Technical Documentation

Search and Rescue - Underwater

Version 1.0

Author: Alexander Roser
David Andersson
Hampus Frick
Isac Lundin
Ken Dahl
Oscar Holm
Oskar Philipsson

Date: December 19, 2022

Status

Reviewed	-	-
Approved	-	-

Project Identity

Group E-mail:	searchandrescueunderwater2022@gmail.com
Homepage:	http://www.grphomepage.se
Orderer:	Gustav Zetterqvist, ISY
	E-mail: gustav.zetterqvist@liu.se
Customer:	Andreas Gällström, Saab Dynamics
	E-mail: andreas.gallstrom@saabgroup.com
Customer:	Jonatan Olofsson, Saab Dynamics
	E-mail: jonatan.olofsson@saabgroup.com
Supervisor:	Daniel Bossér, ISY
	E-mail: daniel.bosser@liu.se
Supervisor:	Philip Andersson, Saab Dynamics
	E-mail: philip.e.andersson@saabgroup.com
Supervisor:	Erik Söderberg, Saab Dynamics
	E-mail: erik.soderberg@saabgroup.com

Group Members

Name	**Responsibility**	**E-mail**
Alexander Roser (AR)	Head of testing	alero223@student.liu.se
David Andersson (DA)	Head of documentation	davan957@student.liu.se
Hampus Frick (HF)	Project leader	hamfr643@student.liu.se
Isac Lundin (IL)	Head of hardware	isalu162@student.liu.se
Ken Dahl (KD)	Head of software	kenda091@student.liu.se
Oscar Holm (OH)	Head of information	oscho082@student.liu.se
Oskar Philipsson (OP)	Head of design	oskph717@student.liu.se

Document History

Version	Date	Changes made	Sign	Reviewer
0.1	2022-12-09	First draft.	DA, IL, OH, OP, AR, HF, KD	DA
0.2	2022-12-15	Second draft.	DA, IL, OH, OP, AR, HF, KD	DA, IL, AR
0.3	2022-12-19	Third draft.	DA, IL, OH, OP, AR, HF, KD	DA
1.0	2022-12-19	First version.	DA, IL, OH, OP, AR, HF, KD	DA

Contents

1 Introduction **1**

1.1 Parties . . . 1

1.2 Background . . . 1

1.3 Definitions . . . 1

2 System overview **2**

2.1 Hardware . . . 2

2.1.1 UAS . . . 2

2.1.2 Navigation Sensors . . . 2

2.1.3 Hydrophone . . . 2

2.1.4 Barometer . . . 3

2.1.5 Sound card . . . 3

2.1.6 Pi computer . . . 3

2.1.7 Manual controller . . . 3

2.1.8 UUV . . . 3

2.1.9 Emergency pinger . . . 4

2.1.10 Base station . . . 4

2.2 Communication . . . 4

2.3 Communication scheme for the data collection . . . 4

2.4 Intended communication scheme . . . 5

3 Software architecture **6**

3.1 ROS2 . . . 6

3.1.1 Custom interfaces . . . 7

3.1.2 Sound simulator/recorder . . . 7

3.1.3 Estimator . . . 7

3.1.4 Offboard Controller . . . 8

3.1.5 Task planner nodes . . . 8

3.2 GUI - QGroundControl . . . 8

3.3 Mechanics simulation . . . 9

3.3.1 UAS without cable . . . 9

3.3.2 UAS with 1 segment . . . 10

3.3.3 UAS with 2 segments . . . 10

3.3.4 UAS with 9 segments . . . 10

3.3.5 Water . . . 10

4 Sound simulation **12**

5 Task planner **12**

5.1 Simulation . . . 12

5.1.1 Task-planner node . . . 12

5.1.2 Semi-planner node . . . 12

5.1.3 Emergency-planner node . . . 13

5.1.4 Battery node . . . 13

5.2 Reality . . . 13

5.3 Emergency protocol . . . 14

6 Preprocessing of the emergency signal **15**
6.1 Filtering . . . 15
6.2 Generate logarithmic power measurements . . . 15

7 Position estimation of the emergency pinger **16**
7.1 Received signal strength model . . . 16
7.2 Estimation of P^0 and β . . . 16
7.3 Minimizing the loss function . . . 17
7.4 Likelihood function of RSS . . . 17

8 Motion planner **19**

9 Control algorithms **19**
9.1 Position and velocity control . . . 19
9.2 Angle and angular rate control . . . 20
9.3 Hydrophone oscillations . . . 21

10 Future work **22**
10.1 Get hardware fully working . . . 22
10.2 More advanced sound simulation . . . 22
10.3 Adaptive measurement point selection algorithm . . . 22
10.4 More advanced signal processing algorithms for stationary submersion . . . 22
10.4.1 Doppler effect - stationary submersion . . . 23
10.4.2 Received time difference - stationary submersion . . . 23
10.4.3 Received signal strength with triangulation - stationary submersion . . . 24
10.5 More advanced signal processing algorithms for mobile submersion . . . 25
10.5.1 Doppler effect - mobile submersion . . . 25
10.5.2 Received signal strength - mobile submersion . . . 25
10.6 Analog filters / look at another sound card . . . 25
10.7 Better Control for UAS . . . 26

1 Introduction

This document is the technical documentation for the project Search and Rescue Underwater in the course TSRT10, Reglerteknisk projektkurs, CDIO, given at Linköping University.

1.1 Parties

In this project there are three parties involved; the customers, Andreas Gällström and Jonatan Olofsson at Saab Dynamics, the client Gustav Zetterqvist at Linköping University, an advisor Daniel Bossér at Linköping University and a project group working on the project. The members of the project group, their roles, and contact information can be found in the beginning of the document.

1.2 Background

One of Saab Dynamics' largest departments designs underwater crafts of various types. These are used for inspecting underwater infrastructure. In the unfortunate case of a submarine somehow gets stuck during a mission, an emergency pinger is activated to be able to find the submarine and retrieve it again. You often then have to go out by boat and lowering hydrophones to find out where the emergency ping sound is coming from. In stressful environments, this process takes a very long time to search the area. Therefore, instead of searching an area manually, an autonomous Unmanned Aircraft System could complete the task. Thus, the goal is to find a distressed Unmanned Underwater Vehicle that transmits an emergency signal, with an Unmanned Aircraft System and hydrophone.

1.3 Definitions

- UAS - Unmanned aircraft system, i.e., drone.
- GUI - Graphical user interface.
- ROS2 - Robot operating system.
- Session - The time span from which the UAS is turned on until the UAS is turned off.
- Mission - Contains all tasks produced by the task planner. Multiple missions can be executed in one session.
- UUV - Unmanned underwater vehicle.
- Emergency pinger - Transmitter of the emergency signal relayed by the UUV.
- Semi-autonomous mode - A mode where the user is coordinating where the UAS should go.

2 System overview

The full system is divided into multiple subsystems. These are; UAS, ground station and underwater craft, in which these subsystems also can be divided into smaller subsystems.

The UAS contains of a flight controller which controls the UAS with additional sensors and a Raspberry Pi which takes in the information from the sound card and calculates/transmits the data to the ground station. On board the UAS, there is a hydrophone which gathers the data (the ping) from the distressed UUV which is recorded on the the sound card.

The UUV has an emergency pinger which it activates when it is in distress.

The ground station does the more advanced calculations that the Raspberry Pi onboard the UAS cannot do.

Additionally, there is also a simulation environment that is used for development and evaluation purposes.

2.1 Hardware

This section contains descriptions of the general hardware and the components that are used.

2.1.1 UAS

The body of the UAS is a *Holybro x500 v2* [4] equipped with a *Pixhawk 6C Flight Controller*, *PM02 V3-12S Power Module*, *M8N GPS Module* and a *SiK Telemetry Radio V3 433/915MHz* (range $\sim$ 300 m).

The UAS carries the hydrophone and sound card. In the simulation environment, the signal processing and control algorithms are implemented on the UAS. In reality, only the control algorithms are implemented on the UAS.

2.1.2 Navigation Sensors

The UAS has different sensors that are used for the navigation. They are as follows:

- GPS – Global Positioning System. The GPS has a NEO-M8N module, with a horizontal accuracy of 2.5 m (CEP 50%)[9].
- IMU – The combination of accelerometer, gyroscope and magnetometer.
- Barometer – Measures the current air pressure that the UAS is experiencing which contributes to the height estimation, see section 2.1.4 for more information.

2.1.3 Hydrophone

The hydrophone that will be used is called *AS-1 Hydrophone*, and is manufactured by Aquarian Scientific [1]. The hydrophone is attached to a nine-meter long cable that will be connected to the UAS. Information about the hydrophone is listed below:

- Receiving Sensitivity: -208dBV re 1 μPa (40μV/Pascal)
- Linear range: 1Hz to 100kHz $\pm$ 2dB,

- Horizontal Directivity(20kHz): ± 0.2dB
- Horizontal Directivity(100kHz): ± 1dB
- Vertical Directivity(20kHz): ± 1dB
- Vertical Directivity(100kHz): + 6dB -11dB

2.1.4 Barometer

The barometer is of model MS5611. It has an altitude resolution of 0.065/0.042/0.027/0.018/0.012 mbar dependent on the oversampling ration, where the oversampling ratios are 256/512/1024/2048/4096. It has an accuracy of ±1.5 mbar at 25°C and 750 mbar and the error band is at -20°C to +85°C and 450 to 1100 mbar: ±2.5 mbar. Its full operating range is 10 to 1200 mbar at -40 to +85°C. The response time is 0.5/1.1/2.1/4.1/8.22 ms dependent on the same oversampling rations as earlier.

2.1.5 Sound card

A sound card is mounted on the UAS and takes the analog data from the hydrophone and send it to the Raspberry Pi. The sound card is of the model *Zoom F3*. Over USB, it has support for sending 2-channel audio at 96 kHz. This is just enough to sample the roughly 45 kHz signal that the pinger is emitting. The sound card's settings can also be controlled over Bluetooth in case physical access to the sound card is not possible.

2.1.6 Pi computer

The computer that is used is a *Nanopi M4*. It is a more powerful Raspberry Pi clone. It has the same GPIO pins as a normal Pi, but a slightly different layout. The pin layout and other details are available on the wiki: `https://wiki.friendlyelec.com/wiki/index.php/NanoPi_M4`. It has Wi-Fi and Bluetooth support, but requires an external antenna. It can also talk over UART with the flight controller using the serial ports in its GPIO pins. To get this working, a custom cable was manufactured to connect the Pi:s debug serial port to the **TELEM 2** serial connection on the flight controller.

2.1.7 Manual controller

When the UAS is driven outside, it must be possible to control manually to be able to abort the session at any time [8]. Therefore, a handheld controller must be connected to the UAS and supervised by a person, during flight time.

The controller used is a *Radiomaster TX16S*. It has a transmission frequency in the range 2.4 – 2.48 GHz and a range of 2 km [3]. Besides controls for throttle and steering, the controller has a switch that change flight mode, one that arm & disarm the UAS and one that kills the UAS. The implemented flight modes are manual- and altitude mode [6].

2.1.8 UUV

The UUV will be transmitting a distress signal using the emergency pinger, see section 2.1.9 for more information about the emergency pinger. The UUV will be sending the distress signal when an accident has occurred and the UUV is immobile.

2.1.9 Emergency pinger

The emergency pinger is of model *LINDA UNDERWATER ACOUSTIC BEACON - MODEL C1A* and transmits sinusoidal pulses with a frequency of 45 kHz. The pulses are about 0.02 s long and have a period time of about 0.7280 s. The transmitted signal may propagate with spherical or cylindrical symmetry. This depends both on the model and the depth in the water.

2.1.10 Base station

The base station consists of a laptop that controls the session. Computations are split between the base station and the Pi on the drone so that the heavy computations needed are done on the base station which has a more powerful processor.

2.2 Communication

This section describes how communication between the different subsystems work. Two different schemes of communication will be explained. First, the communication scheme that was used for the real data collection will be described, then the communication scheme that was originally intended to be used will be explained.

2.3 Communication scheme for the data collection

The communication scheme used for the real data collection is illustrated in Figure 1.

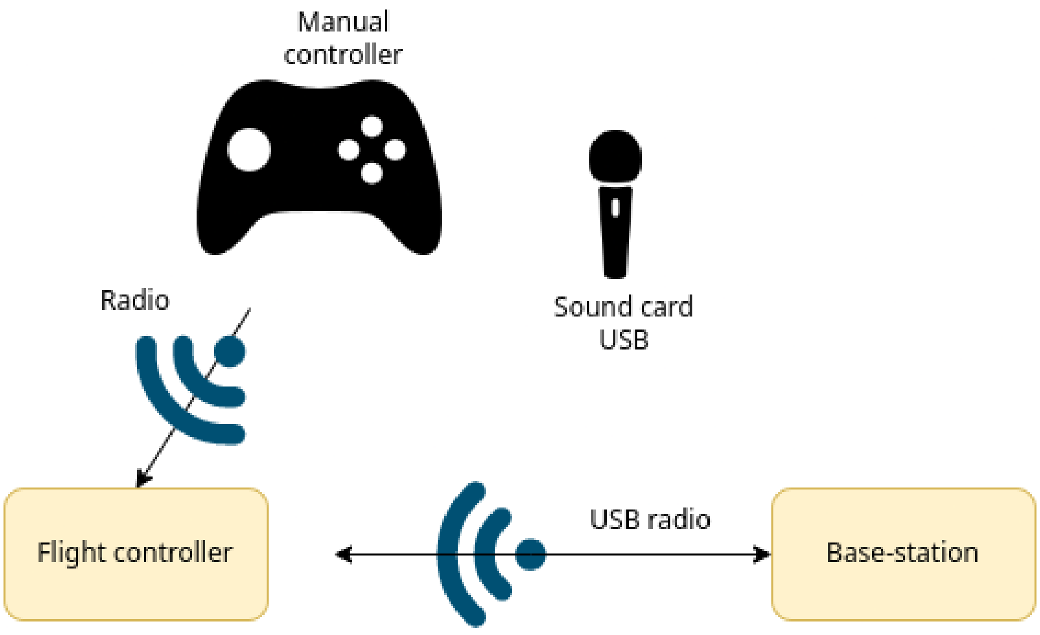


Figure 1: Communications used when collecting real data

The drone was controlled using a USB-radio by the program QGroundContol. In QGroundControl, the wanted waypoints along with geo-fences were added. As a safety and legal measure, a handheld radio controller was used to allow manual control of the drone. The sound card was not controlled during the flight, but was set to record in continuous mode to an SD-card.

2.4 Intended communication scheme

The orignally intended communication scheme is illustrated in Figure 2.

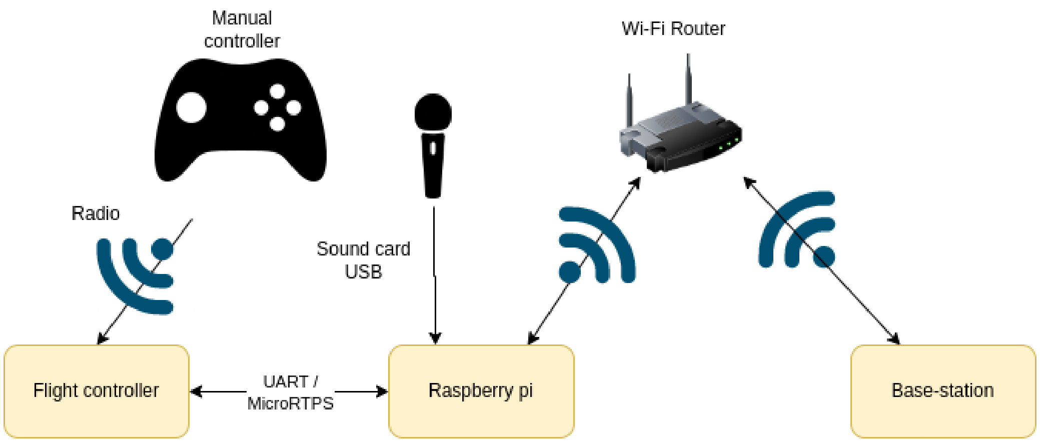


Figure 2: Intended communication between all subsystems in project

On the UAS there is a flight controller and a Raspberry Pi. They talk to each other over a UART port. The actual protocol they talk over is called MicroRTPS, which is a publish / subscribe protocol, similar to ROS2 topics. Over this bus, sensor data from the flight computers sensors are sent, as well as flight instructions to the flight controller from the Raspberry Pi. The Raspberry Pi also talks to the sound card over USB.

The base station and Raspberry Pi communicate with each other through a Wi-Fi router, which allows communication within the local network. The router is created by setting up a hotspot on a mobile phone that both the UAS and base station connects to. The range of the drone will therefore be limited to the range of the Wi-Fi network. A manual controller is also used here for manual control.

This communications scheme has been tested and verified to work at the communication level. However, because of unresolved issues regarding setting waypoints from ROS, it could not be used for the real data collection. The advantages to the intended communication scheme over the practically used one, is that it allows for full automation of all the subsystems. With the scheme used for the real data collection, the sound card could not be controlled during flight. This adds extra steps to the cleaning process of the data before the estimation can be done. Also, in QGroundControl, the drone could not be controlled programmatically. It could only fly to predefined waypoints. This makes it impossible to implement dynamic dipping positions.

3 Software architecture

This project uses an advanced software architecture, which allows for software in the loop testing (SIL). This means that the same code that should be used on the real hardware, can run against a simulation environment where the hardware is replaced with a digital twin. The software should also be run in a reproducible environment so that dependency problems are eliminated. To achieve this, the software has been developed in Docker containers running on Linux. ROS2 was chosen as the main framework for the project, utilizing the already developed simulation tool Gazebo, and visualization tool RViz2.

The overview of the software architecture is shown in Figure 3. It shows that the majority of the code is identical for the simulation as well as when running on real hardware. The OR blocks select whether to simulate hardware or not.

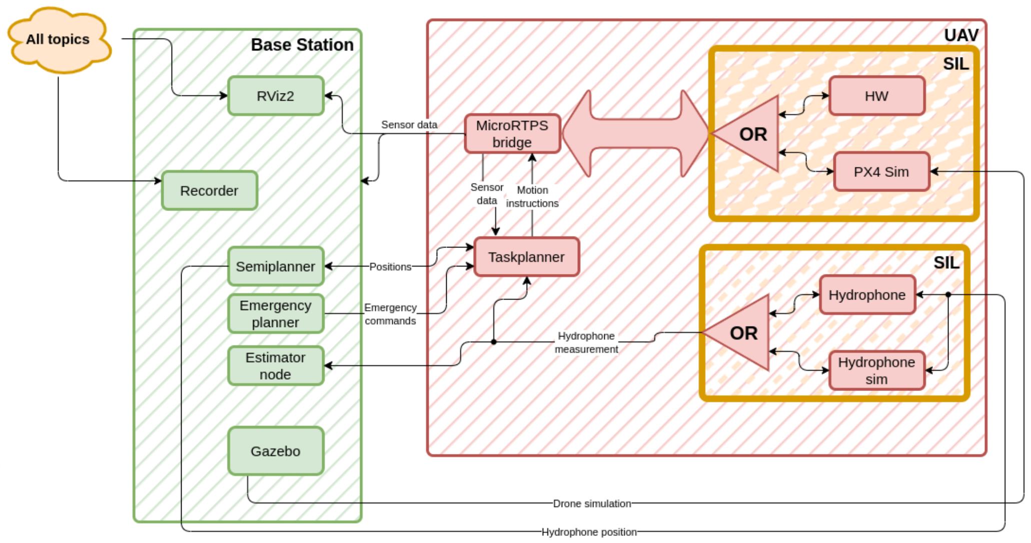


Figure 3: Software architecture overview

The intention was that exactly the same code should be used in the simulation environment as the code running on real hardware. The only difference should be how the docker containers are set up. However, some issues regarding setting waypoints from ROS did not work as expected. Therefore, the majority of software was only used for the simulation environment. However, when the hardware issues are resolved, the same code should work perfectly on the real hardware.

3.1 ROS2

As previously mentioned, the project heavily utilizes the tools from the ROS2 project. The project code is divided up into the main nodes shown in Figure 3. The nodes communicate with each other through one of the three communications methods in ROS2: topics, services and actions. What data is communicated between the nodes are also shown in the same figure.

The nodes that require a lot of computational power, have been placed on the base station and the nodes that need to interact with real hardware have been placed on the UAS.

There also exists a recorder node running on the base station to record all messages sent in ROS2, but it has not yet been used. This is done in ROS2 bags. These bags can then later be used for replaying scenarios for debugging purposes, or replaying events from running on real hardware, in the simulation environment.

Next, all the ROS2 interfaces and groups of nodes used will be listed and explained. Before that, the important ROS topics used in the project are summarized in Table 1.

Table 1: List of important ROS

Topic	**Subscribers**	**Publishers**	**Description**
/sound_rec/start	hydrophone_sim, hydrophone_hw	planner	Start sound recording
/sound_rec/recordings	planner, estimator	hydrophone_sim, hydrophone_hw	Sound recordings
/estimator/plot_likleyhood	estimator	planner	Plot likelihood
/uav_trajectory_setpoint_pose	offboard_controller	planner	Send position to drone
/uav_arm_command	offboard_controller	planner	Arm the drone

3.1.1 Custom interfaces

Interfaces in ROS2 are the definitions of the structures of the messages that are used on topics, services and actions. One custom message was created to be able to send sound samples on over topics. The sound message has the following structure:

- **float32**[] samples: the sound samples in a list of **float32**:s.
- **uint32** sample_rate: the sampling rate used in Hz.
- **int32** time_start_capture: Unix time stamp when the recording started.
- **geometry_msgs/Pose** hydrophone_pose: a ROS2 geometry message describing the pose of the hydrophone at the time of measurement.

3.1.2 Sound simulator/recorder

The sound simulator and recorder consists of a set of two nodes. Their interfaces are identical. They subscribe to the **/sound_rec/start** on which the hydrophones position is published. This is done so that the simulator node can simulate the sound that the hydrophone would have received at that location. For details regarding the sound simulation, see section 4. The real recorder node has no use for this, but is still used to keep the interface consistent. When the start topic is received, the recorder node starts a recording and the sound simulator computes the simulated sound. Then the nodes publish their sound result on the **/sound_rec/recordings** topic.

The sound simulator node has been tested and verified working. The recorder node has only been tried on a laptop using its microphone, but not with the hydrophone as a microphone on the real Pi.

3.1.3 Estimator

The estimator is a single node that takes in the sound measurements that are published to the topic **/sound_rec/recordings** and stores them in a list. Every time new data comes in, estimation is done on all available measurements and estimations are done using three algorithms. Two likelihood based algorithms with different assumptions on distribution and one using a Gauss-Newton approach on a cost function. More details on the estimation are given in section 6. The estimates are printed to the log console. The estimator node also subscribes to the **/estimator/plot_likleyhood** topic. When the boolean `true` is sent to this topic, the estimator node freezes and displays the likelihood plots.

3.1.4 Offboard Controller

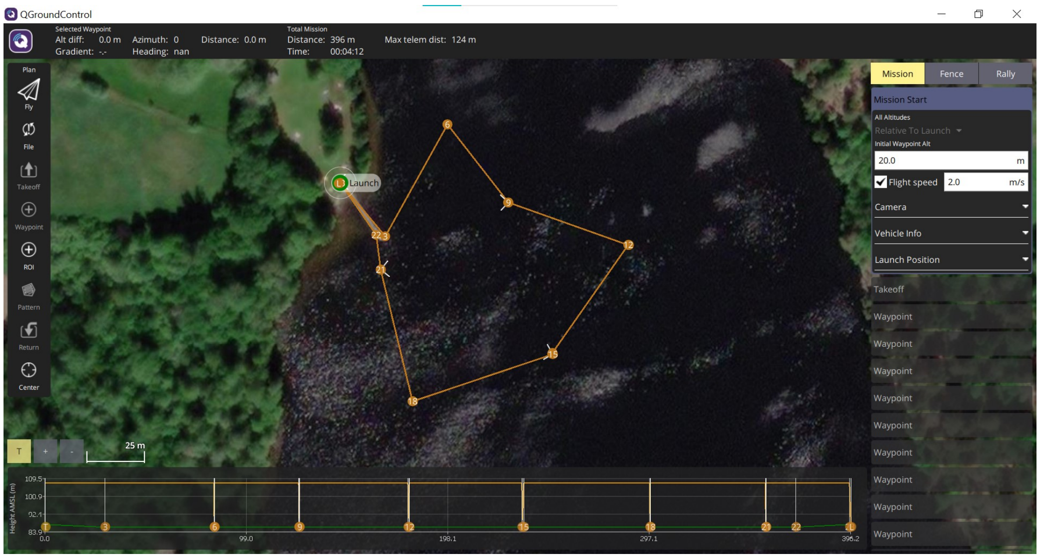
The offboard controller acts as a middle man between our nodes and the PX4 bridge. It listens to the **/uav_arm_command** topic and the **/uav_trajectory_setpoint_pose** topic. It then creates the appropriate messages for the bridge to arm and send positions to the PX4 bridge.

3.1.5 Task planner nodes

The purpose of the task planner nodes is to decide what coordinates the drone should move to next. It is also responsible for handling emergencies like the base station and UAS losing connection or the UAS battery getting too low. It arms the UAS by talking to the PX4 bridge/simulator on the topic **/uav_arm_command**. It also publishes the positions the drone should be at on the topic **/uav_trajectory_setpoint_pose**. It also orders the sound recorder/simulator to take a measurement on the topic **/sound_rec/start**. For more information about the task planner nodes, see section 5.

3.2 GUI - QGroundControl

The GUI runs on the base station through QGC (QGroundControl). The purpose of it is to provide the user with information and the ability to send commands to the UAS. It displays battery level, an estimate of flown distance, elapsed time, and a visualization of the vicinity. Commands include changing between manual and semi-autonomous mode, as well as defining 3D-waypoints to fly to and changing control parameters. A view of how QGC looks like is shown in Figure 4.



Selected Waypoint
Alt diff: 0.0 m Azimuth: 0 Distance: 0.0 m
Gradient: -.- Heading: nan

Total Mission
Distance: 396 m
Time: 00:04:12

Max telem dist: 124 m

All Altitudes: Relative To Launch
Initial Waypoint Alt: 20.0 m
✓ Flight speed 2.0 m/s

Figure 4: A figure that shows how the interface of QGC looks like when planning a mission.

3.3 Mechanics simulation

The mechanical simulations are run in *Gazebo11*, an open-source software library collection with physically realistic 3D-environment simulation capabilities. It allows the user to simply drag and drop 3D shapes into the simulation world without having to provide any mathematical models of the shapes. The virtual UAS can via a terminal receive and execute positional commands to fly to.

Since the hydrophone and the cable change the dynamics of the UAS, four different models of the system are used:

- UAS without cable
- UAS with 1 cable segment
- UAS with 2 cable segments
- UAS with 9 cable segments

The models with cable use universal joints for joining the different parts together, meaning that each cable segment can rotate around both the x- and y-axis.

All models give different results in the simulations, none of which were fully accurate to the real UAS with hydrophone and cable. Still, the models with 2 and 9 segments gave the most accurate results, compared to the oscillations observed in the real-life UAS.

3.3.1 UAS without cable

This model only contains the original UAS model built by PX4, which is mostly used to try the motion and task planner without the added difficulty of the hydrophone. It is also the main model to test the communication between the UAS and ROS2 in the simulation.

3.3.2 UAS with 1 segment

This model is made by adding a 9 m long stiff cylindrical beam with the exact weight of the physical cable with hydrophone, 425 g homogeneously distributed throughout the beam and is shown in Figure 5a This is the simplest model of the UAS with the cable and hydrophone, but the representation of the real world model is fairly bad since the weight and dynamics are not very accurate, since most of the weight is at the bottom of the cable and not homogeneously distributed.

3.3.3 UAS with 2 segments

This model is made up of two segments, one 8.9 m long and one 0.1 m long, which are fused together using a universal joint and is shown in Figure 5b. The 8.9 m segment was attached to the UAS and is meant to symbolize the cable part of the hydrophone and cable, and by that has a homogeneously distributed weight of 265 g. The second segment symbolizes the weight at the end of the cable by the hydrophone and weighs 160 g.

This model was found to be the most useful model. This is because the complexity and the number of errors that occurred are very low, since the number of segments are fairly few.

Even with the much simpler model than later models, the dynamics and weight distribution gave good simulations of the real UAS with the cable and hydrophone.

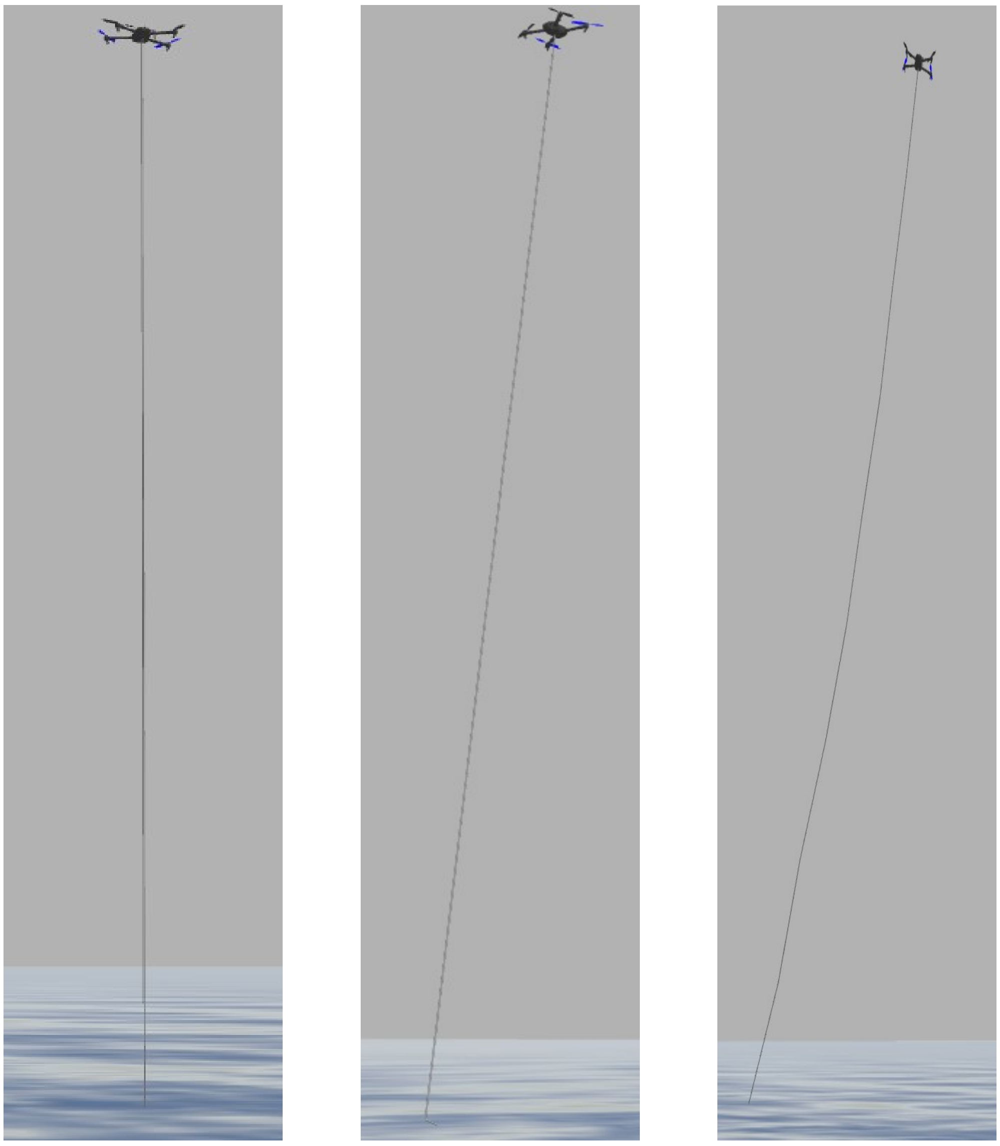
3.3.4 UAS with 9 segments

This model is made by adding together nine 1 m long segments, where the first 8 segments weigh 33.125 g each and symbolize the cable. The last segment weighs 160 g and symbolizes the weight at the hydrophone. The model is shown in Figure 5c.

This should give the most realistic cable since it includes the most amounts of segments, but this also adds some problems. Since the universal joints could rotate 360 degrees, the different segments can flip around and hit the other segments above or below them. This results in that sometimes if the UAS flies too aggressive and by that moves the “cable” too fast, self-oscillations can occur between the segments. Thus, the oscillations will not die out, which sometimes results in the simulation crashing.

3.3.5 Water

To simulate the water in Gazebo in this project, only visual water with a physical bottom is used. This means that visually, the hydrophone can be submerged into the water, but it does not physically affect the hydrophone and drone. This is used to test visually if the hydrophone does get submerged during a mission controlled through the task planner.



(a) The UAS with cable and hydrophone built by one segment.

(b) The UAS with cable and hydrophone built by two segments.

(c) The UAS with cable and hydrophone built by nine segments.

Figure 5: The UAS with the three different cable models.

4 Sound simulation

The sound simulation node generates measurements y, from a real 15 seconds long recorded sample s, according to the following simple model:

$$y = \frac{s}{\|x - p\|^{\frac{\beta}{2}}} + e \tag{1}$$

where β is the path loss coefficient, x is the pinger position, p is the hydrophone position and e is added Gaussian noise. β represents the power loss, and therefore the amplitude loss will become $\beta/2$. So far in the project, β has been assumed to be 2 according to spherical propagation.

5 Task planner

Different systems are used in simulation and reality. In simulation, the task planner consists of several nodes. In reality, the program *QGroundControl* is used, which has a GUI where you can configure most of the required safety-features.

5.1 Simulation

In simulation, the Task Planner-package consists of multiple nodes that communicate with each other over topics. The different nodes are:

- Task-planner
- Semi-planner
- Emergency-planner
- Battery

5.1.1 Task-planner node

This node is the one that communicates with the Px4 controller. It receives coordinates from the emergency- & semi-node and decides what coordinates to pass on to the Px4. The task-planner also pass on commands for measurement request and arm-requests. The task-planner prioritize the commands from the semi-planner, unless the battery gets too low, which the battery-node decides. If, for some reason, the task-planner stops to receive commands from the semi-planner, it will start listening to the emergency node, and fly to the pre-defined home position.

The task-planner node also sends commands to make measurements with the hydrophone if it receives such command from the semi-planner.

5.1.2 Semi-planner node

This node is responsible for coordinating a mission of pre-defined coordinates where the UAS should take measurements.

The semi-planner will start by sending arm commands to the task planner which instructs the UAS to takes off and hover above the home position.

The mission starts when the semi-planner sends the first position to measure to the task-planner. It will continue to do so until the task-planner confirms that the UAS has reached that position. Then, it will request the UAS to submerge the hydrophone, by sending coordinates that are five meters below the current position to the task-planner. When the new position is reached, it will request a measurement and wait until that measurement is done. Then, it will command the task-planner to fly up again, so the hydrophone is above water. Then, it will send the next location and repeat the procedure. When the UAV has visited all locations, the semi-planner will send the first location again and revisit all other locations.

5.1.3 Emergency-planner node

The emergency node will request the task-planner to fly home. When the UAS has flown home, it will send a command to disarm.

5.1.4 Battery node

The battery node simulates battery consumption. When the UAS starts to fly, it will keep track of how long it has been in the air. The battery percentage is then estimated with a predefined maximum fly-time and how long it has been in the air according to

$$percentage = \frac{T_{max} - T_{air}}{T_{max}}, \tag{2}$$

where T_{max} is the maximum fly-time and T_{air} is the time it has been in the air.

The node also estimates how long it can travel by estimating how long time it would take to fly home from the current position. Assuming that the home position is in $(0, 0, 0)$ [m] and the current position is (x, y, z) [m], the estimated time to fly home is given by

$$T_{home} = \frac{|z| + \sqrt{x^2 + y^2}}{0,7}. \tag{3}$$

Here, the assumed speed to fly home is 0,7 m/s.

If the battery percentage gets below 30 % or the remaining fly-time exceeds the time to fly home, the battery node will tell the task-planner that the battery is low.

5.2 Reality

When the mission is performed in reality, the mission is constructed through *QGround-Control* (QGC)[2]. To set up a mission, the user constructs a new plan in QGC. From there, the user chooses positions to fly to. For each position, the user can choose the speed to fly there, height, and if the UAS should stay there for some time. By default, the UAS will land where it launched. It is necessary to define a new "land"-position. This is important to do if you want the UAS to land at a specific position. When the UAS executes an "emergency landing", the UAS will do the following:

1. Fly up to an altitude of 40 meters above ground.
2. Fly back to the land location.
3. Fly down, above the land position, to an altitude of 10 meters.

4. Hover for 10 seconds.
5. Descend back to ground.
6. Disarm.[1]

5.3 Emergency protocol

The UAS can, at all time, be driven manually with a remote controller. In case of manual intervention, the UAS needs to be reset by the base station, in order to operate autonomously again. The remote controller is equipped with a kill-switch, that if used, kills all motors on the UAS. The UAS cannot do anything if the kill-switch is engaged.

The UAS can by itself estimate its battery level. If the battery level gets below 30%, a warning will be shown on the QGC window. If it gets below 10 %, the UAS will execute the landing procedure described above. If it gets below 5 %, it will land wherever the UAS is at that moment. The UAV will also execute the landing procedure if it looses connection with the remote control or base-station. A geo-fence is also defined around the take-off position. This fence defines the volume from where the UAS can fly. The fence is defined as a cylinder with height 80 meters and radius 100 meters.

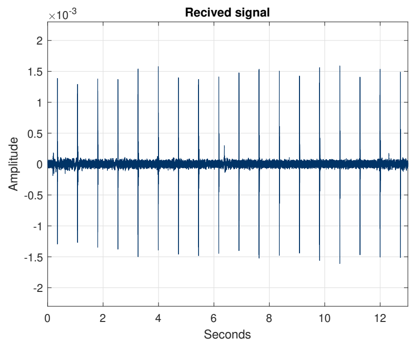
[1]If the UAS does not disarm, use the kill-switch on the remote controller.

6 Preprocessing of the emergency signal

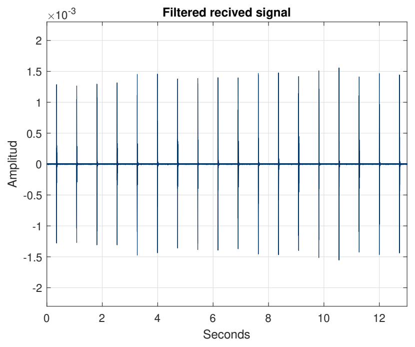
To be able to determine the location of the UUV, relevant information is needed to be concluded from the signal emitted by the emergency pinger. This section describes how the signal received by the hydrophone is dealt with in order to generate logarithmic power measurements.

6.1 Filtering

The signal transmitted by the emergency pinger is emitted by a pulse every 0.7 seconds with a frequency of 45kHz, as seen in Figure 6a. A bandpass Butterworth filter of order six was created between 44KHz and 46KHz, yielding a frequency response depicted in Figure 7. This was done to minimize the present noise. The resulting filtered received signal can be seen in Figure 6b.



(a) Received signal



(b) Filtered received signal

Figure 6: Transmitted signal of the emergency pinger

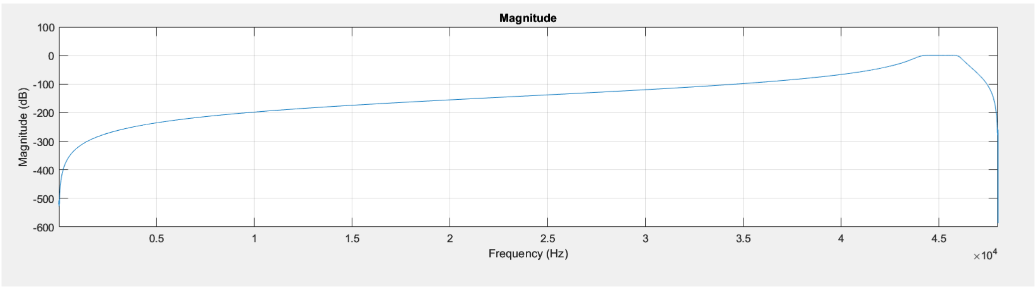


Figure 7: Frequency response of the Butter filter

6.2 Generate logarithmic power measurements

The measurements are obtained by calculating the power of the filtered signal according to

$$P = \frac{1}{N}\sum_{n=1}^{N} s_n^2 \tag{4}$$

Here, P is the power, s is the filtered signal and N is the total number of samples in s. Since the emergency pinger emits pulses with a constant time interval, a time window containing a fixed number of pulses, m, is considered in each measurement. Therefore, N is determined by the number of samples in the time window, which depends on how m is chosen. For example, $m = 4$ appears to be sufficient. In this manner, several measurements are obtained from each recorded signal. Finally, in order to fit the measurement model described in section 7.1, the logarithm of the power is calculated.

7 Position estimation of the emergency pinger

In this section, the measurement model is first given. This is followed by the two methods that are used to estimate the position of the emergency pinger.

7.1 Received signal strength model

The model that is used in this project to enable position estimation of the emergency pinger is the received signal strength(RSS) model. The RSS measurement model is determined by equation,

$$y_k = P^0 - \beta \log\left(\|x - p_k\|\right) + e_k. \qquad [7] \tag{5}$$

In accordance to this project, P^0 is emitted power from the emergency pinger, β is the path loss coefficient, x is the position of the emergency pinger and p_k is the position of the hydrophone at measurement k whilst e_k is additive measurements noise. Here, the measurement y_k is assumed to be the logarithmic power of the signal, i.e. $y_k = \log(P_k)$ when comparing to equation (4).

7.2 Estimation of P^0 and β

As can be seen in equation (5), the measurement model is linear in the parameters P^0 and β which may be utilized in several ways. In this project a calibration for the two constants is made. That is, we gather measurements when the position of the emergency pinger is known. Then, since $\|x - p_k\|$ in (5) can be seen as a constant, the model is completely linear with parameters P_0 and β. Thus, the two parameters can be estimated with the weighted least square (WLS) approach according to

$$\begin{bmatrix} \hat{P^0} \\ \hat{\beta} \end{bmatrix} = \left(\sum_{k=1}^{K} H_k^T R_k^{-1} H_k\right)^{-1} \sum_{k=1}^{K} H_k^T R_k^{-1} y_k, \tag{6}$$

where $H_k = [1 \ -log(\|x - p_k\|)]$, $R_k = Cov(e_k)$, y_k is the measurement and K is the total number of measurements [7]. By an investigation during the project when inspecting the noise in different signals, it appears that the measurement noise e_k can be seen as an independent and identical distributed process. That is, $Cov(e_k) = diag(R_1, ..., R_N) = diag(R, ..., R)$, which means that the WLS estimate is equivalent to the least square (LS) estimate [7]. Therefore, the estimation of P^0 and β is reduced to

$$\begin{bmatrix} \hat{P^0} \\ \hat{\beta} \end{bmatrix} = \left(\sum_{k=1}^{K} H_k^T H_k\right)^{-1} \sum_{k=1}^{K} H_k^T y_k \tag{7}$$

7.3 Minimizing the loss function

Minimizing a loss function by adjusting the parameters in a model to fit the measurements is a classical approach in parameter estimation theory. In this case, we want to estimate the hydrophone position, x, in the model given in equation (5). This corresponds to minimizing the loss function, that is solving the nonlinear optimization problem

$$\hat{x}^{NWLS} = \underset{x}{\operatorname{argmin}} \frac{1}{2} \sum_{k=1}^{K} (y_k - h_k(x))^T R_k^{-1} (y_k - h_k(x)) \tag{8}$$

Here, y_k is the measurement, $h_k(x) = P^0 - \beta log(\|x - p_k\|)$ and $R_k = Cov(e_k)$ [7]. With the same arguments as stated in Section 7.2, the nonlinear weighted least square (NWLS) estimate is equivalent to the nonlinear least square (NLS) estimate given by

$$\hat{x}^{NLS} = \underset{x}{\operatorname{argmin}} \frac{1}{2} \sum_{k=1}^{K} (y_k - h_k(x))^T (y_k - h_k(x)) \tag{9}$$

In order to solve this equation, it is assumed that the parameters P^0 and β are already estimated as described in Section 7.2. The problem is solved by using a variant of the Gauss-Newton algorithm, with a step size parameter to reduce the possibility of divergence due to large steps in the wrong direction during the iteration. The search for the minimum is made in the two-dimensional space which means that the estimate will be given by $\hat{x}^{NLS} = [\hat{x}_1^{NLS}\ \hat{x}_2^{NLS}]$. A three-dimensional estimate can also be made, but it turned out that it requires a good initial guess for the algorithm to converge. Therefore, it was not considered practical to use with our implementation in the project.

7.4 Likelihood function of RSS

A likelihood function can be constructed to estimate the position of the transmitter. This can be done when various measurements have been gathered as specified in section 7.1, and when the parameters P^0 and β has been determined as described in section 7.2.

The logarithmic power, see section 6.2, of the filtered recordings depicted in Figure 6b will be denoted y_{k_meas}. y_{k_meas} is calculated for every measurement k, and paired to the corresponding position of the hydrophone, p_k.

Furthermore, a prediction of the received power $\hat{y_k}$ can be calculated by an equation similar to (5),

$$\hat{y_k} = P^0 - \beta \log(\|x - p_k\|). \tag{10}$$

e_k in equation (5) is not present in equation (10). Still, noise will be present in y_{k_meas}. This yields that when the position of the emergency pinger, x, is the true one, the noise e_k can be expressed as,

$$e_k = y_{k\_meas} - \hat{y_k}. \tag{11}$$

Furthermore, as explained in section 7.2, e_k can be seen as an independent and identical distributed process.

After an investigation, the measurement noise in the filtered signal, i.e. before calculating the logarithmic power of the signal, appears to be Normal distributed. Although, considering how the measurements are obtained, other distributions for the noise may occur. The most convenient ones for this situation are a Log-normal distribution or a Skew Normal distribution shifted to the right with only positive numbers.

Then, with this distrubution, the mean μ and the variance σ are estimated. That is, we consider $e_k \sim dist(\mu, \sigma)$, where $k = 1, ..., K$ and $dist$ stands for the chosen distribution. Then, The likelihood as a function of x is given by

$$\mathcal{L}(x) = \prod_{k=1}^{K} PDF(e_k). \tag{12}$$

Here, PDF is the probability density function for the corresponding distribution. The maximum likelihood estimation is then

$$\hat{x} = \underset{x}{\operatorname{argmax}}\, \mathcal{L}(x). \tag{13}$$

The problem can be solved by evaluating the likelihood over a two-dimensional position grid for x. Then $\hat{x} = [\hat{x_1}\ \hat{x_2}]$ is the position which corresponds to the maximum value of the likelihood. The evaluation over the grid also makes it possible to provide a three-dimensional plot for the likelihood function with respect to x.

8 Motion planner

The motion planner is in control of how the UAS is supposed to fly in between the different submerging points, i.e., planning the route. It is also responsible for planning the motion when descending and ascending during hydrophone submersion. Since the environment is essentially free of obstacles, a straight line is calculated between points. This is done by PX4's built in mission mode.

9 Control algorithms

The purpose of the control algorithms is to make sure the UAS follows the trajectory given by the motion planner. It is also responsible for basic flying characteristics like; moving, starting, landing, keeping the UAS stable and also lowering the UAS, resulting in submerging the hydrophone in water.

The UAS is controlled by a mix of P and PID controllers with position and attitude set-points as inputs and thrust as output, see Figure 8. The position and angle are controlled by P controllers, while the velocity and angular rate are controlled by PID controllers. Other than controllers, the control loop consists of an acceleration to yaw and attitude converter, as well as a thrust mixer.

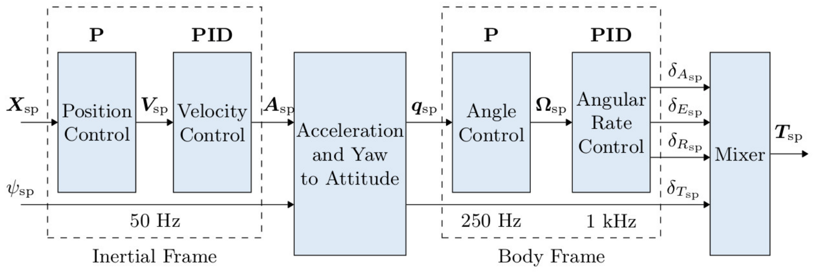

P
PID
P
PID
$\boldsymbol{X}_{\mathrm{sp}}$
Position Control
$\boldsymbol{V}_{\mathrm{sp}}$
Velocity Control
$\boldsymbol{A}_{\mathrm{sp}}$
Acceleration and Yaw to Attitude
$\boldsymbol{q}_{\mathrm{sp}}$
Angle Control
$\boldsymbol{\Omega}_{\mathrm{sp}}$
Angular Rate Control
$\delta_{A_{\mathrm{sp}}}$
$\delta_{E_{\mathrm{sp}}}$
$\delta_{R_{\mathrm{sp}}}$
$\delta_{T_{\mathrm{sp}}}$
Mixer
$\boldsymbol{T}_{\mathrm{sp}}$
ψ_{sp}
50 Hz
250 Hz
1 kHz
Inertial Frame
Body Frame

Figure 8: The control architecture of the UAS. Figure from [5].

9.1 Position and velocity control

The position controller receives the position setpoint and outputs a velocity setpoint, saturating the output on the way, see Figure 9. A PID controller with anti-windup for the position has been implemented in an attempt to improve the position reference following. The addition of the integral och differential part did, however, not yield a desirable behavior since most often a PID controller for positioning needs a higher order system and more aggressive acceleration to be able to properly follow the position. Thus, the I and D parts are set to zero.

The velocity PID controller, given the velocity setpoint, calculates an appropriate acceleration setpoint to output. This is then converted into attitude and thrust setpoints, which are sent to the angle controller and thrust mixer, respectively.

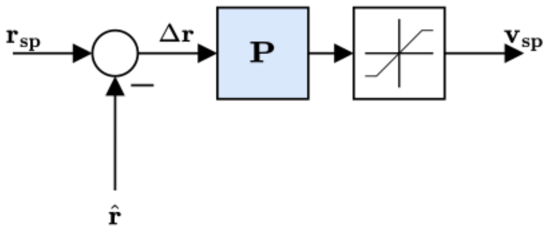


Figure 9: Diagram of the position control for the UAS. Figure from [5].

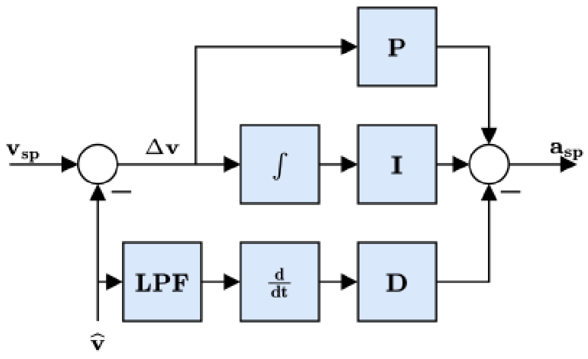


Figure 10: Diagram of the velocity control for the UAS. Figure from [5].

9.2 Angle and angular rate control

The angle controller receives an attitude setpoint in the form of a quaternion and outputs an angular velocity setpoint, saturating it on the way out as seen in Figure 11. This is sent to the angular rate controller which, using PID control, calculates an angular acceleration setpoint, see Figure 12.

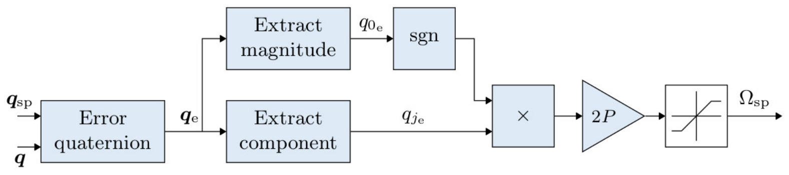


Figure 11: Diagram of the angle control for the UAS. Figure from [5].

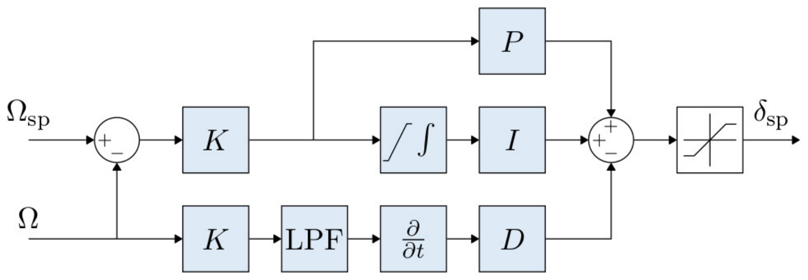


Figure 12: Diagram of the angular rate control for the UAS. Figure from [5].

9.3 Hydrophone oscillations

Since the implementation of a new PID controller for the UAS did not work, a new method for controlling the oscillations of the hydrophone and cable would be desirable. From the simulations with the implemented cable and hydrophone, it is clear that the current tuning of the UAS cannot properly fly with the hydrophone in the simulation. This is due to the cable oscillating, resulting in the UAS not being able to hold its position very well, or it oscillates so much it crashes. And since the implementation of a new PID controller, or tuning of the already existing PID/P controller only gives marginal improvements, limiting the speed and acceleration is the solution.

To find suitable limits, multiple simulation tests were done, where the oscillations of the hydrophone were plotted in both x- and y-directions but also the angle of the hydrophone in the simulation environment.
From these tests it was concluded that the maximum velocity of the UAS should be 2/3 m/s to minimize the oscillations of the hydrophone to less than 45°. Since the velocity is very low the acceleration does not have to be reduced and can keep its initial value of 3 m/s^2.

10 Future work

This sections details possible future work for this project.

10.1 Get hardware fully working

Right now, some of the hardware is not setup fully. So far, the UAS has been autonomously flown through QGroundControl instead of through ROS when collecting data. It is necessary to get everything working through ROS for more advanced control. The ROS2 to RPPS bride over UART seems to work, and we have been able to arm it through ROS, but the UAS immediately crashes on takeoff.

Also, the recorder node has yet to be tested on the Pi. So far it has only been tested on a laptop using its build in microphone.

10.2 More advanced sound simulation

It is possible to extend the sound simulation. Right now, it is done purely as a ROS node. One could look into integrating the sound simulation into Gazebo to get a more unified simulation environment. One could also look into making more advanced sound models using either static AR models to add distortions to the original sounds, or simulate sound reflections on surfaces in Gazebo. Another possibility is to simulate the layering in water and thus simulating different propagation speeds and bending of sound at different depths.

10.3 Adaptive measurement point selection algorithm

To make the UAS fully autonomous, it should select the measurement points based on data from previous measurements and estimates of the UUV position.

There are multiple ways to execute this. One way would be to choose the estimate of the UUV:s position as the next measurement point. To get an unambiguous estimate, three measurements are needed at different locations. Thus, the first three measurement points must be chosen more manually/semi-autonomously in order to get an initial estimate. These three points could be fairly close to each other in order to get a good sense of direction to the UUV. Another way is to choose the points far between, trying to enclose the UUV.

To determine which of these methods are most useful, one could decide depending on the range of the first measurements. If the first measurement indicates that the UUV is far away, it could be more time- & energy efficient to have the following two points closer to the first point. If the first range indicates that the UUV is close, it could be more valuable to enclose the UUV:s, thus having the measurement points further apart.

10.4 More advanced signal processing algorithms for stationary submersion

This section deals with alternative strategies for estimating the position of the UUV when the UAS is submerging the hydrophone at a stationary point.

10.4.1 Doppler effect - stationary submersion

The Doppler effect takes advantage of how the frequency is perceived while moving. Basically, the frequency of a signal will be observed differently if the observer is moving relative to the transmitter. The relationship between transmitted frequency f, and received f' is

$$f' = f\frac{v \pm v_R}{v \pm v_T}. \tag{14}$$

In equation (14), v is the velocity of the signal, v_R is the speed of the receiver relative to stationary and v_T is the same for the transmitter. The velocity v of the signal is the speed of sound in the present medium, in this case water. Since the UUV is the transmitter of the emergency signal here, it can be assumed that $v_T = 0$.

In a scenario when the UAS is submerging the hydrophone with constant velocity vertically, the Doppler effect may be used. For example, consider two different positions for the hydrophone, p_1 and p_2. Further, assume that the UUV is located stationary at position p_0 and that $||p_1 - p_0||_2 < ||p_2 - p_0||_2$. Then, when the UAS is submerging the hydrophone vertically with constant velocity, the relative velocity between p_1 and p_0 will be greater than between p_2 and p_0. Therefore, according to equation (14), the hydrophone will measure a higher frequency for the case when the hydrophone initially was located at p_1 compared to p_2. In this manner, one can tell which submerging point that was closest to the UUV.

10.4.2 Received time difference - stationary submersion

Figure 13 displays a simplified version of how soundwaves propagate through a medium, in this case water. The emitted signal from the emergency pinger is a soundwave; therefore, it will propagate in the shape of a sphere. Evidently, the further away from the emitter, the greater time the signal has been traveling. An illustration of this can be seen in Figure 14. The distance U → A is longer than U → C and U → B.

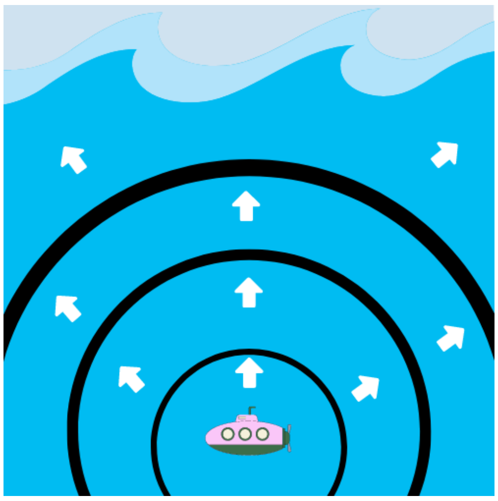


Figure 13: The propagation of soundwaves in water

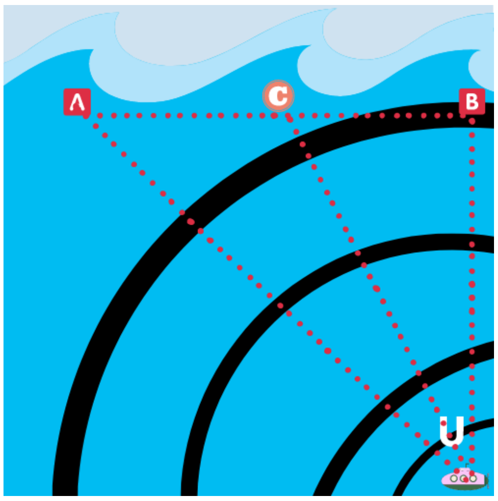


Figure 14: The propagation of soundwaves in water with distance

With the measurements A, B and C in Figure 14, a phase shift of the static pulses emitted from the transmitter will arise. This is due to the increase of length traveled for the signal.

If a positive phase shift occurs between two measurements, then the second measurement is further away from the origin. And the opposite if the phase shift is negative. An example of a phase shift between two different measurements can be seen in Figure 15.

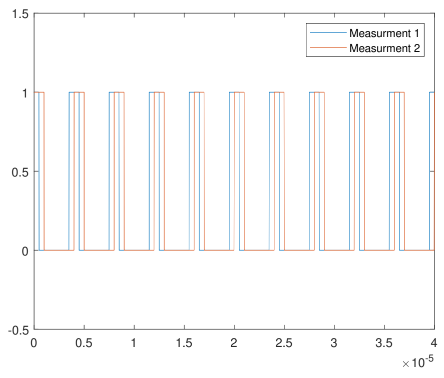


Figure 15: Example of a phase shift of two measurements

To be able to measure a change of phase for the emitted signal, a couple of things are needed to be fulfilled. The emergency pinger must emit its signal with a consistent and accurate rate of pulses. The receiving computer, a Raspberry Pi in our case, will need to keep track of its internal clock with a sufficient accuracy.

10.4.3 Received signal strength with triangulation - stationary submersion

A distance with equation (5) can be calculated between the hydrophone and the emergency pinger, that is we consider $\|x - p_k\| = d_k$ as the parameter we want to estimate. Then, if several distances are estimated from the measurements, a triangulation can be configured to give an estimation of the UUV's location. Figure 16 depicts how a triangulation can be made when the hydrophone has received the signal from the emergency pinger at position A,B and C. The red dot represents the UUV.

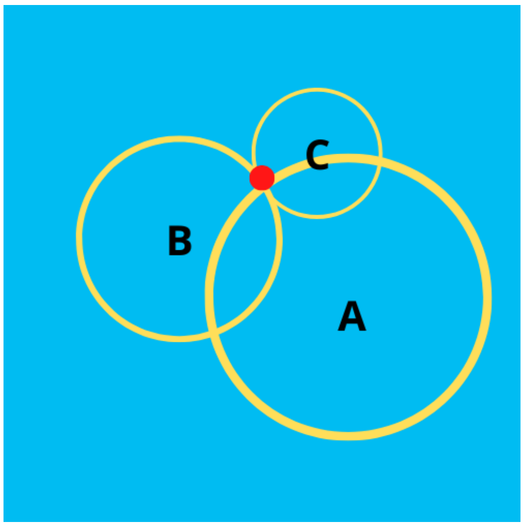


Figure 16: Triangulation of three measurements

10.5 More advanced signal processing algorithms for mobile submersion

The strategies outlined below require the UAS to be able to fly horizontally with the hydrophone tailing submerged. An explanatory picture of the UAS with the tailing hydrophone is seen in Figure 17.

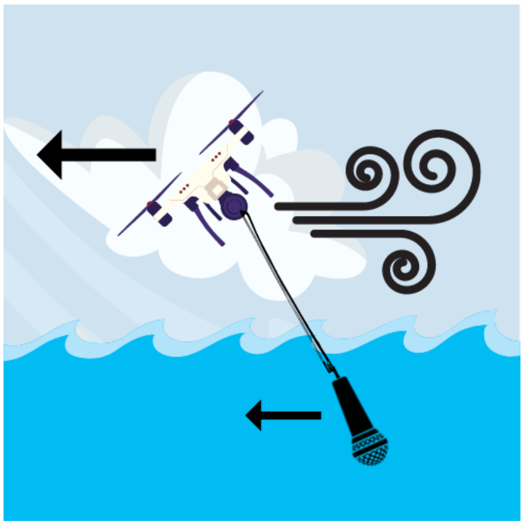


Figure 17: A sketch of the UAS flying with the hydrophone submerged

10.5.1 Doppler effect - mobile submersion

When the UAS fly horizontally with the hydrophone towed in the water, the Doppler effect may be used in a different, but similar approach compared to what was explained in section 10.4.1. Assume that the hydrophone is moving in some direction with a velocity v. Then, if the hydrophone is moving towards the UUV which transmits the emergency signal, the hydrophone will measure a higher frequency than the emitted signal according to equation (14), since v_r is positive in this case. In contrast, the hydrophone will measure a lower frequency if it is moving away from the UUV. Therefore, this strategy enables the UAS to calculate in which direction the UUV is located.

10.5.2 Received signal strength - mobile submersion

When the UAS has the hydrophone tailing submerged, the received signal strength explained in equation (5) can be used. The difference of the technique explained in section 7.1, is that the UAS can determine if it is moving towards the UUV or not depending on how the strength of the signal changes.

10.6 Analog filters / look at another sound card

Some limitations with the sound card were found. While the sound card seems to be good in terms of resolution and sampling rate, there are some issues with it. Firstly, it is really heavy and is designed to be used for filmmaking and podcasts, not to sit on a drone. A much lighter one would be preferred. It can record up to 192 kHz sampling rate, but only to its internal SD card. To use it over USB, the maximum is 96 kHz which is barely twice the frequency of the pinger which emits a frequency of roughly 45 kHz. A higher streamable sampling rate would be preferred. Another possibility is recording the SD card and then using some serious Bluetooth hackery to reverse engineer how the android app can be used to control the sound card. Then you could control the sound card over Bluetooth and make it into a hard drive so that the files can be transferred to the Pi. This is a terrible solution, but then it is possible to autonomously record at 192 kHz with the current sound card.

Another possibility is to modify the filtering done internally on the sound card. It has an inbuilt adaptive anti-alias filter. This is normally good, but aliasing could in this case be useful as a way of getting away from the sampling rate problem and processing the signal with the knowledge that it is aliased. To do this, some analog filtering is probably needed to remove the noise before sampling the raw signal.

10.7 Better Control for UAS

Since the maximum velocity of the UAS is fixed to a fairly low value of 2-3 m/s future work could be to analyze if it could be possible to control the oscillations of the hydrophone at accelerations and braking. If that were to be possible, one could increase the speed of the UAS and by that increase the amounts of measurements or increase the search area with risking depleting the battery.

One possible way could be to implement an IMU at the hydrophone and measure the angle and acceleration, and then calculate how the UAS should fly to cancel the oscillations. Another way could be to model the dynamics of the hydrophone cable to estimate the position of it. There are state-space models of quadcopters online which possibly could be combined with a hydrophone model and controlled with a state-feedback controller or an LQ controller.

References

[1] As-1 hydrophone. URL https://www.aquarianaudio.com/as-1-hydrophone.html.

[2] Qgroundcontrol user guide. URL https://docs.qgroundcontrol.com/master/en/index.html. Accessed: 22-12-02.

[3] Radiomaster radio comparison chart. URL https://www.radiomasterrc.com/blogs/radiomaster-press/radiomaster-radio-comparison-chart. Accessed: 22-10-03.

[4] X500 v2 full kit. URL http://www.holybro.com/product/x500-v2-kit/. Accessed: 22-10-05.

[5] Px4 user guide - controller diagrams, . URL https://docs.px4.io/main/en/flight_stack/controller_diagrams.html.

[6] Px4 flight modes overview, . URL https://docs.px4.io/main/en/getting_started/flight_modes.html. Accessed: 22-12-02.

[7] F. Gustafsson. *Statistical Sensor fusion*. Studentlitteratur, 2018.

[8] Transportstyrelsen. Drönare, 2022. URL https://transportstyrelsen.se/sv/luftfart/Luftfartyg-och-luftvardighet/dronare/. Accessed: 22-10-05.

[9] *NEO-M8 u-blox M8 concurrent GNSS modules Data sheet*. u-blox, 9 2021. R11.