52833 range test 2 locations.

To simulate a sheep head, we used a filled water container as depicted in figure 5.5. The water container was 12 cm wide and placed about 70 cm above the ground. Each measurement was done with and without the water container in front of the nRF52833.

Figure 5.5: nRF52833 range test 2 water container setup.

Additionally, to learn how the orientation of the radio chip would affect the measurements, we completed each measure with seven different nRF52833 directions. The orientations are described in figure 5.6.

(a) nRF upward, antenna forward. (b) nRF leftward, antenna forward. (c) nRF rightward, antenna forward. (d) nRF downward, antenna forward.

(e) nRF upward, antenna backward. (f) nRF forward, antenna upward. (g) nRF forward, antenna leftward.

Figure 5.6: nRF52833 development kit orientations used during range test 2.

Results

The complete results are presented in table 5.2. These results show that a reduction in signal strength is present when the radio signals travel through the water container. As we can see from the average of signal strengths recorded at position 1 and 2, the lost signal strength is on average $\frac{-67.1+-74.7}{2} - \frac{-74.6+-78.5}{2} = 5.65$ dBmW when obstructed by the water container.

Table 5.2: nRF52833 range test 2 results.

	RSSI (dBmW)			
	Pos. 1 35m		**Pos. 2** 133m	
nRF Orientations (from figure 5.6)	**Air**	**Water**	**Air**	**Water**
a	-62.6	-74.1	-71.9	-73.3
b	-72.3	-73.4	-82.2	-79.7
c	-74.6	-80.3	-76.5	-81.0
d	-61.8	-73.2	-68.6	-75.3
e	-67.4	-77.5	-69.4	-73.2
f	-61.0	-74.4	-74.1	-80.5
g	-69.9	-69.4	-80.4	-86.4
Average	**-67.1**	**-74.6**	**-74.7**	**-78.5**

By comparing these results in the scatter plot in figure 5.7, we can see that the lost signal strength is pretty consistent. The 100 meters between positions 1 and 2 do not seem to affect the signal loss caused by the water obstacle. The nRF52833 orientation seems to be just as important to the loss of signal strength as the water container obstacle. The water container at the orientation *b* measurement seems to have a low impact at both positions, which might indicate that tweaking the orientation of the nRF52833 can mitigate lost signal when attached to sheep.

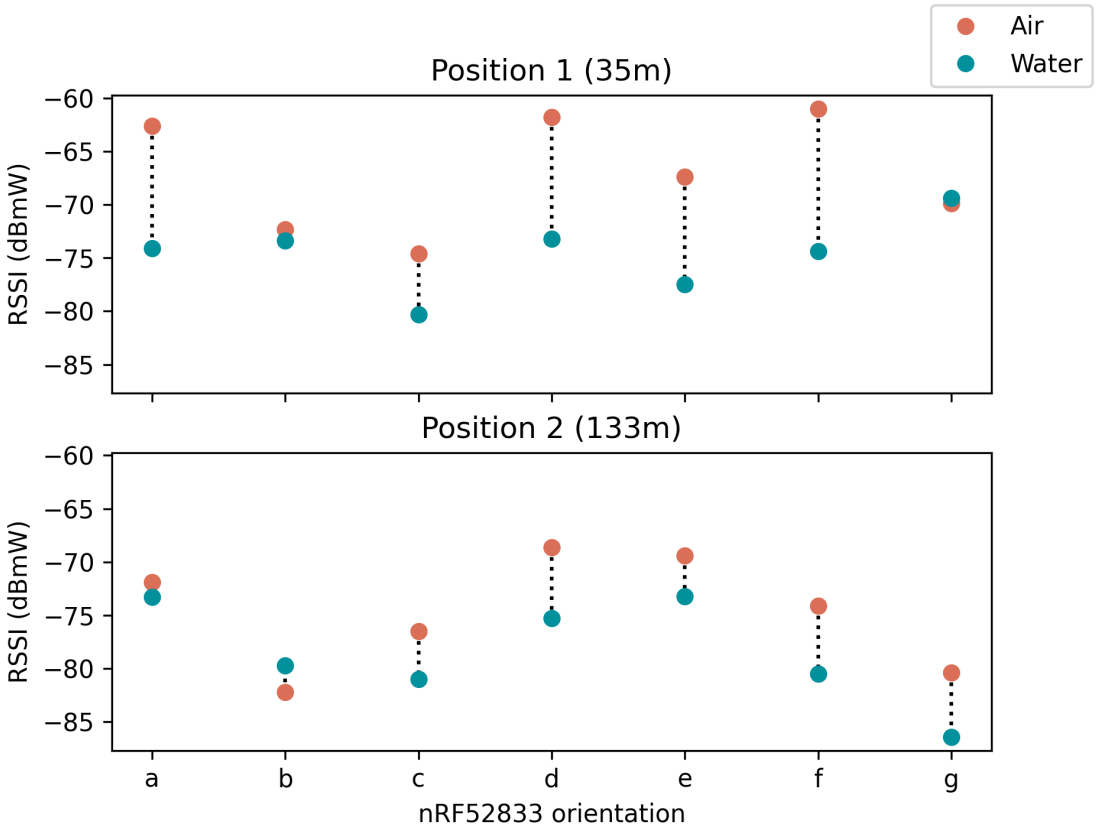


Figure 5.7: Range test 2 results.

We see that the signal strengths in orientation g in position 1 and orientation b in position 2 are higher when passing through the water obstacle. We consider this data noise which interference from nearby radio sources might have caused.

Key Takeaways

- Signal strength is reduced by around 5.65 dBmW when a sheep head obstructs the signal.
- The nRF52833 orientation is an important factor in mitigating signal loss.
- Signal obstruction caused by a sheep head does not increase noteworthy with distance.

5.1.3 Initial Flight Test

The initial flight test was the first experiment where Radio Sheep GCS conducted a flight for the autonomous drone all by itself. The goal of this experiment was to determine if the GCS and the drone were able to communicate with each other appropriately in a real-life scenario and if the drone were able to follow the instructions provided by Radio Sheep GCS. To perform the test, Radio Sheep GCS would follow the mission planning procedure we can recall from figure 4.5 on page 43. The drone's onboard flight log provides data used to evaluate this test.

Test Setup

The drone was powered on and then placed on the ground in the middle of a field. The takeoff zone is depicted in figure 5.8a. A virtual geofence was loaded into the drone using Mission Planner GCS, as Radio Sheep GCS does not support this feature. The geofence is a safety tool to stop the drone from leaving a pre-defined area. We configured the geofence to trigger a Return to Launch (RTL) if the drone would fly outside of the field or if it were to ascend more than 30 meters above launch.

A flight plan was created in Radio Sheep GCS as shown in figure 5.9a. The flight path had a total length of about 400 meters, and we set it to be 20 meters above the ground. Radio Sheep GCS instructed the drone to fly at a speed of 10 meters per second.

(a) Initial placement of the drone. (b) Drone just after takeoff.

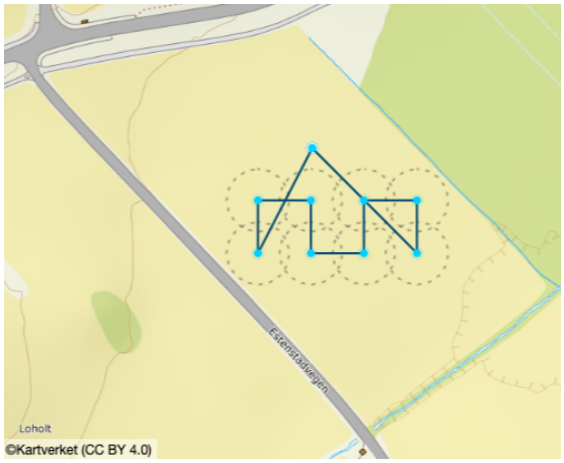
Figure 5.8: Photographs of the drone during the initial flight test.

Results

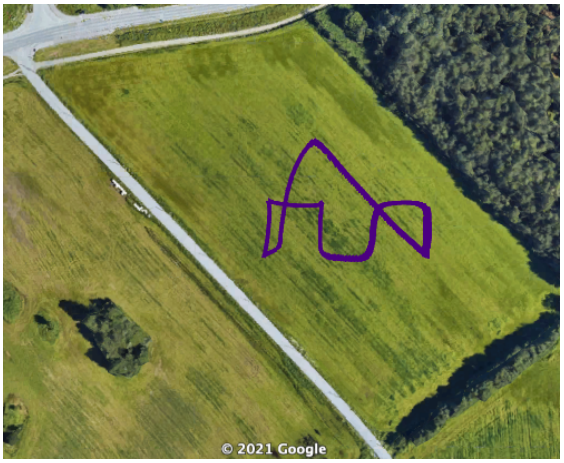
The drone and Radio Sheep GCS successfully established a stable connection. Radio Sheep GCS had no issues transferring the flight plan to the drone. Arming the drone and starting the mission was working as intended.

On the first flight attempt, there was an issue where the drone immediately ascended beyond the geofence threshold and was forced to perform an RTL. Radio Sheep GCS having an incorrect format on the MAVLink messages encoding the waypoints caused this issue. The problem was that Radio Sheep GCS provided MSL elevation data in a MAVLink message expecting relative elevation data. This issue made Radio Sheep GCS instruct the drone to fly 175 meters up relative to the ground instead of the correct 20 meters. We resolved the issue on the field, and we made a second flight attempt.

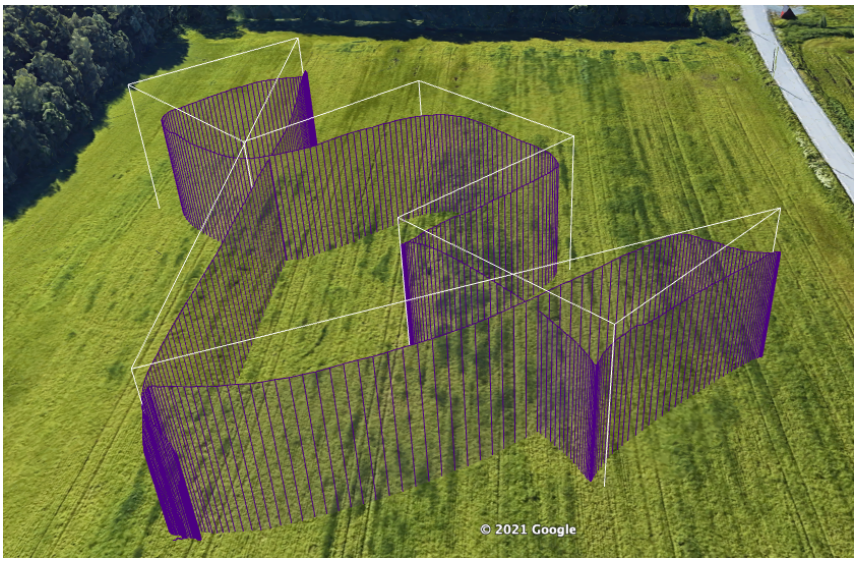
During the second flight, there were no concerning issues. The recorded flight path can be seen in figure 5.9b and 5.9c. By comparing figure 5.9a and 5.9b, we can see that there are no major deviations from the planned flight route and the performed flight. Gusts of wind and uncertainty in GPS measurements are most likely causing the more minor deviations from the flight plan.



(a) Route planned in Radio Sheep GCS.



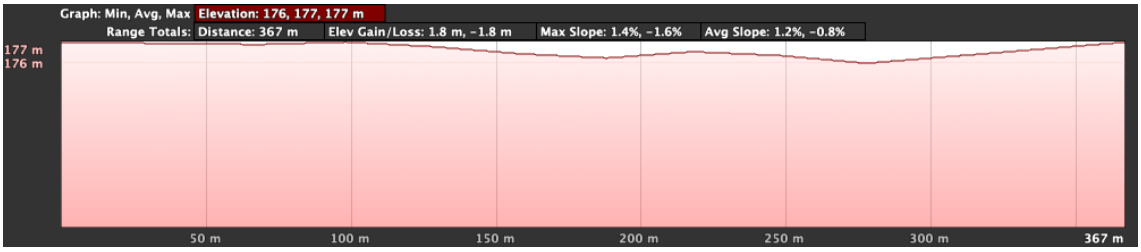
(b) Actual flight performed by the drone.



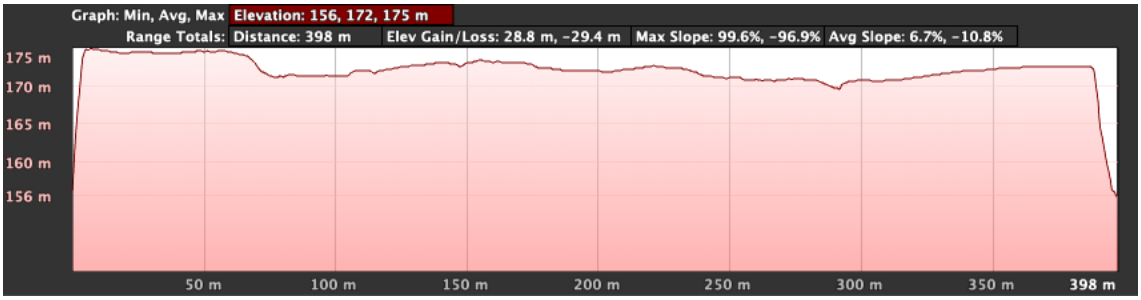
(c) 3D view of the route and flight.

Figure 5.9: Initial flight test route and flight.

As we can see in figure 5.9c, the drone's path (the purple line on the figure) does not match the specified altitude. The white lines represent the waypoints at a correct elevation, and it is prominent that the drone is constantly a few meters below the specified height. In figure 5.10, we can see a comparsion of the planned elevation profile and the recorded elevation profile. Note that the planned elevation profile in figure 5.10a is for the route 20 meters above the ground and does not reflect that the drone has to ascend and descend at the starting point. The planned elevation profile is within 176 meters and 177 meters MSL, while the drone moves within the range of 170-175 meters MSL when filtering out the ascension and descension. These results suggest that the drone can keep a reasonably consistent elevation during the flight, with a deviation from the plan within about 6 meters.



(a) Planned elevation profile.



(b) Recorded elevation profile.

Figure 5.10: Elevation profile of the initial flight test.

There are multiple possible sources of errors for these elevation results. The planned elevation data provided by the Norwegian Mapping Authority can differ from the actual elevation. Since the area in this test is cropland, the elevation will alternate with the seasonal growing crops. Another possible source of error is uncertainty in the drone's sensors, which is more likely to cause the minor inconsistencies seen in the recorded elevation profile.

The last aspect of this test to consider is the drone's ability to keep the speed of 10 meters per second as instructed by the GCS. The drone's speed profile is graphed in figure 5.11. The speed recorded is GPS-based, so it does not cover climbing speed. We can see that the drone struggles to reach the specified speed of 10 m/s. The drone has to turn at every corner, which means that it will have to slow down before reaching its next waypoint. Since the planned route had relatively short paths between the waypoints, the drone had to slow down before reaching top speed. From the graph, we see that the drone reached 8 m/s at the most extended straight lines at the beginning and end of the planned route and reaching about 5 m/s at the shorter paths.

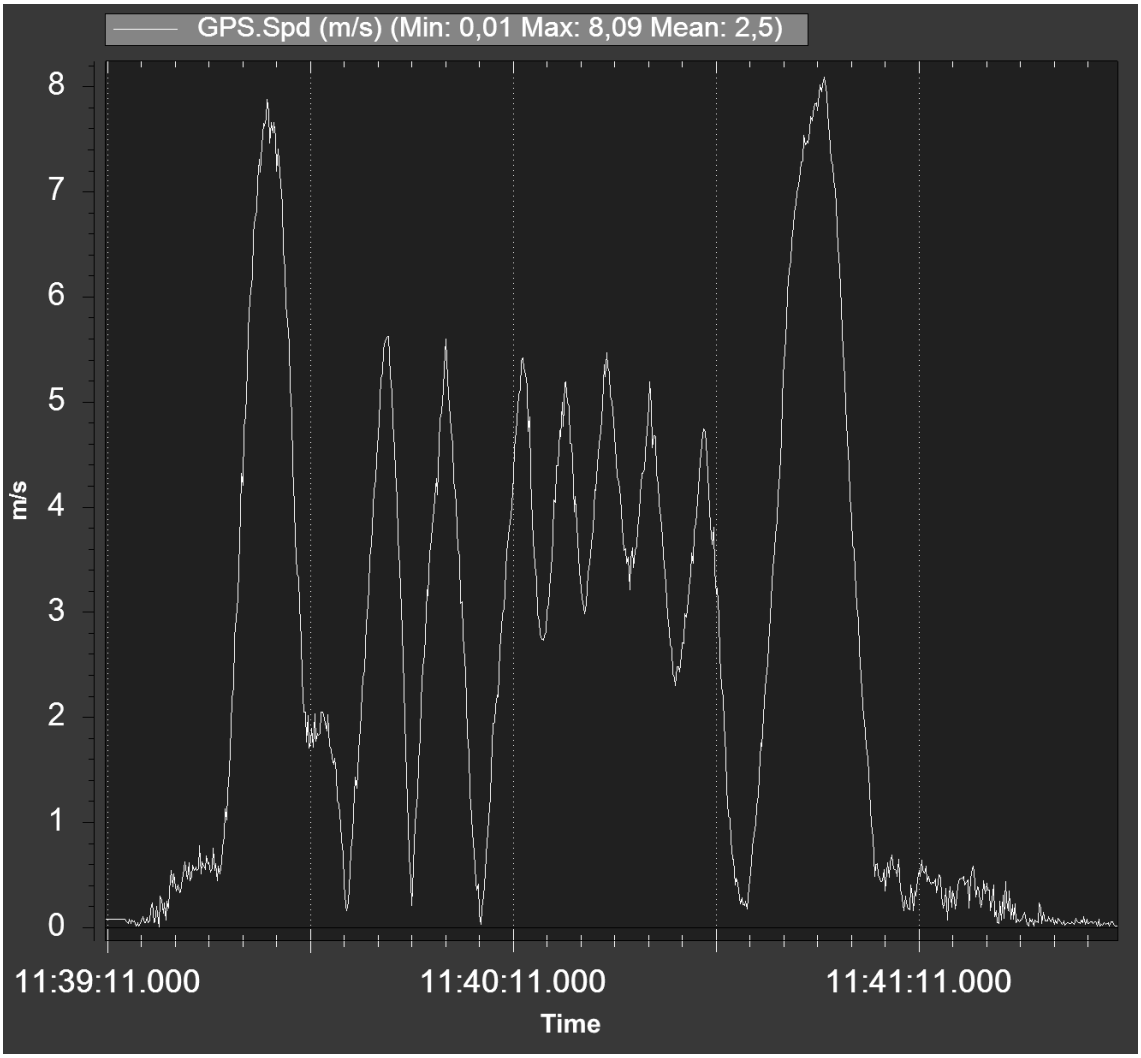


Figure 5.11: Recorded speed profile of the initial flight test.

These results are not problematic, but we see that the drone requires longer distances to reach its target speed.

Key Takeaways

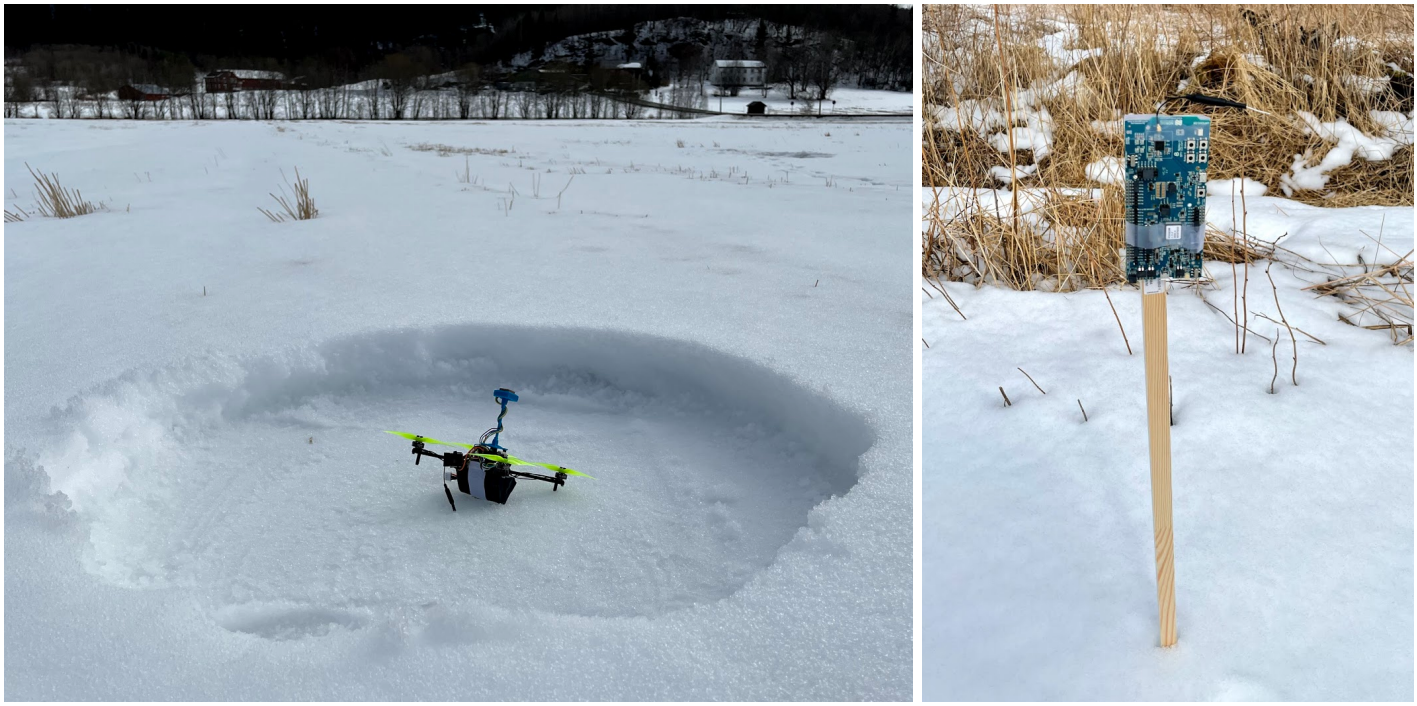
- Radio Sheep GCS can create a valid flight plan.
- Radio Sheep GCS and the drone can establish a reliable connection and negotiate the MAVLink version.
- Radio Sheep GCS can successfully upload a mission to the drone.
- Radio Sheep GCS had an issue causing it to set the wrong elevation.
- Radio Sheep GCS can successfully arm the drone and start a mission.
- The drone can follow a flight path provided by Radio Sheep GCS.
- The drone can keep a fairly consistent elevation within about 6 meters from the flight plan.
- The drone needs longer straight distances to reach higher speeds.

5.1.4 Small-Scale Prototype Test

Before testing the system on a larger scale, we needed to verify that all the parts of our composite system worked together as intended. This test was the first experiment where all three system components worked together to attempt locating sheep.

Test Setup

We conducted this test on the same field as the previous experiment. To simulate sheep, we used poles with nRF52833 chips attached about 70 cm up. The drone had an nRF52833 attached to it, programmed to listen to the sheep signals. Figure 5.12 shows our physical hardware setup in this experiment.



(a) Drone and attached nRF with antenna hanging down. (b) nRF52833 simulating sheep.

Figure 5.12: The small-scale prototype test hardware setup.

Four simulated sheep were placed on the field. Two of them were standing alone, and two were placed together. The sheep were configured to advertise every tenth second.

Radio Sheep GCS generated a route for the drone to fly. Figure 5.13 show the drone's flight plan and the four sheep locations. We instructed the drone to fly at a 30-meter altitude above the terrain at a speed of 5 m/s.

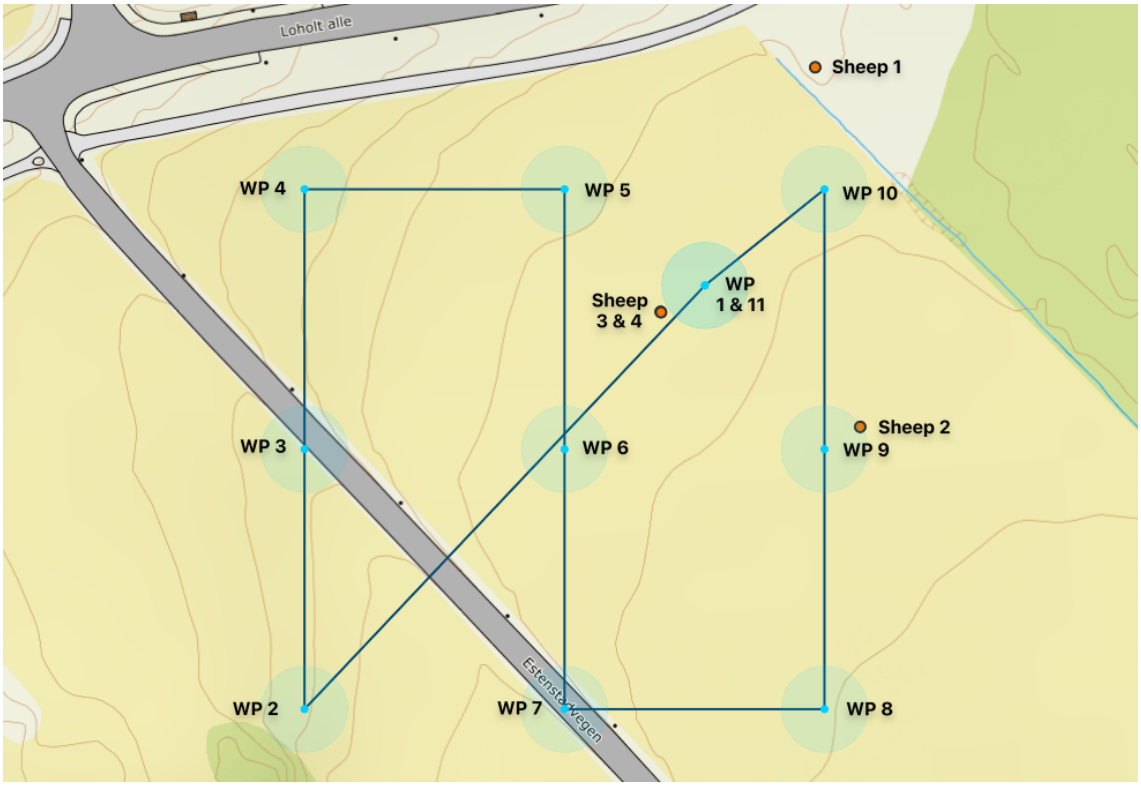


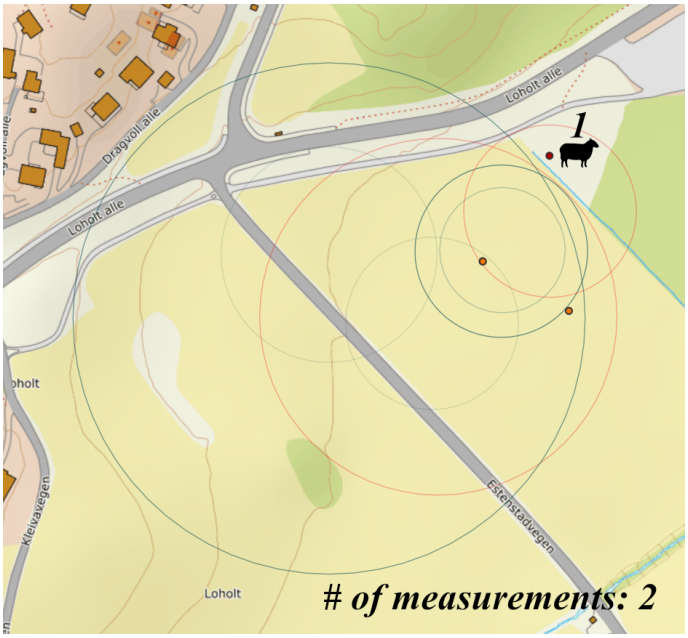
Figure 5.13: Route plan and sheep locations in the small-scale prototype test.

The generated waypoints are marked with "WP" in the figure. The zone for takeoff and landing is marked by WP 1 and 11. The blue areas around each waypoint represent the acceptance radius for the drone. The drone needs to be within this boundary to accept the waypoint as visited to proceed to the next one. The sheep placements were not in focus during this test, as the goal was to verify that every part of the system did its job and could communicate with the other components.

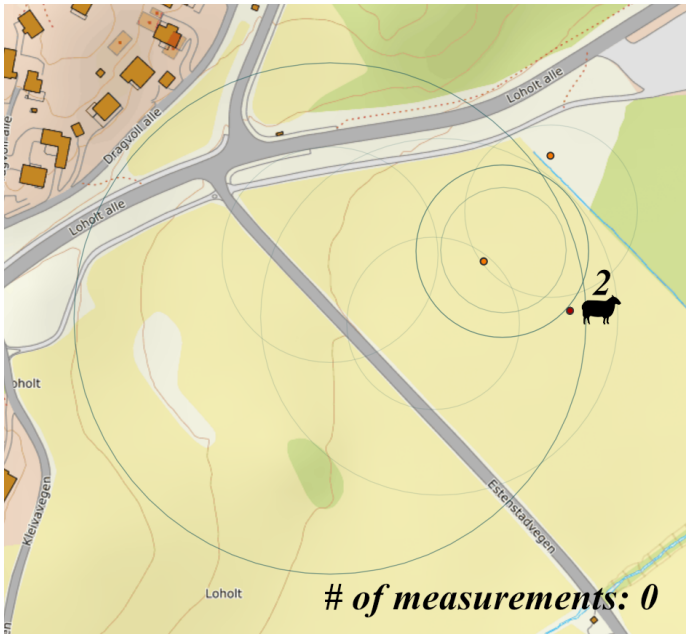
Results

Radio Sheep GCS successfully established a connection to the drone and started the planned mission. The drone had no issues during the flight. The nRF52833 radio chips performed distance measurements and sent them to Radio Sheep GCS without any issues.

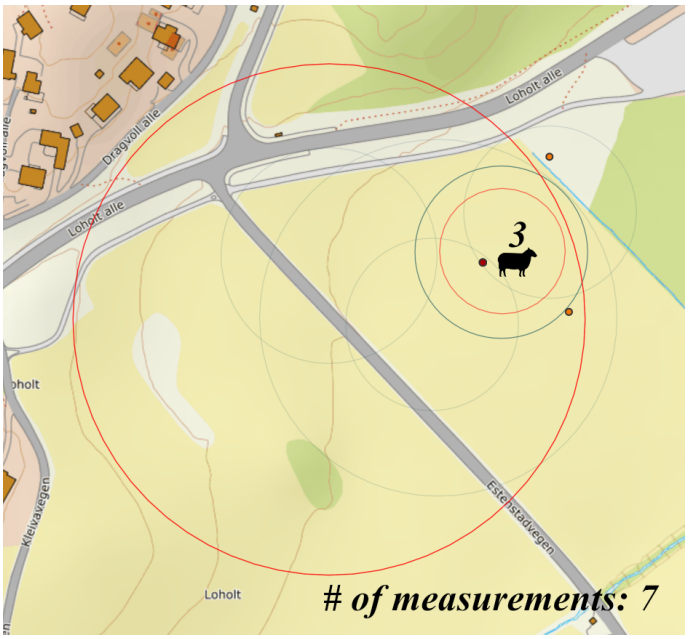
Figure 5.14 shows a visualization of the measurements. The thin circles illustrate the measurements from the drone's point of view. The drone's GPS location is the center of the circle, and the measured distance is the circle's radius. The red circles are the selected sheep's belonging measurements.



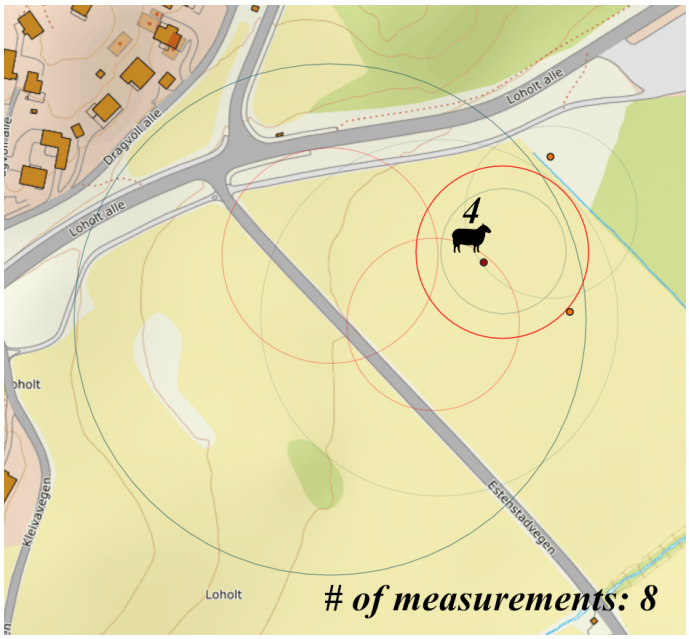
(a) Sheep 1 measurements.



(b) Sheep 2 measurements.



(c) Sheep 3 measurements.



(d) Sheep 4 measurements.

Figure 5.14: The small-scale prototype test's results.

As we see from figure 5.14b, the drone could not locate sheep 2. It turned out to be a battery issue with that particular nRF52833.

Due to the small scale of the experiment, we received a low number of measurements. The sheep's suboptimal placements and lack of measurements made Radio Sheep GCS's location estimation algorithms insignificant. However, we believe that these results show that the system is ready to be tested on a larger scale.

Key Takeaways

- The drone can detect and measure the distance to multiple sheep.
- Radio Sheep GCS and the drone can successfully exchange the sheep measurements.
- The system is ready to be tested on a larger scale.

5.1.5 Medium-Scale Prototype Test

This project's final outside field experiment was to demonstrate the complete composite system with all components working together on a larger scale. The goal of this test was to simulate a realistic execution of the project at a slightly smaller scale than a real-life scenario.

Test Setup

The location chosen for this test was a cropland field where we would have a relatively large open space with a continuous line of sight to the drone at all times. We placed four nRF52833 boards acting as sheep at different locations of the area, which the map in figure 5.15 shows. Radio Sheep GCS would then generate a flight plan to cover the area containing the sheep locations. The zone for takeoff and landing is marked by WP 1 and 12.

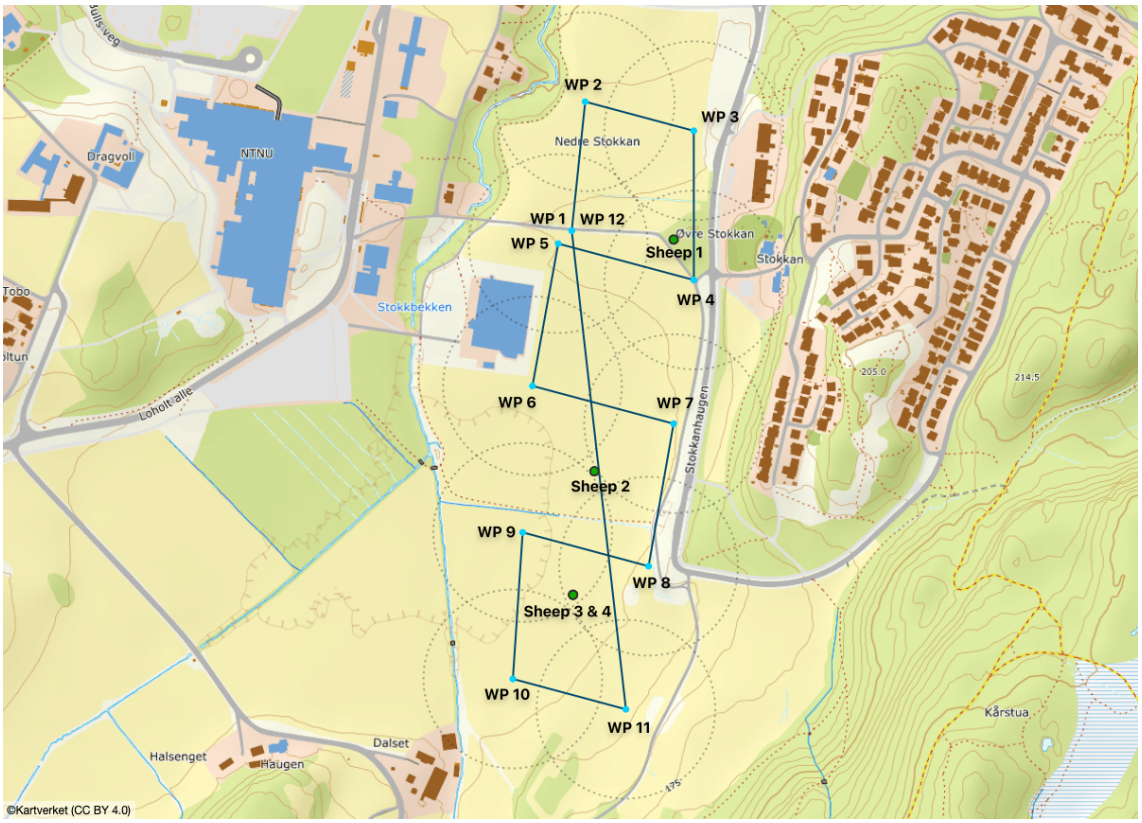


Figure 5.15: Route plan and sheep locations in the meduim-scale prototype test.

The dotted circles around each waypoint represent the drone's search radius in that particular waypoint. In this case, we set the search radius to 100 meters. Note that this is not the actual search radius of the drone in this project but is used to generate an ideal flight path over the test field.

Radio Sheep GCS instructed the drone to fly at an altitude of 30 meters above the ground at a speed of 8 meters per second.

We attached the nRF52833 boards about 70 centimeters up on a pole and placed them in the ground so that the radio chip would be at roughly the same altitude

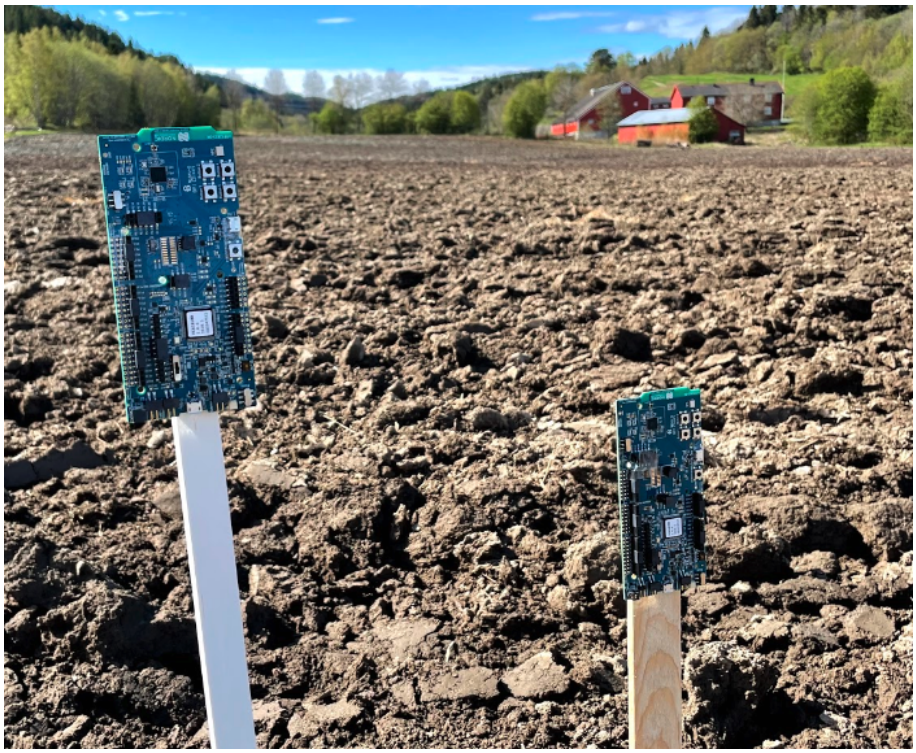
as when attached to a real sheep. In figure 5.16, we can see the sheep placements during the experiment. Sheep 1 in figure 5.16a was placed in an area with trees and bushes to simulate a scenario where vegetation might obstruct a sheep's radio signal. Sheep 2 in figure 5.16b was placed alone on the field to simulate an ideal scenario with no radio interference. Sheep 3 and 4 in figure 5.16c were placed together on the field to see if sheep close to each other might interfere with their radio signals.

Radio Sheep GCS would instruct the drone to fly the generated flight path and receive the sheep measurements upon the drone's return. To conclude the experiment, Radio Sheep GCS would estimate the sheep locations based on the range-only measurements. The estimation uses both the particle filter method and the multilateration method to compare the performance of the two algorithms. Since the size of the area restricted the test and not the size of the drone's search radius, any sheep detected outside of the selected 100-meter search radius was filtered out of the results to simulate how the system would work on a larger scale. If we were to use all the unfiltered measurements, the generated flight path would be unideal, creating a mismatch of the system's proportions.

The nRF52833 boards were programmed to advertise presence once per second, ten times faster than intended for a production-ready system. Because of this, we chose to use only every tenth RTT measurement in the location estimation to more accurately simulate a large-scale version of the system. Effectively, this gave us ten separate datasets to evaluate separately. Only the first dataset will be presented in detail in the result section, while we show aggregated numbers for all ten datasets.

(a) Sheep 1 placed among shrubs.

(b) Sheep 2 placed alone on the field.



(c) Sheep 3 and 4 placed together on the field.

Figure 5.16: The sheep placements during the medium-scale prototype test.

Results

We had initial trouble completing this test. The Wi-Fi module on the drone kept failing a while after takeoff. This issue resulted in Radio Sheep GCS being unable to retrieve the measurements from the drone. The test had to be aborted and repeated a few days later. The issue was resolved by connecting Radio Sheep GCS and the drone with a USB cable and using proxy middleware to forward serial port data over UDP.

As we can see from the flight log visualized in figure 5.17, the drone had no problems following the generated flight path.

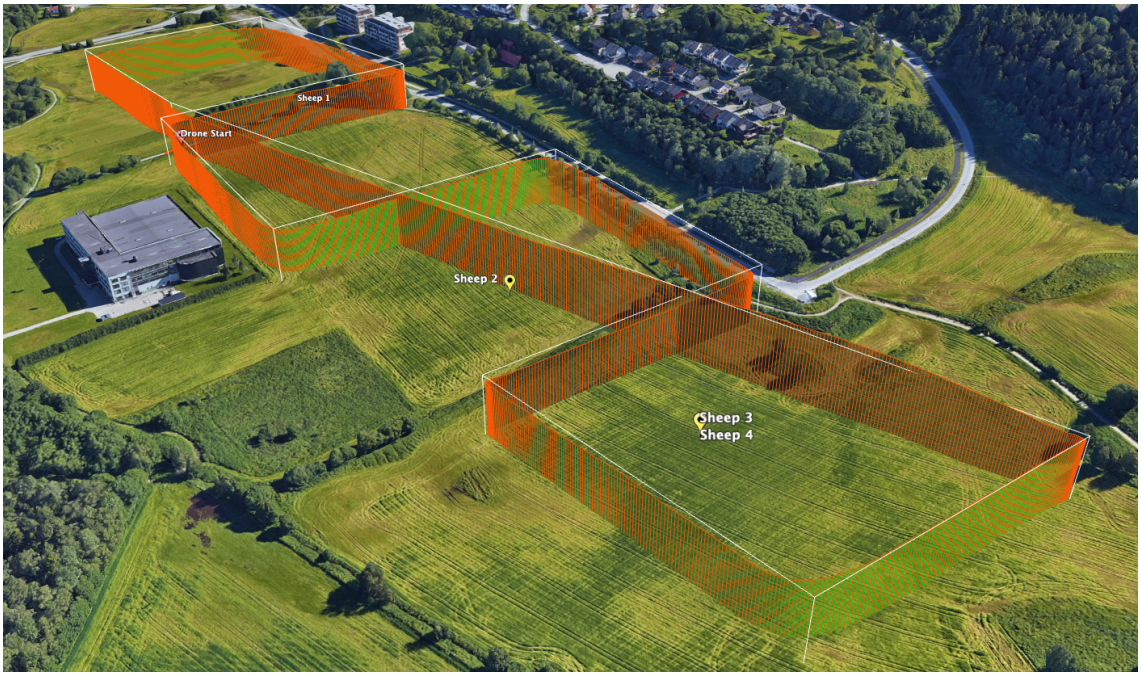


Figure 5.17: 3D visualization of the drone's flight log from the medium-scale prototype test.

The drone collected 1032 samples from the four sheep, where 91 of them were advertisement packets that the drone could not measure the RTT from. A sample of this kind suggests that a sheep is within range, but the drone cannot determine the distance. These samples are not usable in Radio Sheep GCS's current state, so they were filtered out before the sheep location estimation. Table 5.3 presents the amount distribution of measurements to the four sheep.

Table 5.3: The distribution of measurement amount in the medium-scale prototype test.

	Measurement count		
Sheep	**Usable**	**Unusable**	**Total**
1	171	39	210
2	270	13	283
3	245	20	265
4	255	19	274
Totals	941	91	1032

From this table, we can see the impact of radio signal obstruction. Sheep 1 placed among vegetation has the fewest total measurements and the highest count of unusable measures. Sheep 2 placed alone with no obstacles has the most measurements and the lowest number of unusable measures. Sheep 3 and 4 placed together had a slight drop in measurement count and slightly more unusable measurements.

As stated earlier, we chose to divide these measurements into ten separate datasets to fit a realistic large-scale scenario better. By filtering out measurements with distances exceeding the artificial search radius cap of 100 meters, we get the results in figure 5.19. The orange dot with a black outline represents the actual location of the sheep, and the dark circular lines represent the RTT distance measurements from the drone's point of view. The orange hollowed circle represents Radio Sheep GCS's estimated location of the sheep, and the colored area within the dashed circle around it shows the uncertainty radius σ. The error distance between the actual point and the estimated location is denoted as e. Figure 5.18 shows a summary of the legend used to present the results.

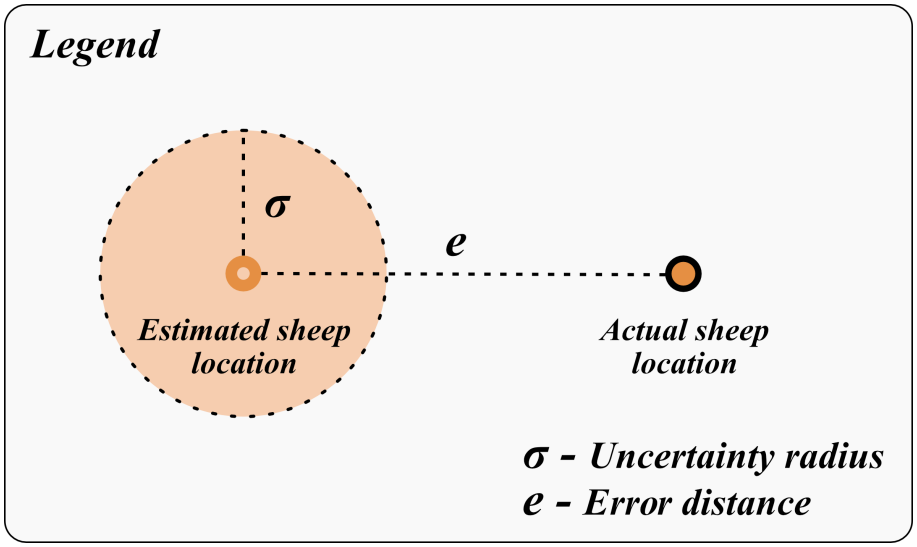
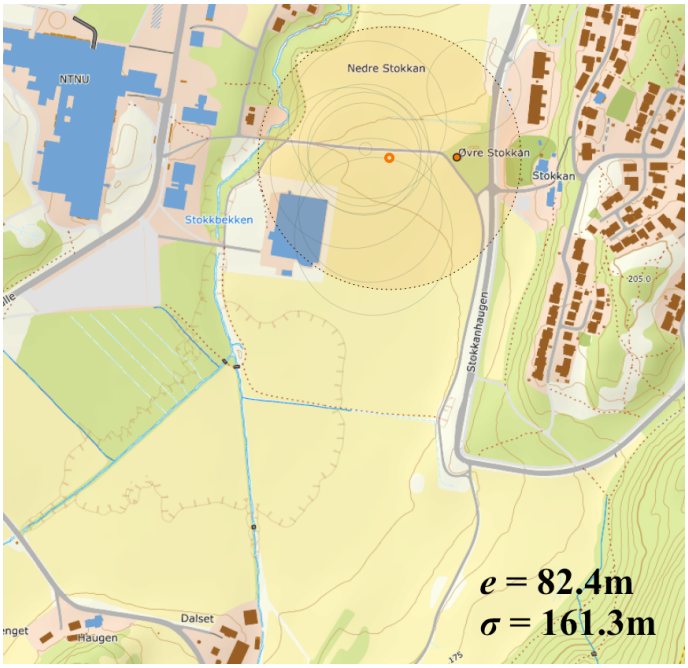
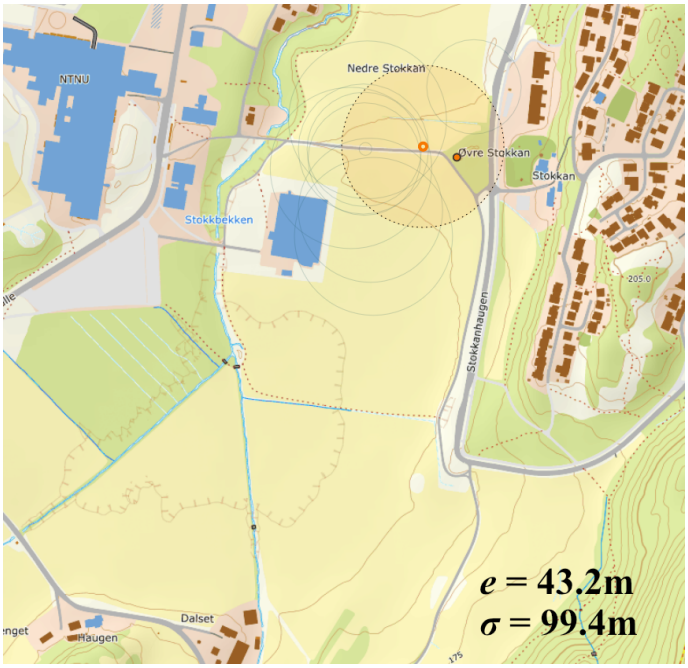


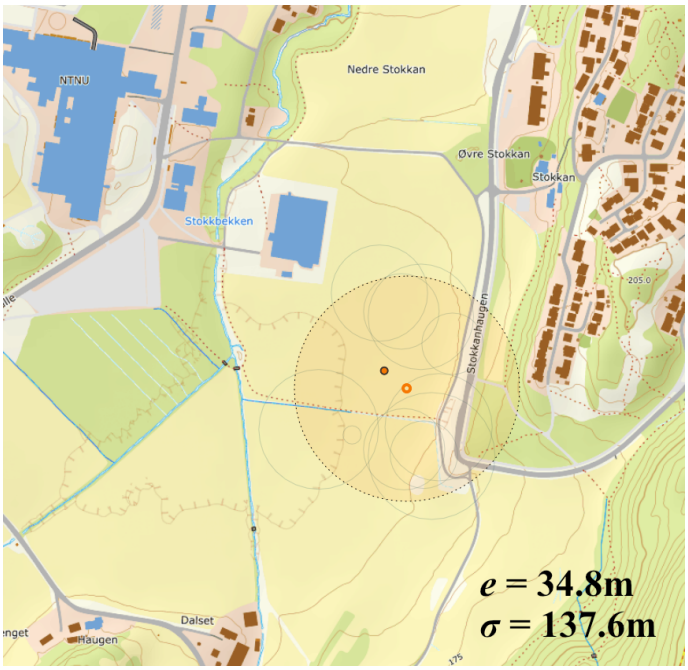
Figure 5.18: Location estimation legend.



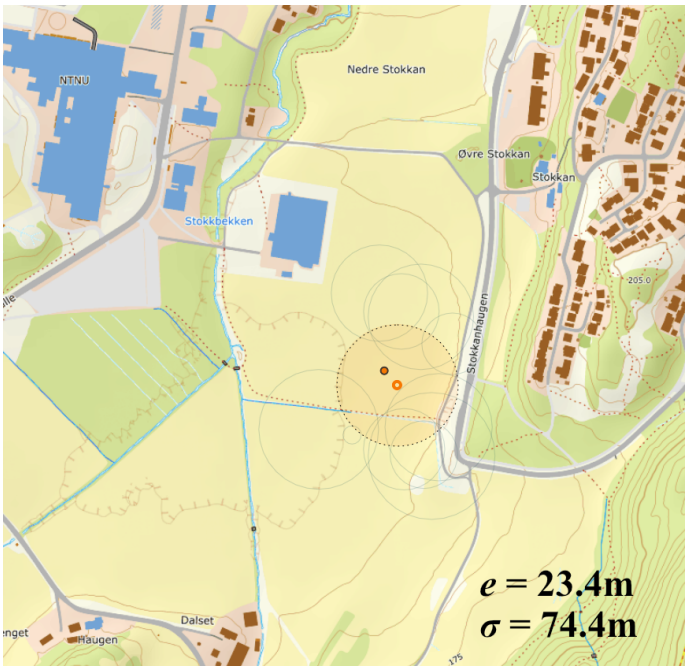
(a) Sheep 1 estimated with particle filter.



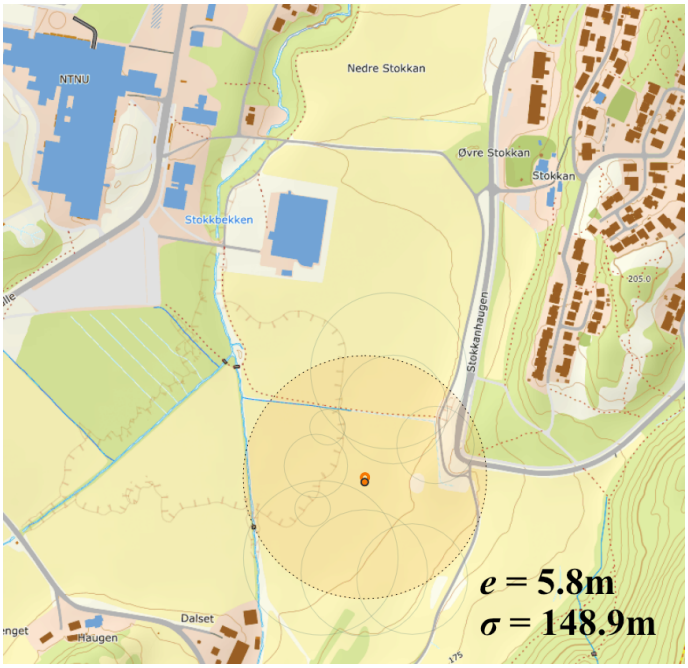
(b) Sheep 1 estimated with multilateration.



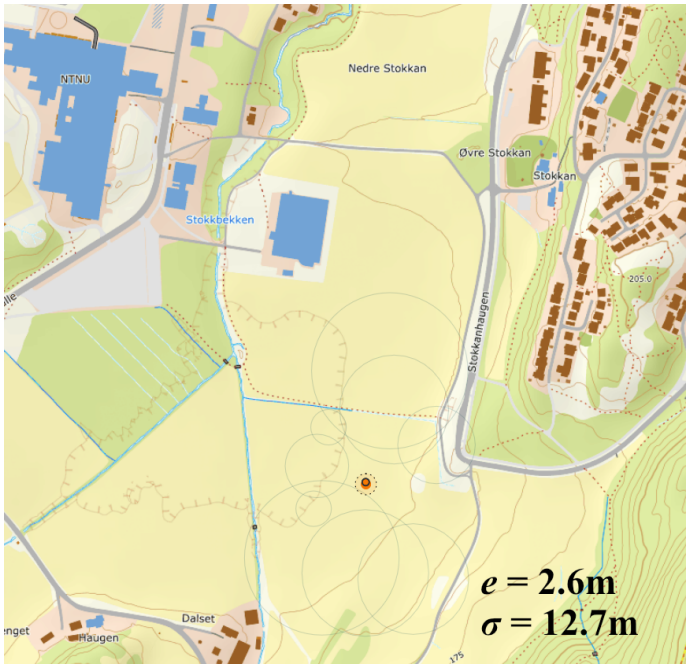
(c) Sheep 2 estimated with particle filter.



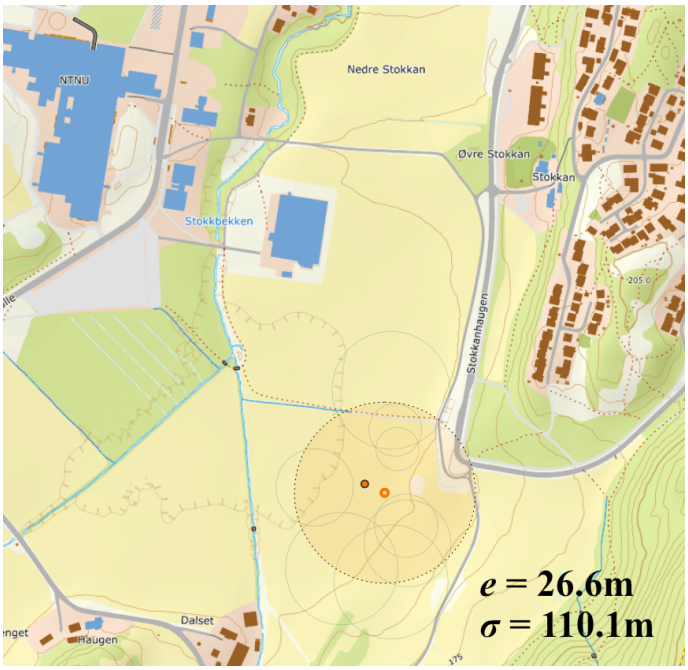
(d) Sheep 2 estimated with multilateration.



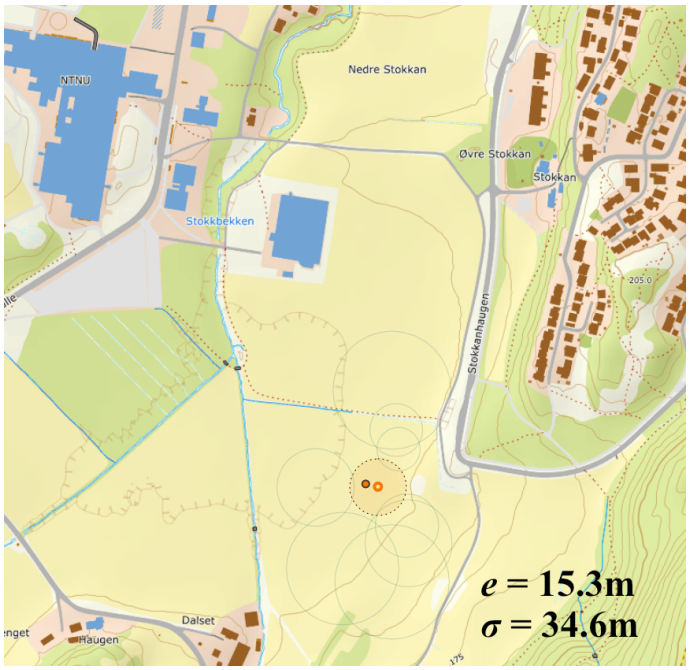
(e) Sheep 3 estimated with particle filter.



(f) Sheep 3 estimated with multilateration.



(g) Sheep 4 estimated with particle filter.



(h) Sheep 4 estimated with multilateration.

Figure 5.19: The location estimation results of dataset 1 in the medium-scale prototype test using both particle filter and multilateration.

These results show that both estimation algorithms can find locations sufficiently close to the actual sheep locations. From figure 5.19e and 5.19f, we see that both methods are estimating a location within a few meters of the target, but the particle filter has a much larger uncertainty radius.

By analyzing the data from all ten datasets, we see that the particle filter method performs worse than the multilateration method on a general basis. Figure 5.20 shows the estimated uncertainty from both algorithms.

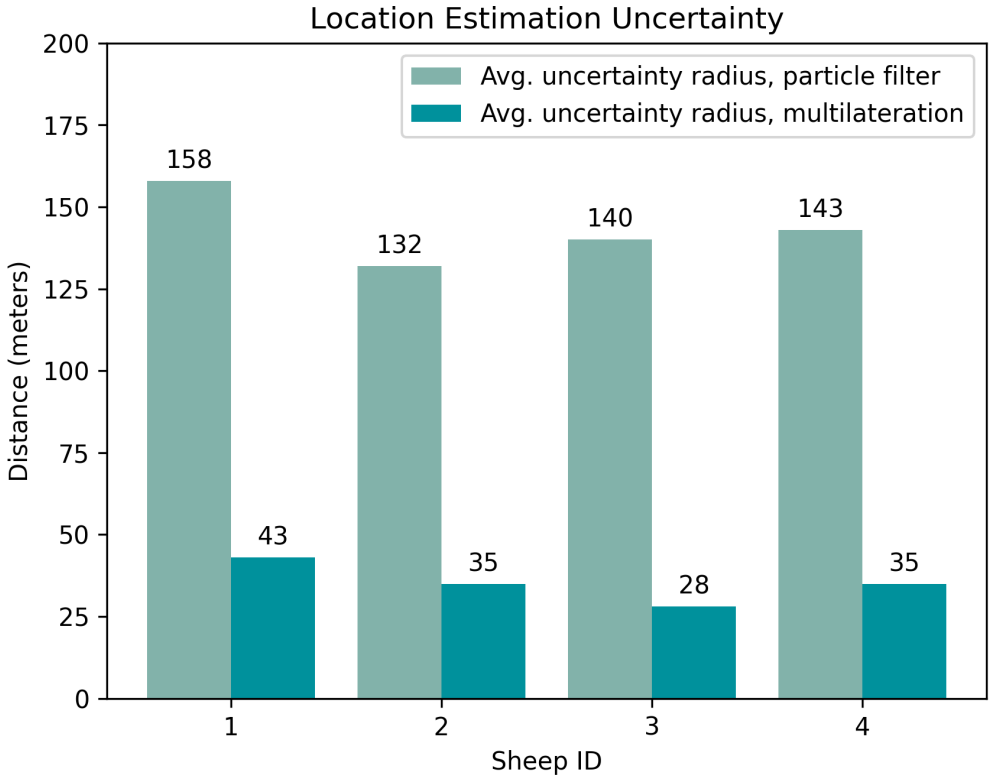


Figure 5.20: The average uncertainty of Radio Sheep GCS's location estimations in the medium-scale prototype test using all ten datasets.

Given that the estimated point always is within the uncertainty radius, both methods can locate sheep with sufficient accuracy, but the multilateration method is undoubtedly performing better.

The following chart in figure 5.21 shows how the location estimation methods performed by showing the error distance from the estimated point to the GPS-measured sheep location.

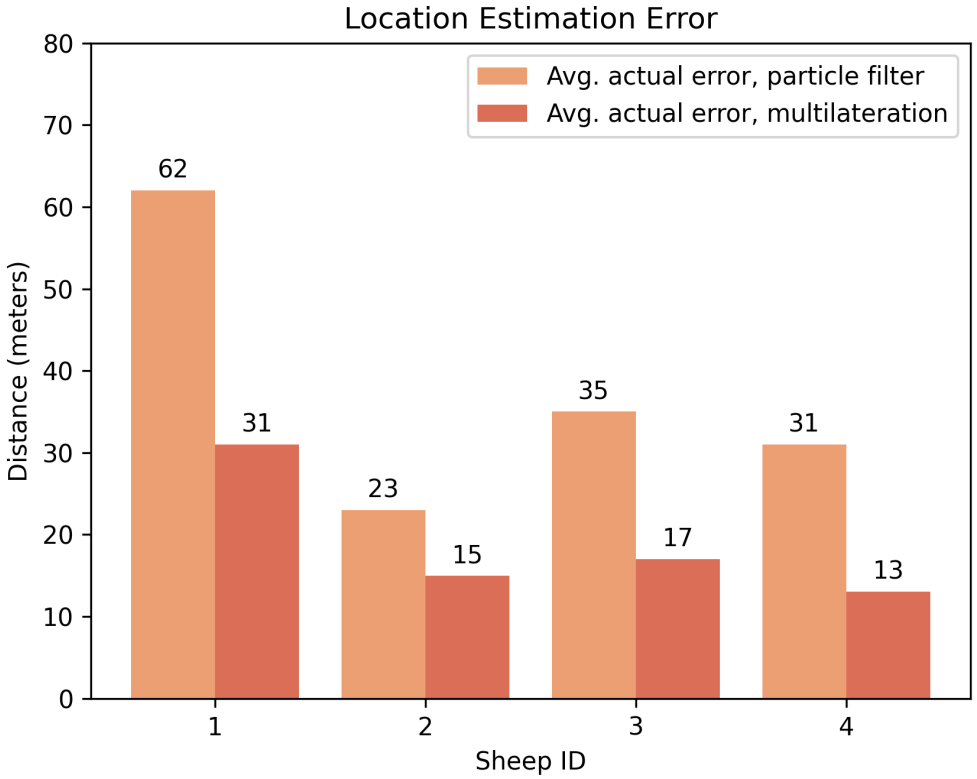


Figure 5.21: The average error of Radio Sheep GCS's location estimations in the medium-scale prototype test using all ten datasets.

The particle filter method and the multilateration method perform well, but we see that multilateration has better performance. The average error across all datasets are **38 meters** for the particle filter and **19 meters** for multilateration.

To evaluate the reliability of the methods, we need to verify that the actual sheep location is within the estimated uncertainty radius. Figure 5.22 presents a comparison of the estimated uncertainties and actual errors.

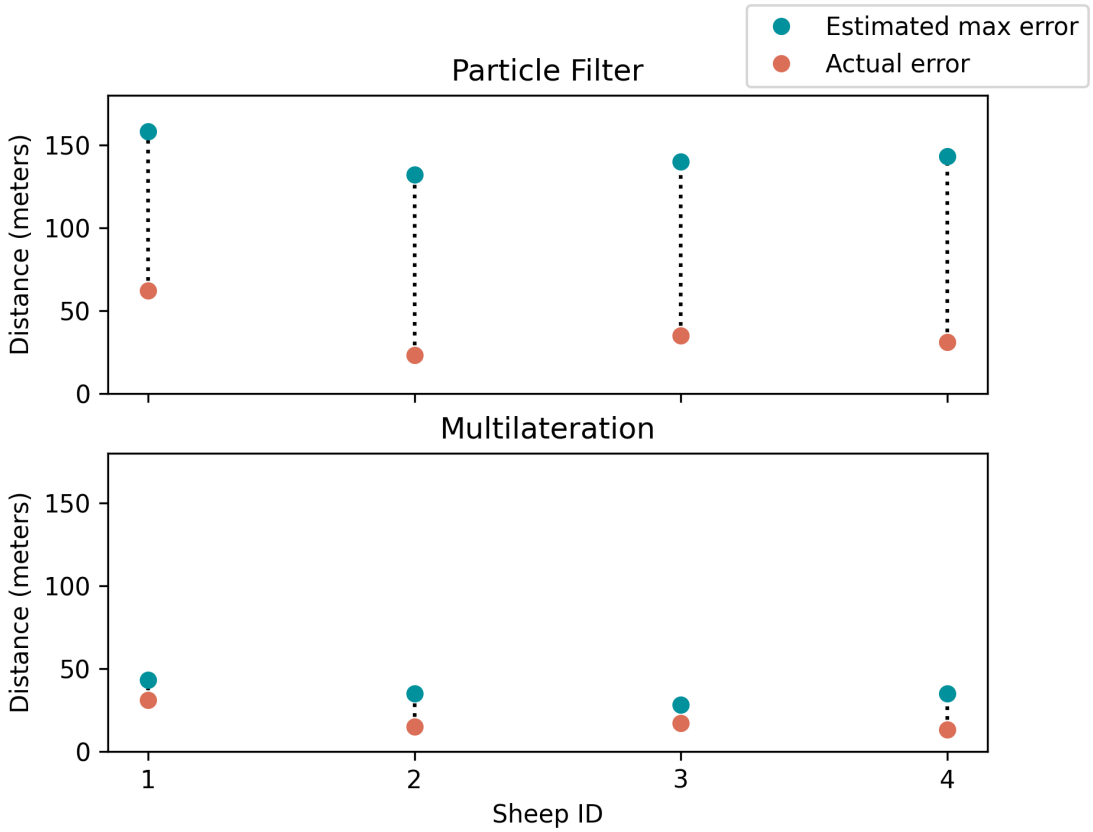


Figure 5.22: The difference of estimated max error and actual error of the localization algorithms in the medium-scale prototype test using all ten datasets.

On average, the actual error distance is always within the estimated uncertainty radius, which is vital for the system's credibility.

Overall, the multilateration method has a lower error and a smaller uncertainty radius. The results from this medium-scale experiment suggest that this system would be able to locate sheep in a large-scale scenario, making radio-tracking of sheep a viable concept.

Key Takeaways

- The drone can fly larger distances on its own and collect sheep data.
- About 10% of all collected sheep samples is without a distance measurement.
- Radio signal obstruction slightly reduces measurement quality.
- Radio Sheep GCS can localize sheep with a sufficiently small margin of error.
- The multilateration method has a smaller uncertainty radius than the particle filter method.
- The multilateration method has less error than the particle filter method.
- On average, the system could locate sheep with an error distance of 19 meters using multilateration.

5.1.6 Large-Scale Simulator Test

To investigate how the localization algorithms' performances were affected by the scale of the search operation, we wanted to use test our system in a large-scale environment. Due to BVLoS restrictions of drone usage in Norway, we could not conduct a large-scale experiment with our physical prototype. The solution was to simulate the drone and the sheep and use Radio Sheep GCS without any modification. We configured the sheep simulator based on previous data gathered in this project to do a simulation as close to a realistic scenario as possible.

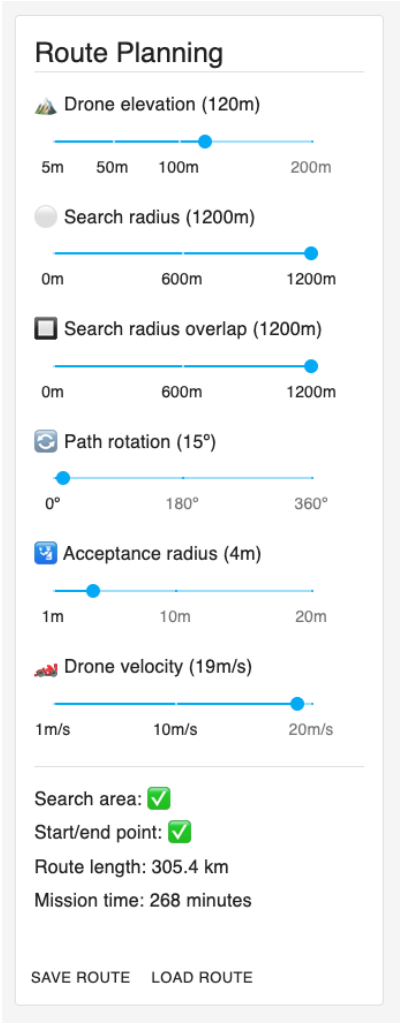
Test Setup

To perform this experiment, Radio Sheep GCS was placed in a simulated environment using the ArduPilot quadcopter SITL and a Python script simulating the sheep nRF52833 radio chips. The sheep simulator source code is available on GitHub in [48]. With Swiderski's experiments in [30], we saw results where the nRF52833 was able to measure distances up to 1200 meters with a consistent uncertainty of ± 50 meters. We used these findings as a basis for the simulator configuration. Twenty-five sheep were randomly generated over a 20km x 20km area and set to advertise their presence once every ten seconds.

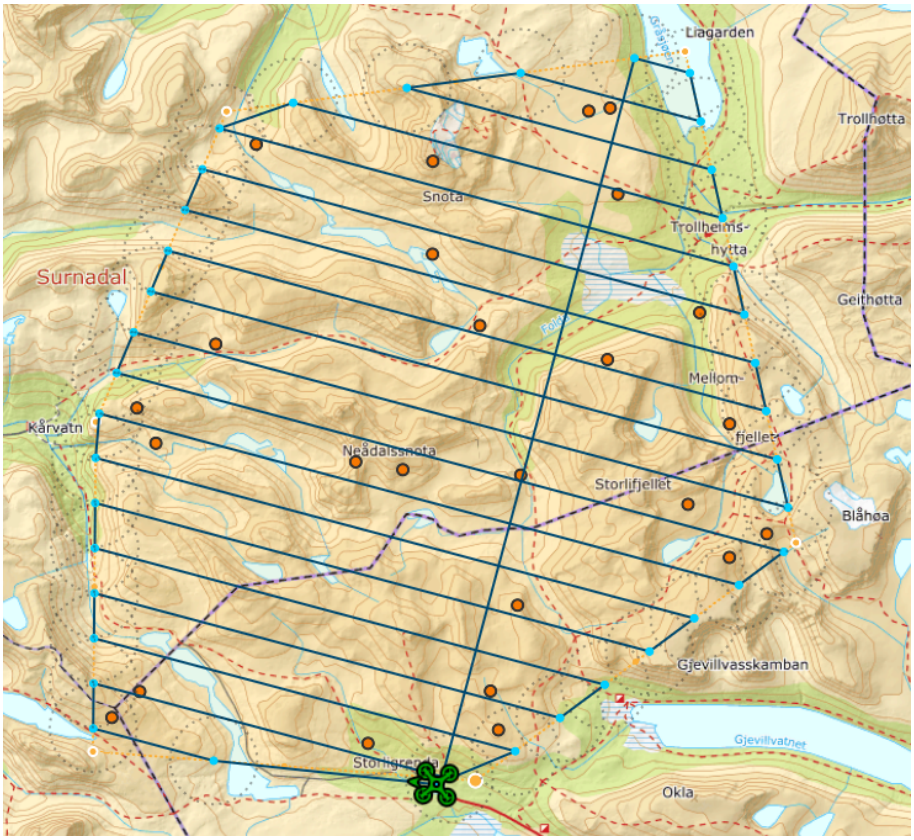
Radio Sheep GCS was used to set up a flight path to cover all the sheep. Figure 5.23 shows the configured flight plan in Radio Sheep GCS. The flight parameters (fig. 5.23a) were set to fly the drone 120 meters above the terrain at a top speed of 19 m/s. The 268 minute mission time provided by Radio Sheep GCS assumes a constant speed of 19 m/s, which the drone cannot maintain at all times. The route (fig. 5.23b) is about 300 km long and ensures that at least two sweep lines detect every sheep. The distance between each sweep line is 1200 meters.

Due to restrictions of the simulators, we were not able to incorporate the terrain as an obstacle in this simulation experiment. The lack of obstacles in the simulation causes the collected sheep data to be unreasonably high quality, making the results somewhat optimistic.

After the data collection, Radio Sheep GCS performed the two location estimation algorithms without filtering out any data.



(a) Flight parameters in Radio Sheep GCS



(b) Route planned and sheep locations in Radio Sheep GCS.

Figure 5.23: Large-scale simulator test plan in Radio Sheep GCS

Results

The localization algorithms' results for all 25 sheep will not be presented individually but introduced as aggregated results. Figure 5.24 shows an overview of the particle filter method's results, and figure 5.25 shows the results for the multilateration method.

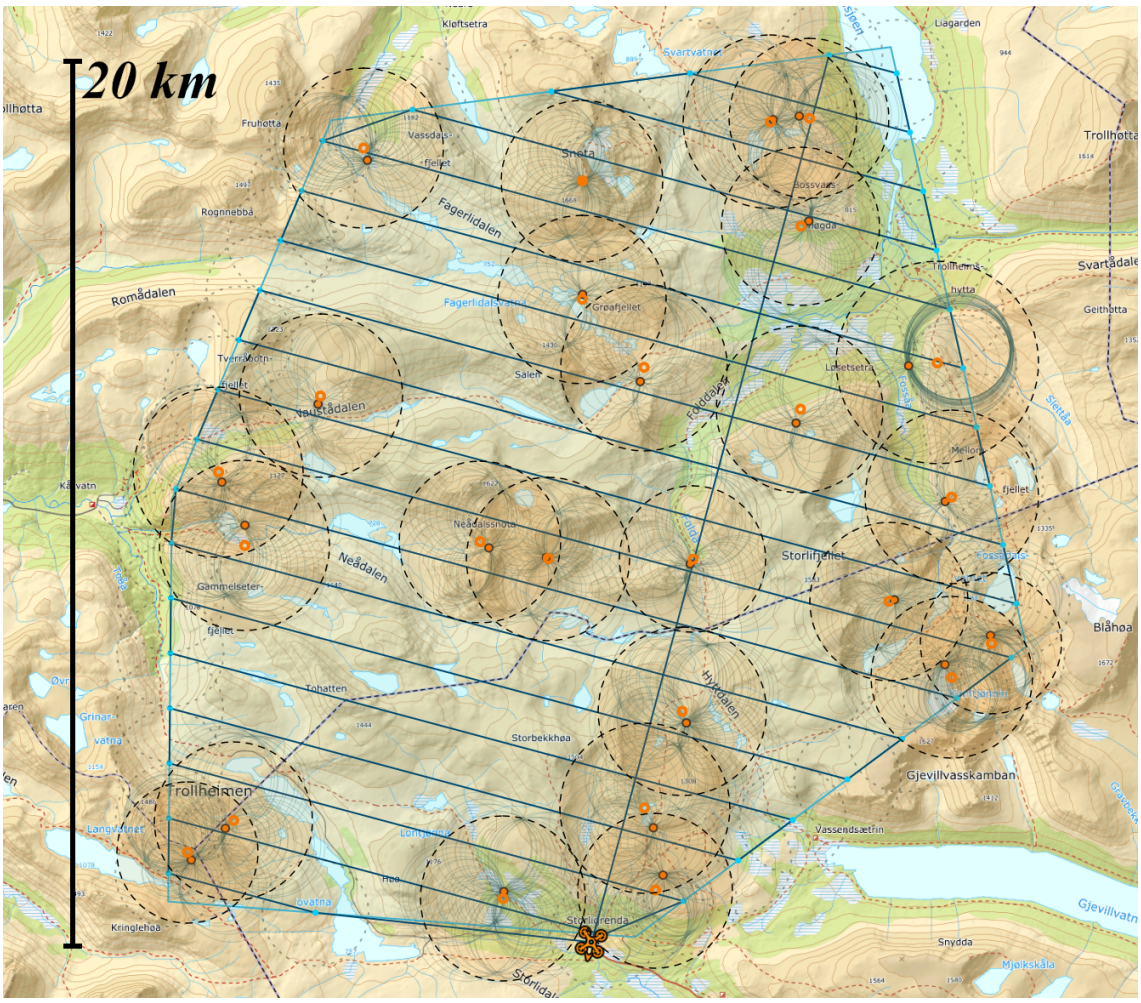


Figure 5.24: Locations estimated in the large-scale simulation using particle filter.

Due to the map's scale in this overview, the estimated locations seem relatively close to the actual positions, but the distances are pretty high. The uncertainty perimeters represented by the dashed circles are unsatisfactorily high. We can see that all the uncertainty areas span multiple sweep lines, and there is a 1.2 km gap between each line. This is not an ideal distance to localize sheep.

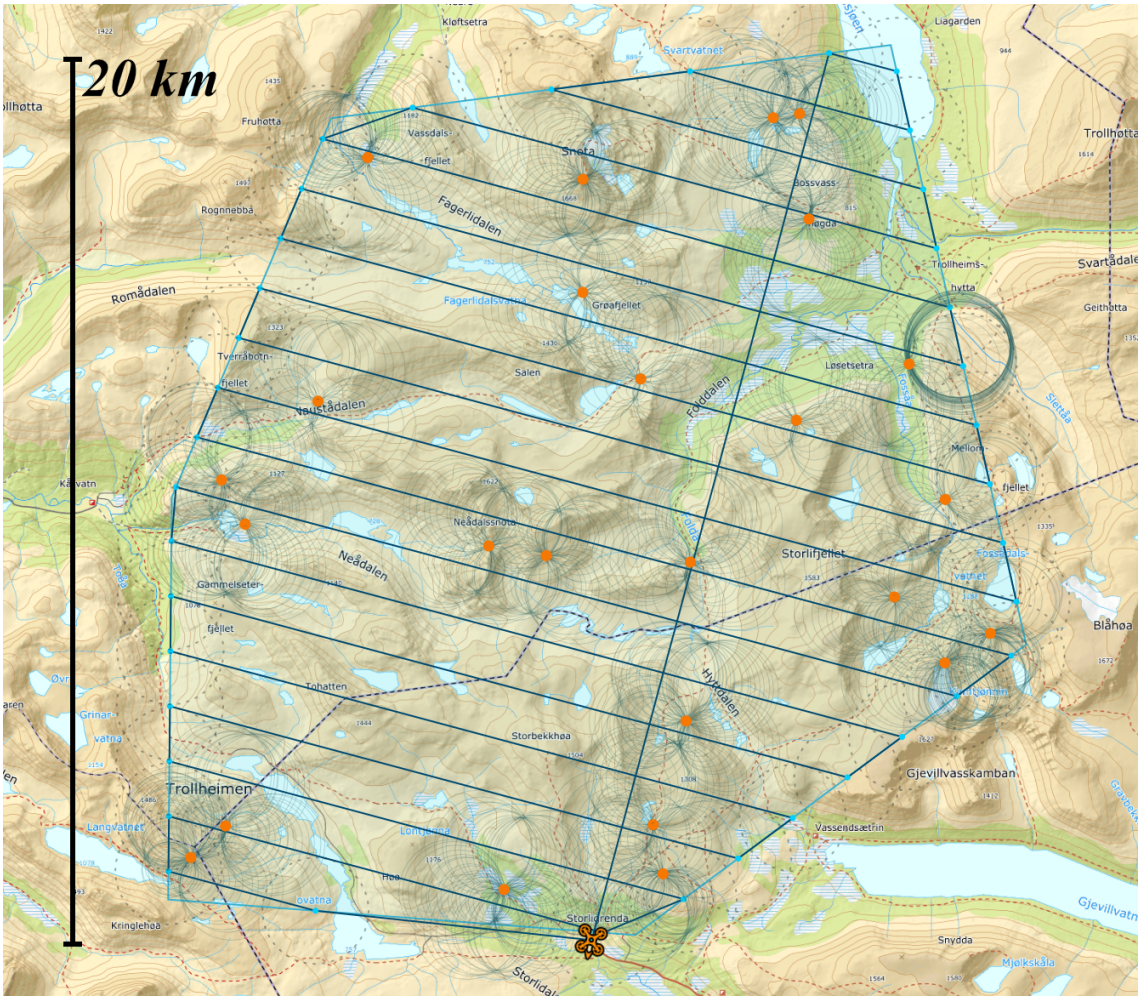


Figure 5.25: Locations estimated in the large-scale simulation using multilateration.

These results look very promising. All the estimated position markers cover their belonging actual position marker. No uncertainty areas are possible to see with the map scale, implying that they are pretty small.

The chart shown in figure 5.26 presents an average of the results for all 25 sheep using both localization algorithms.

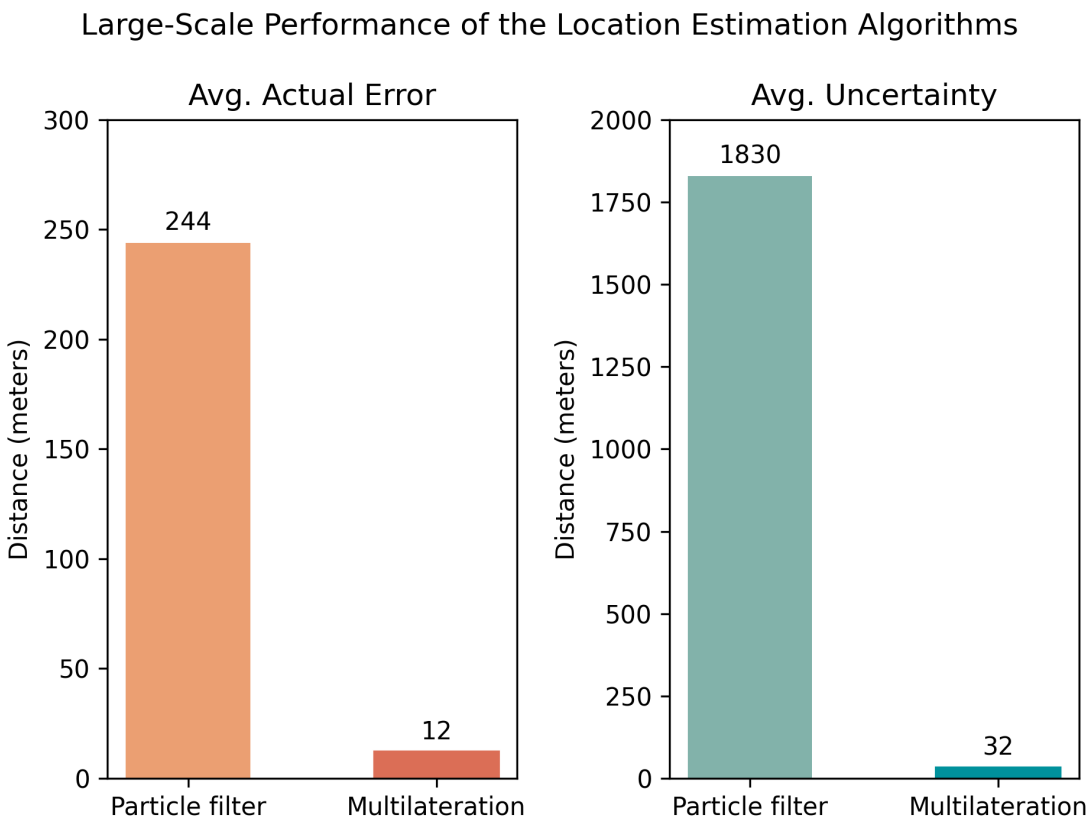


Figure 5.26: Performance of particle filter and multilateration in the large-scale simulator test.

These results suggest that the particle filter method's performance scales poorly with an increased scale of the search mission, while the multilateration method's performance is maintained. The multilateration performance seems to resemble what we observed in the medium-scale prototype test for the sheep placed unobstructed in the open field. This simulator experiment did not account for signal obstruction by the terrain but only the uncertainty of the distance measurements. We cannot be confident that the achieved result will remain this good if tested in a real-world scenario, but it indicates that it might be possible.

Key Takeaways

- The multilateration method performs significantly better than the particle filter method on a large scale.
- The particle filter method's error and uncertainty increase with the scale of the search mission.
- The multilateration method's error and uncertainty do not increase with the scale of the search mission.

5.2 Discussion and Summary

This section discusses the project results and evaluates the feasibility of the system. We will look at the ground control station capabilities, sheep locating performance and compare our solution to existing alternatives.

5.2.1 Ground Control Station Capabilities

Radio Sheep GCS has proven to be a full-fledged ground control station for use in this project. It can communicate with and command our custom-built drone running the ArduPilot autopilot software using the MAVLink protocol. The sheep location estimation algorithms implemented in Radio Sheep GCS have displayed good accuracy and reliability.

Radio Sheep GCS is not a viable alternative to use as a multi-purpose GCS in its current state. It lacks compatibility with other autopilots, multiple UAVs, and support for MAVLink version 1. The route planning in Radio Sheep GCS is limited to coverage path planning only, meaning that the operator cannot place custom waypoints. Radio Sheep GCS is also limited to a maximum of 50 waypoints due to a limitation in the Norwegian Mapping Authority's elevation API. The route planning functionality has been sufficient for conducting our experiments but falls short for most other use-cases.

We considered offline maps and geofence capabilities nice to have for this prototype but not strictly necessary to demonstrate the system. For safety reasons, when conducting our experiments, we chose to use Mission Planner to set up a geofence before Radio Sheep GCS took control over the drone.

Radio Sheep GCS is not yet a feasible option for multi-purpose UAV operations compared to other GCS alternatives. Still, it has fulfilled all requirements needed to create a proof-of-concept prototype for radio-tracking of sheep.

5.2.2 Sheep Localization Range and Accuracy

Our system shows promising results for locating sheep. With our medium-scale prototype test, we showed capabilities of locating sheep with an average accuracy of 19 meters using multilateration. We cannot be confident that this number will stay this low in a larger scale version of the system. Still, our simulator testing suggests that the multilateration performance will remain on the same level. Due to regulations of BVLoS drone usage, we have not been able to test our system in a large-scale, real-world scenario. This kind of experiment would uncover the impact of terrain and other possible obstructions.

In Swiderskis thesis in [30], he shows that the nRF52833 radio chips are capable of measuring distances up to 1.2 kilometers with an uncertainty of ± 50 meters. The range could be even higher with further development of the system.

Radio Sheep GCS has implemented two different localization methods. The first is the geometry-based multilateration method, and the second one is a statistical particle filter method. Both methods compensated for the height difference between the drone and the ground, but neither did localization in three dimensions. Due to

the project's time frame, most of the development time had to be spent early on to create a functioning GCS prototype required to gather data to the localization algorithms in the first place. Despite performing only 2D localization, the methods showed promising results. The multilateration method performed very well. The particle filter method performed ok but struggled when measurements covered larger areas. I believe the reason for this is the lack of assigning weights and filtering of the particles. The particle filter method's current state favors large counts of very similar measurements unreasonably high, but it should instead mitigate their impact. Further development of the method might overcome this issue.

5.2.3 Comparison to Alternative Solutions

Our system offers a new way of tracking sheep without external networks such as NB-IoT or low-orbit satellite networks as dependencies. The network independence of this solution is a tremendous advantage when locating sheep in areas with unstable network connections. However, this system cannot offer real-time tracking or ad-hoc location checking. The drone must travel a long distance to cover the desired search area, often requiring several hours to retrieve valuable data. This delay causes another uncertainty in the location data, as the sheep would most likely move before the drone's return. This system would still be a handy aid in the sheep retrieval process compared to traditional search. Further iterations of this system might enable to drone to connect to mobile networks, allowing it to transfer telemetry and sheep measurements in real-time. This would eliminate some of the travel delays when localizing the sheep.

It is suggested by Nerland in [29] that a radio-enabled ear tag prototype could cost around 100 NOK, which is up to 20 times cheaper than a GPS collar. The tag's weight is estimated to be around 10 grams, which is also a significant reduction compared to existing solutions. The price and weight make it feasible for a farmer to equip his entire sheep herd with our radio tags, which is not the case with GPS collars.

The reliability of our system is highly dependent on the drone's capabilities. A drone failure in inaccessible terrain could be a significant setback for the convenience and trustworthiness of the system. The use of drones in this type of mission would also require certification by the authorities, which will have to be investigated further.

Chapter 6

Conclusion and Future Work

This project has aimed to create a proof-of-concept system for radio-tracking of sheep consisting of a ground control station, an autonomous drone, and a set of nRF52833 Bluetooth-enabled radio chips. In collaboration with Nerland building the drone and Swiderski programming the radio chips, we have successfully implemented and evaluated a prototype displaying this concept's feasibility.

This chapter concludes the work done towards the research and development of Radio Sheep GCS, the localization algorithms and summarizes the whole project.

6.1 Conclusion

This thesis has documented the research and development of a MAVLink-enabled ground control station, named Radio Sheep GCS, with capabilities of locating sheep based on range-only measurements. It was accomplished by first researching the MAVLink protocol, localization techniques, and state-of-the-art ground control stations, and existing sheep tracking solutions. The research was followed by programming the GCS in parallel with performing field experiments providing valuable insights into the development. When the programming concluded, we conducted a series of experiments to evaluate and demonstrate our proof-of-concept prototype.

Radio Sheep GCS has implemented core GCS functionality, allowing it to conduct a UAV flight mission all by itself. Radio Sheep GCS can create customizable flight plans to cover the operator's desired area, and it automatically fetches accurate elevation data from the Norwegian Mapping Authority. The GCS has implemented its communication routines according to the MAVLink protocol. We have developed a custom MAVLink subprotocol to distribute sheep measurements from the drone to the GCS. Radio Sheep GCS proved sufficient to conduct evaluation tests with the complete prototype system but is not ready for multi-purpose GCS usage.

Two different range-only localization algorithms have been implemented in Radio Sheep GCS. The first method was a particle filter variant using statistical calculations to estimate coordinates, and the second algorithm used geometrical multilateration to approximate the sheep locations. Both methods proved capabilities of locating sheep based on range-only measurements with sufficiently small error margins on a smaller scale, but the multilateration method performed consistently better. The particle filter method performed worse than the multilateration method and had issues causing its uncertainty to be unreasonably high. When simulating

a large-scale scenario, the particle filter's performance decreased significantly, while the multilateration performance stayed the same.

Through tests and experiments, we have demonstrated the system's capabilities of locating sheep. Using our physical prototype and the multilateration method, we could locate sheep within 19 meters of the actual location on average. With a large-scale simulated environment, the same method could locate sheep within 12 meters on average.

When comparing our system to other sheep-tracking solutions, we can offer a much cheaper and network-independent solution at the cost of reduced convenience for the user. Our system uses a lightweight radio chip in the sheep's ear tag, while competitors rely on bulky GPS collars. Other systems can track sheep wherever there is a network connection, while our system can only track sheep where the operator chooses to search with the drone. Compared to instantly locating sheep with a GPS collar, our system introduces an unfortunate delay with the drone's travel time, but this can be reduced by connecting the drone to mobile networks for real-time sheep data transfer.

6.2 Future Work

This section proposes some ideas of how to develop our system's capabilities further.

6.2.1 GCS Capabilities

Radio Sheep GCS proved capable of controlling our drone and conduct our experiments, but it is not yet a full-fledged multi-purpose ground control station. To further develop Radio Sheep GCS, the support of multiple AutoPilot types can be implemented. This feature can be achieved by listening to the autopilot flag in the MAVLink heartbeat messages to identify the type and adjusting the communication routines accordingly.

A feature lacking in Radio Sheep GCS is the possibility to download map data for offline usage. This addition would make field experiments more convenient, as the absence of Wi-Fi networks outside caused quite a bit of frustration when we tested our system out in cropland fields.

The route planning functionality should also be further expanded. Placement of arbitrary waypoints and spline waypoints would be an excellent addition to the existing route planning functionality. Complete customizability of each part of the mission in the same way as Mission Planner and APM Planner 2.0 would provide more flexibility when creating a route.

Further investigation of automatic route optimization could be fascinating to conduct. Since the drone often would traverse uneven terrain, it should be possible to optimize the speed and power consumption by doing less vertical travel. The current solution might generate several sweep lines over a mountain ridge, forcing the drone to ascend and descend on every sweep.

6.2.2 Localization Methods

The localization methods implemented in Radio Sheep GCS have shown promising results but are far from perfect. Their most significant limitation is the lack of tracking in three dimensions. The current implementation uses a simple height compensation before feeding the data to the algorithms, but this would cause incorrect results in uneven terrain. Future versions of the implementations should incorporate the height differences in the algorithms and not in pre-processing of the data.

From our medium–scale prototype test, we learned that about 10% of all sheep data packets were missing a distance measurement. This data is still usable, as we can determine that sheep is in the area around the drone. With knowledge of the drone's maximum search radius and introducing processing of distance uncertainty in the estimation algorithms, we can use the empty sheep data packets to help determine the location.

One idea that I did not have time to develop was to use "negative" sheep measurements to determine positions where the sheep cannot be. If the collected sheep distance measurements entail multiple possible locations and the estimation algorithm cannot be sure of which one, we can use the drone's previous GPS data to eliminate areas where the drone did not detect sheep. This solution could effectively double the required range between the search sweep lines, significantly reducing the total flight path's length. A shorter flight path would reduce power consumption and provide an increased range of the system.

6.2.3 System Functionality

Future versions of the system could expand on the functionality it offers. By equipping sheep with various sensors, the system could monitor the sheep's health data. An accelerometer can determine if the sheep has been moving, and if a sheep hasn't moved over an extended period, the system could flag it as deceased.

Our prototype could be expanded upon with further development and evaluation experiments. Testing the system in a large-scale real-world scenario has not been done. This type of testing requires certification from the Civil Aviation Authority of Norway, so we did not have time to do this. Developing an ear tag prototype and attaching it to real sheep would give significant insights into this project's feasibility.

References

[1] MAVLink. *MAVLink Developer Guide*. n.d. URL: https://mavlink.io/en/ (Accessed: 3. Dec. 2020).

[2] MAVLink. *MAVLink – Micro Air Vehicle Message Marshalling Library*. 2010. URL: https://github.com/mavlink/mavlink (Accessed: 4. Dec. 2020).

[3] MAVLink. *MAVLink Versions*. n.d. URL: https://mavlink.io/en/guide/mavlink_version.html (Accessed: 12. Mar. 2021).

[4] MAVLink. *Packet Serialization*. n.d. URL: https://mavlink.io/en/guide/serialization.html (Accessed: 12. Mar. 2021).

[5] MAVLink. *Routing*. n.d. URL: https://mavlink.io/en/guide/routing.html (Accessed: 12. Mar. 2021).

[6] MAVLink. *Heartbeat/Connection Protocol*. n.d. URL: https://mavlink.io/en/services/heartbeat.html (Accessed: 16. Mar. 2021).

[7] MAVLink. *MAVLINK Common Message Set*. n.d. URL: https://mavlink.io/en/messages/common.html (Accessed: 19. Mar. 2021).

[8] MAVLink. *Mission Protocol*. n.d. URL: https://mavlink.io/en/services/mission.html (Accessed: 16. Mar. 2021).

[9] MAVLink. *Command Protocol*. n.d. URL: https://mavlink.io/en/services/command.html (Accessed: 16. Mar. 2021).

[10] MAVLink. *Parameter Protocol*. n.d. URL: https://mavlink.io/en/services/parameter.html (Accessed: 16. Mar. 2021).

[11] Norwegian Civil Aviation Authority. *The Civil Aviation Authority of Norway's main objective is to contribute to safe civil aviation in Norway*. n.d. URL: https://luftfartstilsynet.no/en/drones/new-eu-regulations/ (Accessed: 30. May 2021).

[12] Norwegian Civil Aviation Authority. *New EU-regulations*. 2020. URL: https://luftfartstilsynet.no/en/drones/new-eu-regulations/ (Accessed: 4. Dec. 2020).

[13] Norwegian Civil Aviation Authority. *Open Category*. 2020. URL: https://luftfartstilsynet.no/en/drones/new-eu-regulations/open-category/ (Accessed: 4. Dec. 2020).

[14] Frank Dellaert et al. "Monte Carlo Localization for Mobile Robots". In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA99)*. May 1999.

[15] Jerry C. Hamann. *Exploring the Mathematics of Multilateration*. University of Wyoming - Department of Electrical and Computer Engineering. 2007. URL: http://w3.uwyo.edu/~hamann/TrilatShow.pdf.

[16] Xoneca. *Example of Geometric Dilution Of Precision (GDOP) for simple Triangulation*. 2013. URL: https://commons.wikimedia.org/wiki/File:Geometric_Dilution_Of_Precision.svg.

[17] QGroundControl. *QGroundControl - Intuitive and Powerful Ground Control Station for the MAVLink protocol*. n.d. URL: http://qgroundcontrol.com/ (Accessed: 8. Dec. 2020).

[18] DJI. *DJI GS pro - Mission-Critical Flight Simplified*. n.d. URL: https://www.dji.com/no/ground-station-pro (Accessed: 8. Dec. 2020).

[19] DJI. *About Us*. n.d. URL: https://www.dji.com/no/company (Accessed: 8. Dec. 2020).

[20] ArduPilot. *Mission Planner Home*. 2020. URL: https://ardupilot.org/planner/ (Accessed: 8. Dec. 2020).

[21] ArduPilot. *ArduPilot - Versatile, Trusted, Open*. 2016. URL: https://ardupilot.org/index.php/about (Accessed: 9. Apr. 2021).

[22] ArduPilot. *APM Planner 2 Home*. 2020. URL: https://ardupilot.org/planner2/ (Accessed: 8. Dec. 2020).

[23] Sky-Drones. *SMARTAP GCS*. 2018. URL: https://sky-drones.com/smartap-gcs (Accessed: 8. Dec. 2020).

[24] AirMap. *AirMap - Airspace Intelligence to Power the Drone Economy*. 2015. URL: https://www.airmap.com (Accessed: 29. May 2021).

[25] Findmy AS. *Findmy - Model 2*. 2019. URL: https://www.findmy.no/nb/model2 (Accessed: 1. June 2020).

[26] Telespor AS. *Telespor - Radiobjella.* 2021. URL: https://telespor.no/produkt/ (Accessed: 1. June 2020).

[27] Smartbells AS. *Smartbjella 2*. 2021. URL: https://smartbjella.no/product/smartbjella-2-kjop/ (Accessed: 1. June 2020).

[28] Nordic Semiconductor. *nRF52833*. 2019. URL: https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52833 (Accessed: 9. Apr. 2021).

[29] Trygve Nerland. "Radio-Tracking of Sheep - Developing MAVLink enabled sensors, MAVLink control and the basis for MAVLink enabled UAVs". MA thesis. NTNU: Norwegian University of Science and Technology, June 2021.

[30] Grzegorz Swiderski. "Radio-Tracking of Sheep: Low-Cost Energy-Efficient Coarse Distance Estimation using Bluetooth Low Energy Transceiver". MA thesis. NTNU: Norwegian University of Science and Technology, June 2021.

[31] The Norwegian Mapping Authority. *About the Norwegian Mapping Authority*. n.d. URL: https://www.kartverket.no/en/about-kartverket (Accessed: 14. Apr. 2021).

[32] H. Butler et al. *The GeoJSON Format*. RFC 7946. Aug. 2016. DOI: 10.17487/RFC7946. URL: https://rfc-editor.org/rfc/rfc7946.txt.

[33] M Coombes et al. "Optimal Polygon Decomposition for UAV Survey Coverage Path Planning in Wind". In: *Sensors (Basel)* (July 2018). DOI: 10.3390/s18072132. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6068989/.

[34] Emanuele Goldoni et al. "Experimental analysis of RSSI-based indoor localization with IEEE 802.15.4". In: May 2010, pp. 71–77. DOI: 10.1109/EW.2010.5483396.

[35] Sajina Pradhan and Suk-Seung Hwang. "Mathematical analysis of line intersection algorithm for TOA trilateration method". In: *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and*

15th International Symposium on Advanced Intelligent Systems (ISIS). 2014, pp. 1219–1223. DOI: 10.1109/SCIS-ISIS.2014.7044849.

[36] OpenJS Foundation. *Electron - Build cross-platform desktop apps with JavaScript, HTML, and CSS*. 2013. URL: https://www.electronjs.org/ (Accessed: 4. Dec. 2020).

[37] OpenJS Foundation. *Node.js*. 2011. URL: https://nodejs.org/en/ (Accessed: 27. May 2020).

[38] Google. *Chromium*. 2008. URL: https://www.chromium.org/Home (Accessed: 27. May 2020).

[39] Facebook. *React*. 2013. URL: https://reactjs.org/ (Accessed: 4. Dec. 2020).

[40] Dan Abramov. *Redux - A Predictable State Container for JS Apps*. 2015. URL: https://redux.js.org/ (Accessed: 27. May 2021).

[41] Material-UI. *Material-UI - React components for faster and easier web development.* 2018. URL: https://material-ui.com/ (Accessed: 4. Dec. 2020).

[42] Google. *Material Design*. 2014. URL: https://material.io/design (Accessed: 4. Dec. 2020).

[43] Microsoft. *TypeScript: Typed JavaScript at Any Scale*. 2012. URL: https://www.typescriptlang.org/ (Accessed: 27. May 2020).

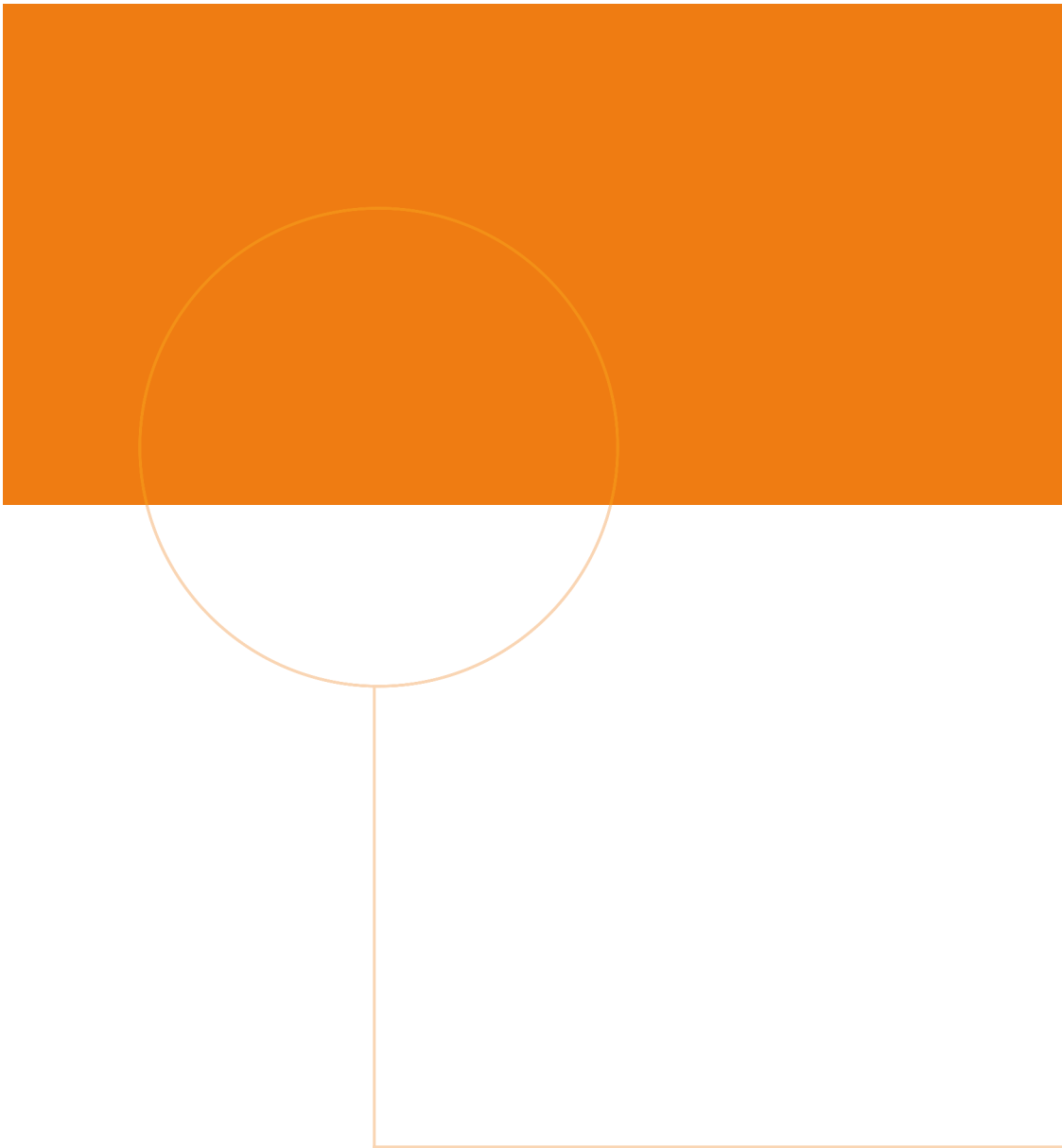
[44] Mapbox. *Mapbox - Maps and location for developers*. 2010. URL: https://mapbox.com/ (Accessed: 4. Dec. 2020).

[45] Turf. *Turf.js - Advanced geospatial analysis for browsers and Node.js*. 2013. URL: https://turfjs.org/ (Accessed: 27. May 2020).

[46] Pascal Groß. *node-mavlink*. 2019. URL: https://github.com/ifrunistuttgart/node-mavlink (Accessed: 28. May 2021).

[47] ArduPilot. *MAVProxy*. 2019. URL: https://ardupilot.org/mavproxy/ (Accessed: 28. May 2021).

[48] Trygve Nerland and Gard Steinsvik. *sheep-2021-emulator*. 2021. URL: https://github.com/trygve55/sheep-2021-emulator.



NTNU

Norwegian University of
Science and Technology