Julie Støverud Skjøy

Smart Fish Crowding Sensor

Master's thesis in Cybernetics and Robotics
Supervisor: Jo Arve Alfredsen
July 2021

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

Abstract

The aquaculture industry suffers from substantial losses in correlation with crowding processes. The process gathers a part of the salmon biomass in a fish farm cage with a sweep net to streamline pumping out fish for slaughtering, delousing, or sorting. Crowding is associated with significant risks concerning fish welfare, mortality, and product quality. The consequent risks substantiate the demand for a smart sensor observation system that monitors the fish behavior and process equipment during a crowding process.

This report describes the prototype of a sensor fusion system. It relies on a Raspberry Pi as a microprocessor and five sensor units. An Xsens motion tracker for tracking orientation and rotation of the system, a MicroElektronika GNSS tracker for tracking the position, a LAUMAS load cell for measuring tension in the rope of a sweep net, an Aanderaa oxygen optode for measuring environmental parameters in water, and a Raspberry Pi camera for visual observation. The prototype software consists of Python drivers that extract measured data from all the sensors simultaneously to capture external influence on the system and store the data persistently for retrieval after testing the prototype.

Test of the prototype in an environment resembling the one in a crowding process showed measurement readings indicating a difference between noise and simulated fish movements. In addition, the position of the system, the tension in the operational equipment (the top rope of a sweep net), and changes in water environment variables were prominent and accurate. The measurement responses demonstrate the possibilities for further development of the prototype. A good base is offered for the implementation of a smart sensor crowding system during a crowding process in the future.

Sammendrag

Havbruksnæringen lider av betydelige tap av Atlantisk laks i forbindelse med trengeprosesser. Prosessen går ut på å samle deler av den totale biomassen i en fiskeoppdrettsmerd for å effektivisere utpumping av fisk for slakting, avlusing eller sortering. Trenging er forbundet med stor risiko knyttet til fiskevelferd, dødelighet og produktkvalitet. Den påfølgende risikoen underbygger etterspørselen fra næringen etter et smart sensor observasjonssystem som overvåker laksens adferd og prosessutstyr under en trengeprosess.

Denne rapporten beskriver prototypen til et sensorfusjonssystem. Den er avhengig av en Raspberry Pi som mikroprosessor og fem sensorenheter. En Xsens bevegelsessensor for sporing av orientering og rotasjon av systemet, en MikroElektronika GNSS sensor for sporing av systemets posisjon, en LAUMAS lastcelle for måling av spenningen i tauet til en orkastnot, en Aanderaa oksygenoptode for å måle miljøparametere i vann og et Raspberry Pi kamera for visuell observasjon og validering. Prototypens programvare består av Python drivere som uthenter data fra alle sensorene samtidig for å oppdage ekstern påvirkning av systemet og lagre dataen permanent for uthenting etter test av prototypen.

Test av prototypen i et miljø som samsvarer det under en trengeprosess indikerer tydelige forskjeller mellom støy og simulert bevegelse fra fisk. I tillegg var systemets posisjon, strekk i prosessutstyr (tau til orkastnot) og vannkvalitet målt med fremtredende og nøyaktige endringer. Målt respons fra sensorene viser muligheter for videre utvikling av prototypen. Det er lagt et godt fundament for fremtidig implementering av et smart sensor trengesystem under en trengeprosess.

Acknowledgements

I want to express my gratitude to my supervisor, Jo Arve Alfredsen, who gave me the opportunity to work with an engaging project of high interest for the aquaculture industry and project partners in Crowdguard. I would like to thank Crowdguard project leader Birger Venås (SINTEF), who has provided me with great insight into the crowding process and connections within the Crowdguard project. Also, Ragnhild Bekk (Nærøysund Aquaservice AS) for being helpful towards a potential test at a crowding facility. I wish to acknowledge the help provided by Asle Tomren (Polyform AS) for developing the floatation devices for the prototype.

The assistance provided by Nikolai Luvrås, Jan Leistad, and Åsmund Stavdahl has been appreciated during the process of hardware acquirement and software development.

I am particularly grateful for the assistance given by Terje Haugen and the employees at the Department of Engineering Cybernetics' mechanical workshop for always being available and wanting to help me with the mechanical integration of the prototype.

Finally, I would like to thank my family and friends for their support throughout the thesis.

Trondheim, July 30, 2021

Julie Støverud Skjøy

Contents

List of Figures

List of Tables

Listings

Acronyms

ADC Analog to Digital Converter. iii, 24, 26, 34, 42, 43, 55, 68

CAN Controller Area Network. 16

CSI Camera Serial Interface. 33–35, 44

CSV Comma-separated values. 31

DCE Data Communication Equipment. 16

DOF Degree Of Freedom. 12

DTE Data Terminal Equipment. 16

GNSS Global Navigation Satellite System. 9, 22

GPIO General-purpose input/output. 16, 33, 43, 45

GUI Graphical User Interface. 29, 30

I2C Inter-Integrated Circuit. 16

IDE Integrated Development Environment. 30

IMU Inertial Measurement Unit. 22

MEMS Micro-Electro-Mechanical Systems. 22

NOOBS New Out Of Box Software. 29

NTP Network Time Protocol. 10

OOP Object-Oriented Programming. 31

OS Operating System. 29, 33

PD Power Delivery. 27, 35

PWM Pulse Width Modulation. 17, 26, 34, 45

RDBMS Relational Database Management System. 31

RPM Revolutions Per Minute. 80

SPS Samples Per Second. 43

SQL Structural Query Language. 31

UART Universal Asynchronous Receiver/Transmitter. 15

USB Universal Serial Bus. 16

UTC Coordinated Universal Time. 10, 23, 49

WFE Whole fish equivalent. 1

Glossary

Arduino An open-source electronics platform based on easily usable hardware and software. An Arduino boards executes instructions sent to it, using the Arduino porgamming language and the Arduino Software (IDE), based on processing. 55

Full Duplex Communication between two devices is allowed simultaneously. 39

MikroBUS The standard defines mainboard sockets and add-on boards used for interfacing microcontrollers and microprocessors with integrated circuits and modules. It specifies the physical layout of pins, the communication and power supply used, the size and shape of the add-on boards, ant the positioning of the bus socket on the mainboard. 39

Raspberry Pi OS A free operating system based on Debian, optimized for the Raspberry Pi hardware. 32, 62

Sensor Fusion A concept of bringing together inputs from multiple sensors to produce more reliable information with less uncertainty. 22

Thurlby Thandar Power Supply Variable series regulated DC power supply designed for operation in both constant current and constant voltage modes. 27, 41

Chapter 1

Introduction

The thesis background and motivation are initially presented to establish the relevance of the thesis and the problem it pursuit to cover.

After that, an introduction to fish cages, crowding processes, and fish welfare indicators. The importance of welfare indicators for fish and further development in the aquaculture industry are also discussed.

Given that this thesis is a continuation of the preceding Specialization project "Smart Fish Crowding Sensor" [1], the previously done work and the basis for the rest of this report are presented. Then follows the scope of the thesis, with the framework and limitations of the work.

Lastly is an outline of the thesis chapters and a brief description of their content.

1.1 Background and motivation

The continuous increase in the world's population pressurizes the food-production industry to satisfy a growing food demand. According to DNV GL's Energy Transition Outlook 2020, the world's population will reach 9.4 billion people by 2050, an increase of 1.6 billion people from today's population [2]. The aquaculture industry, which was the most rapidly growing food-producing sector involving animal species globally in 2018, can satisfy some of the demand [3]. In Norway, commercial Atlantic salmon farming began around 1970, which laid the foundation of the modern fish farming industry we know today [4]. The total yearly stock of salmon was about 160,000 tons in 1994 and has quintupled to approximately 800,000 tons in 2019 [5]. In 2015, Norway contributed with over 50% of the world's total Atlantic salmon production, which substantiates the position Norway has in the world aquaculture market [6]. The ratio between exported Atlantic salmon (118.300 tonnes (Whole fish equivalent (WFE))) and total sales (121.900 tonnes (WFE) including domestic consumption), is 97% export of the Atlantic salmon produced in Norway in October of 2018 [7]. Preserving these resources by improving existing and develop new methods to expand the industry even further in the future is of high interest.

The salmon fish farming industry is experiencing a need for improving existing methods because of substantial losses linked to critical operations in production processes. In 2019, over 61 million individual salmon were lost during production, which equals 14.5% of

the total salmon stock [5]. While there is uncertainty regarding the exact amount of individual fish that are lost due to certain events and operations, it is still crucial to reduce the number of losses related to critical operations. One of the most critical operations performed at fish farms are crowding processes, a frequently occurring operation in the salmon farming industry. The operation is associated with significant risks concerning fish welfare, mortality, and product quality. However, it is at the same time an essential and unavoidable operation inherent to many farm practices such as sorting, delousing, and harvesting. The critical operations must be improved on operational methods. Therefore, efficient monitoring and control tools that hold the crowding process within acceptable limits regarding fish welfare are in great demand.

Today, fish behavior observations and monitoring are done manually by fish farm workers. It consist of visual observation of fish density and surface activity. When unwanted activity is observed, measures are made to reduce the fatal outcome that can arise from a crowding process. However, there are challenges that arise from humans observing an ecosystem below the water surface. Most of the fish's environment is below the water surface and only a percentage of fish behaviour is possible to observe for a fish farm worker from above the surface. It result in unrepresentative assessment of fish behaviour and welfare. Given that good fish welfare contributes to less loss of fish, the industry wants reliable and qualitative observations from sensors in addition to the existing method to improve the fish welfare conditions during crowding processes.

Few alternatives for monitoring fish crowding processes have been researched or implemented. However two methods are worth mentioning. Firstly, an analytical sampling of environmental variables during a crowding process showed that the water quality was relatively acceptable with great access to oxygen and constant values in water temperature and salinity. Visual observations of fish behavior indicated high density throughout the process, although mostly without surface burst activity or excessive white (fast) muscle swimming. However, underwater cameras indicated contact between fish or between fish and net when density was high. Samples of blood chemistry and white muscle biochemistry portrayed considerably elevated ventilation rates and cortisol levels. The study illustrated that visual observation of a crowding process is unrepresentative for fish welfare conditions. The study presented no straightforward method of monitoring fish behavior during crowding processes, but that new technology might not be the answer. Therefore, further monitoring has to be done with caution [8]. On the other hand, the use of acoustic telemetry for monitoring fish during critical operations like fish crowding provided new knowledge about how the behavior of Atlantic salmon may be affected by critical processes [9]. In the study, data collected automatically from acoustic transmitter tags placed in several fish indicated increased swimming patterns, revealing that the crowding process is stressful for the fish. It also portrays the possibility of supplying a crowding process with new technology for fish welfare observing properties.

Because of the limited available alternatives for monitoring fish crowding processes, development and new solutions are in high demand. The solution presented in this thesis is a sub-project of the SINTEF project, "Crowdguard", that seeks to increase the breeder's control during crowding processes. By developing and validating new and unique technology for data capture during crowding of salmon.

In addition to being based the preceding Specialization project, this thesis has great interest in satisfying the requirements that "Crowdguard" seeks and aims to answer if it is possible to capture fish behaviour and process changes with a system composed of a set of sensors,

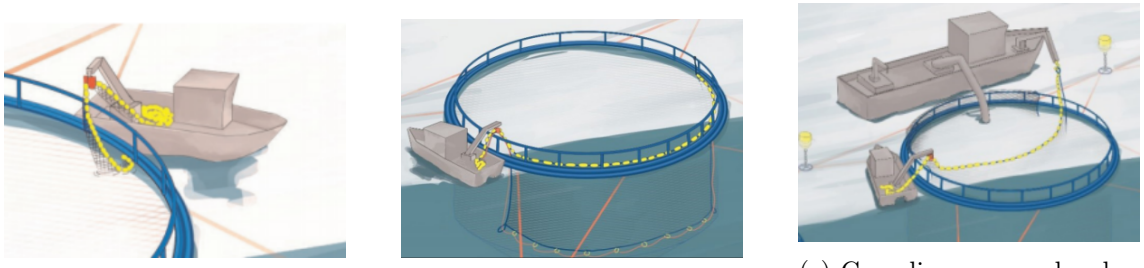
attached to a sweep net during a crowding process?

1.1.1 Fish cages and crowding processes

Fish cages are underwater net cages containing up to 200,000 individual fish [10]. It equals approximately 1,000 tons of biomass with an average fish weight of 4.5kg. The size of the offshore fish cages that is relevant for this thesis is 50 m in diameter and 25 m deep, resulting in a net volume of approximately 49,000 m^3.

Net cages consist of four principal components; net structure, cage collar, ropes/chains, and sinker tube. They contribute to keeping fish inside a designated area for containment for feeding and care of farmed fish. When fish needs to be slaughtered, deloused, or sorted, they are extracted from the fish cage with a fish pump and transferred to another cage or confinement. On its own, the pump cannot reach the entire cage, and extraction of fish would not be possible within a reasonable time frame. Therefore, the concept of fish crowding is introduced as an essential operation to reduce the time it takes to pump out fish from the fish cages. A separate net, called a sweep net, is lowered into the cage which catches an unknown amount of fish. The aim is to gather fish from the fish cage to a smaller area to increase efficiency when pumping out fish from the cages.

The sweep net is winched out from a service boat and lowered inside the fish cage (see Figure 1.1a). The net is kept afloat by floaters (yellow circles in Figure 1.1) and weighted down by weights. The sweep net is then dispersed over the entire side of the fish cage (see Figure 1.1b) before the bottom is dragged up towards the operational boat (the larger boat) in Figure 1.1c. The closed-off area with an unknown amount of fish is now separated from the rest of the fish cage, and a suction pump is lowered approximately 1 m below the water surface and starts pump out fish (see Figure 1.1c). Then, the net is winched in to compress the free-swimming area around the pump, which decreases the time consumption of the process. As fish are removed from the closed-off area, the net is gradually compressed even further to minimize time consumption.



(a) Sweep net lowered

(b) Sweep net dispersed

(c) Crowding process has begun

Figure 1.1: Preparation of sweep net before crowding process (Source: Egersund Net AS)

As less free swimming space for fish and the pump result in faster pumping than pumping without crowding, it is preferred. Considering that welfare conditions of fish decrease over extended periods while crowding and the economic value for aquaculture industries, a shorter crowding process entails. Although shorter crowding processes are preferred, it is also important not to excessively or hurriedly crowd fish to comply with recommended fish welfare indicators (described further in Section 1.1.2).

To this day, surveillance of crowding processes is experience-based and depends entirely on how the fishermen experience and interpret the process and fish behavior. Consequently,

the type and degree of measures are non-trivial and vary between each crowding process and fish farm worker. Typical measures are oxygen-supplement in the crowded fish area to increase oxygen levels in the populated fish area. In addition to supplying oxygen, loosening the fish sweep net to increase free swimming space for fish is efficient for increasing fish welfare. The net's tension is regulated with a winch that workers operate at the fish farm. The issue with the winches is the unknown tension applied on the net's top rope, which is only controlled by the operator's intention. It results in inconsistent operations, which is unwanted, and the need for a new solution that increases consistency is essential.

Given few to no other surveillance methods than those described above, and most notably, the substantial loss of fish during crowding processes, equipping the crowding processes with a sensor system to obtain data about fish welfare status is of great demand. Consequently, it also increases the consistency of crowding processes and the overall fish welfare.

1.1.2 Fish welfare indicators

Discussions regarding fish welfare in aquaculture have grown considerably over the last few years. Both because of the continuous increase in fish production volume each year and society awareness regarding sustainable and safe food production. Scientists debate whether fish can feel pain and emotions, which is an interesting discussion that affects further development in technology implemented in the aquaculture industry.

Creating a scenario of fish being emotion- and painless, the industry steers towards being efficient but having the main principle of saving money without respecting the wastage of resources. The scenario would be damaging to the industry's reputation and integrity.

Because both fish welfare and resource wastage are of high interest to the "Crowdguard" project partners, the assumption that fish can feel pain and emotions is utilized when developing a sensor system solution in this thesis. The premise is advantageous both concerning caged fish welfare as well as the overall economic and environmental effects [11].

According to the handbook on welfare indicators for Atlantic salmon in fish farms, Fishwell, developed by the Norwegian food research institute Nofima, there are four sections of welfare indicators contribute to the overall welfare status of salmon. These are; available resources, a suitable water environment, good health, and freedom to express behavior. For normal operations in aquaculture facilities, all sections are necessary to observe. However, for the critical operation of crowding fish, the water environment and freedom to express behavior can be prioritized. This is because neither the available resources nor good health can be regulated during such a process [12].

The water environment has four subcategories; respiration, osmotic balance, thermal regulation, and good water quality. Respiration is the uptake of oxygen and the release of carbon dioxide, which is essential for aerobic metabolism and maintaining pH in the fish's body. When oxygen saturation is below the required aerobic metabolism level, called hypoxia, fish switch to anaerobic glycolysis. Eventually, substances available for glycolysis deplete and cause a build-up of anaerobic by-products, leading to death. Hypoxia can also cause stress response in salmonids. Therefore, sufficient dissolved oxygen in water is a crucial welfare condition for salmon. Osmotic balance is access to water with adaptable salinity and pH levels. Seawater fish cages cannot increase or decrease the salinity or pH levels because of the open fjords and sea location. However, the open water environment will change in quickly adaptable increments. The same applies to thermal regulation, and

water quality.

The behavior of fish is grouped into five categories; behavioral control, social contact, rest, exploration, and sexual behavior. Fish must have the freedom to control their body movements to move away from danger and have buoyancy control. In fish farming, fish panic is seen when fish are crowded and handled. Characteristics of potential panic are avoidance behavior, increased oxygen consumption, adrenaline hormones, cortisol, and serotonin levels, all indicating stress and possible fear. As crowding processes are known to be stressful and harmful for fish, a delicate process in terms of duration and strength is vital to observe and regulate. Given that most farmed fish species live in groups, social contact is related to safety. Fish can seek protection among equals, and information is shared about food and dangers. During a crowding process, fish may have excessive social contact, but for a limited period. The category can be interesting to observe visually. Rest and restitution are not applicable during a crowding process but instead in between these processes. Hence, crowding should not be performed too frequently or too vigorously. Both exploration and sexual behavior relates to fish in free waters and rivers and are not relevant during crowding processes in fish cages.

1.1.3 Previous work

The work presented in this report builds upon the preceding Specialization project, "Smart Fish Crowding Sensor" [1]. The project covered a discussion of different measurement principles and the necessary system requirements to develop a fish crowding sensor system with the chosen prinsiples.

Six measurement principles would form the framework of the system. Sensors based on the chosen principles were acquired and intended to be interfaced with a system microprocessor and tested in the preceding project. Table 1.1 presents the acquired sensors, including type, model, and the information it provides.

Sensor	**Type and model**	**Provided information**
Motion sensor	Xsens MTi-670	Orientation of system Rotation of system Position of system
Load cell	LAUMAS FUN	Force in top line of net
Oxygen optode	Aanderaa 4531D	Environment variables in water
Camera	RPi HQ Camera	Visual view of process under water

Table 1.1: Sensor overview

The motion sensor was interfaced with the system microprocessor (a Raspberry Pi) and conceptually tested. The test included data observations of responses from diversion in orientation, rotation, and position. The results acquired gave a brief introduction to how the sensor responded to the different movements. Based on these, it was reasonable to conclude that the sensor would be sufficient for measuring deviations in motion and position during a test in a real environment during a crowding process. This was because of the rapid response and logging from the sensor when tests were conducted. The remaining three sensors were not interfaced due to time constraints and no further results were acquired.

Four system requirements were discussed in the preceding project; positioning, size, data saving and retrieval, and the operational life. Given that this thesis is a part of the SINTEF project "Crowdguard", resources that provide insight about a crowding process

were made available. It includes videos of an entire crowding process which illustrates the general practice of how the sweep net is handled. Observations of the videos portrayed a placement that followed the crowding process dynamics at the top line of the sweep net. It allows for direct contact with the water, a very exposed area in terms of both fish and water movements (orange oval in Figure 1.2).

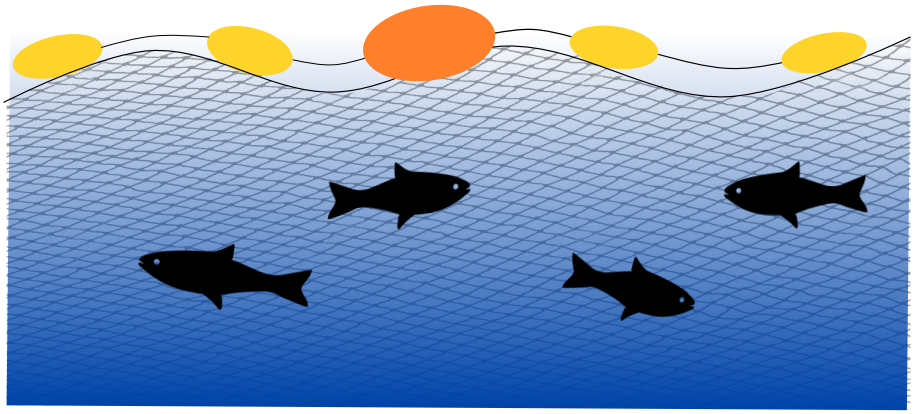


Figure 1.2: System position on sweep net in water (floating orange oval)

The positioning presents limitations of system size. It had to be compact and light enough to follow the dynamics of the crowding process. It was neither desirable for the industry to have a sizeable additional system mounted to the existing equipment. A custom-made flotation device from Polyform AS holding all necessary equipment to float the system was planned to be used. Most of the electrical components were not water-resistant and required to be placed in a watertight enclosure which the flotation device would hold. The load cell and the oxygen optode were IP-certified to be water-resistant and did not require to be placed in the enclosure.

The desired area of use of the system is a real-time data collector that transfers the data to a remote operating station. An operator can surveillance the data and implement risk-reducing measures based on the data presented. A step towards a real-time collector is to store obtained data locally on the system computer for data retrieval after being measured during an entire crowding process. Finally, the data analysis should conclude whether the system can be used to further be developed to a real-time system.

The system must log sensor data for an entire crowding process. Therefore, the operational life of the system that lasts an entire process is a requirement. Since the system will be positioned in the middle of a fish net, a cord with a power supply increase the risk of complications and reduce safety. A powerbank serves as the power supply, resulting in wireless supply to the system.

Although the system requirements were established during the preceding Specialization project, they were not implemented in any way due to time constraints.

1.2 Scope of work

As mentioned in the previous section, most prototype developments remained after the preceding Specialization project. However, the work established the foundation of this thesis.

This thesis include integration of sensors, development of software and the physical prototype including mechanical integration, and testing the prototype. The mentioned areas of development are described in detail in this thesis.

Testing of the prototype consists of two parts. First, a land-based system verification test that verifies the functionality of the prototype. It will ensure that the system works as intended and stores logged data from each sensor in a persistent way. The main goal of the thesis was to test the system throughout a crowding process, but due to seasonal and capacity difficulties towards the end of the thesis, a full scale test was not possible. However, a fisrt system verification test on water was conducted to verify logging or reasonable data in an environment similar to a crowding process. The fabricated test is conducted in Trondheimsfjorden outside of Børsa.

1.3 Outline of report

Chapter 2 Theory present theory regarding topics that is used further in the thesis and provide a fundamental theoretical understanding in order to follow the concepts and basis for decision in Chapter 5 as well as analyse of results.

In **Chapter 3 System Requirements** the functional, non-functional requirements of the system is listed to inform about the necessary implementation requirements. How they are met is presented in Chapter 5.

Chapter 4 System components and sensors describe each sensors measurement principle and purpose. The purpose and relevance of the other system components are also presented.

Chapter 5 System Design and Implementation gives an in depth description of the design and implementation of both the hardware and software of the system. In addition to how the initial setup of the system is done, the available storage of the device, and how the physical prototype is designed and implemented.

In **Chapter 6 Testing of prototype** the method for testing the prototype both on desk (land-based) and during a crowding process is described and presented with results aquired.

Chapter 7 Discussion and conclusion included a discussion of to which extent the thesis has met the criteria from the project description and a general conclusion.

Chapter 2

Theory

2.1 Positioning and time

2.1.1 Geographic coordinate system

The Geographic coordinate system is a subcategory of the Terrestrial coordinate system that is earth fixed and rotates with the earth. It is used to define the coordinates of points on the earth's surface, with an origin near the earth's center. The primary pole, which defines the axis of symmetry of the coordinate system, is aligned to the earth's axis of rotation. The primary plane is perpendicular to the primary pole, meaning the equatorial plane. The intersection between the equatorial plane and the plane containing the Greenwich meridian is the primary axis. The Geographic coordinate system consists of parallel East to West lines of latitude and North to South lines of longitude. Because of this, it is more commonly known as the "Latitude and Longitude" coordinate system [13].

The latitude (θ) of a point P on earth is the angle between the primary pole and the equatorial plane. Latitude is zero if P is at the equatorial plane, and it increases towards the North Pole (θ = +90°) and decreases towards the South Pole (θ = -90°). Longitude (λ) is the angle between the Greenwich meridian and the meridian ellipse containing point P. The equatorial plane from the Greenwich meridian measures λ = 0, and eastwards through λ = +180°or westwards through λ = -180°. The coordinates can have different formats, but the most accurate holds the numeric values for degrees, minutes, and seconds. Plus, the specified direction N, S, E, or W, e.g., 51°28'40.5408"N, 0°0'5.5620"W. Positional representation of this kind is well suited for small changes in the position of a device [14].

A GPS receiver receives signals simultaneously from all the visible satellites and calculates the correct position. There are six different types of Global Navigation Satellite System (GNSS) Constellations that a GPS receiver can obtain coordinates from, which are produced in different parts of the world. These are GPS (US), QZCC (Japan), BeiDou (China), Galileo (EU), GLONASS (Russia), and IRINSS (India). The main reason for all satellite constellations is availability and redundancy. If one system fails, another constellation can take over. Satellites belonging to the different constellations have specific numbers of satellites and operational frequencies [15].

2.1.2 UTC

Coordinated Universal Time (UTC) is a standard for timekeeping that defines the world's time zones. It is based on the combined output of atomic clocks which are highly stable. In addition, leap seconds are added or subtracted, if necessary, at two opportunities every year to adjust UTC for irregularities in Earth's rotation [16].

UTC on devices is often obtained automatically with a Network Time Protocol (NTP). The synchronization of clock time is enabled by a network request sent to a service that synchronizes time for devices. For devices that do not have an available network connection, UTC time can be obtained from satellites via a GNSS receiver [17].

In addition to UTC time, UTC date is automatically obtained with either of the two methods described above. The date is determined according to the Gregorian calendar.

2.2 Orientation and rotation

The orientation and rotation of an object describe how an object is placed in the space it occupies. An object can be composed of multiple parts that are connected either by a stiff or non-stiff connection. Two stiffly connected parts will directly reflect each other's movements because they represent one object. Opposite, two non-stiff parts represent two separate objects (attached by, e.g., a rope) that can move differently on what influences the object's orientation and rotation. There are multiple ways to describe orientation and rotation, below is three commonly used representations.

2.2.1 Rotation matrix

The rotation matrix describes the rotation of an object in 3D space with a fixed coordinate system. The basis of a rigid body is three orthogonal unit vectors fixed to the body. Specifying the coordinates of a vector of this basis in its current position will describe the rotation in terms of the reference coordinate axes. The three unit vectors can be represented as:

$$\mathbf{R} = \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} \end{bmatrix} \in \mathbb{SO}(3) \tag{2.1}$$

Each column is a 3D unit vector described in a coordinate frame with orthogonal vectors. The initial frame acts as a reference when an object changes its orientation to a new frame. Both frames are in the Cartesian coordinate frame where the vectors are directed along the x-, y-, and z-axis, respectively. $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are in the set of real values in three dimensions. The norm of each vector is one because they are unit vectors. All the columns are orthogonal, meaning that they are orthogonal to each other, magnitude is one, and they are at 90° to each other. The rotation matrix follows the right-hand rule, meaning that the rotation around each axis can be defined. A rotation can be constructed from rotations about the primary axes, given by the matrix:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.2}$$

Each column (r_{1j}, r_{2j}, and r_{3j}) represents the direction of the unit vectors **x**, **y**, and **z**. The nine numbers represent the coordinates of the object's orientation, which relative to the reference frame defines how the system has moved in space.

The matrix elements of the rotation matrix r_{ij} specify the i-j-element within the matrix. It is given by the dot product between the i-th basis vector after rotation and the j-th basis vector before the rotation. Assuming a passive rotation (rotation of coordinate frame axes) that is orthogonal and normalized gives:

$$\mathbf{R}_{ij} = \hat{e}'_i \cdot \hat{e}_j \tag{2.3}$$

A counter-clockwise rotation in a right-handed system about the z-axis (Figure 2.1) result in new basis vectors that are related to the old basis vectors via sine and cosine functions:

$$\hat{e}'_1 = cos\varphi\hat{e}_1 + sin\varphi\hat{e}_2 \tag{2.4a}$$
$$\hat{e}'_2 = -sin\varphi\hat{e}_1 + cos\varphi\hat{e}_2 \tag{2.4b}$$
$$\hat{e}'_3 = \hat{e}_3 \tag{2.4c}$$

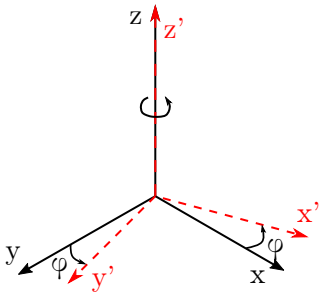


Figure 2.1: Rotation about the z-axis in a 3D Cartesian coordinate system

Matrix elements are calculated according to eq. (2.3) and result in the following equation that represents a rotation about the z-axis:

$$\mathbf{R} = \begin{bmatrix} \hat{e}'_1 \cdot \hat{e}_1 & \hat{e}'_1 \cdot \hat{e}_2 & \hat{e}'_1 \cdot \hat{e}_3 \\ \hat{e}'_2 \cdot \hat{e}_1 & \hat{e}'_2 \cdot \hat{e}_2 & \hat{e}'_2 \cdot \hat{e}_3 \\ \hat{e}'_3 \cdot \hat{e}_1 & \hat{e}'_3 \cdot \hat{e}_2 & \hat{e}'_3 \cdot \hat{e}_3 \end{bmatrix} = \begin{bmatrix} cos\varphi & -sin\varphi & 0 \\ sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

A combined rotation that do not only rotate about one axis has nine parameters that express the rotation of an object relative to the initial frame (eq. (2.2)) [18].

2.2.2 Quaternions

Quaternions are represented by a vector of dimension four and are an extension of complex numbers to work in a three dimensional space. There are one real and three imaginary coordinates written as:

$$q = \alpha + \beta_1 i + \beta_2 j + \beta_3 k \quad \alpha, \beta_1, \beta_2, \beta_3 \in \mathbb{R} \tag{2.6}$$

The quaternions that describe the 3D rotation in space are the unit quaternions, meaning that the sum of the squares of the four real numbers equals one. When either of the

β-components is equal to one, it will result in the other β-components and α being zero. Either of the $\beta_{1,2,3}$ equal to one, means that unit quaternion is directed along with either of the Cartesian axes, x, y, or z. The i-component corresponds to the x-axis, j to the y-axis, and k corresponds to the z-axis.

The advantage of viewing orientation measurements as quaternions is the minimal space it takes in memory compared to a rotation matrix. Because quaternions only require four values compared to the minimum of nine numbers of a rotation matrix. Also, it does not suffer from Gimbal Lock, where one Degree Of Freedom (DOF) is lost.

2.2.3 Euler angles

A rotation matrix describes the orientation of a frame b with respect to a frame a. The rotation matrix is a 3x3 matrix with nine elements. The orthogonality of the matrix gives six constraints on the matrix elements, and three independent parameters describe the rotation matrix. A widely used set of parameters for the rotation matrix is the Euler angles.

Euler angles are given as a composite rotation of selected combinations of rotations about the x, y, and z axes. One representation used is the Roll-Pitch-Yaw angles, often used to describe the motion of rigid bodies that move freely. A rotation form a to b is described as a rotation ψ about the z_a axis, then a rotation θ about the current (rotated) y axis, and finally a rotation ϕ about the current (rotated) x axis [18]. The resulting rotation matrix is:

$$\mathbf{R}_b^a = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \tag{2.7}$$

A disadvantage with the Euler angle representation is Gimbal lock. It arises through an unfortunate rotation sequence combination and angles where one DOF is lost [19]. The advantages include interpretation of rotation about each axis in the Cartesian coordinate system, where rotation about x (Roll), y (Pitch), and z (Yaw) are easy distinguishable.

2.3 Tension and pulling forces

Tension forces are forces existing either within or applied by a string or wire.

A rope is assumed to be attached to an object at one end (point B) and pulled on the other end (point A). It is also assumed that it has negligible mass. The coordinate system of the rope has a $\hat{j}$-vector pointing upward in the normal direction to the surface, and a $\hat{i}$-unit vector pointing in the positive x-direction (Figure 2.2).

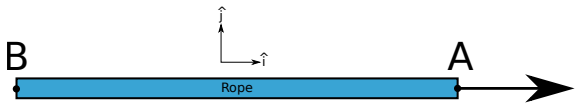


Figure 2.2: Force acting on a rope pulled on end A

Because the rope is not accelerating, Newton's Second Law applied to the rope requires that:

$$F_A - F_B = m * a \tag{2.8}$$

The force dragging the rope in both directions, meaning one force in positive x-direction and one force in negative x-direction, must equal zero. Because the mass is negligible (m = 0), we get:

$$F_A - F_B = 0 \tag{2.9}$$

The applied force is transmitted through the rope. For example, suppose an imaginary slice at point P of the rope, a distance x_p from point B that divides the rope into two sections (L and R) as seen in Figure 2.3.

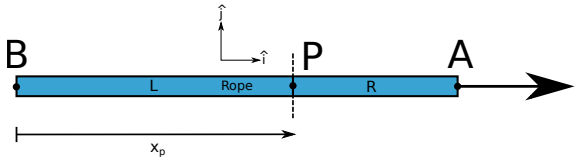


Figure 2.3: Force acting on a rope pulled on end A with imaginary slice at point P

There is a pair of forces acting between the left and the right section of the rope. Denoting the force acting on the left section as $\overrightarrow{\mathbf{F}}_{R,L}(x_P)$ and the force acting on the right section $\overrightarrow{\mathbf{F}}_{L,R}(x_P)$. Newton's Third Law requires that the force in this interaction pair is equal in magnitude and opposite direction.

$$\overrightarrow{\mathbf{F}}_{R,L}(x_P) = -\overrightarrow{\mathbf{F}}_{L,R}(x_P) \tag{2.10}$$

A force diagram for the left and right sections is shown in Figure 2.4.

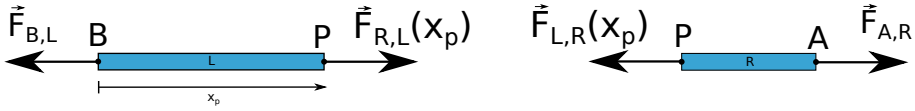


Figure 2.4: Force diagram for left and right section of rope after a slice

The tension T(x_P) at a point P in a rope lying a distance x from the left end of the rope is the magnitude of the action-reaction pair of forces acting at the point P:

$$T(x_P) = |\overrightarrow{\mathbf{F}}_{R,L}(x_P)| = |\overrightarrow{\mathbf{F}}_{L,R}(x_P)| \tag{2.11}$$

For a rope with negligible mass, under tension, the sum of horizontal forces applied to the left section and the right section of the rope is zero. Therefore the tension is uniform and is equal to the applied pulling force [20]:

$$T = F_{A,R} \tag{2.12}$$

The tension in a rope with a pulling force (e.g., from a winch) at one end can be measured at any point of the rope. The force measured at point P will be the same as the pulling force applied at point A.

2.4 Water environment variables

Water environment variables are measurable variables that indicate the quality and condition of the water.

Given that possible measures of reducing highly critical stages of a crowding process can be supplying oxygen to the water, it is interesting to measure the level of oxygen in the water. According to the handbook Fishwell, dissolved oxygen is a vital fish welfare variable to observe during most fish farm operations. In addition, the water temperature has an effect on both environmental and health requirements which makes it an important water environment variable [12].

2.4.1 Dissolved oxygen

Oxygen is essential to all forms of aquatic life and is a fundamental part of water quality assessment since oxygen is involved in, or influences, nearly all chemical and biological processes within water bodies [21]. However, oxygen is likely more limiting for aquatic organisms than terrestrial organisms because of the lower availability in water. The high density and velocity in water also increase the cost of oxygen extraction. An insufficient supply of oxygen that does not meet the minimal energy demands of essential functions leads to suffocation of organisms.

Fish are poikilothermic, which means that their metabolic rates and oxygen requirements increase at higher temperatures. The adverse effects that declining oxygen saturation is harmful effects on fish welfare. Therefore, requirements for dissolved oxygen, or % of air saturation, differ with increasing temperature in the water. Table 2.1 present the lower limits for oxygen saturation with maximal feed intake (DO_{maxFI}) and limiting oxygen saturation (LOS) for Atlantic salmon post-smolts of 300-500g [22].

Temperature	**DO_{maxFI}**	**LOS**
7 °C	42%	24%
11 °C	53%	33%
15 °C	66%	34%
19 °C	76%	40%

Table 2.1: Lower limits for oxygen saturation with maximal feed intake (DO_{maxFI}) and limiting oxygen saturation (LOS) for Atlantic salmon post-smolts of 300-500g.

The advantage of measuring dissolved oxygen is the easy and rapid both to measure and interpret. On the other hand, oxygen levels may vary significantly in the space measured, resulting in a possible miss of low levels.

2.4.2 Temperature

As mentioned in section 2.4.1, fish are poikilothermic. In conjunction with temperature, their physiological and metabolic systems need to adapt to their offered temperature range. Fish's different life stages will affect the preferred temperature, but since only post-smolts are kept in fish cages during crowding processes, the temperatures for this stage will be presented. In general, the temperature for Atlantic salmon should have access to temperatures between 6 °C and 18 °C, but Table 2.2 below presents different temperature thresholds and the implications that follow.

Temperature	**Implications for salmon** [a]
$>$ 20°C	Growth stops, mortality increases and feed intake decreases
16 to 20 °C	Reduced welfare and growth rate, increased stress and mortality
14 to 16 °C	Higher risk of reduced health and welfare conditions
11 to 14 °C	Optimal growth and feed intake, good welfare conditions
7 to 11 °C	Slightly increased risk of reduced health and welfare conditions
$<$ 7 °C	Reduced welfare, increased stress and mortality

[a] [23], [24], [25], [26], [27]

Table 2.2: Temperature thresholds for salmon production

Temperature is cheap, easy, and rapid to measure and interpret (similar to dissolved oxygen). It also explains many aspects of behavior, welfare, and the performance of salmon. However, in a fish farm, it is difficult or even impossible to change the temperature [12].

2.5 Interfaces

The construction of an embedded system presents many possibilities regarding interfaces between components. The following sub-chapters highlight some of the advantages and disadvantages of a few interface devices and standards and their area of usage.

2.5.1 UART

Universal Asynchronous Receiver/Transmitter (UART) is an asynchronous communication device for exchanging serial data between two devices. It allows a receiver and transmitter to communicate without a synchronizing signal. The logical signal produced by the digital UART typically oscillates between zero volts for a low level and five volts for a high level. Different communication standards define the voltage levels, e.g., RS-232 signals range from -3V to 25V and 3V to 25V (section 2.5.2).

The UART device transmits data by sending each bit sequentially, while the receiving UART device re-assembles the bits to the original data [28]. There are three necessary connection wires for a UART device; Transmit Data (Tx), Receive Data (Rx), and Ground (GND). The sequential transfer highlights the importance that the transmitting and receiving end has the same serial communication setup. The setup includes, e.g., baud rate, number of data bits, number of stop bits, parity, and flow control.

2.5.2 RS-232/422

RS-232 was one of the most widely used techniques to interface external equipment to computers that use the UART device. It is a form of asynchronous serial data transmission that sends one bit along a line at a time. The transmission mode for RS-232 is single-ended, meaning that one wire carries a varying voltage that represents the signal while the other is connected to a reference voltage (ground, often 0V). Signals are defined with maximum and minimum voltages corresponding to binary '0' and '1'. A positive voltage ranging from 3V to 25V equals a logic '0', while a negative voltage ranging from -3V to -25V equals a logic '1'. Any voltage between -3V and 3V has an indeterminate logic state.

The standard is suitable for data transfer over short distances. Cables have a maximum length of 20 meters with a bit rate of 20kb/s.

There are two types of RS-232 equipment: Data Terminal Equipment (DTE) and Data Communication Equipment (DCE). DTE's are the computer end of transmission, while DCE's are the equipment (e.g., sensors). Communication between the two equipment requires three signal wires; Transmit Data (Tx), Receive Data (Rx), and Ground (GND).

RS-422, a newer standard, does not have a single-ended transmission mode and can have up to ten receivers on one line. It also allows for longer cables up to 1,6km with a bit rate up to 1 Mb/s. [29].

2.5.3 CAN

Controller Area Network (CAN) networks are based on a shared-bus topology, where buses have to be terminated at each end with resistors. It was initially developed for the automotive industry and is widely used in cars today.

The CAN bus is a twisted two-wire serial communication bus standard that broadcast messages, meaning that all connected nodes can hear all transmissions. It is a synchronized serial communication protocol that enables the communication between microcontrollers without a host computer. CAN rely on the non-return-to-zero (NRZ) bit encoding, which features very high efficiency in that synchronization information is not encoded separately from data.

Because of the bus structure of the standard, it is especially suitable when data is needed by more than one location and in an application that requires a large number of short messages [30].

2.5.4 USB

Universal Serial Bus (USB) technology is used to connect computers with peripherals or input/output devices. While there are around 20 different device classes, a system undergoes an enumeration when a device is connected. An enumeration is a complex back-and-forth stream of messages that supplies the host with the type of device, which commands it supports, and what data it can provide when it is connected [31].

2.5.5 I2C

Inter-Integrated Circuit (I2C) standard was designed to simplify communication between individual devices in highly integrated hardware design. It is designed to support multiple master nodes connected to a single bus. They automatically sense each other's activity and avoid a collision because of distinct chip addresses.

The I2C bus consists of two wires, SDA and SCL, both pulled up to the power supply voltage rail (making them logic HIGH by default). The Serial Data Line (SDA) carries the information, and the Serial Clock Line (SCL) tells the slave devices when it is safe to read it [32].

2.5.6 PWM

A General-purpose input/output (GPIO) device provides a method for sending digital signals to external devices. It can be useful to control devices that have two states: on and off. Pulse modulation is the idea that the computer sends a stream of pulses to the device. The device acts as a low-pass filter, which averages the digital pulses into an analog value. By varying the percentage of time that the pulses are high, versus low, the

computer control how much average energy is sent to the device. It is called a duty cycle and is referred to as modulation. Pulse Width Modulation (PWM) is a type of modulation where the frequency of pulses remains fixed, but the pulse of the positive pulse (the pulse width) is modulated [33].

Chapter 3

System Requirements

Given that the project involves designing and implementing a physical prototype, it is essential to divide the development process into stages. The first stage of the prototype design is to decide the system requirements, which define the framework of an embedded system. They describe the necessary functions and features to conceive, design, implement and operate such a system. When constructing an embedded system, it is crucial to set precise requirements and not confuse them with facts or goals. Otherwise, it can lead to several unnecessary mistakes, a lot of additional work, and economic consequences.

Requirements describe what the system shall do, facts or declaration of purpose describe what the system will do, and at last, goals tell what the system should do [34]. An example of a requirement is:

- The system **shall** have a mass of five kilograms or less.

A fact or declaration can be expressed as:

- The system **will** include three separate devices.

An example of a goal for the system is:

- The system **should** use Bluetooth for transferring messages.

There are several different types of requirements ranging from high-level concept-focused to low-level for specific parts. Two of the most relevant for the development of the prototype are described in each of the next subchapters.

3.1 Functional requirements

A functional requirement is a task or action that must be accomplished to provide an operational capability and define the basic system behavior. These requirements focus on the system concept and are often specified before acquiring components. They lay the general foundation of how the system shall operate. The following list is based on a discussion with the project supervisor, Jo Arve Alfredsen (NTNU), and Research Scientist, Birger Venås (SINTEF), regarding the properties and possible system issues present during a crowding process.

- The system **shall** have sufficient operational life to operate through an entire crowding process.
- The system **shall** have an integrated power source so that it can operate as a standalone device.
- The system **shall** initiate its software at reboot of the system without the need of user influence.
- The size of the system **shall** not influence the crowding process in such a way that it increases the risk of not being able to operate.
- The size of the system **shall** not prevent it from following and capturing the dynamics of the crowding process.
- The system **shall** obtain and store logged data in a persistent way for easy retrieval after a crowding process.
- The system **shall** have enough memory to store a significant amount of sensor data.

3.2 Non-functional requirements

A non-functional requirement is a system specification that describes the operational capabilities and constraints that enhance its functionality. Contrary to functional requirements, the non-functional requirements do not affect the basic functionality of the system. Even if the non-functional requirements are not met, the system still performs its primary purpose.

- The system status indicator **should** be visible from a distance of 30 meters in line of sight.
- The code **should** not depend on a wireless network to obtain system clock time.
- Measured data **should** be stored in a database which enables extraction of Comma-separated values (.CSV) files.

Chapter 4

System components and sensors

So far, six system components has been presented. A Raspberry Pi as system computer and five sensors as data collectors. Each of the sensors measurement principles, as well as the purpose and relevance is described in this chapter. In addition, there are six aiding components that complete the system. These are also presented with their purpose and relevance.

Figure 4.1 is a sketch of sensor placement. It forms the basis for the discussed purpose and relevance. The additional components are placed inside the enclosure part of the system (cylindrical structure attached to the orange floater in the figure), which is the basis of the discussion of purpose and relevance.

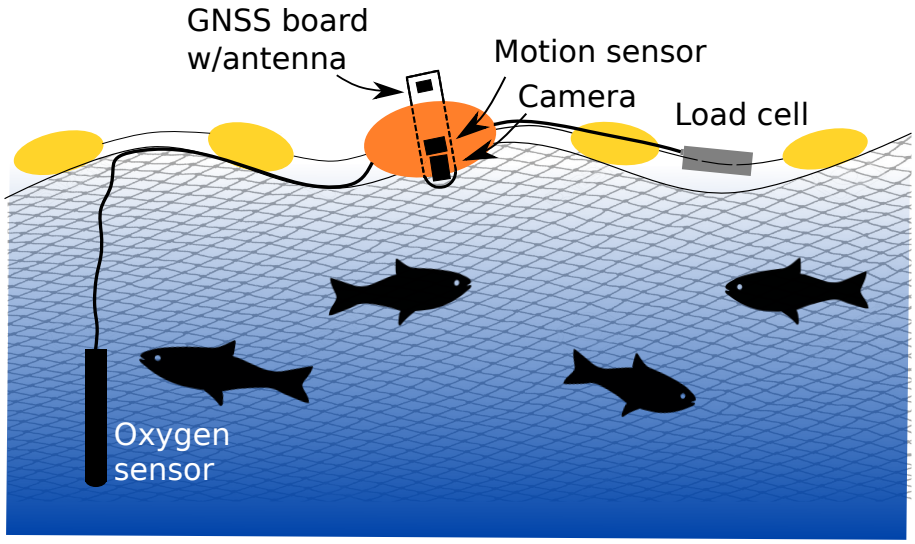


Figure 4.1: Placement sketch of the five sensor components of the sensor system

4.1 Motion tracker

Measurement principle

The motion tracker is an Xsens MTi-670 with an external GNSS receiver. It combines orientation estimates with velocity and sub-meter position data from the external receiver.

The motion tracker is based on Sensor Fusion algorithms combined with Micro-Electro-Mechanical Systems (MEMS)-based sensor measurements. These sensors are microscale integrated devices that are a combination of electronics, electrical and mechanical elements. The Sensor Fusion algorithms combined with synchronized inputs from onboard measurement units with data from an external GNSS-receiver determine the output. The measurement units include a 3-axis gyroscope, a 3-axis accelerometer, a 3-axis magnetometer, and a barometer.

A gyroscope measures changes in rotational motion, both rate of rotation and angular position around a particular axis. It measures or maintains orientation and angular velocity for three axes in the Cartesian coordinate system. The rotation and orientation of the motion tracker can either be presented as a rotation matrix, quaternions, or Euler angles.

Accelerometers measure the acceleration, or the rate of change in velocity, of an object using either the Piezoelectric effect or change in capacitance. The Piezoelectric effect is based on the compression of crystal structures when there is a change in rate. The crystals generate voltage when change occurs, indicating the acceleration present. Change in capacitance uses the same mass compressing concept, but instead of crystals generating voltage, a capacitor changes its capacitance. It allows the applied force to be measured and calculated to acceleration using Newton's second law of motion.

The gyroscope and accelerometer form an Inertial Measurement Unit (IMU) that combined with the aiding sensors, a magnetometer, barometer, and GNSS, achieve accurate tracking, orientation, and rotation estimates.

A magnetometer measures the magnetic field strength and direction to which the object is exposed.

Barometers provide the atmospheric pressure. Based on the pressure measured, readings indicate the units altitude both above or below sea level.

Purpose and relevance

The motion tracker provides information about the rotation, orientation, and position of the enclosure part of the system. The stiff connection to the crowding net means that the system will reflect the net's movements measured by the motion tracker. Movements can come from sporadic contact between salmon and the net, which is assumed to be observed as a slight deviation in amplitude in both directions compared of the reference of the axis. A continuous increase in rotation about one axis can indicate that fish are forced towards the net due to lack of free swimming volume. It is an unwanted consequence of harsh crowding that cannot easily be observed from above sea level.

4.2 GNSS board and antenna

Measurement principle

The Mikroelektrika GNSS 5 CLICK board utilizes an antenna that receives signals from

satellites, while the board translates the incoming signals. GNSS module NEO-M8 is built on the u-blox M8 GNSS engine with the features presented in Table 4.1. The GNSS antenna, type 33-4217NM from Tallysman, is active and provides accurate reception for all upper L-band GPS, GLONASS, BeiDou, and Galileo signals. These signals determine the UTC and the specific position of the device.

Feature	**NEO-M8N**
Category	Standard Precision GNSS
GNSS	GPS/QZSS GLONAS Galileo BeiDou
Number of concurrent GNSS	3
Features	Progammable (Flash) Data logging Additional SAW Additional LNA RTC crystal Timepulse = 1
Grade	Professional

Table 4.1: NEO-M8N features

Purpose and relevance

The external GNSS receiver tracks the position of the enclosure throughout the crowding process. In addition, UTC is obtained from the atomic clocks that are implemented in satellites. The UTC is used to configure the system clock, providing synchronized time stamps for all sensors. Time synchronization is beneficial for real-time data analysis during a crowding process and essential for past crowding process analysis.

The positional data contributes as a reference for the other sensors measurements. E.g., significant deviations in orientation and rotation at the start of the crowding process cannot indicate tightly packed fish because crowding has not began. It can rather indicate water currents dragging on the net or measurement results indicating orientation deviation occurring when deploying the net.

4.3 Load cell

Measurement principle

A load cell is a transducer that converts applied force into measurable electrical output. The LAUMAS FUN load cell utilizes two strain gauges positioned to measure the force acting on it from a wire rope. Measurements are analog variations in force that have to be converted to digital signals for a computer to interpret the measurement. The weight on the load cell is measured by a fluctuating voltage caused in the strain gauge when it undergoes deformation. A load cell consists of either a purpose-designed strain gauge or strain gauges in a *Wheatstone formation*.

A *Wheatstone formation* is a measuring bridge that consists of four strain gauges or resistors in shape shown in Figure 4.2. The load cell used in the system is a resistor based

Wheatstone bridge. It consists of a power source that provides a constant potential V (at points AC of the circuit) independent of resistance variation. The diagonal BD (V_G) contains an instrument that measures the balance of the bridge. The instrument used in combination with the load cell is an ADC that converts the measured resistance to a digital signal that the Raspberry Pi can read and interpret.

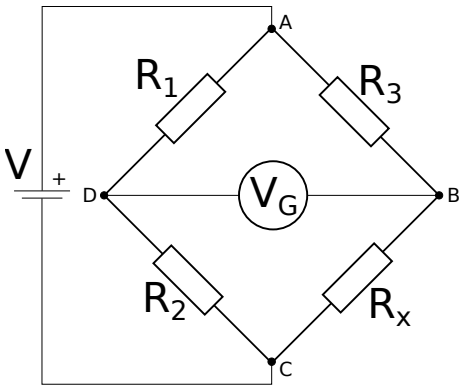


Figure 4.2: Wheatstone bridge circuit with power source at AC and measurement point DB

Measurements from the breakout board include the change in resistance of the load cell. With some calibration, an accurate weight measurement can be acquired.

Purpose and relevance

The sensor is placed on the sweep net's top line and measures the tension applied to it. It reflects the pulling forces in the line, which is created from winches on the service boats surrounding the fish cage (as seen in Figure 1.1c).

The force directly reflects the tension in the top net caused by either increased winch force applied, fish forced towards the net due to lack of space, or a combination of the two. Either one of the causes leads to unwanted consequences. Real-time measurements of the sweep net's tension can serve as a corrective reference to whether the net must be loosened to reduce a potentially critical situation and increase fish welfare.

4.4 Oxygen optode

Measurement principle

The sensor consists of a sensing foil for measuring the O_2-concentration, a temperature sensor for measuring temperature. In addition, the air saturation is derived from the partial pressure of the dissolved oxygen.

The Oxygen Optode uses the principle of dynamic luminescence quenching to detect the O_2-concentration in μM (1 Molar = 1 mole/liter). The principle consists of how specific molecules can influence the fluorescence of another molecule. Fluorescence is the ability of a molecule to absorb light of certain energy and later emit light with lower energy. For example, the luminophore molecule will enter an excited state after absorbing a photon with high enough energy. When the luminophore emits a photon of lower energy, it returns to its initial state. Whether the molecule is uninterrupted or collides with other molecules affects how long it takes for a photon to be emitted. The type of luminophores used will affect the lifetime of a photon before it is emitted. For the type used in the selected oxygen

optode, collision from O_2-molecules will shorten the lifetime of the photons. It is causing the luminophore to return to the initial state quicker than without a collision. The effect is called dynamic luminescence quenching.

The specific luminophore used in the optode is called platinum porphyrin, embedded in a polymer layer. A black gas permeable coating acting as an optical isolation layer protects the indicator layer from the surroundings or direct sunlight. Below the indicator layer is a support layer which is the final component of the sensor sensing foil.

The air saturation (in %) is calculated from the sensors partial pressure and measures temperature (in °C). The partial pressure of the oxygen is calculated by using a two dimensional polynomial that is based on the calibrated phase and temperature. The calibrated phase is calculated based on the difference between the phase obtained with blue and red light from the sensor's raw data.

Additionally, the sensor sensing foil is only permeable to gas and not water, therefore it cannot sense the effect of salt dissolved in the water. Hence, the optode always measures as if immersed in fresh water. If salinity varies significantly, a correction of measured oxygen concentration data should be compensated (Aanderaa provides an excel spreadsheet containing equations for post compensation).

Purpose and relevance

The oxygen optode measures the environment variables; temperature, oxygen concentration, and air saturation in the water.

Given that sensor placement is a few meters below the water surface fastened to the net, it is very close to a possibly dense concentration of fish. Therefore, it is assumed to be the optimal placement to measure dissolved oxygen and temperature. Again, this is because of the dense concentration that can appear in the area and cause a decrease or shortage of dissolved oxygen.

Although the temperature is difficult or even impossible to regulate, it explains many aspects of salmon's behavior, welfare, and performance.

In contrast to temperature regulations, fish farm equipment developed to supply oxygen in waters could be regulated based on the measurement. It can substitute the existing method of experience-based oxygen supply based on visual observations.

4.5 Camera

The Raspberry Pi High-Quality camera consists of a Sony IMX477 sensor. It is a 12.3 Mega-pixel CMOS active pixel-type stacked image sensor with a square pixel array. It achieves high-speed image capturing by column-parallel A/D converter circuits with high sensitivity and low noise image through the backside illuminated imaging pixel structure.

Purpose and relevance

The camera is placed in the enclosure and aimed down towards the bottom of the net structure. It will record the underwater activity during the crowding process.

In contrast to the other sensors, no quantifiable data is provided from the camera. Instead, it is a visual view of the fish density and behavior that can verify or falsify interpreted situations from the other sensors.

4.6 Analog to Digital Converter

Given that the output from the load cell is analog and the Raspberry Pi is unable to read and interpret analog signals, it has to be converted to a digital signal. A SparkFun Qwiic Scale (NAU7802) is a 24-bit ADC with built in gain and I2C output to amplify and convert the readings from a standard load cell.

Purpose and relevance

Without an ADC, measurements from the load cell would be impossible to extract to the Raspberry Pi. In addition to converting the signal to a digital sequence of data, it can be amplified with the onboard programmable gain amplifier. It is an advantage when working with low voltage load cells that can produce signals with low outputs, where different measured weight is less distinguishable.

4.7 Magnetic sensor

Measurement principle

The magnetic sensor is a 55100 Miniature Flange Mounting Proximity Sensor and has a three-wire digital voltage output. It contains a switch that is triggered by the south pole of a magnet. The maximum distance between the sensor and a magnet is 18,0mm for a switch to be registered which can be read by the microprocessor on a digital input pin.

Purpose and relevance

Figure 4.1 shows system component position within the system, where most of the components are placed inside the enclosure (the cylinder attached to the orange floater). Since the system floats on water, is exposed to water, and most electronic components get damaged if exposed to water, the enclosure must be water-tight. The risk of water infiltration increases if the enclosure is opened to be turned on in an environment around a fish cage (e.g., on surrounding boats) and is unwanted. In addition, it is undesirable to start the system logging for a long time in advance of the crowding process because of increased unnecessary data sets and power consumption. The magnetic sensor serves as an interrupt that can be triggered from the outside of the enclosure with a magnet. Logging of sensor data starts after the interrupt is detected by the microprocessor and will reduce the unnecessary data collection.

4.8 RGB LED board

A Sparkfun RGB LED Breakout (WS2812B) is implemented as a system status indicator. Based on a sequence of 24 bits, eight bits for each of the colors red, green, and blue light up the LED. A PWM-signal can adjust the brightness of each color to one of the 256 different levels. Zero indicates no light, and 256 indicates the maximum brightness. The levels of brightness of each color give in total $256^3 = 16.777.216$ different colors.

Purpose and relevance

The microprocessor cannot indicate the system working status on its own. It means that usage of the system on its own presents some difficulties. E.g., it cannot indicate if the system has started logging sensor data or if an error has occurred during usage. Therefore, the only way to establish if the system has logged through an entire crowding process is

by viewing the data in the database afterward. The method of system quality assessment after a crowding process is both inconvenient for usage of time and resources.

Since there are numerous color combinations, distinct colors can indicate system status. E.g., startup of the system, when the system has started logging, and if an error has occurred. The LED functionality is not necessary for the system to work, but is very useful for simple and efficient usage.

4.9 Powerbank

A powerbank of type Ansmann is used to power the microprocessor and belonging electronics of the system. It has a capacity of 10000mAh, which equals power capacity of 37Wh at the nominal voltage of 3.7V. To charge the powerbank, either a micro USB or a type C-socket with both Power Delivery (PD)-standard and QC3.0 (Quick Charge) can be used. A power button turns the unit on and a LED indicator shows the available capacity in 25% intervals. Outputs include two USB type A with QC3.0 and the type C-socket. The PD-standard enables three output levels, 5-6V@3.0A, 6-9V@2A, or 9-12V@1.5A, which are enabled with a PD-device that communicates and negotiates the wanted voltage level.

Most of the components are supplied by the Raspberry Pi which enables a power drain of maximum 5V@680mA, but also enables 3.3V@50mA. According to the oxygen optode's datasheet it require a minimum power supply of 5V, however, conducted tests showed that stable data logging requires 9V with a current drain of 1mA. It was established by using a Thurlby Thandar Power Supply (TS3021S). Because of the two separate USB standards (PD and standard) it was the intention to have a 5V output on one of the USB-A and 9V at the USB-C PD. Testing the described configuration revealed it to be impossible and a different solution had to be implemented (see Section 4.10).

Purpose and relevance

Given the system requirement of an integrated power source so that the system can operate as a stand-alone device a small, portable, and rechargeable battery is a beneficial option.

4.10 Boost voltage regulator

A 9V boost (step-up) voltage regulator from Pololu (U3V12F9) is used to generate a higher output voltage from an input voltage as low as 2.5V. The available output current is a function of the input voltage, output voltage, and efficiency, but can be as high as 1.4A.

Purpose and relevance

The regulator enables a voltage of 9V which is essential to supply the oxygen optode for a stable measurement.

4.11 Overview

Table 4.2 is an overview of the components described in this chapter, including the sensor type and the information it contributes with or the functionality it has within the system.

Sensor	**Type**	**Provided information/purpose**
Motion tracker	Xsens MTi-670	Orientation of system Rotation of system
GNSS board with antenna	Mikroe GNSS 5 CLICK with Tallysman 33-4721NM	Position of system System UTC
Load cell	LAUMAS FUN	Force in top line of net
Oxygen optode	Aanderaa Oxygen Optode 4531D	Environment variables in water
Camera	RPi HQ Camera	Visual view of process under water
Analog to Digital Converter	SparkFun Qwiic Scale NAU7802	Convert and amplify load cell measurements
Magnetic Hall sensor	Littelfuse 55100	Start and stop of data logging
RGB LED board	Sparkfun WS2812B	System status
Boost voltage regulator	Pololu 9V Step-Up Voltage Regulator	Sensor power supply

Table 4.2: Sensor and component overview with type and provided information/purpose

Chapter 5

System Design and Implementation

This chapter describes how the sensor system is designed, both concerning hardware and software, and the necessary steps to implement the system.

To begin with, the initiation and configuration of the Raspberry Pi, including Operating System (OS), database setup, and software development tools are presented.

In addition, the systems storage alternatives are described with the chosen solution.

Then, a hardware connection overview with a detailed description of each component with sensor specification and functionality. Also, a hardware connection diagram that includes all of the system components and how they are connected to the Raspberry Pi.

Furthermore, the software design is presented with main software program and sensor drivers.

Lastly, a presentation of the prototype enclosure including component placement and the flotation devices design and functionality.

5.1 Initial setup and configuration

The first step of the initial setup and configuration of the sensor system is the installation of an OS that runs on the Raspberry Pi. The acquired Raspberry Pi included a kit containing a microSD card, preloaded with New Out Of Box Software (NOOBS). NOOBS is an operating system installation manager with various different operating system options, such as Raspberry Pi OS, Windows 10 IoT Core, OSMC, and Lakka. Given that the Raspberry Pi OS is officially supported by the Raspberry Pi Foundation and has active community support the Raspberry Pi OS (Debian Buster v.4.19) was chosen.

Before developing code, the necessary interface options must be enabled so that the Raspberry Pi recognises devices via the particular type of connection. A console-based configuration tool can enable the different interfaces for the Raspberry Pi via a Graphical User Interface (GUI). It is initiated by running the command *sudo raspi-config* from the command line. The *sudo*-command lets the user run programs with security privileges of another user, by default the *superuser*.

The camera, I2C, and serial communication interfaces needs to be enabled in the configuration tool and require a reboot afterwards to be available for usage.

The recursive process of software code development portrays the importance of a development tool. The Raspberry Pi OS has several desktop graphical text editors with varying usability and functionality. Given that Python was chosen as the preferred programming language in the preceding Specialization project, an editor that supports Python is unavoidable. Python was chosen because of the expansive open-source library and module availability. It is also a high-level, general-purpose programming language that can build and run applications fast, which is advantageous when developing software for a prototype.

As of the year 2020, Python 2 was discontinued. Therefore, Python 3 is used for the prototype in this project. However, there are multiple versions of Python 3 that are continuously updated and renewed. The status of Python branches are available on Python's website where the latest version with a secure status is chosen, which was Python 3.7 (per. 11.01.21). Python version 3.7.7 was downloaded as default to the Raspberry Pi when manually downloading Python 3. Since there are minor differences between Python 3 versions, it was not changed or renewed.

Recall to the development tool. Despite mentioning that Python is a programming language, it is also a scripting language which is interpreted and compiled. There are two widely used ways of running Python. Either by creating an interactive session or running finished scripts from the command line. An interactive session use the command-line for scripting. The disadvantage with the method is that the written code is lost whenever the session is closed. It makes it an unsuitable development tool for the purpose of prototype software design. In contrast, using the command line to run code requires already developed code, e.g., from a text editor. Thonny and Geany are both pre-installed Python Integrated Development Environment (IDE)'s that enable editing code. However, whereas Thonny is developed with beginners in mind with a built-in decoder, Geany has a more advanced GUI with a variable and function viewer to easily navigate the code. Geany also provides syntax highlighting and line numbering, which is convenient for sizable code structures. Based on this, Genay was used for code development. A Python code is run by typing the command, *python3*, followed by the path to the script in a terminal window.

The default installation of Raspberry Pi OS is equipped with Python 3.7 and some packages, including the *pip* tool. The tool enables installation of packages from Python Package Index (PyPi), a repository of software for Python. The necessary packages for the prototype are:

- General purpose: time, multiprocessing, os, sys, logging, re, pprint, datetime, getopt, glob, tabulate
- GPIO control: RPi.GPIO
- LED control: rpi_ws281x
- Database control: mysql-connector-python
- I2C compatibility: smbus2
- Serial compatibility: serial
- Camera compatibility: picamera

If the sensor system is connected to a wireless network, the system clock will be automatically obtained. To prevent automatic sync of clock time towards wireless network, the command *sudo timedatectl set-ntp false* must be run in a terminal window.

5.2 Storage

A vital functional requirement is storing data persistently for retrieval after a crowding process cycle. One of the objectives for the thesis is observation, interpretation, and discussion of the collected data sets to conclude on the relevance of the data obtained. It is therefore, essential to consider the different possibilities of storing acquired data.

There are numerous alternatives for storing data, including textfiles, Comma-separated values (CSV)-files, or in databases, to name a few. The relatively large quantity of data acquired from each sensor during a crowding process cycle substantiates the usage of a database, making data accessible in tables. Given that both text- and CSV-files are flat, meaning that capturing relationships across single data files is not possible, it is not suitable for the intended usage of the stored data beyond the scope of this thesis.

Databases can be classified as, amongst a few, relational (Structural Query Language (SQL)), non-relational (NoSQL), hierarchical-, network- or object-oriented-based. Each one has properties of how data is stored and accessed via commercial or open-source solutions across several programming languages.

In a database, tables and their associated rows and columns can be sorted into either relational or non-relational databases. Relational databases have the option of explicitly structure data and define connections between tables based on keys, e.g., common timestamp. Non-relational databases cannot form connections between tables, but it allows for more variant inputs than relational databases. Hierarchical-based models differ from relational by the implicit joining of segments to its parent and child segments. The model is non-beneficial when many-to-many relationships are required; instead, a one-to-many relationship is preferred when using the method. Similarities can be drawn between hierarchical and network-based models, but the latter can have child segments with multiple parent connections. It increases complexity and maintaining issues. Object-oriented databases have similar properties as objects in Object-Oriented Programming (OOP). Data is loaded into objects and stored in memory. Few programming languages support object-oriented databases and it does not have a standard, as Relational Database Management System (RDBMS) have SQL.

Based on the fact that relational (SQL) databases can form connections between tables based on keys and has easily connection options with Python, it is the chosen option.

5.2.1 Database server

A database server consists of hardware (Raspberry Pi) and software that run a database. The software side of a database server (called the database instance) is the back-end application. The application represents a set of memory structures and background processes accessing a set of database files. The hardware side of the database server is the server system used for database storage and retrieval.

There are multiple options to software that run a relational database. However, access via a Python application is vital because the prototype utilize Python as the programming language. It narrows down the options to SQLite, MariaDB (replaced by MySQL in Raspberry Pi OS), or PostgreSQL database. SQLite is fast, powerful, and has a compact library size, but it does not support multi-user capabilities. Given that the sensor system is intended to transfer real-time data wireless, multiple user access to the database is a necessity. MariaDB does support multi-user features and is easy to install and use, with

a broad community compared to PostgreSQL. The performance of MariaDB is adversely impacted by bulk insertions and long-running select statements. PostgreSQL is not adversely impacted by parallel processing capability, but the memory usage of the database makes it a less suitable option because of the limited memory available on the Raspberry Pi. Therefore, the relational database used for data storage is a MariaDB database.

MariaDB organizes data into tables with the option of the relationship between data types. Comparing data from each sensor and analyzing it against each other at timestamps supports the usage of relational databases. The timestamp acts as a primary key with sensor data as elements. MariaDB also provides a multithread pool without having to purchase additional packages or upgrade the product. Implementation of this is accomplished by having separate database connections for each thread/process. It results in non-overlapping execution of database insertion in the different threads, as long as two threads does not insert data into the same table.

The MariaDB server has to be installed and a database inside the server has to be created. Then, a connection from python is established with the *connect* constructor which utilize a set of communication parameters. It includes the user (must have permission to edit the database), password, host (localhost if database is stored locally on computer/microprocessor), port (3306 is default for database connection), and the database name. Tables for data insertion are created in the main script and does not require to be created before running the main script.

5.2.2 Memory card

A 16GB Class 10 microSD card includes the Raspberry Pi OS and the available storage on the Raspberry Pi. The class rating are defined as a Speed class and refer to the absolute maximum sustainable write speed. A card with class 10 has a speed of 10MB/s.

The SD-card is used for storage of Python files, database files (which both take up minimal amount of storage that does not affect the free storage significantly) and video recordings from the camera. Depending on the camera configurations and the duration of the captured video, the required storage space is determined. As a rough calculation of the required storage, a video storage calculator was used (`https://www.seagate.com/gb/en/video-storage-calculator/`). Inputs of the calculator are:

- Number of cameras: 1
- Frames per second (fps): 15
- Hours per day: 2
- Number of days stored: 1
- Resolution: 5MP
- Video quality: High
- Compression type: H.264

The resolution in megapixels (MP) is given by the chosen resolution screen multiplied and divided by 1.000.000. E.g., a resolution of 2592x1944 results in 5,1MP.

The required storage, based on the inputs in the list above, is 6,31GB. Given that the available storage on the Raspberry Pi SD-card was 12GB when the prototype was ready for testing, only one sequence of testing is possible to record video of with the memory card (an upgrade of the memory card is possible).

5.3 Hardware

Before describing each system component's specifications and functionality, it is convenient to present a conceptual block diagram of the system. It is a simple representation of how the system is composed and interconnected. Figure 5.1 shows which components the system consists of and how they are interfaced towards the microprocessor.

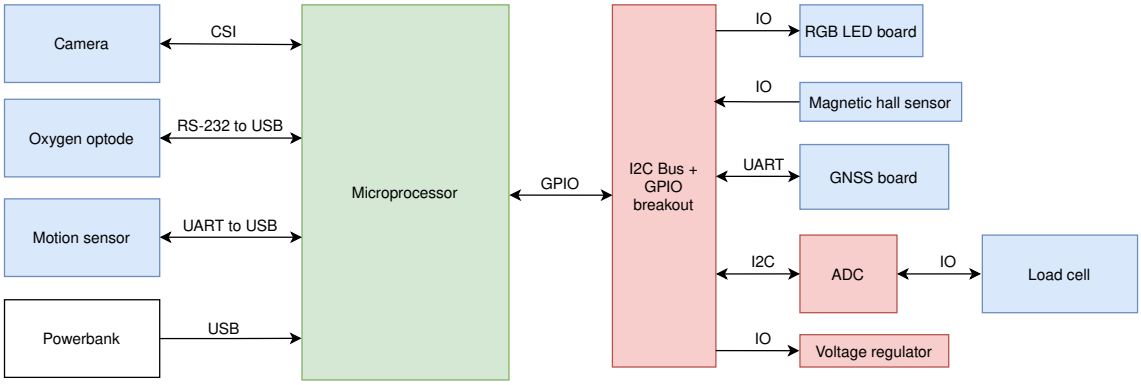


Figure 5.1: Conceptual block diagram of system components and their interface towards the microprocessor

5.3.1 Microprocessor

A Raspberry Pi 4 serves as the microprocessor. It is the system's brain and has the role of fetching addresses of instructions that are to be executed from the program counter in the memory and performs these instructions. First, the fetched addresses have to be converted into binary code so that the microprocessor can understand the instructions, a process called decoding. The instructions can then be executed. After the execution, the results are stored in memory.

The Raspberry Pi 4 has a 64-bit quad-core processor at 1.5GHz. The four cores enable the division of tasks between multiple components, an advantage when operating with several sensor components that require distinct specifications in terms of, e.g., sampling frequency.

The microprocessor provides a set of connection options. These include a micro SD card slot used by the SD card that stores all the Raspberry Pi's files and OS. A USB-C port is designated for power supply, four USB-port for device connection, and 40 GPIO pins with standard (e.g., ground and 3.3V) and designated (e.g., I2C, PWM, and UART) input and output pins. In addition, there is a Camera Serial Interface (CSI) port, a Display Serial Interface (DSI) port, a stereo audio port, and an Ethernet port. Finally, two High Definition Multimedia Interface (HDMI) ports that enable the connection of two monitors.

While developing the prototype, two USB ports were used to interface a keyboard and a mouse. Also, a monitor was connected to an HDMI port. The devices are used for viewing and editing code directly on the Raspberry Pi. Since two USB ports are used by sensors and the remaining two used by the keyboard and mouse, all available USB ports are used during development and on-desk testing. However, only two of the ports are used during prototype testing. Therefore, expanding the prototype with multiple sensor devices is possible. It is important to not limit an embedded system on any specifications because it is often necessary to expand or change, regardless of how well the prototype is designed and planned.

As seen in Figure 5.1 there are five direct component connections to the Raspberry Pi. These include:

- Powerbank for power supply of the Rasberry Pi using a USB-A to USB-C cable.
- Camera for visual capturing using a CSI.
- Oxygen optode for measuring environmental variables using an RS-232 to USB-A cable.
- Motion tracker for measuring system motion using an UART to USB converter and a micro USB to USB-A cable.
- I2C bus and GPIO-pins breakout (component called a Qwiic HAT) using a 40-pin flat cable.

Each component interface and specification is described in the related subchapter.

The Qwiic HAT's function is to transfer signals through either the I2C bus or the broken out GPIO-pins. Qwiic HAT originates from the distributors Qwiic connect I2C system that uses 4-pin JST connectors. Connections using this system do not require any soldering, and connectors are polarized to ensure the correct connection between Qwiic devices [35]. The ADC utilize one of the four I2C bus's Qwiic connectors with belonging Qwiic cable.

The remaining five components connected to the Qwiic HAT use broken out GPIO-pins, meaning that some pins from the Raspberry Pi are made electrically accessible. These are:

- RGB LED board utilize the 3.3V, ground and a digital output pin to receive a PWM-signal.
- Magnetic Hall sensor utilize the 5V, ground and a digital input pin to transmit an interrupt signal.
- GNSS board utilize the 3.3V, ground, as well as RX and TX for transmitting and receiving signals.
- ADC utilize the 4-pin JST connector in addition to 5V and ground.
- Voltage regulator utilize the 5V and ground.

5.3.2 Power supply

A power supply provides electric power to an electrical load. The systems microprocessor and belonging system components are electrical loads and will not operate without a power supply.

As the functional requirements state, the system should have an integrated power source to operate as a standalone device and have sufficient operational life to operate through an entire crowding process. Based on the requirements, a powerbank is used. It immediately satisfies the requirements of a standalone device with an integrated power source. Each system component with the required voltage and current levels determines the individual power drain at all times. Sufficient operational life is ensured by calculating the system's total power drain throughout an entire crowding process (Table 5.1).

From these calculations, it is observed that the system's total power consumption is 6.8854W at all times. Multiplying it by the wanted duration of operation will establish the system's power consumption over time. As mentioned before, a crowding process

Description	**Hardware**	**Vs [V]**	**I [mA]**	**P [W]**
CPU	Raspberry Pi 4 4GB	5	680	3.4
Motion tracker	Xsens MTi-670	5	106	0.53
GNSS board	MikroElektrika GNSS 5 CLICK	3.3	67	0.2211
GNSS antenna	Tallysman TW4721	3.2	10	0.032
Oxygen optode	Aanderaa 4531D	9	100	0.9
Load cell	LAUMAS FUN	9	25	0.225
Camera	Raspberry Pi HQ cam	5	250	1.25
Magnetic Hall sensor	Littelfuse 55100	5	10.6	0.053
RGB LED board	SparkFun WS2812B	3.3	60	0.198
ADC	SparkFun NAU7802	3.3	2.1	0.0693
Voltage regulator	Pololu U3V12F9	5	1.4	0.007
Total			**1312.1**	**6.8854**

Table 5.1: Power consumption of each system component

can last up to two hours. However, to increase the margins and ensure operational life throughout an entire crowding process, the specification is increased to three hours. The power supply must be able to deliver 3h * 7W = 21Wh (rounding the total power consumption up to 7W). According to the power bank's datasheet, the possible total energy based on the 10,000mAh capacity and 3.7V nominal voltage is 37Wh. It is a significant positive difference between the possible total energy that the power bank can deliver and the necessary power needed by the system of 16Wh. In theory, the system can be powered for 5 hours (5h * 7W = 35Wh).

Observing the required voltage (V_S) for each component in Table 5.1 reveals the need for four different voltage levels.

The Raspberry Pi is supplied by the powerbank's USB-A port at 5V. The rest of the components, except the oxygen optode and the antenna, is supplied by the Raspberry Pi's GPRIO power supply pins (3.3V or 5V). The oxygen optode is supplied with the 9V output from the voltage regulator, whereas the antenna is supplied by the GNSS board through the SMA connector. The camera has sufficient power (5V) and signal connection through the CSI connector.

Since the powerbank support the PD standard from the USB-C input/output, a multi-voltage extraction from the unit was intended to be used. A voltage of 5V from one of the USB-A outputs and a voltage of 9V from the USB-C output through a Power Delivery Board supporting the PD standard. The board would negotiate the wanted voltage (9V) from the powerbank and retrieve it to an output on the board. The method was tested independently and a 9V voltage was available on the output. However, an additionally connected device on the USB-A output would pull down the voltage on the Power Delivery board output to 5V, causing the solution to be insufficient. The demand for a step-up voltage regulator was unavoidable.

5.3.3 Sensor interface and output

This subchapter describes and discusses the possible and selected interfaces between sensors and the microprocessor. In addition, the potential sensor outputs are presented.

Motion Tracker

The motion tracker is a sensor module that can be used as a standalone device for a minimal connection, or attached to a larger Development Board (provided with the acquired version, MTi-670DK). The two possible solutions present some differences in host interface connection and size (see Figure 5.2). While the sensor module as a standalone component requires a connection through the sensor's communication pins, the Development Board enables RS-232, RS-422, and CAN through DSUB9 connectors and UART with the external connector pins on the board. The sensor module is connected to the Development Board with a MTi-600 connector and fastened with three screws. For enabling the different possible host connections, the board is equipped with two switches that have to be set according to the wanted host peripheral.

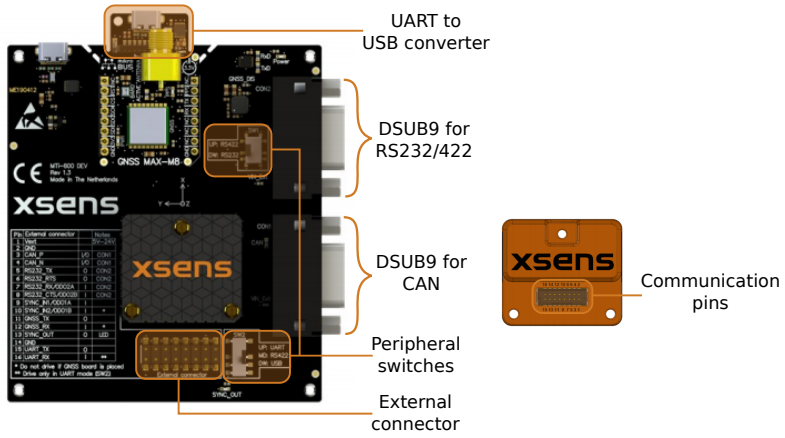


Figure 5.2: Xsens motion tracker, relative size comparison between sensor with Development Board (L) and without Development Board (R)

Using the Development Board presents some advantages. There is no need for additional communication components except cords, a microBUS compatible GNSS extension socket to connect a GNSS module, and exposed pins via the external connector making it easier to test all sensor features. The many peripheral options, on the other hand, present a disadvantage in size. In addition, the Development Board will not fit in the selected watertight enclosure and is therefore not used.

The following discussion of host interface connections is based on a standalone usage of the sensor.

As a standalone sensor module, host connection options include RS-232, CAN, or UART interfaces, or USB using a UART to USB converter. The UART to USB connector is a small part of the Development Board that can be broken off and used separately with the sensor.

All of the available interface options (except the UART to USB converter) require connection to the GPIO-pins of the microprocessor via the sensor's communication pins. For example, both RS-232 and UART communication interfaces would connect to the serial

ports transmitting (TX) and receiving (RX) on the Raspberry Pi for connection and power input voltage of 5V and ground.

If the motion tracker was to use CAN interface, it would be the only node of a CAN bus as none of the other sensors have CAN interface option. A CAN bus would require a standalone CAN Controller with an additional interface to the Raspberry Pi like SPI.

The UART to USB converter is a part of the Development Board, converting UART signals to be available through USB. It can be broken off the larger board and serve as a UART to USB standalone board. It is significantly smaller in size than the Development Board and a better alternative in the confined space of the watertight cylindrical enclosure. The sensor can be connected to the breakout board with the exposed pinouts via a ribbon cable. The breakout board is connected to the Raspberry Pi via a USB cable which supplies the sensor with a power input voltage of 5V and makes sure that the sensor is grounded.

When two devices are connected, corresponding communication parameters must be set to ensure that the signals are sent and received in the same format and rate. If such parameters are not set, a message sent from a sensor could be read at the wrong speed (because of different baud rates), or the message length could be read wrong (because of different numbers of data bits or stop bit). In addition, the device's connection port must be specified. For the motion tracker that interfaces with the host with USB, the specific USB port works as a connection parameter for the sensor. It is vital to connect the device to the same port on the Raspberry Pi for each test of the prototype since the parameter is hardcoded. The connection parameters that have to be specified for the motion tracker are:

- Communication port: USB 0
- Baudrate: 115200

The communication port is defined in Python code by stating the device location and name. The device can be found in the "dev"-folder, which is the location of special files or device files, under the name "ttyUSB0". The "tty" is short for teletype (known as terminal) and is a device that allows interaction with the system by passing on the data to the system and displaying the output produced by the system.

The motion tracker provides a GNSS/INS solution offering a time, position, and velocity output in addition to orientation estimates. However, using the motion tracker without the Development Board requires a separate connection of the GNSS solution. The module could be used by itself and a separate GNSS board (as described in Section 4.2) would not be required. This solution was firstly implemented, but due to user error of the 9V power supply the board broke and was unusable. Therefore, a new GNSS board was acquired and used instead.

The raw sensor signals from the motion tracker is combined and processed at a high frequency to produce a real-time data stream of the outputs up to 400Hz. However, a data stream with such high frequency is assumed to be unnecessary for the intended use.

Table 5.2 shows an overview of the possible sensor output parameter options. It is vital to notice that choosing maximum frequency sampling of more than five parameters results in an overload of information from the communication object to the Raspberry Pi. Reasonable frequency for the wanted parameters are between 20-30Hz and is used when testing the system in a natural environment (the specific frequency was found by conducting a trial-and-error test).

Parameter group	Parameter type with unit*	Maximum frequency [Hz]
Temperature	Temperature [°C]	1
Timestamp	UTC Packet counter Sample time fine Sample time coarse [s]	**
Orientation data	Rotation matrix Quaternions Euler angles [deg]	400
Pressure	Barometric pressure [Pa]	50
Acceleration	Delta velocity [m/s] Acceleration [m/s^2] Free acceleration [m/s^2]	400
Position***	Altitude Ellipsoid [m] Position ECEF [m] Latitude and longitude [deg]	400
Angular velocity	Rate of turn [rad/s] Delta Q	400
Magnetic field	Magnetic field [a.u.]	100
Velocity	Velocity XYZ [m/s]	400
Status	Status byte Status Word Device ID Location ID	**

Table 5.2: Motion tracker output parameter options

* Unit N/A where no unit is specified.
** Output frequency ignored; if enabled, this data will accompany every message. Output frequency is 0xFFFF.
*** Only accessible with the external GNSS board provided in the Development Kit.

GNSS board and antenna

An external GNSS board from MikroElektronika, type GNSS 5 CLICK, carries a NEO-M8N GNSS receiver module from u-blox. The GNSS receiver utilizes concurrent reception up to three GNSS systems (GPS/Galileo together with BeiiDou or GLONASS) and recognizes multiple satellite constellations simultaneously, providing high accuracy.

The board requires a 3.3V power supply and can communicate with a microcontroller through USB, Display Data Channel (I2C compliant), or UART interface. In addition, pins for resetting (RST), pulse interrupt (INT), and external interrupt (PWM) are available on the boards MikroBUS.

The USB supply voltage of the device require a voltage of maximum 3.6V. Given that the voltage on the USB port of the Raspberry Pi is 5.1V, it it not a suitable option because it requires a step-down regulator.

Since it is already established that the system has an I2C bus (from the Qwiic HAT) with four available ports, it would be reasonable to attach another I2C device to it. However, the GNSS board from MikroElektronika is not equipped with the Qwiic port system, which means that the device would be connected to the broken-out I2C pins on the Qwiic HAT instead. There is no difference in circuit complexity between the DDC (I2C) and UART interface, however the DDC have to be addressed in the software and does not support Full Duplex communication. UART does not require software addressing and supports Full Duplex communication. Therefore, serial interface with the UART pins, transmit (TX), and receive (RX), is chosen. The communication parameters for the GNSS board are:

- Communication port: Serial 0
- Baudrate: 9600
- Data bits: 8
- Parity: None (Used to determine if the data character being transmitted is correctly received by the remote device)
- Stop bit: 1

An antenna is required to receive signals from the satellites, while the receiver translates the signals. The receiver is the GNSS board. Given that the motion tracker Development Kit included an antenna with the same SMA-connector as the GNSS board, it is utilized to receive signals. The antenna is a Tallysman active antenna (33-4721NM) with an SMA connector that operates at frequency 1.599GHz.

Output parameters from the GNSS board are based on the NMEA-0813 standard, where GGA, GLL, GSA, GSV, RMC, VTG, and TXT messages are activated at the startup of the device. The most relevant message version is RMC (Recommended Minimum Specific GNSS data) which contains the output parameters in Table 5.3. The wanted output format must be extracted in the software application.

Output parameter	Description
UTC time	UTC time in hhmmss.sss format (000000.000 ∼235959.999)
Status	Status 'V' = Navigation receiver warning 'A' = Data Valid
Latitude	Latitude in dddmm.mmmm format. Leading zeros are inserted.
N/S indicator	'N' = North; 'S' = South
Longitude	Longitude in dddmm.mmmm format. Leading zeros are inserted.
E/W indicator	'E' = East; 'W' = West
Speed over ground	Speed over ground in knots (000.0 ∼999.9)
Course over ground	Course over ground in degrees (000.0 ∼359.9)
UTC date	UTC date of position fix, ddmmyy format
Mode indicator	Mode indicator 'N' = Data not valid 'A' = Autonomous mode 'D' = Differential mode 'E' = Estimated (dead reckoning) mode
Checksum	Checksum

Table 5.3: GNSS NMEA RMC output

Oxygen optode

The oxygen optode is of type 4531D from Aanderaa with a free end cable, meaning that the wire ends are exposed. The specific version can be interfaced with a host using the RS-232 standard. Other versions of the sensor are equipped with an analog output option, but the design of the sensor and cable does not differ. It results in three wires designated for analog output that are unavailable and unused.

For the Raspberry Pi to retrieve the digital signals from the sensor, the RS-232 standard has to be implemented in a way that the microprocessor can interpret. There are multiple options, including an RS-232 expansion module for Raspberry Pi, a direct connection to the UART pins of the Raspberry Pi, and an RS-232 to USB cable including an FTDI chip (see Table 5.4 for comparison).

RS-232 option	**Disadvantages**	**Advantages**
RS-232 module connected to Raspberry Pi via UART pins	- Large in size - UART pins already in use - RS-232 cable extension on oxygen optode cable end	- Enables easy connection with a DE9 female cable
Connect free end of oxygen optode to UART pins	- Less robust hardware connection compared to the others - UART pins already in use	- Minimal connection w/no additional components - Small in size
RS-232 to USB cable connection	- Additional wire that use one USB port	- Minimal connection - Small in size

Table 5.4: RS-232 connection options to Raspberry pi w/advantages and disadvantages

After considering the three alternatives, the RS-232 to USB cable connection is the most advantageous option. Even though it requires additional wiring, equivalent to the other alternatives, the already used UART pins on the Raspberry Pi portray a complication that needs a workaround for the two other options.

The chosen RS-232 to USB cable use FTDI's FT232RQ USB to serial UART integrated circuit device. It converts the serial RS-232 signal from the sensor to a serial signal that the microprocessor can read through a USB port. The free end sensor cable is soldered to the free end USB cable as in Figure 5.3.

Pinouts of the 6-pin FTDI cable are; RTS, RX, TX, 5V, CTS, and GND. However, only the minimum number of pinouts for RS-232 communication is used. These are RX (Receive signal), TX (Transmit signal), and ground. The signal transmitting pin (TX) of one device is connected to the signal receiving pin (RX) on the other device. This is essential for the communication between the two devices.

A positive voltage supply is necessary to power the sensor unit. As seen in Figure 5.3, a positive supply pin is available on the USB cable. Since the oxygen optode's datasheet stated a minimum supply voltage of 5V and the voltage supplied on the VCC pin of the USB cable is 5V, a direct connection between the two was tested. However, 5V was insufficient for the sensor to output a continuous data output. A Thurlby Thandar Power Supply was used to test the necessary voltage, which revealed that at least 8V at 1A current had to be supplied. A 9V power supply from the voltage regulator is used to ensure the sensor transfers data successfully. As described in section 5.3.2, a step-up voltage regulator is used to get the wanted voltage level.

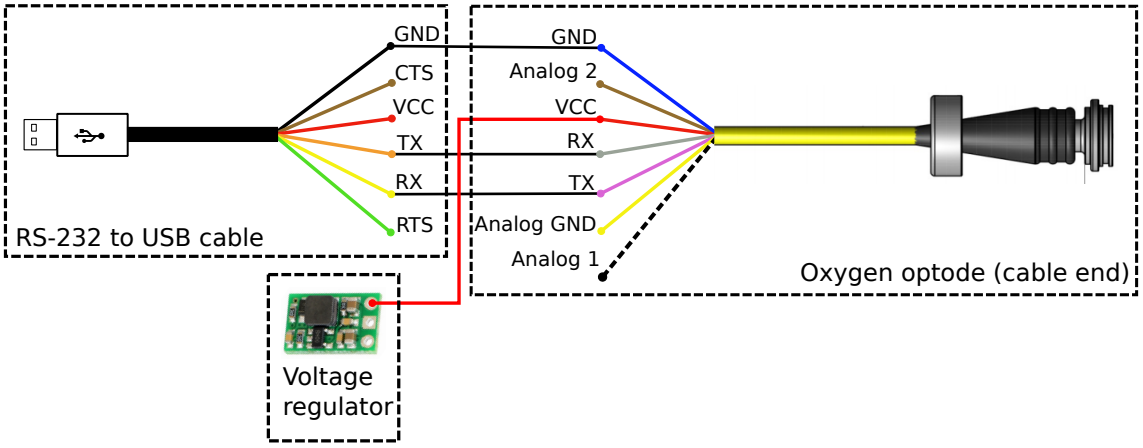


Figure 5.3: Oxygen Optode 4531D connection with a RS-232 to USB cable and a voltage regulator

The oxygen sensor has the following specifications in terms of communication with a third party program:

- Communication port: USB 1
- Baudrate: 9600
- Data bits: 8
- Parity: None
- Stop bit: 1
- Flow control: Xon/Xoff (Sending of data transmission (X) control characters along the data stream, transmission on or transmission off)

These specifications enable logging of the optode directly from a PC with autonomous sampling between 2 seconds and 255 minutes. The possible sensor parameter outputs are listed in Table 5.5 with the corresponding measurement units.

Output parameter	**Unit**
O_2-concentration	μM (Mol)
Air Saturation	%
Temperature	°C
Oxygen raw data	N/A
Temperature raw data	N/A

Table 5.5: Aanderaa 4531D Oxygen optode output parameters and corresponding units

Load cell

The LAUMAS FUN load cell is based on the concept of strain gauges in a Wheatstone formation. The signals from the formation are analog, which the microprocessor cannot interpret. Therefore, an ADC is necessary to convert the analog voltage from the load cell to a digital value that the Raspberry Pi can read. The ADC used is a 24-bit converter of type NAU7802 from Sparkfun. It has a spring terminal that connects the four wires from the load cell without any soldering. The four wires are power supply (red wire), ground (black wire), positive (green wire), and negative (white wire or dotted line in figure Figure 5.4) output signal. The analog signal between the signal wires represents the electrical resistance in response to the applied tension on the sweep net's top rope.

The differences in measured resistance between the positive and negative output signal are relatively small and require an amplifier. Fortunately, the ADC has a programmable amplifier gain range from 1 to 128 included. The analog signal is converted and amplified with the selected amplifier gain and sent on the serial data pin of the I2C bus available on the Raspberry Pi. The ADC is connected to the Qwiic HAT with a Qwiic cable, making it the only device connected to the I2C bus. When multiple devices are connected to the same I2C bus, all devices must have a unique device address. The address is the only thing that the microprocessor can differentiate devices on and read or write to the correct device.

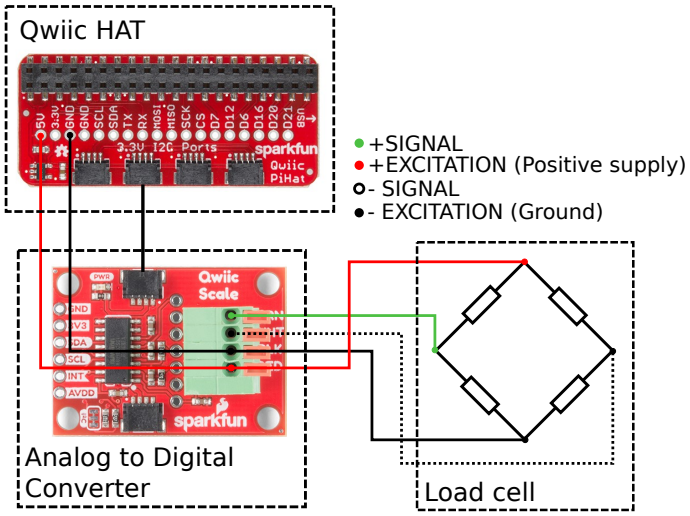


Figure 5.4: Loadcell connection to ADC and Qwiic HAT

The communication parameters that are set for the load cell:

- Communication port: I2C-port 1
- Device address: 0x2A

In addition to the programmable gain, the ADC has programmable output data rate, Low-Drop-Out voltage, and ADC channel are. The output data rate is the ADC sample rate, which has the options 10 (default), 20, 40, 80, and 320 Samples Per Second (SPS). Since the tension in the top rope of the crowding net is assumed to increase slowly, an excessive amount of samples per second are not necessary, and the default value is used. The ADC has an onboard Low-Drop-Out voltage regulator, which derives lower output voltages from a main supply or battery. It is convenient but not necessary for the ADC to derive a lower output voltage. Therefore, the Low-Drop-Out voltage regulator is not activated. It is possible to use the dual-channel mode to perform two conversions at the same time, but it will not be necessary for the intended use of the ADC.

According to the datasheet, the maximum supply voltage for the load cell without damage is 15V. Given that a voltage of 5V is available on the broken out GPIO-pins from the Qwiic HAT, it is used.

The load cell has one output, the tension in a rope, which is represented by a number between zero and 2^{24} (= 16,777,215). The specific number comes from the ADC being 24 bits, where 24 binary combinations results in 2^{24} possible output combinations. The measured bit value does not represent a voltage or weight directly, it has to be calibrated and calculated. Given that load cells have a linear response, meaning that the bit value

changes in proportion to the weight applied to it, the weight can be determined by the following linear equation:

$$Weight = Calibration\ factor * Bit\ reading + Zero\ offset \tag{5.1}$$

The measured value when zero weight is applied on the load cell is the zero offset (measured to be 5,260). If the bit value of a known applied weight (e.g., 10kg) is 14,852, the calibration factor is calculated by:

$$Calibration\ factor = \frac{14625 - 5260}{10} = 936.5$$

Every time the digital bit reading increase by 936, the weight has changed with 1kg. If the bit value is 60,000 the wight is:

$$Weight = \frac{60,000 - 5,260}{936.5} = 58.48kg$$

The load cell will measure the tension in the top rope of the crowding net, and the sensor is occasionally immersed in water and most definitely exposed to water during a crowding process. Hence, the chosen load cell has protection class IP67, meaning protection against dust and the effects of temporary immersion in water on a depth between 15 cm and 1 m.

According to the datasheet, the rated output is 3mV/V ±0.1%. The rated output is the sensors electrical signal which is proportional to the applied excitation voltage. Since the excitation voltage is 5V, the rated output is 15mV. Full load on the load cell will produce a 15mV signal on the output. A gain of 128 will amplify the signal on the SDA line of the I2C bus connection to 1.92V. High and low signal in the line will be easier to differentiate.

Camera with lens

The Raspberry Pi High-Quality camera is connected to the Raspberry Pi's camera connector with the CSI and a 15-pin ribbon cable. There are no other connections needed to power, configure, or extract captured images or videos from the camera.

The camera requires a lens to capture a motive. A wide-angle lens with a 6mm focal length and 3MP resolution is used with a C-mount. The focal angle of the lens is 63° which defines the focal view. Since the camera does not support auto focus, the lens has manual adjustment rings for back focus and aperture (the opening of the lens where light passes to enter the camera).

The Raspberry Pi High-Quality camera has a set of available configuration options for resolution and framerate, which is set to default if not defined. The options are included in Table 5.6.

Magnetic hall sensor

The magnetic hall sensor is a three-wire sensor with a switch that is triggered by a hall plate exposed to a magnetic field. Two of the wires are the supply voltage and ground. The sensor requires a 5V voltage, which the Raspberry Pi delivers on GPIO pin 2 broken out on the Qwiic HAT. The last wire is the signal wire that is connected to a digital pin on the Raspberry Pi, with a pull-up resistor to the positive supply. A pull-up resistor will ensure a known state of the sensor signal is digital HIGH. When the sensor is triggered with a

Resolution	Aspect ratio	Framerate	Field of view
2592x1944	4:3	1-15fps	Full
1296x972	4:3	1-42fps	Full
1296x730	16:9	1-49fps	Full
640x480	4:3	42.1-60fps	Full
640x480	4:3	60.1-90fps	Full
1920x1080	16:9	1-30fps	Partial

Table 5.6: Raspberry Pi High Quality Camera resolution and framerate options

magnet, the sensor signal will be digital LOW which is read and used as an interrupt in the software implementation. The value of the resistor is calculated with Ohms law:

$$R_{pu} = \frac{VDD}{Io} = \frac{5V_{dc}}{10mA} = 500\Omega \tag{5.2}$$

The current used in the calculation is 10mA even though the maximum output current is up to 20mA. This is because the hall sensor should not have the maximum possible current through it at all times to reduce the load on the sensor. A resistor of 500Ω was not available at the university facilities, so the closest possible resistor was used instead, a resistor of 470Ω. The slight difference between the calculated and implemented resistor does not change the current through the magnetic hall sensor considerably. The resulting current is 10.6mA with the implemented resistor.

RGB LED Board

The LED board has six pins, where three are designated for input, and three for output. The design makes the chaining of multiple LED boards easy and accessible. Since only one indicating signal is necessary for the system, the chaining functionality is not used. The output side power supply and ground pin are wired to the corresponding pins on the input side, while the output pin is unconnected. The board is connected to the Qwiic HAT with the power supply connected to the 3.3V pin, the ground to ground, and digital input pin to a GPIO-pin with the option of PWM on the Raspberry Pi (pin 12).

5.3.4 Hardware connection overview

A hardware connection overview of how the sensors, components, and the system computer are connected is presented in Figure 5.5.

Blue wires indicate digital signals, and orange wires indicate the USB power connection. Red wires are positive power supply, and black is ground. UART connection wires (RX and TX) are indicated with the colors yellow and purple, respectively.

There are three separate power supplies with different voltage levels. The 3.3V and 5V are directly extracted from the Raspberry Pis GPIO pins through the Qwiic HAT board. The 9V is delivered on the output of the voltage regulator and supplies the oxygen optode.

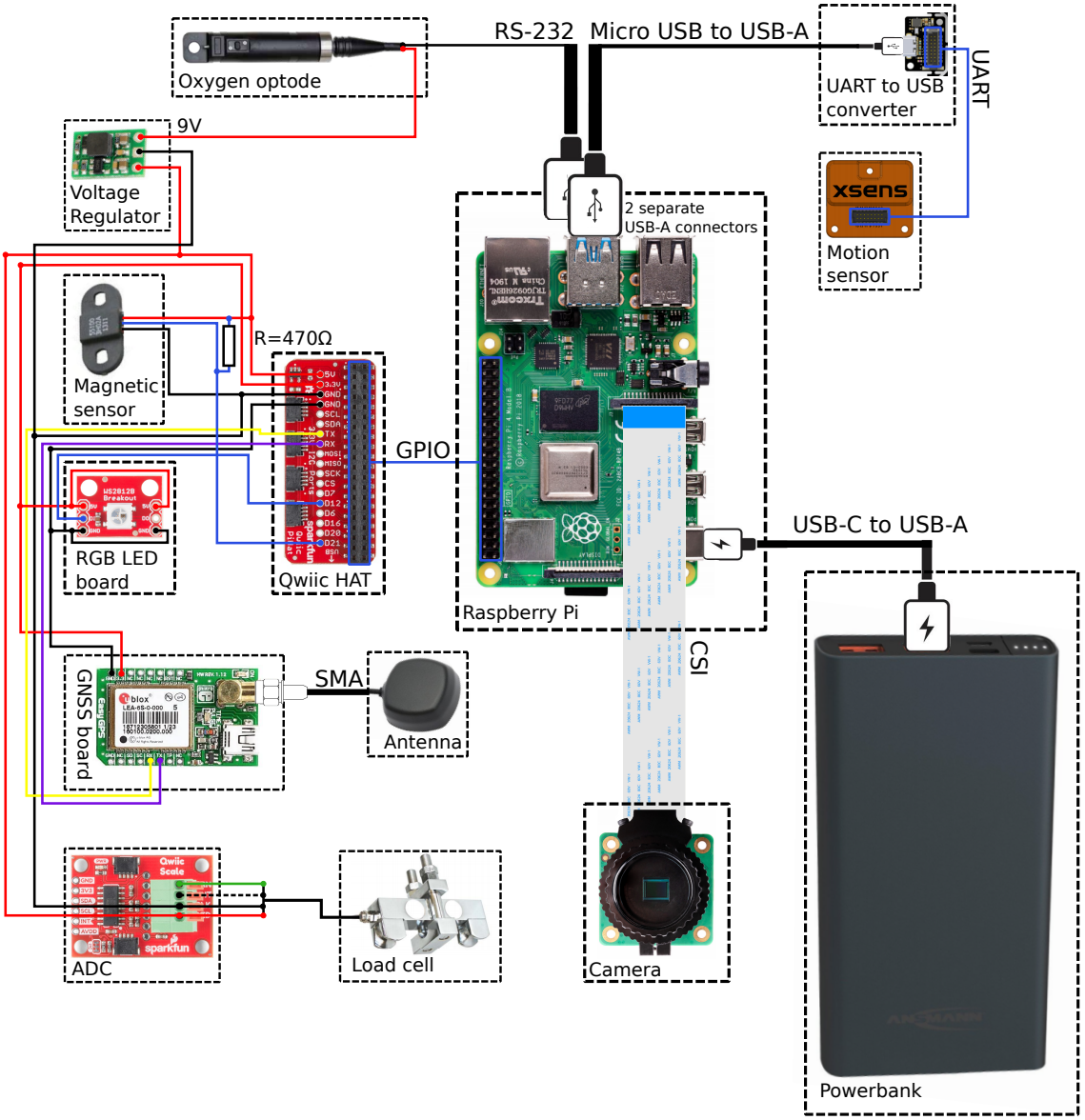


Figure 5.5: Hardware diagram of all system components and the connection between them

5.4 Software

Designing and implementing software differ considerably from hardware. For example, the chosen hardware host interfaces and soldered connections are more permanent than variable names and string output variations in code. Software development is also a recursive process of writing, testing, and editing code to customize it for a specific use-case. As improvement of software always should be considered, time consumption can increase if a set of predetermined functionalities to satisfy are not set before beginning software development (the key functionalities are described in the next sub-chapter).

When it is apparent that code structures will reach a considerable length, it is reasonable to structure the code in different modules. In this thesis, separate Python scripts are constructed to divide the software into prominent parts that share equal resources and acts on the same hardware device. The advantages of structuring the code into subparts are that it reduces the effort required to edit or extend it and increases the efficiency when maintaining and using the code. The main script can access the other scripts and use the necessary classes and functions from these to operate.

The software for the prototype consists of the following list of python files that include the main program, database connection driver, and sensor drivers:

- sensorsystem.py - Main program: set sensor parameters, initiate camera module, and create processes for each sensor
- terminalmenu.py - Displays and edits sensor configurations
- db.py - Establish and close database connection
- mtdevice.py - Motion tracker driver
  - mtdef.py - Contains protocol variables for the motion tracker
  - mtnode.py - Wrapper to get messages from motion tracker as fast as possible
- gnss.py - GNSS board driver
- lcdevice.py - Load cell driver
- o2device.py - Oxygen optode driver

The procured sensors were researched before purchased, which revealed available Python drivers for all sensors, except the GNSS and camera module. The drivers are developed to output measured sensor data to a terminal, but did not have the exact wanted functionalities for database insertion. It is often experienced when reusing open-source code, where customization of the code is required for the driver to work as desired. Each of the sensor drivers are presented in corresponding chapters.

The methodology of developing a cohesive system was to test each sensor driver separately to understand the functionality. After that, the drivers were adjusted and tested to improve to the desired functionality. Before developing any software, it was essential to establish a set of features for each hardware component to ensure that no critical functionality was lost or missed. The features are described for each script in the corresponding chapters.

Sensor driver testing began during the Specialization project with the motion tracker being interfaced with the Raspberry Pi. Because of a lack of understanding of how the motion tracker driver worked and how the sensor could be configured, it was not rewritten or edited. After reviewing the sensor driver again when software development started during this thesis, it was evident that too much data was sent at one time between the motion tracker and the Raspberry Pi. That meant an excessive amount of data with too high of a sampling frequency to transmit data was sent continuously and the transmission stopped.

Reducing both the amount of data and the sampling frequency solved the issue.

Then, drivers for the load cell and oxygen sensor were downloaded, tested, and adjusted to the wanted functionality. Thereafter, the GNSS driver was constructed to extract position and time from the external receiver. When the sensor drivers had the wanted functionality, connection to the database and insertion of the measured data was implemented. Firstly, a single database connection where data from each sensor was inserted at the same time was created. It quickly enlightened a timing issue where, e.g., the motion tracker output sampled with a frequency of 20Hz, was inserted at the oxygen sensor output frequency of 5 seconds. The substantial delay between the sampled output and insertion in the database resulted in unusable data that could not capture the motion tracker's distortion in movements. Implementation of multiprocessing, which in Python are separate flows of execution, became necessary. Each sensor process is executed independently of each other, which enables sensor logging and insertion in database at different intervals. Tasks take advantage of the multiple CPUs of the Raspberry Pi. It has four processing cores that can execute different tasks simultaneously.

A separate database connection for each process was created to access the database for insertion when necessary. The timing issue that one connection presented was no longer an issue. It is important to note that multiple database connections are only possible when data insertion does not conflict. Two connections cannot access the same table simultaneously, but since the different processes insert data in separate tables, there are no conflicting issues to consider.

An important functional requirement for the system is that it initiates the software at reboot without the need of user influence. It means that the software runs in headless mode, without a monitor connected. There are multiple ways to run a program at a Raspberry Pi at startup, two of them are to use the rc.local file or cron. Both of the methods were tested separately, but the cron method was preferred because a log directory for potential errors while running the wanted file was easily accessible. The cron method utilize the editor for cron schedule expressions, crontab, which contains the instructions for the cron daemon (a program that runs a background process in a system that runs multiple processes). Even though the cron daemon enables execution of scripts at specific times, it is sufficient to execute at reboot for the system. A launcher (shell) script containing the commands for reaching (the path) and launching the main script is made executable and available in the crontab file. At reboot the crontab file will execute the launcher script which executes the specified script (in this case, the main script of the prototype software implementation).

The following chapters will present the features, functionality, and working principles of each script mentioned in the list above.

5.4.1 Sensor system main script

The main script is the software module that holds the main software functionalities. It has the overall responsibility of creating connections, setting sensor configurations by accessing modules and functions from the sensor drivers. As mentioned before, predetermining which functionalities each specific module has to meet decreases the risk of developing unnecessary and incorrect software.

Taking this into account, the following functionalities have to be satisfied by the main script:

- Start sensor logging on interrupt
- Set sensor communication parameters
- Set system clock time
- Establish connection with the database
- Set sensor configuration parameters
- Initiate camera and begin recording
- Create and start processes for sensor logging
- Insert logged data into database
- Stop sensor logging on interrupt

After the launcher script initiates the main script, a continuous while loop checks for an interrupt at the digital pin connected to the magnetic hall sensor. If an interrupt is detected, a callback function is called for initiating sensor configurations.

The sensor communication parameters were listed for each sensor in section 5.3.3. These are predetermined and cannot be changed as they are configured by the manufacturer. Therefore they are hardcoded. USB-ports for both the motion tracker and the oxygen optode can differ from what is hardcoded, therefore it is vital to have corresponding hardware connection to the defined software parameter. Without the right connections, the system will fail to read measurements from both of these sensors, resulting in a massive loss of potential data collection. The possible user error can be taken into account by implementing a USB-port reader, but it was not implemented due to time constraints.

After the communication parameters are set, the time and position must be extracted from the external receiver with a valid data sequence before the database connection is established. The GNSS board is the most crucial sensor to retrieve information from first. It is because the system time is set from the acquired UTC, which all of the other sensor measurements use as a synchronized timestamp when logging the data and inserting it into the database.

It is essential to establish a connection to the database to store the logged data. Sensor measurement readings are useless if they are not stored locally or sent to a remote surveillance station, where they can be accessed after or observed during logging. The functions from the database driver (`db.py`) creates a communication object for each of the sensors (camera excluded) for insertion of data in the database. The communication object is used to create a cursor, a handler structure for insertion or updating data. How data is inserted will be presented in the sensor drivers.

The camera module is initiated from the main script after the connection parameters are set. Given that the camera module is a Raspberry Pi product, a user guide to getting started with the module is available on the Raspberry Pi home page. It consists of a step-by-step guide that goes through the connection and control of the module with Python code. The Python library *picamera* allows the creation of a camera object with the *PiCamera* class that provides a pure Python interface to the Raspberry Pi's camera module. Because the distributor of the camera recommends the library, it is a clear choice for camera connection.

After creating a camera object, the camera can start recording to a specific Raspberry Pi storage location. It is ensured that each video file path includes the date and start time of a data logging sequence. Also, annotation with continuous updates of time passing is assured. The *picamera* module enables a function called *annotate_ text* that takes a defined string of a maximum of 255 characters and sets the text annotation for all output. Before

initiating video recording, the camera resolution and framerate are set.

There are two options regarding video formats when using the picamera class; 'h264' and 'mjpeg' (Moving JPEG). Both are video compression formats, but 'mjpeg' is a series of individually compressed pictures. Given that 'mjpeg' has the disadvantage of taking about 5-20 times the hard drive space compared to 'h264', the latter was chosen. The 'h264'-format also enables recordings in much higher resolution than 'mjpeg'.

The sensor configuration parameters are sent to each corresponding sensor to tell it which output parameters to measure and the frequency to measure at. These parameters can change every time the system starts a logging sequence, but not in the middle of logging. The parameters are extracted from a text file that is stored locally on the Raspberry Pi. It is a convenient way of storing data that change relatively often and where only the lastly stored parameters is necessary to store.

After the above functionalities are implemented, each sensor can start logging. Five processes, one process for each sensor, are created to operate simultaneously. These are independent sequences that start sensor logging, store the data to a table in the database, and stop the process if an interrupt is triggered. When the interrupt has been triggered, sensor data logging stops, and the database connection is closed.

The above description of the script functionalities compose the program flow (see Figure 5.6 for activity diagram).

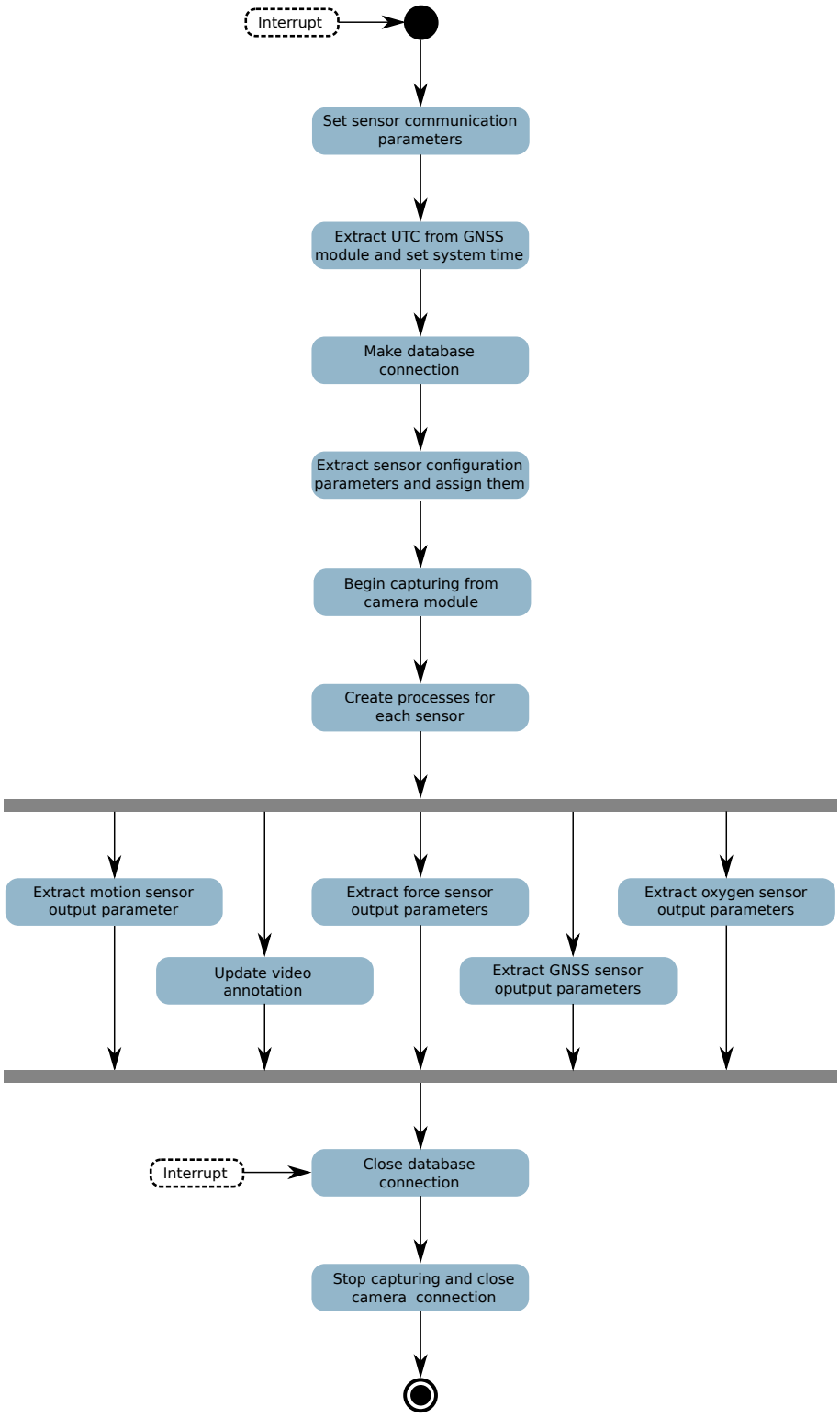


Figure 5.6: Activity diagram of main script program flow

5.4.2 Terminal menu

Inside each sensor driver are functions that utilize configuration parameters to output a specified output at a given frequency. These are often not easily accessible to a user. Therefore, a text-based terminal menu with the functionality of displaying and change sensor configurations was constructed. It requires a monitor and a keyboard connected to the Raspberry Pi to display and navigate through the menu. The simple interface uses the following set of standard keyboard inputs to navigate:

- 'q' for quitting the program
- 's' for starting the program
- 'b' to go back to previous menu
- Numbers to select menu option
- 'Enter' to verify selected configuration

In addition to the standard keyboard inputs, there is an implementation of unique character input in each sensor configuration menu to change or remove specific configurations. These are described in the menu and does not require any prior knowledge on the menu usage.

The currently configured sensor configurations are read from the same text file described in section 5.4.1. It contains four lines where each line represents configurations for one of the four sensors, GNSS board excluded.

When any sensor configuration is changed, the text file is updated. At an initial startup of the main script, the text file is read and the configurations corresponding to each sensor are set.

The structure of the terminal menu is:

- Main menu (Figure B.1)
  - Display sensor configurations (Figure B.2)
  - Choose sensor to configure (Figure B.3)
    - Motion tracker (Figure B.4)
    - Load cell (Figure B.5)
    - Oxygen optode (Figure B.6)
    - Camera (Figure B.7)
  - Start sensor system
  - Quit program

5.4.3 Database driver

A Python application can establish a connection to a SQL database by utilizing a SQL driver, e.g., *MySQL Connector* (actually MariaDB). Creation of the database connection require the code in Listing 5.1.

Listing 5.1: MySQL database connection

```
1 import mysql.connector
2
3 mydb = mysql.connector.connect(
4     host="localhost",
5     user="yourusername",
6     password="yourpassword",
7     database="youdatabase"
```

```
8 )
```

The connect() constructor creates a connection to the MariaDB server and returns a *MySQLConnection* object to a specified database (*mydb* in Listing 5.1). Simultaneous insertion of data using the same connection object is not possible. Therefore, each sensor (camera excluded) has a separate connection object to insert data in the database. The main script calls a function from the database-script, which returns the individual communication objects. These can be used further to create a separate cursor for each communication object that is used in the main script to insert data in the database.

5.4.4 Motion tracker driver

The motion tracker driver is based on a code downloaded from GitHub (`https://github.com/rpng/xsens_standalone`). It contains three separate python scripts with the property of communicating with the motion tracker and extract and output the measured parameters. The main driver is called `mtdevice.py` with a main class, which creates a motion tracker communication object. The communication object initiates serial port communication with a specified port and baud rate. Then a set of low-level and high-level communication functions are defined. They include read and write message functions and device information, configuration, and output extraction functions. The high-level utility functions include the function that reads the measurement and parses the message based on the selected output parameters. Neither of the function groups are edited as they work according to the wanted usage.

Then the main driver includes a documentation guide for standalone usage with command descriptions on how to run and configure the device. The standalone use creates a communication object and extracts, parses, and prints each sampling frequency's measured output.

All definitions for constants and messages are gathered in a separate file called `mtdef.py`. The last file is `mtnode.py`, which is a wrapper that extracts a new message from the device as fast as possible. Neither of the two files are edited.

Incorporated in the driver are specific database functions. These are developed to create a table every time the system is started and append the data extracted from every measurement sample. Because the measurement output can change between every sampling sequence, the function loops through the initial set configuration parameters and creates a table with columns based on these. The table's name includes an identifier, "Xsens", and the date that the sample sequence is taken. Equal behavior for the insertion function where each configuration parameter is looped through, extracted from the measurement message and inserted into the correct column in the table. If parameters are configured with different sampling frequencies, a lower frequency parameter will insert a *NULL*-value in the rows for the specific parameter in the unsampled rows. A field with a *NULL*-value in a SQL table is a filed without a value. Since the function loop through each configuration parameter and insert one at a time, a column designated for the sample number is added as a verifying element, ensuring correct insertion for each sample.

Two sensor system measurement sequences with equal configuration parameters on the same day would originally overwrite the previously stored data set. To avoid the problem, the registered date is also supplemented with a counter that increases if the current date is equal to the last stored date. Equal date creates a new table in the database supplemented with a version, e.g., "Xsens_2021_06_07_v2".

5.4.5 GNSS driver

As established in section 5.3.3, the GNSS board is connected to the Raspberry Pi's serial UART interface. A serial module is available for Python applications by importing the serial module and establishing a communication object as in Listing 5.2. The communication object can be used with a module function called *readline()* to extract the messages sent from the board on the transmitting UART channel.

Listing 5.2: Oxygen optode serial connection object

```
1  import serial
2
3  gnss_port = 'dev/serial0'
4
5  ser = serial.Serial(gnss_port,
6                          baudrate = 9600,
7                          parity = serial.PARITY_NONE,
8                          stopbits = serial.STOPBITS_ONE,
9                          bytesize = serial.EIGHTBITS,
10                         xonxoff = True,
11                         timeout = 0.05,
12                         writeTimeout = 0.01)
```

The output of the GNSS board is the time, the position, and the fix-related data of the receiver. It is supported by the NMEA-0183 standard, a serial communications protocol that defines how data is transmitted in predefined sentences that are broadcasted to multiple listeners. The signals from the GNSS board are binary structures and therefore decoded to a string element. Since all of the possible structures in section 5.3.3 are sent on the serial line, the preferred one is extracted while the rest is ignored. To accomplish the extraction of the one message sequence, the message structure has to be known. Thereafter, specific conditional statements can extract the wanted message.

According to the NMEA standard, the sequence has the structure as in Figure 5.7.

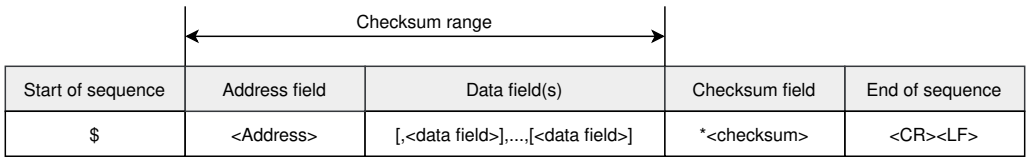


Figure 5.7: NMEA sequence structure

The sequence begins with a "$" and the address field, which defines the message type. The wanted address field is "GNRMC", where "GN" is the talker identifier for Global Navigation Satellite System and "RMC" is the sequence type identifier. It is the address field that set the condition for the extracted message structure. Then a comma (",") separates the data field (see Table 5.3 for the data fields included in an RMC sequence). Then a checksum delimiter ("*") separates the checksum from the rest of the fields. Lastly, a carriage return "<CR>" and line feed "<LF>" defines the end of the sequence.

The UTC date and time is extracted with a function assigning a set of date and time variables to the extracted values from the sequence. Equally, the position is extracted with a function and assigned to variables for longitude and latitude. These are a part of the main script, but mentioning them was reasonable in correlation with explaining the sequence structure.

5.4.6 Load cell driver

The load cell driver is based on a code downloaded from GitHub (`https://github.com/longapalooza/nau7802py/blob/master/nau7802py.py`). It is a Python library for the ADC (NAU7802) using the I2C interface, developed from the manufacturer's Arduino library.

Firstly, multiple register definitions are used in the code to configure the ADC. These are defined based on the ADC's datasheet. The rest of the driver is the main class for extracting data from the specified I2C address with corresponding operation and configuration functions.

A constructor for communication with an I2C device is created as in Listing 5.3 with the Python module *smbus*. Firstly, a method called *__init__*, is created. The method is reserved in a Python class and is also known as a constructor. The first parameter in the constructor is *self*, which represents the instance of the class. Further use of the *self* keyword in other class methods access the attributes of the class.

The *smbus*-module enables the definition of the distinct I2C peripheral that the device is connected to (i2cPort = 1 in the listing). In addition, the device address, which the manufacturer predefines, is hardcoded (0x2A). If multiple I2C devices with the same address are connected to the same bus, a different device address must be configured.

Listing 5.3: Load cell I2C constructor

```
1 import smbus
2
3 def __init__(self, i2cPort = 1, deviceAddress = 0x2A, zeroOffset = False,
4                  calibrationFactor = False):
5         self.bus = smbus.SMBus(i2cPort)
6         self.deviceAddress = deviceAddress
```

The main script uses four functions in the load cell driver to configure the ADC and then extract the bit value representing the resistance in the strain gauge. The function *begin* check the communication with and initialize the sensor. If the connection is invalid (returns FALSE), a red light appear on the LED board and hardware wiring should be checked. Then a function sets the sensor parameters gain, low-dropout voltage, samples per second, and ADC channel from the text file containing sensor configurations. After the parameters are set, the sensor is calibrated by setting a status bit to know when calibration is complete. A process for reading the force measurement is called from the main script if a function called *available* returns true. The function reads the "CR" (Cycle Ready) bit in the control register and enables measurement extraction if the bit reads boolean logic true.

According to the datasheet, there are two options for reading the complete 24-bit ADC conversion result. Either using an I2C burst read of 3 bytes or an I2C single read. The downloaded code had used option 1, which remained unchanged. As there are minor differences between the methods and they require the same amount of coding, the choice between the two options are indifferent.

The function *getReading* returns a 24-bit reading of the voltage representing the tension in the rope being measured. The weight is extracted by calibrating the reading with a calibration factor (see for further explanation Section 6.1.4).

5.4.7 Oxygen optode driver

The oxygen optode is based upon a driver that is downloaded from GitHub (`https://github.com/stanleylio/fishie/blob/master/drivers/aanderaa.py`) with the corresponding parser for the specific sensor version (`https://github.com/stanleylio/fishie/blob/master/drivers/aanderaa_4531d.py`). The file has a license that states that all rights are reserved to the creator. Therefore permission to copy and modify the code was asked directly to the creator by e-mail and granted for usage in this thesis.

Since the sensor driver is designed to extract data from four different oxygen sensor versions from Aanderaa, it is generic. It establishes a serial connection object with the same method as the motion tracker, but port "/dev/ttyUSB1" instead. By writing the encoded version of the string message "\r\n set interval (f)\r\n" to the sensor, the message sample interval is set. The interval (f), is a float number with the specified interval in seconds. The "\r" is a Carriage Return (CR), a control character used to move a cursor to the beginning of the line without advancing to the next line. The "\n" is a Line Feed (LF); it moves the cursor down to the next line without returning to the beginning of the line. The two combined (CR+LF) move the cursor to the front of the line and down to the next line. The command creates a sequence of lines on the serial output, read by using the serial object with the serial module function "readline()". For a readable output, the binary read line has to be decoded. After it is decoded, the message has the following structure:

MEASUREMENT	4531	1559	O2Concentration[uM]	f	AirSaturation[%]	f	Temperature[Deg.C]	f

Figure 5.8: Message sequence from Oxygen Optode 4531D after decoding

The grey fields are constant and do not change between every sample. The white fields are the measured values for oxygen concentration, air saturation, and temperature. The first field after the keyword MEASUREMENT, is the sensor version (4531). The third field is the serial number of the specific sensor (1559).

The message is a string that has to be parsed to extract the wanted numbers. To do this, the "match" function from the Python module that implements regular expression operations (*re*). A predefined string is created to match the read serial line from the sensor that finds the wanted numbers and returns them as a list with four fields.

The driver also includes a function for inserting the data in a database table. It extracts data from the list after parsing and assigns the four different values to a separate variable. These variables are used to insert the in the corresponding table column with the current timestamp.

5.5 Prototype

As mentioned earlier, most of the components are not resistant to water and require an enclosure since the sensor system will most certainly be partly or fully immersed in water for time periods through a crowding process. This means that components that are not enclosed in a watertight environment will be exposed to water. Only the oxygen optode and load cell is certified to withstand water immersion, which means that 83% of all system components needs to be in the enclosure. It is therefore viewed as a critical step of finishing the prototype. For the system enclosure to float on water, it needs to be attached to a buoyant object that can hold the weight of the system.

This chapter will explain the procedure of designing and implementing both the enclosure and the buoyant object, called a flotation device.

5.5.1 Watertight enclosure

An acrylic cylindrical tube with the dimensions 298.45mm in length and 76.2mm in inner diameter with a 10mm thick cylinder cast is used as an enclosure. One end is a clear acrylic dome cap, suitable for placing the camera with direct view out from one end. The other end is an aluminum end cap with five holes, suitable for cable threading for the two sensors that are placed outside of the enclosure. Both of the end caps are inserted in the tube on opposite sides and can be removed for easy access to the system. In addition, three o-rings on both sides of the enclosure keeps water from entering the tube, making the system watertight.

The methodology of physical prototype design was to complete the hardware and software design before insertion in the acrylic cylinder. It was done to ensure a working prototype prior to mechanical integration. However, the cylinder dimensions portrayed difficulties when all of the system components was to be inserted in the tube. The limited space available inside the tube lead to optimization on usage of space and some rewiring and resoldering was necessary to fit every component inside the enclosure.

When designing a prototype it it important to consider the component functionalities and make sure that they not depleted. Therefore a set of critical requirements for placement of components was established:

- Powerbank: the power button should be reachable from the cable end cap, when it is detached from the enclosure, to ensure easy startup of the system.
- Camera: camera view facing down towards the dome end of the tube below the enclosure towards the sweep net and potential fish is in field of view.
- Magnetic hall sensor: the magnet sensitive plate placed close to the cylinder wall to ensure that the magnet can trigger the interrupt from outside of the enclosure.
- RGB LED board: the LED placed out towards the cylinder wall to ensure that the light can be seen from a distance.

In addition to the critical placement requirements, it is beneficial that the motion tracker is positioned with one of the three planes in the Cartesian coordinate system (xy- or xz-plane) perpendicular to the earth's surface that does not cause Gimbal lock. This is because analysis of the acquired rotation and orientation data after a crowding process will be easier to interpret. A plane will then represent the water surface and deviation from that plane represents how the sensor system is oriented in the space it occupies.

A suggestion of component placement (see Figure 5.9) was introduced to the institute's mechanical workshop at NTNU, as a guide to insert all the components into the enclosure, as well as the importance to implement the critical placement requirements.

A sketch of a prototype will often deviate from the finished product. Reason being misjudgement regarding component placement possibilities, as well as omitting of additional wiring in the sketch. It results in necessary adjustments in positioning each component, both to make all the components and the required wiring fit in the enclosure.

Describing the approach of composing the prototype is easier if the dome end of the enclosure is viewed as the bottom of the prototype, while the wire end is the top.

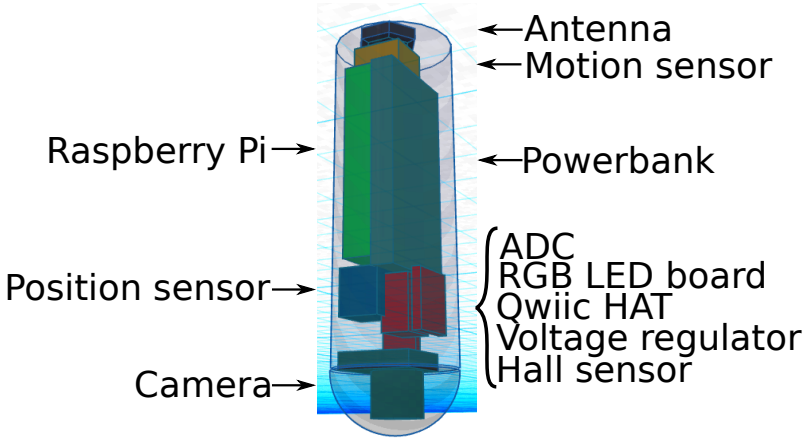


Figure 5.9: Component placement suggestion in the watertight enclosure

The camera is positioned to capture the view directed down from the bottom, out of the clear dome cap. It is mounted to a plastic circular plate that fits at the bottom of the cylinder between the tube and the dome cap. Above the circular plate is a vertically mounted metal plate that holds most of the components, except the camera, the powerbank, the position sensor, and the antenna. The components are screwed on the plate to ensure minimal movement to increase robustness when testing the prototype during a crowding process. The powerbank is positioned on the other side of the metal plate to which most of the components are screwed onto. The GNSS sensor is loosely placed to fit between the enclosure cast and the powerbank. At the wire end of the enclosure a plastic end cap is created to use the top space of the enclosure more efficiently. It replaces the aluminium cap mentioned above, where wire holes were placed towards the center of the plate. The plastic plate has two cable glands, one for the oxygen optode cable and one for the load cell cable. These are positioned towards the edge of the plate to fit the antenna underneath. A similar plate to the camera mounted plate is mounted at the top end of the enclosure below the end cap. It has a cutout where wires connecting the oxygen optode and the load cell to the system is led through. The antenna is mounted on the remaining part of the plate (see Figure 5.10.

5.5.2 Flotation device

Two floaters from Polyfrom will ensure that the enclosure and load cell are fastened to the top line of the sweep net and stays buoyant in the water. The enclosure fits in the middle of the flotation device to the right in Figure 5.11 which is fastened with a hose clamp on the incision on the outside of the floater. It is fastened to a rope with cable ties The load cell fits inside the flotation device on the left in Figure 5.11, where the top rope of the sweep net is inserted into the slit in the floater and goes out from either side. This has multiple advantages, e.g., the net line does not have to be cut and the system can be placed at multiple different places on the line.

As the enclosure is a made for the specific purpose of the sensor system and does not have standardized dimension, a unique floater had to be developed. This was done by customizing an existing floater to the specific enclosure and load cell dimensions.

The floater encapsulating the load cell was created to ensure fish and net safety. Because of the sharp edges and protruding parts of the component, it was introduced as a necessity

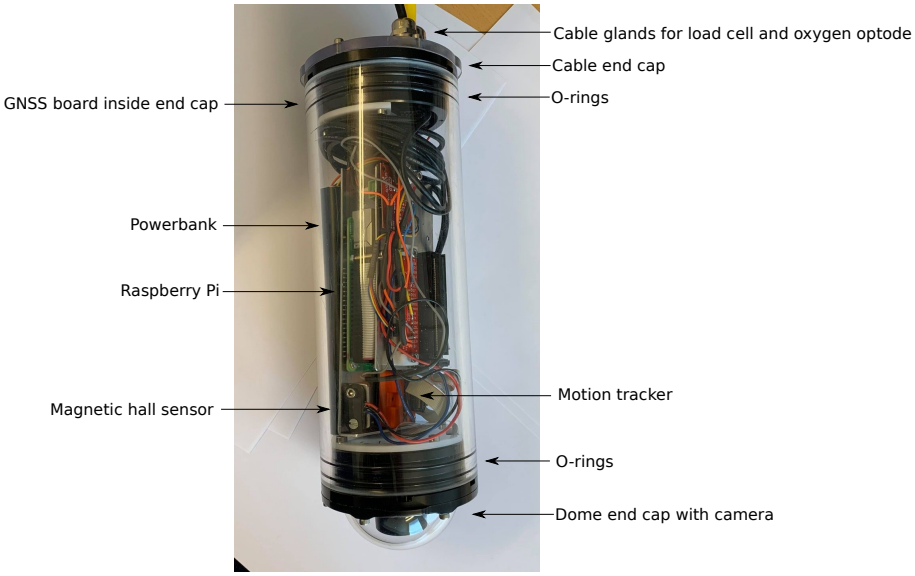


Figure 5.10: Finished prototype (enclosure)

Figure 5.11: Flotation devices for load cell to the left and for enclosure on the right

to encapsulate the load cell. The protecting encapsulation protects the fish from damaging the skin if there is contact with the device.

Chapter 6

Testing of prototype

So far, assumptions have been made regarding the system response (measurement response from each sensor combined) from external influence by fish and operational equipment during a crowding process. As stated in the thesis project description (Appendix A), tests in different scales was to be planned and carried out. Due to time constraints, the specific season that the test would be conducted, and the insufficient capacity of the test facility, a full-scale test during a crowding process was not possible.

However, two types of tests in smaller scale was planned and carried out. There are two primary purposes of conducting smaller-scale tests before a full-scale test. Firstly, the system must fulfill the functional requirements before deploying it during a crowding process to avoid exploiting time and resources. It is essential to circumvent system malfunction or deficiencies that could cause a stop or delay of a crowding process because of the criticality it would produce. Secondly, expected and assumed external influence on the system can be tested, which produces specific results. The results are comparable to full-scale data results acquired during a crowding process test.

This chapter presents two different small-scale tests. The system verification tests verify the working principles of each sensor and provide a basis for further testing. The first verifying test on sea tests the entirety of the system in an environment that replicates a crowding process.

Analyzing the data from each test is difficult within the local database and on the Raspberry Pi. Sensor data is therefore extracted from the database to csv-files and read into MATLAB at a stationary computer. Each column in the database is stored in separate arrays plotted in two-dimensional line plots suitable to interpret sensor responses.

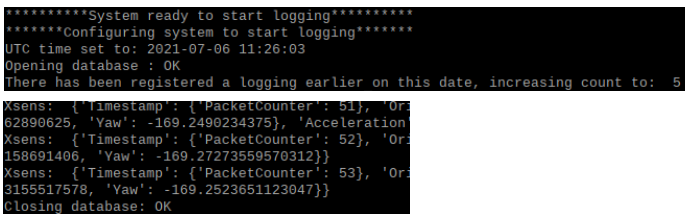
6.1 System verification tests

The purpose of system verification tests is to verify the system functionalities in correlation with initiation and stop procedures of the system and correct logging of data from each of the sensors.

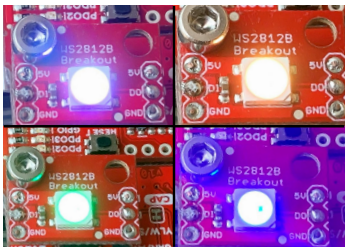
6.1.1 Initiation and stop procedure

The initiation procedure test aims to verify that the system starts, sets all of the sensor parameters, and retrieves sampled data from each sensor simultaneously. The procedure was presented in Section 5.4.1 as a step-by-step startup sequence (see Figure 5.6). A successful initiation can be observed in two ways depending on how the system is used. On-desk experiments with a screen connected to the Raspberry Pi enables readable string verification outputs in a terminal window. A full-scale test does not allow for a connected screen and requires an alternative method of verification. The RGB LED board serves as an alternative verification method. The different steps throughout the startup sequence are differentiated with light signals (pulses and colors).

The first step is to power the system on by turning the powerbank on and wait for the launcher script to start up. When the program has started successfully, a string output displays "System ready to start logging" (see the top figure in Figure 6.1a). The LED on the board will flash three times and stay white until further operations are made on the system (see the top left figure in Figure 6.1b). A magnet held close to the magnetic hall sensor initiates the configuration procedure, indicated by the string output "Configuring system to start logging" and an orange light on the LED board. After the configurations are set, the system starts to log data from the sensors. It is indicated by the printed output from each sensor and a green light on the LED board. Lastly, a magnet switch held close to the magnetic hall sensor again stops logging and closes the database connection. A string output "Closing database: OK" and a blue blink before the LED board turns off indicates a successful system stop.



(a) Readable string verification output. Beginning of logging at top, end of logging at bottom



(b) Visual verification from RGB LED board

Figure 6.1: Readable and visual startup and stop verification from system

Both the string output and LED indicating method were verified during on-desk tests. However, string outputs are unavailable when the system has an initial startup after a manual reboot of the Raspberry Pi. It is because the launcher script does not initiate a terminal window with the string outputs. Instead, the system's main script must be run with a command in the terminal window to portray the verifying string outputs or the crontab log file can be inspected.

The RGB LED has a slight fault that reduces the overall performance of the system during on-desk testing. When the system is manually rebooted from the Raspberry Pi OS desktop, it initiates the first LED indicator (three white blinks and steady white) but does not change according to any other step of the startup sequence. However, the system still initiates configuration and logging even though the LED indicator indicates otherwise. Verification of a working system was ensured by observing the crontab logfile and the inserted sensor data in the database. Given that the scenario is an on-desk fault that does not appear in procedures that replicate startup before a full-scale test, it is neglected.

6.1.2 Orientation and rotation

It is assumed that measured data from a crowding process would be difficult to associate with specific distortions in orientation and rotation without a reference to or knowledge about how the motion tracker output relates to physical movements. Therefore, particular movements expected to influence the orientation and rotation of the system during a crowding process are tested and interpreted below.

Euler angles are chosen to represent rotation and orientation of the device because of the easily understandable decomposition of rotations into individual degrees of freedom about the x-, y-, and z-axis (Roll, Pitch, and Yaw). Although the disadvantage of Gimbal lock can occur when utilizing Euler angles, the floater that holds the enclosure part of the system afloat on water ensures that none of the rotation axes can interlock and cause one degree of freedom to be lost.

Isolated Euler angles

Each Euler angle was tested separately to ensure a correct understanding of the motion tracker's orientation and rotation. Before conducting the test, it was essential to determine the reference frame of the motion tracker to understand how the system is physically oriented and respond to different movements. The motion tracker follows the right-hand rule and is oriented in relation to the sensor as in Figure 6.2.

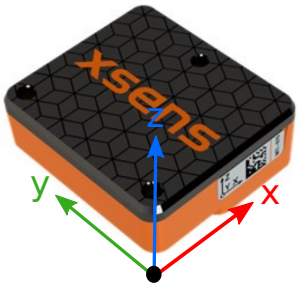


Figure 6.2: Motion tracker internal coordinate system in relation to sensor orientation

The method of testing each rotation for the Euler angles Roll, Pitch, and Yaw was to rotate about the specific axis that correlates with the Euler angles, with the two remaining axes forming a plane perpendicular to the particular axis held still. In addition, all of the angles were ensured to be zero prior to rotating about an axis. Firstly, a test of response on the Euler angle Roll with the following sequence:

1. Hold the sensor completely still for 30 seconds
2. Rotate counterclockwise one round about the x-axis
3. Hold the sensor completely still for a few seconds
4. Rotate clockwise one round about the x-axis
5. Hold the sensor completely still for 30 seconds
6. Rotate clockwise and counterclockwise in an oscillating motion for 60 seconds
7. Hold the sensor completely still for 30 seconds

There are clear distinctions between when the system is held still and rotating about the x-axis (Roll in Figure 6.3). Given that the sampling frequency is set to 20Hz, the measured rotation follows the movements with high accuracy. One rotation about the x-axis is a 360° rotation, but it is represented as 0° to 180° and -180° to 0°. The characteristics of a complete rotation increase to 180° and before shifting to -180° and decrease to 0°. After

that, the same characteristic is observed but reversed for the clockwise rotation.

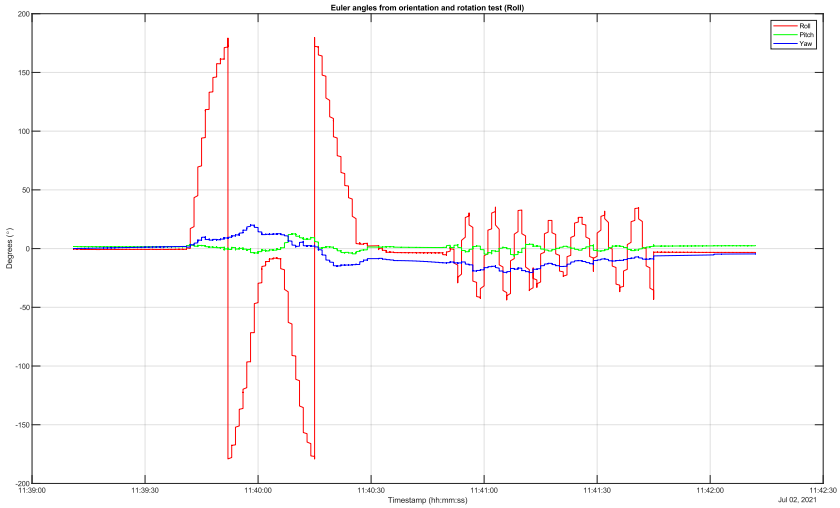


Figure 6.3: Euler angle response during the land-based test of Roll

The same test sequence was conducted for both Pitch and Yaw to compare all Euler angles' accuracy and continuity.

Figure 6.4 contains two different plots. One with all the Euler angles in one plot, and the other with only the Pitch. It is presented in this way to enlighten both that the Roll and Yaw are affected when the system only rotates about the y-axis. It is because of Gimbal Lock. When the second rotation, in this case, Pitch, reaches a rotated degree over 90° or below -90° one degree of freedom is lost (the x- and z-axis interlock). It can be observed from the top plot of Figure 6.4 where Roll and Yaw do not stay constant at a zero degree rotation even though the system is rotated about the y-axis only. The bottom plot of Figure 6.4 portrays the characteristic of Pitch, which resembels to the response of Roll on Figure 6.3.

Lastly, a similar response is observed as rotation about the z-axis (Roll in Figure 6.5), confirming continuity in rotation about each axis.

The small step-by-step increments in plotted degrees (y-axis of the plots above) in relation to time (x-axis of plots above) enlightens that the sampling frequency is too low to capture every change in rotation. However, the characteristics are of such high accuracy that any distortion from the initial frame is assumed to be easily distinguishable on orientation plots. The following test will verify or falsify the assumption.

Combined Euler angles

After each Euler angle was tested separately, expected movements caused by external effects during a crowding process were tested and analyzed. The same internal coordinate system in relation to sensor orientation as in Figure 6.2 still holds, and the system follows the right-hand rule.

How the motion tracker is placed inside the enclosure determines the orientation of the internal coordinate system, also called the reference frame. Firstly, the motion tracker was mounted in the enclosure where the yz-plane was parallel to the earth's ground and

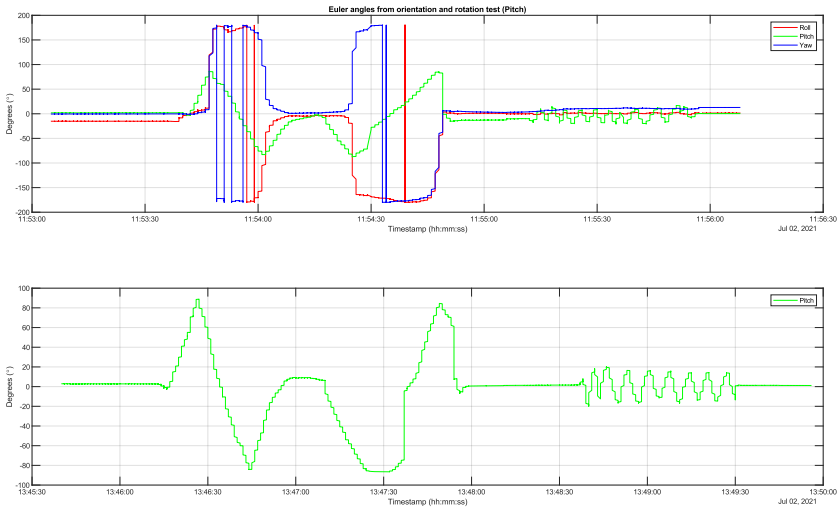


Figure 6.4: Euler angle response during the land-based test of Pitch

the x-axis pointing upwards. The orientation is achieved by rotating the device 90° in a clockwise direction about the y-axis (-90°). Since the Pitch was at the boundary of causing Gimbal Lock (the x- and z-axis interlocked), the reference frame was changed. Instead, the motion tracker was mounted with a 90° counter-clockwise rotation about the x-axis and a 45° counter-clockwise rotation about the z-axis so that the reference frame is as in Figure 6.6.

After establishing the reference frame, a specific test sequence (the list below) was conducted.

1. Hold the sensor completely still for 1 minute
2. Move system up and down (5cm distortion in both directions from origin) along the x-axis for 1 minute
3. Hold the sensor completely still for 1 minute
4. Sporadic collision that moves the system along the z-axis (5cm distortion in positive direction with momentarily relocation to origin) for 1 minute
5. Hold the sensor completely still for 1 minute
6. Sporadic collision that moves the system along the 45° angle between both the y-axis and z-axis (5cm distortion away from origin with momentarily relocation back to origin) for 1 minute
7. Hold the sensor completely still for 1 minute

Data from the motion tracker is stored in a database and can be retrieved to a .csv-file. Each sample is extracted from the file in MATLAB, and the degree of rotation for Roll, Pitch, and Yaw is plotted on the y-axis in relation to the sample's timestamp on the x-axis. Figure 6.7 is the plotted response from the sequence described above. The first minute of the plot portrays the initial orientation of the device compared to the reference frame. As expected, the initial orientation is characterized by 90° rotation about x-axis (Roll), approximately 0° rotation about the y-axis (Pitch), and a 45° rotation about z-axis (Yaw). The degrees are considered set points for each angle.

Movement up and down along an axis (in this case, the x-axis) should not affect Roll, Pitch,

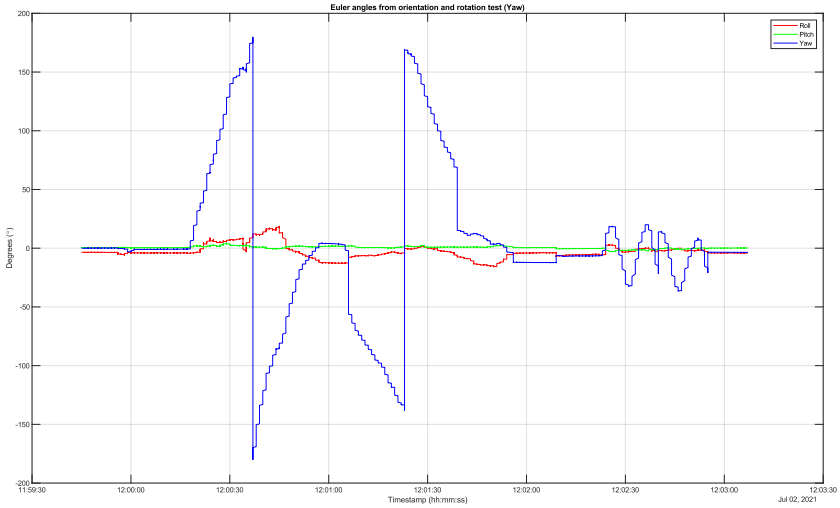


Figure 6.5: Euler angle response during the land-based test of Yaw

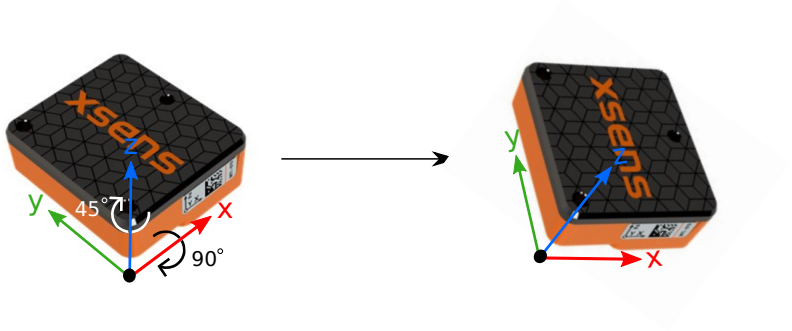


Figure 6.6: Motion tracker internal coordinate system in relation to sensor orientation with a 90° clockwise rotation about the x-axis and a 45° clockwise rotation about the z-axis

or Yaw significantly. Some variations will occur because of how the test is conducted, where the system was held by hand, which will cause some measurement error. It is confirmed in Figure 6.7 where there is a slight deviation from set-point for both Roll and Yaw, but not more than 7°. The periods with no movement are easily distinguishable from the moving periods, especially for Pitch. The periods with sporadic collision are very prominent for orientation about both Roll and Pitch, where the amplitude of the oscillating rotation is noticeably higher than the periods without sporadic collision. Results from further testing in larger scale and during a crowding process will focus on observing such differences.

6.1.3 Position of predetermined area

The position of the system is constrained to the area inside the fish cage during a crowding process. The floating collar on the surface - a circle of 50 meters in diameter, confines the area. As mentioned before, the position of the system in relation to other sensor data is essential to verify or falsify if, e.g., measurements of orientation and rotation should be used to determine the risk level for fish during a crowding process. Given that the system is constrained to a known area, the test is performed at an approximately equally sized

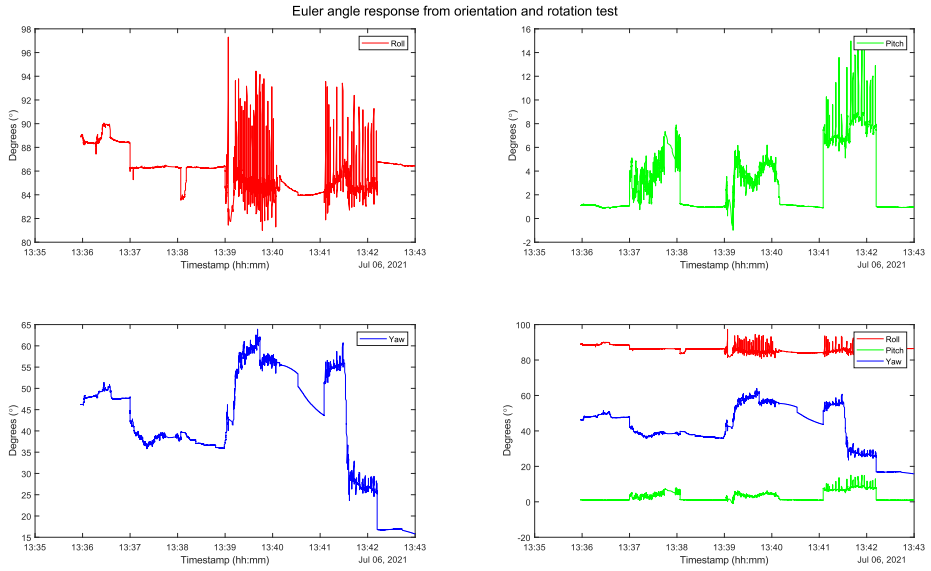


Figure 6.7: Euler angle response during the land-based test of combined Euler angles

measurement area.

A grass field with natural boundaries (road surrounding the area) outside NTNU Gløshaugen (45x70m) was used as a measurement area. The system was started and moved in a predetermined pattern. The logged position would reveal if the system position followed the actual movement. The logging began at point 1 in Figure 6.8a, then moved to point 2, 3, 4, and 1 again (in a rectangle around the grass field). After that, another round in the same patten before the system was moved to each of the points 5 up to 12 in ascending order following the marked lines.

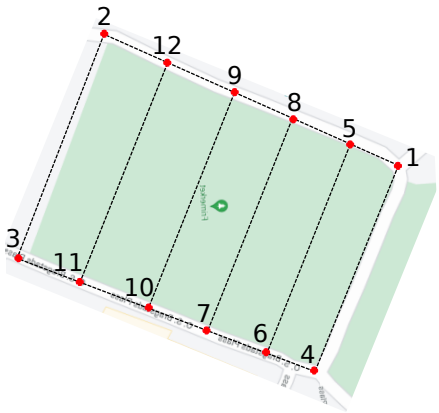
After the test, the logged positional data was retrieved to a .csv file and plotted in MATLAB. The method for plotting was to import the file, create two separate arrays of latitude and longitude values, then plot the latitude and longitude values in a two-dimensional plot. Figure 6.8b is the result of the position test. The first and last registered position is marked with a red point and timestamp (first: 08-Jul-2021 08:11:40 at latitude 63.2514 and longitude 10.2418, last: 08-Jul-2021 08:24:30 at latitude 63.2511 and longitude 10.2420).

Compared to the predetermined pattern in Figure 6.8a, the measured position is very accurate. Given that the measurement area represents the equivalent size of a fish cage and portrays the system's precise position, it is expected to measure the correct system position during a crowding process.

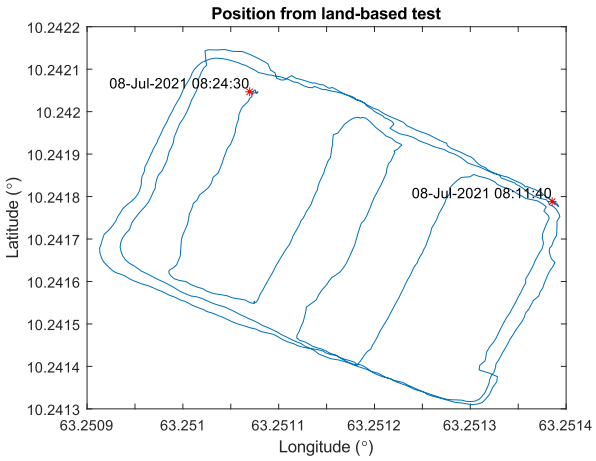
6.1.4 Tension in rope

The load cell will be mounted on the top rope of the sweep net, where tension is applied with a mechanical winch. The applied tension has an unknown value, but 2,000kg, which is the maximum reading range of the load cell, is assumed to be an acceptable range (must be verified during a full-scale test).

Test of the load cell was conducted at the institute's mechanical workshop at NTNU. A rope of 1.5cm in diameter was fastened to a fixed point at one end and attached to a weight



(a) Pattern of measurement area and route during position test



(b) Plot of measured position during land-based test

Figure 6.8: Position test route pattern and measured position during land-based position test

scale on the other. The weight scale was a handheld digital scale with a measurement range of 0-50kg. The load cell was tightly secured in between the two points. By winching the rope tighter, the measured weight on the scale increased.

The digital bit value from the ADC was read and stored in the database. Since the bit value does not represent a unit of measurement (e.g., weight or voltage), it was necessary to calibrate it with a calibration factor. The method of the calibration was presented in Section 5.3.3, but calculations with exact numbers are derived below.

Bit values that are necessary for calculating the calibration factor are:

- Bit value at zero tension: 29.660
- Bit value at a known tension (50kg): 10.089

$$\begin{aligned} Calibration\ factor &= \frac{Bit\ value\ at\ zero\ tension - Bit\ value\ at\ known\ tension\ above\ zero}{Tension\ above\ zero\ in\ kg} \\ &= \frac{29.660 - 10.089}{50} \\ &= -391.46 \end{aligned}$$

The calibration factor is calculated to be -391.46, which has to be used to calculate the tension of every sampled measurement with the following equation:

$$Tension = \frac{Measured\ bit\ value - Bit\ value\ at\ zero}{Calibration\ factor}$$

Each bit value for the measured tension applied on the load cell is stored in the database with a timestamp. The values are extracted to a .csv file and loaded into MATLAB, where the timestamp and measured bit values are stored in two separate arrays.

The test followed the following sequence:

1. Tension increased to approximately 50kg (measured by weight scale)
2. Held the tension at approximately 50kg for 20 seconds
3. Decreasing tension to 0kg
4. Tension increased to approximately 25kg
5. Held the tension at approximately 25kg for 20 seconds
6. Decreasing tension to 0kg
7. Tension increased to approximately 50kg
8. Held the tension at approximately 50kg for 20 seconds
9. Decreasing tension to 0kg

Figure 6.9 is the result of the sequence above. The timestamp is plotted on the x-axis while the calibrated tension from bit values are plotted on the y-axis. The maximum deviation from the measured tension and the weight scale is approximately 6kg. Depending on the tension applied on the sweep net rope, it can constitute a minimal or decisive difference in the measurement. The deviation for substantial tension above 50kg does not affect the usability of the measurement. Tension below 50kg will reduce the usability substantially with the deviation because differences in measured tension equal the deviation with a higher percentage.

The tension can also be converted to force in Newton by multiplying the tension in kg with 9.80665, but the result is easier to interpret with values in kg (50kg * 9.80665 = 490.3325N).

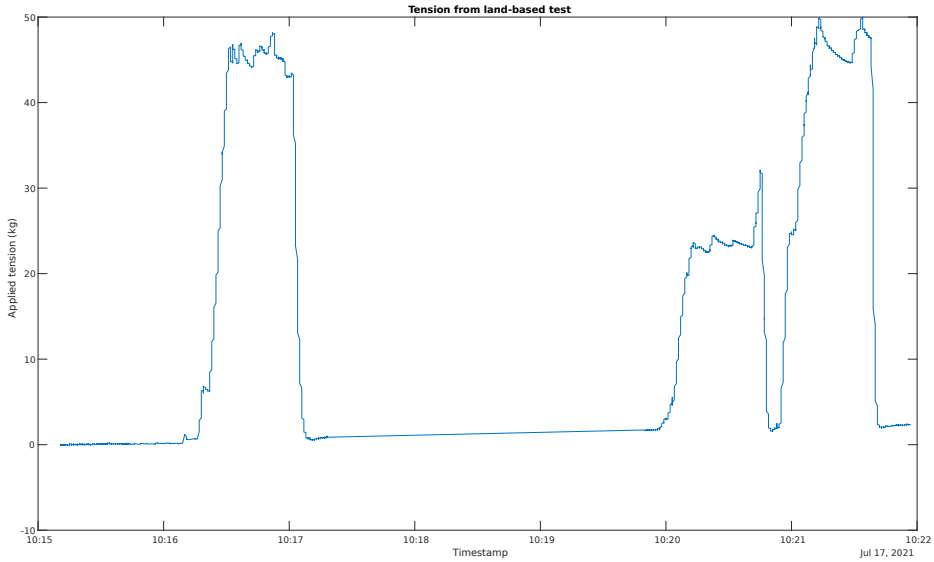


Figure 6.9: Tension response from land-based test

The load cell measured the tension compared to the weight scale accurately because of the calculated calibration factor and the linear response of applied tension on the rope. The tension applied in a rope during a crowing process is assumed to present similar accuracy even though the exact value for weight is known.

6.1.5 Water environment variables

The oxygen sensor will be placed approximately 2-3 meters below the enclosure part of the system, meaning that it is fully submerged in water throughout an entire full scale test. It measures the temperature, oxygen concentration, and air saturation in the water.

The method of testing each of the parameters was to submerge the optode in a bowl of fresh tap water (bowl A) with a known temperature of 31°C, measured with a thermometer. Given that the measurement range of temperature is 0-30°C, it was a goal to keep the temperature within the temperature range. After the optode was submerged in the water, colder fresh tap water from a bowl (bowl B) with a temperature of 12°C was added to bowl A in six approximately 80ml intervals. The cold water was also measured with the oxygen optode, which measured 16°C.

The temperature is expected to decrease with every interval of adding water from bowl B to bowl A. The top plot in Figure 6.10 is the temperature measured in °C as a function of time. As expected, there is a decrease in temperature with every interval of adding water from bowl B. The temperature measured by the thermometer was consistently 2-3°C lower than the oxygen optode measured (also consistent with the measurement of cold water from bowl B).

Each interval is also prominent for measured air saturation and oxygen concentration, but the values increases in contrast to the temperature.

The first sampled air saturation is 101.531%, and it increases with every interval of adding cold water from bowl B (middle plot in Figure 6.10). It means that the water is slightly above the equilibrium point for gases in water. When the water is 100% saturated, it holds as many dissolved gas molecules as possible, and the percentage of oxygen in the water is equivalent to the percentage of gas in the atmosphere. The water can increase to higher than 100% saturated when there are rapid temperature changes. It is assumed to be 100% at depths where the oxygen optode will be placed in ocean water. However, increased respiration rates and oxygen use from fish in a crowding situation can decrease the percentage. Such values are essential to observe during a full-scale test of the system.

Two bodies of water that are both 100% air-saturated can have different concentrations of dissolved oxygen. It is because the oxygen concentration varies depending on temperature, pressure, and salinity. As observed in Figure 6.10, the concentration of dissolved oxygen will increase when the temperature decreases. Since the test was conducted in fresh tap water at a depth of 10-15 centimeters, the salinity and pressure do not affect oxygen concentration. The initial oxygen concentration is 241µM, or 7.715mg/L (calculated with the relationship that 1mg/L = 31.25µM), which is expected at the given temperature with zero salinity. Note that when the concentration is measured in water with salinity and pressure higher than zero needs to be compensated.

Compared to the actual measured sensor response, the expected measurement response confirms that the optode responds to changes in the environment it measures. Neither the air saturation nor the oxygen concentration is verified with another measurement unit. However, the measured values cohere with the changes applied to the measured environment. It was expected to observe an increase in air saturation when additional water was supplied to bowl A, because the added water carries oxygen and supplies the water in bowl A with oxygen from the atmosphere. Likewise, the oxygen concentration is expected to increase because of the additional water supply.

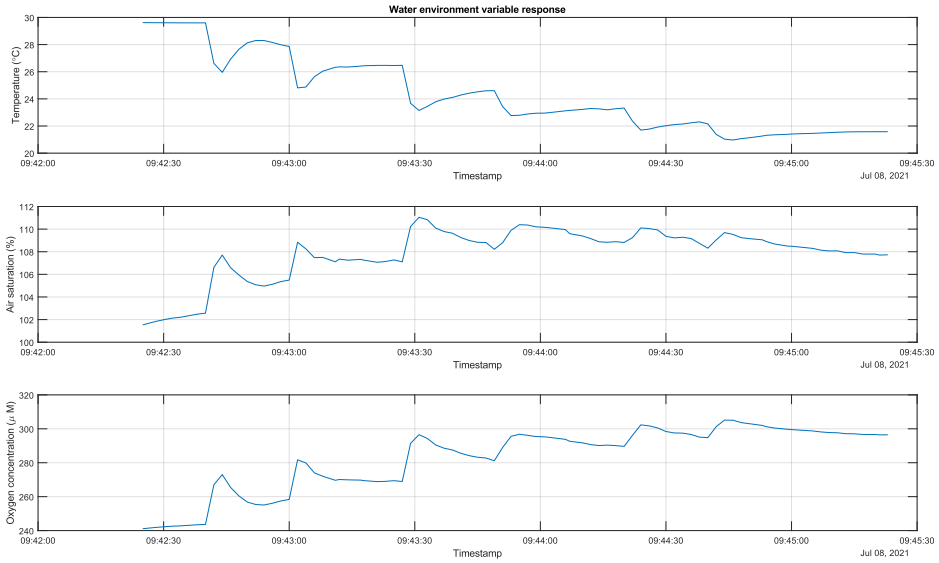


Figure 6.10: Water environment variable response from land-based test

6.1.6 Camera view

The camera is a visual representation of the data from fish behavior below the enclosure. Logged data from the motion tracker, load cell, or oxygen optode implying high or low activity amongst fish in the sweep net is conceivably verified by the observed activity on video recordings.

The field angle of 63° defines the area visible on video recordings. At a distance of 1 meter from the camera, a circular field with a diameter of 1.22m would be visible on a recording.

A downloaded driver (V4L2) that access camera features and provides a live camera view was used to test the camera visibility and quality. The back focus and aperture were manually adjusted with the lens adjustment rings for a clear view. Whereas the correct settings are easily configured manually during an on-desk test, it presents a challenge when submerging the enclosure in water since auto-focus is unavailable with the specific camera (additional components for auto-focus is possible). Since the available light passing to enter the camera is limited below water, it is assumed that the aperture setting must be higher than tested in the land-based test.

An object was placed in front of the camera while recording at a distance of 26cm from the end of the lens and passed the entire field of view (Figure 6.11). The object was barely observed within the field of view at 16cm away from the middle of the lens (Figure 6.11a), which corresponds to calculations using the Pythagoras formula (distance to object from lens*tan(field angle/2)).

Water absorbs different wavelengths of light at different depths. Wavelengths with long waves and low energy will be absorbed first, which reduces the appearance of colors on the visible light spectrum. At a depth of 1 meter, the color red will begin to disappear. However, since the camera will be placed directly below the water's surface, it is assumed to capture the colors slightly muted, depending on the opacity and quality of the water.

(a) Camera view with object at the outer edge of the view

(b) Camera view with object the middle of the view

Figure 6.11: Camera view in land-based test

6.1.7 System operational life

Before system testing of a larger scale was to be carried out, the system operational life was tested. It was evident that the expected order delivery of the voltage regulator would have a belated arrival for testing beyond the already conducted land-based tests. The oxygen optode was removed from the system as it is unusable without the voltage regulator.

A software implementation to log the current time in a text-file was supplemented in the already existing software (and removed after use). A separate process wrote the current time every minute to the text-file. The last registered timestamp in the text file indicates when the capacity of the powerbank was unable to supply the system with power.

The first registered timestamp at 09:30:10 and the last registered timestamp at 16:51:27 conveys an operational life of 7 hours and 21 minutes. The system's operational life is within the functional requirement of operating through an entire crowding process (2-3 hours).

6.2 First verification test on sea

As previously stated, a full-scale test during a crowding process was unachievable towards the end of the thesis. The purpose of a verification test on the sea is to test the sensor system in an environment that replicates the crowding process. In contrast to the land-based tests, where sensor functionality and data logging correctness were the primary goals, the verification test aims to obtain data from events and movements that resemble those during a crowding process and ensure a watertight system that can operate in water.

Considering that the verification test is performed on the water with all the sensors logging simultaneously, it is as close to a full-scale test that was possible within the time frame of the thesis.

The test was conducted in Trondheimsfjorden, right outside of Børsa. Preparing for the test meant checking if the system logged from all sensors simultaneously, correct startup procedure, and charge the powerbank before departing to Børsa.

The test setup was assembled at arrival to Børsa and constructed to replicate an excerpt of the top rope of a sweep net (without the net). Figure 6.12 shows the setup where the enclosure is placed in the middle of the testing rope with surrounding floaters that simulate floaters on an actual sweep net. A test of the floating capabilities of the setup revealed that the enclosure tilted on its side. Therefore, a weight was mounted below the dome end of the enclosure to steadily position it in an upwards direction (cable end of the enclosure pointing up towards the sky). The weight was positioned directly below the camera view, but video recording issues during startup via the shell script resulted in no capturing of visual data during the test (the weight would have disrupted the view). The load cell is of significant weight (2,14kg), and required floaters on both sides to keep it afloat and not drag the enclosure down and disturb orientation measurements. Since the voltage regulator did not arrive in the mail before testing, the oxygen optode could not be supplied with the required voltage and was disconnected from the system during the test. The optode is mounted to the lid of the enclosure with a PG cable gland which keeps the system watertight and cannot be removed. The oxygen optode is therefore mounted on the rope as seen on Figure 6.12, but unusable for logging purposes.

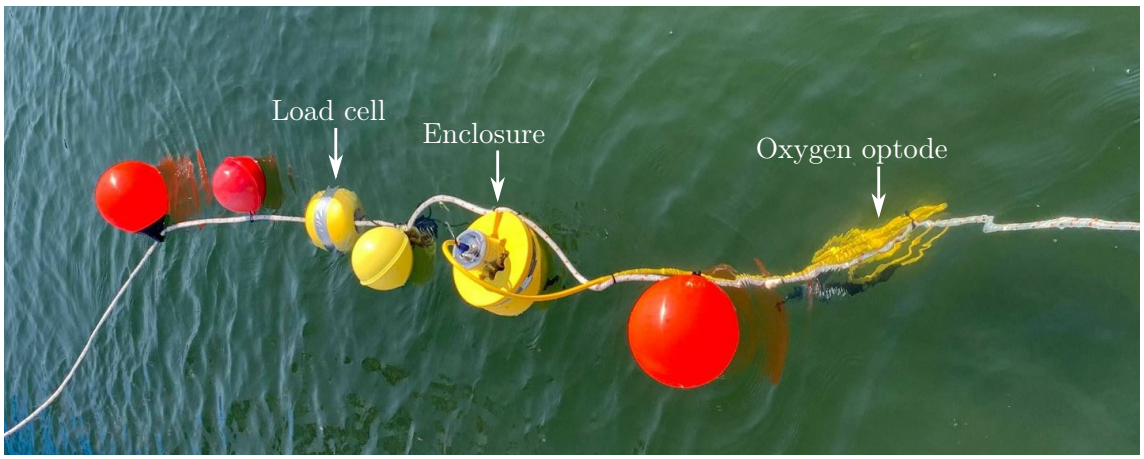


Figure 6.12: System setup for first verification test on sea

After the setup was complete, it was brought in a boat made available from NTNU and transported out to a breakwater structure at Børsa harbor (see Figure 6.13). Three tests

of the system were conducted. Unfortunately, the format of longitude and latitude during the test was incorrect, which resulted in unusable logged position data. Creating the table for logged positional data had the format FLOAT (6 digits for longitude and latitude), while the correct format is DOUBLE (12 digits for longitude and latitude). Six digits for either longitude and latitude are not specific enough to describe the exact position of the system. Correct logging is shown in Section 6.1.3 and was corrected in the code after the verification test. A new test was not possible to perform due to time constraints, but given that the positional data was logged during the entire test and the error was discovered before a full-scale test, it is viewed as successful. The two other tests for orientation and rotation and tension in the rope are described with methodology and results in the chapters below.

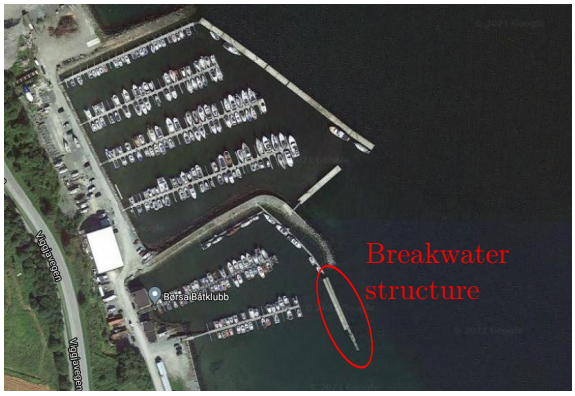


Figure 6.13: Børsa harbor with breakwater structure marked in red

The specific test and noticeable events assumed to be reflected on acquired data after being plotted were written down with timestamps during all three tests. It was essential to obtain a time frame that represents a reference for the logged data. Reference data characterizes the data and establishes permissible values and possible consistency. The test sequence would be challenging to interpret when analyzing the acquired data without the log.

6.2.1 Orientation and rotation

The rope simulating the top rope of the sweep net was fastened to the breakwater structure at one end and with a floater on the other end (orange floater at the visible end of the rope in Figure 6.14). The intention was to observe the orientation and rotation of the system while drifting and being affected by the dynamics of the sea (waves, streams, boats, and simulated fish movements). The system was deployed, and the drift was observed for a given period. The first goal was to acquire a dataset representing the reference frame of the device. It presented difficulties due to the waves and the system drifting towards the breakwater structure. The floater end had to be moved away (thrown away) to prevent the system from drifting underneath the breakwater structure, which influenced the measured reference. A minute of the logged orientation during the time frame of the reference presents that, where the last 10 seconds hold the true reference in Figure 6.15 (based on notes taken during the test). The motion tracker reference frame is a 108° counter-clockwise rotation about Roll and 70° clockwise rotation (-70°) about Yaw. The jump in Yaw is a result of the system being thrown away from the breakwater structure, causing a sudden shift in orientation.

Directly after the reference frame was captured, heavy waves hit the system (from times-

Figure 6.14: Test setup for motion verification on sea with reference frame

tamp 14:06:20). The y-axis (Pitch) is pointing upwards from the enclosure (as seen in Figure 6.14 and the waves hit the system approximately in the middle between the xz-plane. It was assumed that either of the coordinate axes which constitute the xz-plane would be influenced more than observed from the plotted response (fig. 6.16). An oscillating response for all three angles is observed on the plot, but Pitch has the most prominent response. The steady response from the reference frame shifts to an oscillating response with a deviating difference from the highest to the lowest point at almost 19° (15° more compared to the Pitch response from the reference frame). Boundaries for the maximum and minimum response in Pitch is marked with horizontal lines in Figure 6.16 which illustrates the distinctive difference in response. The response is less visible for Roll and even less for Yaw. After analyzing the results, it was concluded that the low acceleration in the waves does not produce high differentiation in orientation and rotation for either of the coordinate axes. The sudden jump in Yaw is also observed while the system was influenced by heavy waves, because the system was thrown away from the breakwater structure.

While the heavy waves hit the system from one side, higher accelerating and lower height waves from a boat driving by the system hit at an approximately 45° angle on the negative x-axis (between the negative x-axis and positive z-axis). The Euler angle response from the event resulted in rapid and substantial differences for Roll, Pitch, and Yaw (see Figure 6.17). The amplitude of each angle had a higher, faster, and more constant response than the heavy waves on their own. The distinction between heavy waves and waves caused by a boat drive-by was effortless, indicating that the system is responsive to deviation in its orientation caused by external influence on the sea.

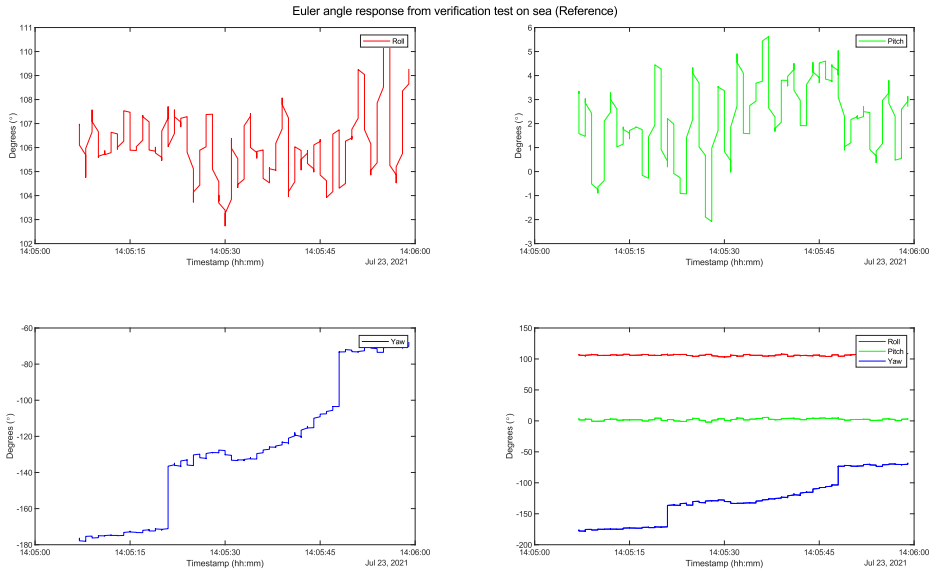


Figure 6.15: Motion tracker response in Euler angles - reference frame

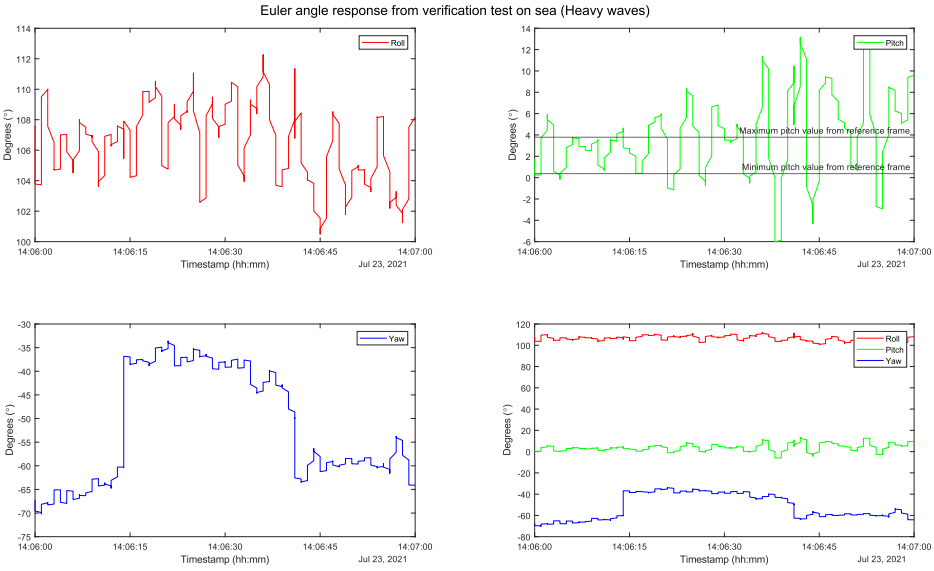


Figure 6.16: Motion tracker response in Euler angles - heavy waves affecting Pitch with line boundaries for comparing to reference frame

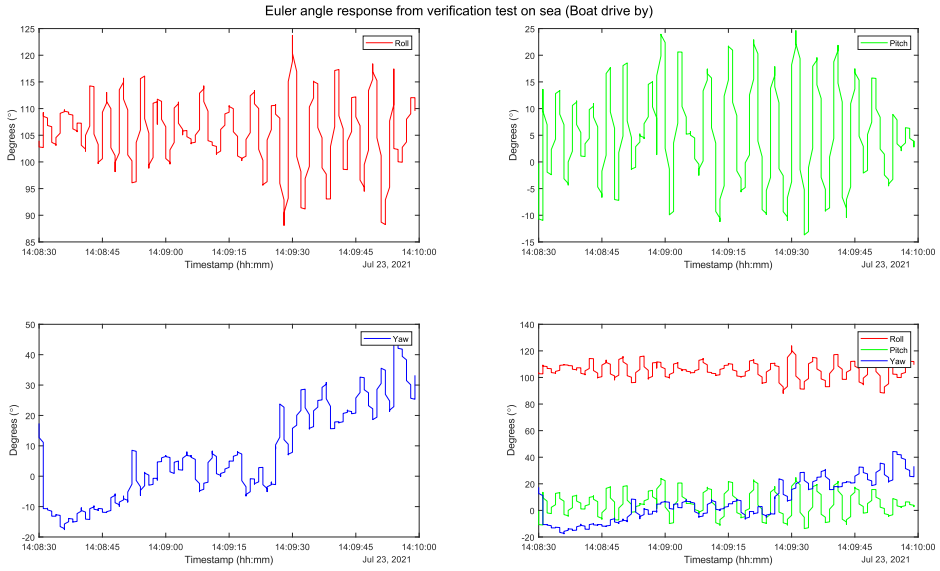


Figure 6.17: Motion tracker response in Euler angles - boat drive by causing rapid and substantial differences in Roll, Pitch and Yaw

Lastly, a simulation of sporadic and rapid fish movements hitting (poking) the enclosure (or the sweep net) with a boat hook as in Figure 6.18.

Figure 6.18: Fish movements simulated by poking the enclosure with a boat hook

The response from simulating fish movements started at approximately 14:11 and is plotted in Figure 6.19. There are minor differences between the response from a boat driving by and simulated fish movements, but the more sporadic, higher valued, and less constant amplitudes are visible. In addition, the slight constant increase in Yaw angle response is supported by the fish hook tilting the system with the boat hook, as in Figure 6.18, increase the rotation about the z-axis (Yaw in Figure 6.19).

The results supports some conclusions that are assumed to influence the orientation and rotation response of the system that a full-scale test during a crowding process would confirm:

1. Movements from fish behavior (fast muscle swimming) against the enclosure cause large deviations from the reference frame with a sporadic and non-constant pattern.
2. Constantly increasing (or decreasing) angle response can be observed, which can imply drag in the net structure from a biomass swimming collectively away from a pump.
3. Movements from other external influence (boats and fish farm equipment) cause large deviations from the reference frame in a constant pattern.

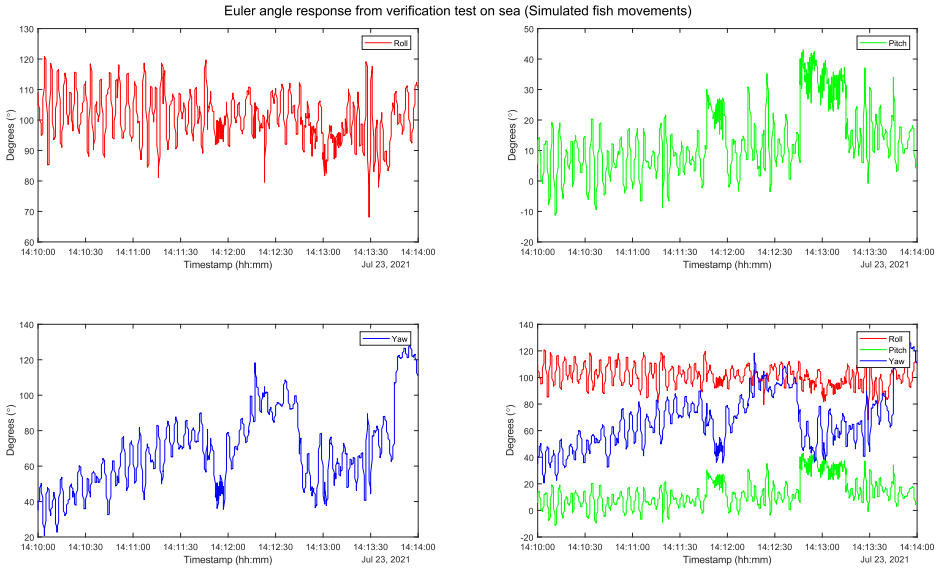


Figure 6.19: Motion tracker response in Euler angles - fish movements simulated with a boat hook

6.2.2 Tension

The rope simulating the top rope of the sweep net was fastened to the breakwater structure at one end and the front of the boat at the other end (see Figure 6.20). To begin with, there was no tension applied on the rope for approximately one minute by observing a slack in the rope. Then, the boat motor was put in reverse at kept steady at five different rotation speeds, defined as Revolutions Per Minute (RPM). The test had the following sequence:

- Start tension test at 15:45
- Slack in rope for 5 minutes
- 700 RPM for 2 minutes
- A jerk in the boat before increasing to 1100 RPM for one minute
- Increased to 1400 RPM for one minute
- Increased to 1700 RPM for one minute (Rope above sea level)
- Increased to 2000 RPM for one minute
- Decreased to a slack rope
- Stop tension test at 15:57

Figure 6.20: Test setup for tension verification on sea

Figure 6.21 is the plotted result of the logged tension throughout the test sequence (with the calibration from the land-based test). The first five minutes of the test indicate a deviation from zero tension at sporadic timestamps, but overall a close to zero measurement. The deviation can appear from the boat drifting with the streams in the water and causing unwanted tension in the rope. Therefore, it is not observed as a measurement error by the system but rather a test setup and implementation error that is difficult to avoid in such an environment. After the first five minutes, there is a steady response of 35-80kg tension at 700 RPM in reverse from the boat for two minutes. At timestamp 13:52, the jerk in the

boat caused by difficulties regarding the fine-tuning of the motor speed is noticeable before the tension stabilizes between 111-180kg for one minute at 1100 RPM. The three levels of increasing RPM are prominent for the rest of the measurement sequence, decreasing to zero at the end.

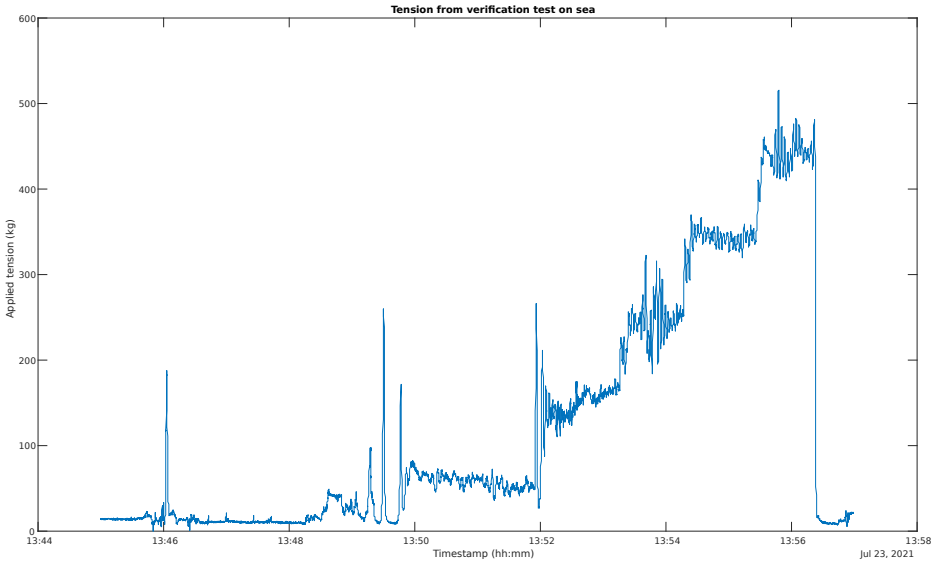


Figure 6.21: Tension response from verification test on sea

Even though the response in tension could not be validated with a known tension from another measurement device, there is a clear distinction between different applied tension. The test also implies that gradually increasing tension is possible to observe from the load cell response, which was the main goal for the test. Further usage of the measured tension can be observed by an operator of the electric winch that controls the sweep net throughout a crowding process.

Chapter 7

Discussion and conclusion

The goal for this thesis was to develop a prototype based on demand from the Crowdguard project partners. The demand was motivated by a real-time fish behavior and crowding process equipment surveillance method that supplements today's existing practice of experience-based observations. This thesis, including the preceding Specialization project, has included constructing an idea for a prototype, the necessary steps to develop the concept to a physical prototype, and lastly testing the finished first draft.

This thesis was a continuation of the preceding Specialization project, where the idea and concept for the prototype were already established. This document presented a detailed account of the design process, focusing on both hardware and software development. The performance of the developed system is evaluated based on the results acquired from system verification tests and a first verification test on sea.

The results from the system verification tests indicate that each of the sensor elements which constitute the prototype is responsive to external influence and change of orientation and rotation, the tension in a rope, the position of the system, water environment variables, and visual data observation during separate tests. In addition, a headless system startup with the extraction of sensor measurement data beginning at an interrupt was successful. The data acquired from each sensor at separate tests and the headless startup of the system indicated a promising outcome of a test on a larger scale in an environment that replicates a crowding process. The first verification test on sea verified that the prototype responded similarly as expected in advance of the conducted test. Since the test was conducted in an environment that was as close to a full-scale test during a crowding process as time allowed, the prototype will operate as expected during such a test. Due to a lack of data on position during the first verification test on the sea, the results cannot confirm that the position was correct. However, the acquired results from the system verification test build upon the assumption that position would be acquired correctly if the data format was precise. The generalizability of the results is limited because neither the oxygen optode nor the camera had sufficient hardware and software implementations during the first verification test on the sea to conclude if they would respond similarly during a crowding process. Further research and testing are needed to establish the assumptions for both the oxygen optode and camera performance.

The results from testing orientation and rotation during the first verification test on sea indicated that differences in external influence on the system were distinguishable. Whereas

amplitude for the oscillating response of Euler angles was repetitive and constant for natural influence from waves, simulated influence by fish movements presented sporadic and more powerful amplitude. The results build upon a simulated environment and cannot conclude that a crowding process and fish behavior during such a process would affect the system. However, it is a leading result that implies that a full-scale test is highly relevant to perform. Based on the results acquired from a full-scale test, further implementations and development of the system can be established.

Results from testing tension in a rope, simulating the top rope of a sweep net, imply that the applied tension resulting from winching is possible to measure and observe. The limited knowledge of the actual tension in the rope during a crowding process limits the possibility of concluding whether the specific load cell can measure the tension in the rope. It substantiates the fact that further testing is required.

To conclude, a prototype was designed and implemented according to the demand from the Crowdguard project partners and preceding Specialization project and is documented in this report. Tests in different scales have been carried out, and the results of the sensor unit have been discussed. Based on the results, it is reasonable to conclude that the system has the potential to operate during a crowding process and retrieve meaningful data about fish behavior and mechanical equipment during a crowding process. It also has the potential to be a part of a real-time wireless data transfer system because it is facilitated to be expanded (the size of any additional hardware components should be considered).

Chapter 8

Further Work

8.1 Implementation of voltage regulator

The smart fish crowding sensor system has the potential to capture water environment variables during test in sea, if the oxygen optode is supplied with sufficient power. A voltage regulator that increases the 5V voltage from the Raspberry Pi and Qwiic HAT to a 9V voltage (with sufficient current) is required to power the oxygen optode. However, the optode is physically mounted to the prototype and the software implementation retrieves data correctly from the optode if it is hardware connected.

Since the implementation of the oxygen optode require another connected USB device to the Raspberry Pi, it is advised to implement a software recognition program that assign the correct USB path to the correct device as it is now hardcoded. Hardcoded variables that could change because of connection differences between each use of the prototype are seen as a overall decrease in quality and functionality of the system.

8.2 Enabling camera recording functionality from headless software launch

The camera recording functionality is unable to capture video from the camera during system startup and launch from a shell script. It is unclear why the functionality is unachievable though the specific use case, but it deduct the performance and analysing material from the system during testing and usage.

It is important to note that the camera functionality is usable if system software launch is performed from a terminal window with a monitor and keyboard connected to the Raspberry Pi. It is therefore assumed that the launcher script causes the malfunction of the camera functionality. The launcher method is suggested to be changed to a suitable method that ensures working functionality of every system component and sensor. Due to time constraints no further inspection on the cause of the problem was conducted.

8.3 Shortcomings of test during a crowding process

The shortcomings of executing a full-scale test during a crowding process has lead to a reduced basis of data which cannot answer the question presented in the introduction of this thesis (Section 1.1). However, the necessary steps are facilitated for and described in the previous sub-chapters. After those are implemented, a full-scale test during a crowding process is possible. A full-scale test also requires planning on methodology with partners in the Crowdguard project that aid a fish farm to conduct the test at, as well as necessary equipment for a crowding process and transportation to the facility. The methodology of the test should answer the following questions before a test is conducted (a set of thoughts are presented below each question).

- How and at what position should the enclosure part of the system be attached to the top rope of the sweep net?
  - Attaching the enclosure to a rope with cable ties was sufficient during the verification test at sea and will allow for easy detachment after a test during a crowding process. A more permanent solution that is attached to the flotation device (e.g., slit in the flotation device that the top rope can slide into) is recommended if the solution is to be implemented for more frequent usage.
  - The enclosure should be positioned to be as close to the middle as possible when the crowding process reaches a point where fish are significantly crowded.
- How and at what position should the load cell be attached to the top rope of the sweep net?
  - Attaching the load cell on the rope has limited possibilities and is advised to be attached as the verification test at sea described.
  - The position of the load cell does not matter, as long as the electrical cord to the enclosure part of the system is long enough and is not under tension.
- When should the system be started and when should the logging of sensor data begin?
  - The system should be started when there is low risk of water entering the system since the top end cap must be detached from the enclosure to turn on the powerbank. Ideally before the system is brought out from mainland.
  - Sensor logging should begin as close to deployment as possible to ensure minimal logging of data prior to the actual crowding process. Within reach, the system should receive an interrupt from the magnetic hall sensor at the beginning of sweep net deployment.

It is advised to record the system behaviour from the water surface throughout the crowding process for analyzing purposes. A camera with vision of the process and the system (above the water surface) should be prepared beforehand with access to timestamped footage. In addition, noticeable events during the process could be noted with timestamp. Given that the dataset reach a considerable size during sensor data logging for 2-3 hours, noted time sequences with noticeable influence on the system that are assumed to be visible on measurements reduce the time searching through data.

After a full-scale test during a crowding process is performed, the following questions should be answered:

- Does an embedded sensor system provide enough information about a crowding process to operate as a real-time surveillance system?
- Can an embedded sensor system to determine when risk-reducing measures need to be enforced so that fish welfare during crowding processes is not reduced?

If the system succeeds to fulfill the questions above, a real-time solution can be implemented. Then a discussion of alternatives to transfer measured data from the Raspberry Pi to an operating station at the fish farm facility is needed. The system requirements state to be standalone, wireless system, and therefore require a wireless data transferring technology. As of today, there are multiple different methods of wireless data transfer technologies, Table 8.1 present some of these with advantages and disadvantages.

Wireless transfer technology	**Advantages**	**Disadvantages**
GSM (2G/3G)	High coverage Large number of operators Low cost of data transfer Free roaming within EU Reliable technology	High energy consumption Low transmission speed Demanding communication
LTE (4G)	High transmission speed Fast building and deployment Free roaming within EU Reliable technology Perspective frequency band	High energy consumption Higher price of modules
5G	Higher transmission speed Effective and efficient	Less coverage Security and privacy issues Could be incompatible with older devices
Bluetooth	Lower consumption Advanced technology	Short range Paid license
WiFi	High transmission speed Advanced technology High security Expandable	High energy consumption Point-point topology
ZigBee	Low consumption Flexible Low cost	Privacy issues Low transmission rate Short coverage

Table 8.1: Wireless data transfer technologies w/advantages and disadvantages

The operating station of the system sets specifications and boundaries on transferring and receiving data. Even though measurement data is stored within a local database on the Raspberry Pi, it is not predefined that the same method of data storage should be used on the user side of the system. However, remote connection to the local database on the Raspberry Pi through a wireless connection is a considerable option. The advantage of storing the data both locally and remotely is portrayed if the wireless connection suffer from failure, because it ensures that data is not lost due to a backup. An equal database structure to the local solution on the Raspberry Pi on the user side would only require extraction of the data sent to a decision making viewing program. The design and GUI of the data and decision making program is already in progress by SINTEF.

References

[1] J. S. Skjøy, "Smart fish crowding sensor," Specialization Project Report, NTNU ITK, 2020.

[2] "Energy transition outlook 2020 - a global and regional forecast to 2050," DNV GL AS, Tech. Rep., 2020.

[3] "The state of world fisheries and aquaculture 2018 - meeting the sustainable development goals," Food and Agriculture Organization of the United Nations (FAO), Tech. Rep., 2018.

[4] B. Misund, *Fiskeoppdrett*, Online; accessed 09 May 2021. [Online]. Available: https://snl.no/fiskeoppdrett.

[5] Fiskedirektoratet, *Akvakulturstatistikk: Matfiskproduksjon av laks, regnbueørrett og ørret*, Online; accessed 09 May 2021. [Online]. Available: https://www.fiskeridir.no/Akvakultur/Tall-og-analyse/Akvakulturstatistikk-tidsserier/Laks-regnbueoerret-og-oerret/Matfiskproduksjon.

[6] A. Iversen, F. Asche, Ø. Hermansen, and R. Nystøyl, "Production cost and competitiveness in major salmon farming countries 2003–2018," *Aquaculture*, vol. 522, p. 735 089, 2020, ISSN: 0044-8486. DOI: https://doi.org/10.1016/j.aquaculture.2020.735089. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0044848619300638.

[7] "Monthly salmon report, november 2018 (demoreport)," KONTALI, Tech. Rep., 2018.

[8] U. Erikson, L. Gansel, K. Frank, E. Svendsen, and H. Digre, "Crowding of atlantic salmon in net-pen before slaughter," *Aquaculture*, vol. 465, pp. 395–400, 2016, ISSN: 0044-8486. DOI: https://doi.org/10.1016/j.aquaculture.2016.09.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0044848616304744.

[9] M. Føre, E. Svendsen, J. Alfredsen, I. Uglem, N. Bloecher, H. Sveier, L. Sunde, and K. Frank, "Using acoustic telemetry to monitor the effects of crowding and delousing procedures on farmed atlantic salmon (salmo salar)," *Aquaculture*, vol. 495, pp. 757–765, 2018, ISSN: 0044-8486. DOI: https://doi.org/10.1016/j.aquaculture.2018.06.060. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0044848617324407.

[10] LOVDATA, *Forskrift om drift av akvakultur (akvakulturdriftsforskriften)*, Online; accessed 12 May 2021. [Online]. Available: https://lovdata.no/dokument/SF/forskrift/2008-06-17-822/KAPITTEL_2#%5C%C2%5C%A713.

[11] F. Berlinghieri, P. Panizzon, I. L. P.-Williams, and C. Brown, "Laterality and fish welfare - a review," *Applied Animal Behaviour Science*, vol. 236, 2021, ISSN: 0168-1591. DOI: https://doi.org/10.1016/j.applanim.2021.105239. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168159121000265.

[12] C. Noble, K. Gismervik, M. K. Iversen, J. Kolarevic, J. Nilsson, L. H. Stien, and J. F. Turnbull, *Welfare indicators for farmed atlantic salmon: Tools for assessing fish welfare*, Online; accessed 19 March 2021, NOFIMA, Tromsø, Norway, Nov. 2018. [Online]. Available: https://nofima.no/wp-content/uploads/2018/11/FISHWELL-Welfare-indicators-for-farmed-Atlantic-salmon-November-2018.pdf.

[13] E.J.Krakiwsky and D.E.Wells, *Coordinate systems in geodesy*, Online; accessed 14 June 2021. [Online]. Available: http://www2.unb.ca/gge/Pubs/LN16.pdf.

[14] T. E. o. E. Britannica, *Latitude and longitude*, Online; accessed 14 June 2021. [Online]. Available: https://www.britannica.com/science/latitude.

[15] B. Hofmann-Wellenhof, *Gnss : Global navigation satellite systems : Gps, glonass, galileo and more*, eng, Wien, 2008.

[16] D. Doody, *Basics of space flight - reference systems*, Online; accessed 13 July 2021. [Online]. Available: https://solarsystem.nasa.gov/basics/chapter2-3/.

[17] D. Liu, B. Barber, and L. DiGrande, "Chapter 2 - the open systems interconnect model," in *Cisco CCNA/CCENT Exam 640-802, 640-822, 640-816 Preparation Kit*, D. Liu, B. Barber, and L. DiGrande, Eds., Boston: Syngress, 2009, pp. 47–82, ISBN: 978-1-59749-306-2. DOI: https://doi.org/10.1016/B978-1-59749-306-2.00006-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781597493062000063.

[18] O. Engeland and J. T. Gravdahl, *Modeling and Simulation for Automatic Control*. Trondheim: Marine Cybernetics AS, 2002.

[19] J. Vince, *Quaternions for Computer Graphics*, eng. London: Springer London, ISBN: 0857297597.

[20] P. Dormashkin, *Tension in a rope*, Online; accessed 25 May 2021. [Online]. Available: https://phys.libretexts.org/@go/page/24470.

[21] "Selection of water quality variables," eng, in *Water Quality Assessments - A Guide to Use of Biota, Sediments and Water in Environmental Monitoring - Second Edition*, London, UK: WHO by F and FN Spon, 2002, pp. 75–110, ISBN: 0419215905.

[22] M. Remen, M. Sievers, T. Torgersen, and F. Oppedal, "The oxygen threshold for maximal feed intake of atlantic salmon post-smolts is highly temperature-dependent," *Aquaculture*, vol. 464, pp. 582–592, 2016, ISSN: 0044-8486. DOI: https://doi.org/10.1016/j.aquaculture.2016.07.037. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0044848616303970.

[23] E. Austreng, T. Storebakken, and T. Åsgård, "Growth rate estimates for cultured atlantic salmon and rainbow trout," *Aquaculture*, vol. 60, no. 2, pp. 157–160, 1987, ISSN: 0044-8486. DOI: https://doi.org/10.1016/0044-8486(87)90307-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0044848687903073.

[24] S. Handeland, Å. Berge, B. Björnsson, Ø. Lie, and S. Stefansson, "Seawater adaptation by out-of-season atlantic salmon (salmo salar l.) smolts at different temperatures," *Aquaculture*, vol. 181, no. 3, pp. 377–396, 2000, ISSN: 0044-8486. DOI: https://doi.org/10.1016/S0044-8486(99)00241-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0044848699002410.

[25] S. Handeland, Björnsson, A. Arnesen, and S. Stefansson, "Seawater adaptation and growth of post-smolt atlantic salmon (salmo salar) of wild and farmed strains," *Aquaculture*, vol. 220, no. 1, pp. 367–384, 2003, ISSN: 0044-8486. DOI: https://doi.org/10.1016/S0044-8486(02)00508-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0044848602005082.

[26] A. Kullgren, F. Jutfelt, R. Fontanillas, K. Sundell, L. Samuelsson, K. Wiklander, P. Kling, W. Koppe, D. J. Larsson, B. T. Björnsson, and E. Jönsson, "The impact of temperature on the metabolome and endocrine metabolic signals in atlantic salmon (salmo salar)," *Comparative Biochemistry and Physiology Part A: Molecular and Integrative Physiology*, vol. 164, no. 1, pp. 44–53, 2013, ISSN: 1095-6433. DOI: https://doi.org/10.1016/j.cbpa.2012.10.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1095643312004837.

[27] M. Hvas, O. Folkedal, A. Imsland, and F. Oppedal, "The effect of thermal acclimation on aerobic scope and critical swimming speed in atlantic salmon, salmo salar," *Journal of Experimental Biology*, vol. 220, no. 15, pp. 2757–2764, 2017, ISSN: 1477-9145. DOI: https://doi.org/10.1242/jeb.154021. [Online]. Available: https://journals.biologists.com/jeb/article/220/15/2757/17866/The-effect-of-thermal-acclimation-on-aerobic-scope.

[28] L. D. Pyeatt, "Chapter 13 - common system devices," eng, in *Modern Assembly Language Programming with the ARM Processor*, Elsevier Inc, 2016, pp. 405–430, ISBN: 0128036982.

[29] W. J. Buchanan, "Rs-232," eng, in *The Complete Handbook of the Internet*, Boston, MA: Springer US, pp. 259–294, ISBN: 9781461349990.

[30] D.-S. Kim and H. Tran-Dang, "Communication using controller area network protocol," eng, in *Industrial Sensors and Controls in Communication Networks*, ser. Computer Communications and Networks, Cham: Springer International Publishing, 2018, pp. 31–41, ISBN: 9783030049263.

[31] T. Gray, "Communication interfaces, operating systems, and drivers," eng, in *Projected Capacitive Touch*, Cham: Springer International Publishing, 2018, pp. 135–142, ISBN: 9783319983912.

[32] W. Gay, "I2c," eng, in *Beginning STM32*, Berkeley, CA: Apress, 2018, pp. 195–221, ISBN: 1484236238.

[33] L. D. Pyeatt, "Chapter 12 - pulse modulation," eng, in *Modern Assembly Language Programming with the ARM Processor*, Elsevier Inc, 2016, pp. 395–404, ISBN: 0128036982.

[34] R. Shishko, *Nasa systems engineering handbook*, [Online], National Aeronautics and Space Administration, Washington, D.C., 1995. [Online]. Available: https://www.nasa.gov/sites/default/files/atoms/files/nasa_systems_engineering_handbook_0.pdf.

[35] *Qwiic connect system*, Online; accessed 15 June 2021. [Online]. Available: https://www.sparkfun.com/qwiic?_ga=2.262819660.1147207308.1623756441-1336105024.1611744551.

Appendix A

Project description

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology and
Electrical Engineering
Department of Engineering Cybernetics

MASTER'S THESIS ASSIGNMENT

Name: Julie Støverud Skjøy
Program: Engineering Cybernetics
Title: Smart fish crowding sensor
Title (Norw.): Smart sensor for fisketrenging
Credits: 30 SP

Project description:

Fish crowding represents a critical and frequently occurring operation in the salmon farming industry. The operation is associated with significant risk with respect to fish welfare, mortality and product quality, but is at the same time an important and unavoidable operation inherent to many farm practices such as sorting, delousing and harvesting. Efficient monitoring and control tools that can help keeping the crowding process within acceptable limits are therefore in great demand. The objective of this project is to investigate and develop a solution for instrumentation of the crowding gear with a smart sensor that can provide relevant and real-time data on the progression of the crowding process and how it affects the state and wellbeing of the fish. The project should also discuss how this information can be exploited to control crowding in a way that minimizes risk of inflicting stress, injuries and hypoxia for the fish. The Master's project builds on the results from a preceding Specialization project on the same topic and includes the following elements:

- Recap of the crowding process and the main results from the preceding Specialization project, including a problem formulation and requirements specification of the smart fish crowding sensor unit. The selected sensor's purpose and relevance for the crowding process should be described in detail.
- Design and implementation of a crowding sensor prototype unit:
    - Detailed investigation of the properties and interfaces of the selected sensors
    - Integration of sensors with embedded computer; electrical interfaces and software drivers
    - Development of overall control software with a solution for persistent storage of data
    - Calculation of power consumption, selection of battery and power supply/voltage levels
    - Mechanical integration in watertight enclosure and floatation device
- Planning and carrying out tests for design verification and validation in different scales. Documentation and discussion of sensor unit performance.
- Suggest and discuss a practical solution for real-time wireless transfer of sensor data.

Project start: 11 January 2021
Project due: 7 June 2021 (postponed to 30 July 2021 due to covid-19 restrictions)
Host institution: NTNU, Department of Engineering Cybernetics
Supervisor: Jo Arve Alfredsen, NTNU/DEC

Trondheim, 22 January 2021
Jo Arve Alfredsen

Appendix B

Terminal menu with belonging sub-menus

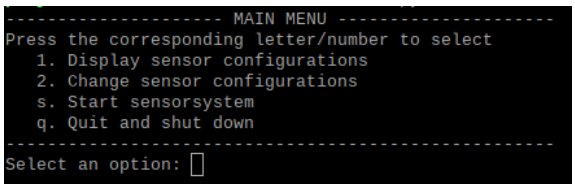


Figure B.1: Main menu in terminal menu

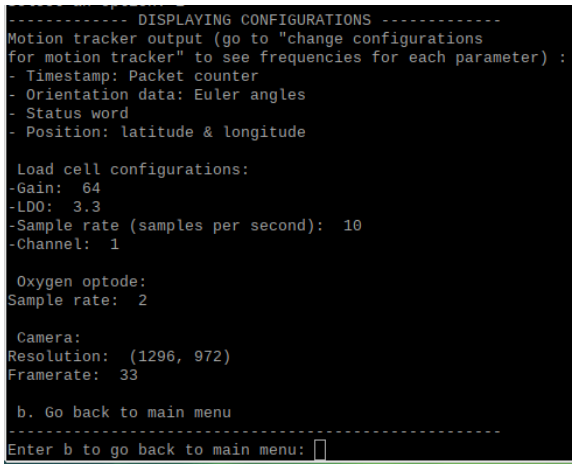


Figure B.2: Menu for displaying current sensor configurations

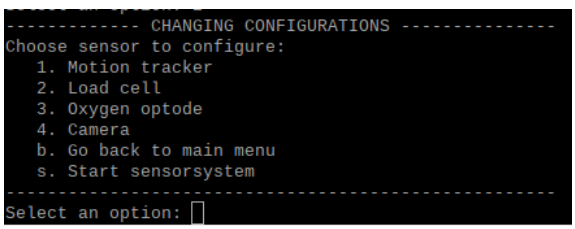


Figure B.3: Menu for choosing which sensor to configure

```
------------------- MOTION TRACKER ------------------
Choose the wanted output by typing its code with
corresponding sample frequency in Hz (between 1
and 30).

Example: iu20,ip20,oe20,pl20,sw20

NOTE: max frequency for
 - Temperature (tt): 1 Hz
 - GNSS (np and ns): 4 Hz
 - GPS (gd, gs, gu, and gi): 4 Hz

SCR = Sensor Component Readout

Press b to go back to previous menu or q to quit program.

| Code   | Description                                | Chosen   |   Sample frequency [Hz] |
|--------|--------------------------------------------|----------|-------------------------|
| tt     | Temperature                                |          |                         |
| iu     | Timestamp: UTC Time                        |          |                         |
| ip     | Timestamp: Packet counter                  | Yes      |                      20 |
| ii     | Timestamp: Integer Time of the Week (ITOW) |          |                         |
| if     | Timestamp: Sample time fine                |          |                         |
| ic     | Timestamp: Sample time coarse              |          |                         |
| ir     | Timestamp: Frame range                     |          |                         |
| oq     | Orientation data: Quaternion               |          |                         |
| om     | Orientation data: Rotation matrix          |          |                         |
| oe     | Orientation data: Euler angles             | Yes      |                      20 |
| bp     | Baro pressure                              |          |                         |
| ad     | Acceleration: delta v                      |          |                         |
| aa     | Acceleration                               |          |                         |
| af     | Free acceleration                          |          |                         |
| ah     | HR acceleration                            |          |                         |
| pa     | Position: altitude & ellipsoid             |          |                         |
| pp     | ECEF position                              |          |                         |
| pl     | Position: latitude & longitude             | Yes      |                      20 |
| np     | GNSS: PVT data                             |          |                         |
| ns     | GNSS: sattelite info                       |          |                         |
| wr     | Angular velocity: rate of turn             |          |                         |
| wd     | Angular velocity: delta q                  |          |                         |
| wh     | Angular velocity: rate of turn HR          |          |                         |
| gd     | GPS: DOP                                   |          |                         |
| gs     | GPS: SOL                                   |          |                         |
| gu     | GPS: time UTC                              |          |                         |
| gi     | GPS: SV info                               |          |                         |
| rr     | SCR: ACC, GYR, MAG, temperature            |          |                         |
| rt     | SCR: Gyro temperatures                     |          |                         |
| mf     | Magnetic field                             |          |                         |
```

Figure B.4: Menu for changing configuration for motion tracker

```
------------------- LOAD CELL -------------------
Choose the wanted parameter to edit by typing its code, then the
corresponding value.
 - Gain: 1, 2, 4, 8, 16, 32, 64, 128
 - LDO: 2.4, 2.7, 3.0, 3.3, 3.6, 3.9, 4.2, 4.5V
 - Sample rate: 10, 20, 40, 80, and 320 samples per second
 - Channel: 1 or 2

Press b to go back to previous menu or q to quit program.

| Code   | Description          |   Value |
|--------|----------------------|---------|
| g      | Gain                 |    64   |
| ldo    | Low-Drop-Out voltage |     3.3 |
| sr     | Sample rate          |    10   |
| ch     | Channel              |     1   |
Choose which parameter to edit:
```

Figure B.5: Menu for changing configuration for load cell

```
----------------- OXYGEN OPTODE -----------------
Change the sample rate by typing the wanted rate between 2 each
second (0.5) and once every 255 min (15300).

Press b to go back to previous menu or q to quit program.

| Code   | Description   |   Value |
|--------|---------------|---------|
| sr     | Sample rate   |       2 |
Choose which parameter to edit:
```

Figure B.6: Menu for changing configuration for oxygen optode

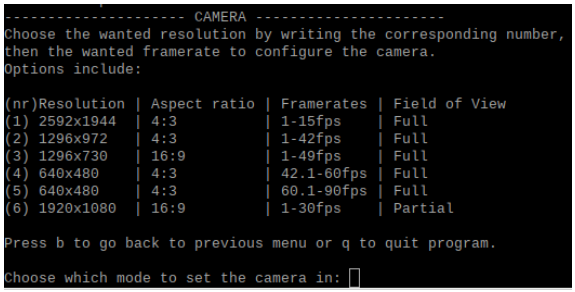


Figure B.7: Menu for changing configuration for camera