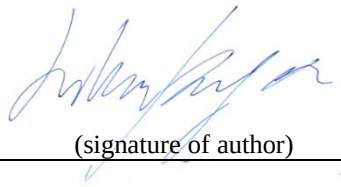
Universitetet i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Robot Technology and Signal Processing	Spring / ~~Autumn~~ semester, 2020 Open/~~Confidential~~
Author: Joakim Andrè Alsaker-Haugen	 (signature of author)
Programme coordinator: Supervisor(s): Karl Skretting	
Title of master's thesis: Attitude and Heading Reference System for a laboratory drilling rig control system	
Credits: 30	
Keywords: Kinematics MEMS Sensor Calibration AHRS design Drillbotics®	Number of pages: 112 + supplemental material/other: …… Stavanger, 13.07.2020 date/year

Abstract

This thesis describes the work done in contribution to the UiS Drillbotics 2020 drill rig design for the annual international Drillbotics competition. Drillbotic is a competition for universities to design and build a miniature drilling rig that autonomously drills through a rock sample, with control algorithms and sensors.

This thesis's objective is to documents the work related to the design of a tracking system for a drill bit on a miniature drilling rig. This design included implementing an Attitude and Heading reference system (AHRS), kinematics, calibration of sensors, and design of a bottom hole sensor card to be implemented in an existing control system. Furthermore, it was worked on changing the existing actuator controllers from an Arduino based system to PLC.

The tracing system was designed for in association with the new guidelines for the annual international Drillbotics competition. One of the new rules implemented in the 2020 Drillbotics competition is the mandatory requirements for the use of both surface and downhole sensors.

In this thesis, I focus on the designing and implementation of a tracing system. This includes the design and testing of different sensor calibration and estimation methods to find a solution for tracking the drill bit in real-time. The developed tracing system should be able to track the drilling path through a known rock sample.

Contents

List of Figures

List of Tables

Terminologi

Notasjon

$\|q\|$	norm of vector q
$\|\|q\|\|$	norm of matrix q
$\hat{q}$	normalized vector q
$\dot{q}$	derivative of a vector q
q^*	conjugate of a vector q

Symbols

$\otimes$	Kronecker product
*	dot product
*	cross product

Abbreviations

AHRS = Attitude and Heading Reference Systems

IC - integrated circuit

IMU = inertial measurement unit

MEMS = Microelectromechanical systems

MCU = Microcontroller unit

MARG = Magnetic, Angular Rate, and Gravity

PCB = printed circuit board

ppm - parts per million

pF = picoFarad

RMSE = Root Mean Square Error

RPM = revolutions per minute

USART = Universal Synchronous Asynchronous Receiver Transmitter

VDD - the label of an IC power supply pin

WOB = weight on bit

1 Introduction

1.1 Background

In the oil and gas industry, it can be observed an increased focus on digitization and automation of their systems to reduce cost and enhance safety. A problem involved in the digitization and automation is directional drilling and position surveillance. Problem considered, it is essential to have good readings from sensors and to have stable tracking of the drill path.

Drilling Systems Automation Technical Section (DSATS) is a part of the Society of Petroleum Engineers (SPE) and works to standardize and support initiatives within drilling automation systems [6]. Drillbotics is one of these initiatives and is an annual international competition for universities, where the goal is to design and build a small drilling rig capable of drilling a directional well through a homogeneous sandstone rock sample provided by DSATS [4]. UiS Drillbotics team has been part of this competition since 2017, and last year got the 2nd place overall. Last year (2019), the team focused on implementing new hardware and software upgrades to drill a deviated well. Some of the implementations were rig upgrades for directional drilling [2] and new software architecture that divides the system into individual modules that run microservices [1]. Since the Drillbotics competition rules allow the re-use and modification of the previous year's rig solution, will this year UiS Drillbotics team use the rig design defined by the 2019 team and use it as the starting point.

This thesis will be center on the design and building of a sensor card that will track the movements of the drill bit in real-time and some minor changes to the hardware.

1.1.1 2019 drilling rig

Directional drilling was implemented in the 2019 competition. Where the goal is to deviate from the center of the rock as much as possible. Therefore many improvements were necessary for the system.[2] [1]. They implemented a pneumatic system to run a pneumatic motor in the downhole assembly instead of a top drive for rotating drill bit, knuckle joint allowing bend in the pipe, more sensors, new actuator system for whipstock and riser, new control system architecture, and an API for third-party usage [1]. The setup for the 2019 drilling rig is illustrated in figure (1), taken from Sander Skjørestad's bachelor thesis (Skjørestad, 2019) [3].Er dette riktig måte å citere

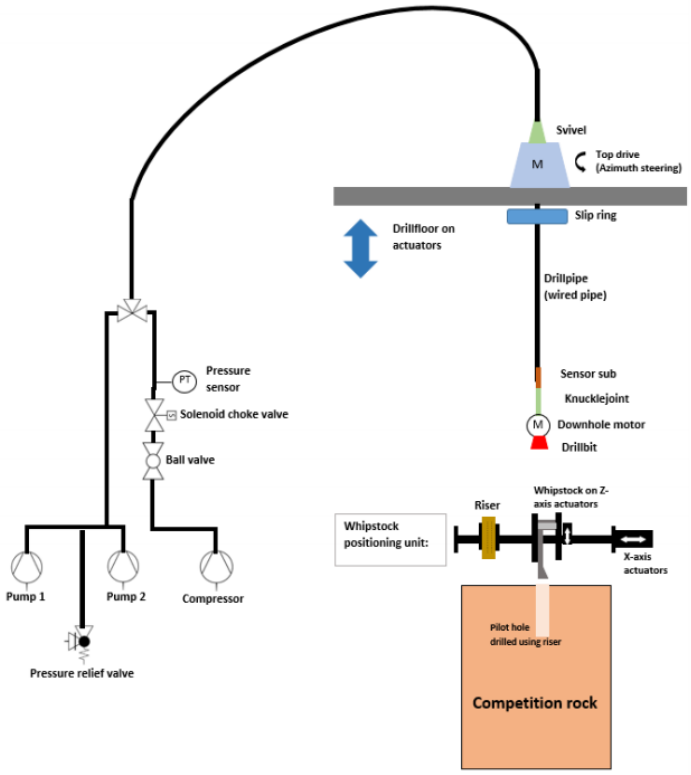


Figure 1: *This illustrates the UiS Drillbotics 2019 setup [3]*

The 2019 drilling rig consists of a total of six hardware systems and a control system. These are; rotation, hoisting, circulation, pneumatic, whipstock positioning, and power systems. Details about these systems and their function can be read in more detail in (Løkkevik and Løken, 2019)[2] and (Skjørestad, 2019)[3]; below some of the systems, necessary for this thesis is described briefly.

The rotation system also called the top drive and is a servo motor that can control the azimuth for the drill bit and is mounted on a metal plate called the top plate. Moreover, this system is located at the top of the rig. The hoisting system moves the top plate by using three actuators connected to the plate, resulting in the drill pipe connected to the top drive to move vertically. Between the connection of the top plate and the actuators are some loadcells which is used to measure weight on bit. There is also a distance sensor that measures the movement of the top plate. The pneumatic system controls the airflow to the downhole motor and will result in controlling the speed and torque of the motor. A solenoid valve whit a pressor sensor is used to measure the pressure and restrict its airflow to control the downhole motor.

For each of the hardware systems, are their one Arduino microcontroller to control it. So there are in total of six Arduinos to collect data and control the whole rig. These Arduinos are connected to one PC that is the center of the control system.

Generally, the control system must have been rewritten for every competition, but last year's system architecture was designed to make it easy to add or remove modules, such that only parts of the system need to be changed. The system was therefore split into multiple modules running a microservice. These modules are constructed as both a server and a client sharing data over a stream connection. A module can subscribe to a module and receive new values that it can process before passing the value to another module. Making a module subscribing or unsubscribe to another module makes it easy to add or remove parts of the control system. More about this control system architecture can be read in Guggedal and Steinstø's bachelor thesis.

1.1.2 Background of study

UiS Drillbotics team has been a part of the annually Drillbotics competition since 2016. Furthermore, the team has needed to change part of the drilling rig every year because of new additions to the competition rules and goals for the teams to complete. Last year (2019), the addition of directional drilling where the goal was to deviate from the center of the rock as much as possible was implemented. In this year's competition, the new goal is to hit multiple targets at varying vertical depths and x,y coordinates. Furthermore, due to the rules of the competition, directional drilling must be drilled autonomously. The autonomous drilling system requires, therefore, to be able to; 1) handle any faults or drilling incidents occurring during the operation, 2) optimizing the drilling parameters for a high rate of penetration (ROP), and 3) steer the drill bit towards the designated target area based on real-time downhole position measurements and a closed-loop control system. New to this year is the mandatory requirements of both surface and downhole sensors. Furthermore, all teams are required to deliver directional survey raw data logs containing the following minimum requirements:

- Timestamp containing year, month, date, hour, minute, and second
- Sensor measured depth
- Downhole sensor value(s) recording of sensor axis
- values, calculated survey qualifier values.

Directional drilling scoring for the competition will be primarily based on how accurate the calculated well trajectory intersects the direction targets. But also on deliverables requirements such as directional survey raw data logs. More about the requirements and competition rules can be read at [5].

As the drilling rig setup was after the 2019 team, Arduino was responsible for controlling all actuators. Using individual Arduinos in earlier years for data

acquisition has been shown to come with some problems of synchronizing the actuator. Since what the control system got from one Arduino could be older measurements in comparison to another. Therefor a measuring amplifier from HMB was implemented early that year to measure the loadcell and distance sensor in the hoisting system in real-time. With this real-time system's success, new input and output PLCs were ordered since the HBM was unsuited for the job. The reason for the purchase of new PLCs was to improve reliability and simplify the integration of new equipment. However, it was not implemented into the system because of time constraints due to late delivery.

Because of the new additional rules about directional drilling and downhole measurements, the rig will need to be upgraded with a directional control system based on data acquired from the sensor(s) in the BHA to get a high score in the competition. Hence one focus for this thesis will be the sensor(s) in the BHA and upgrading the control system with an Attitude and Heading Reference System (AHRS) that uses downhole measurements to survey the directional drilling. However, controlling the rig based on this data will not be a focus, because of insufficient time at the laboratory and the cancellation of the competition in conjunction with the 2020 world crisis. Focusing on the real-time tracing of the directional drilling will be beneficial for future competition because of the new additional rule in the competition of downhole sensors and because the rig does not possess a reliable solution for tracing the directional drilling as of 2019.

1.2 Thesis objective

This thesis's objective is to document the work associated with the design of an attitude and heading reference system (AHRS) to be used for tracing the directional drilling. AHRS design will include both software and hardware needed to accomplish the goal of real-time tracing. The hardware will consist of a sensor card with sensors and a microcontroller, and the software will be composed of filters and sensor fusion algorithms. Results and experiments of different methods and approaches will be presented. An alternative to the AHRS system using kinematics will also be explored as s solution for tracing the directional drilling. Moreover, a hardware upgrade consisting of changing out Arduinos that controls the actuators with PLCs will also be documented in this thesis.

2 Theory

This section will present the relevant theory that the report can base its decisions and assumptions. This chapter will work as a support chapter to create a theoretical understanding of the challenges regarding position tracking and position control. The theory for MEMS magnetometer, accelerometer, and gyroscope, will be presented in this chapter, along with communication protocols, orientation filtering, rigid motions, and kinematics.

2.1 Rigid Motions

Rigid motion is a smaller class from the mathematical transformation of a geometric transformation of a Euclidean space that preserves the Euclidean distance between every pair of points called rigid transformation or Euclidean transformation. Rigid transformations include reflection, rotation, translation, or a combination of these. Rigid motion includes only translation and rotation. In robot kinematics, rigid motion is frequently used to establish various coordinate frames to represent a rigid object's position and orientation [35]. They are also used in transformations among these coordinate frames. To represent the relevant position and orientation of rigid objects with respect to each other will have there own coordinate frames attached and the geometric relationship between these coordinate frames. This section of the chapter will present the relevant theory for this thesis: how to represent position and rotation, rigid motions, and homogeneous transformation.

2.1.1 Representing Position and Rotation

In this thesis, Cartesian coordinates are used at such explaining how to represent position will be base on the use fo this type of system. Moreover, to represent the position in a coordinate system, it is necessary to specify the coordinate reference frame. If there is more than one coordinate frame, a point, p, in space will be represented with respect to either frames. Consider two coordinate frames, frame zero and one, then a point in space can be represented in either in respect to zero, p^0, or with respect to one, p^1. Also, since the origin of these two coordinate frames is just a point in space, they can be represented as coordinates that are the position of the origin of one coordinate frame with respect to the other. Frame zero in respect to frame one will be denoted as o_0^1, and frame zero with respect to frame one as o_1^0.

Vectors are used to represent the point as direction and magnitude from a different point in space. Thus point, p, is not equal to the vector, v. And by assigning coordinates to vectors will make it possible to represent the displacement from the origin frame to the point. Consider point one, p^1, that have

a displacement from frame zero, this vector will then be assigned the same notational convention used for assigning coordinate frames, also v_1^0.

To describe the orientation of one coordinate frame relative to another frame can be done using a rotation matrix. The way to do this is to project each of the axes in one frame onto another coordinate frame. Since taking the dot product of two unit vectors gives a projection of one onto the other, the unit vector of the coordinate frames is used to construct the rotation matrix. At such the resulting rotation matrix between, e.g., frame zero and one, where the orientation of frame one with respect to frame zero is given by:

$$R_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix} \tag{2.1.1.1}$$

The column vectors of R_1^0 are of unit length and mutually orthogonal; thus, this is an orthogonal matrix. These type of n x n matrices is referred to as Special orthogonal group of order n, SO(n). If then define: If then define:

$$\begin{aligned} x_1 \cdot x_0 = cos\theta \qquad & y_1 \cdot x_0 = -sin\theta \quad z_1 \cdot x_0 = sin\theta \\ x_1 \cdot y_0 = sin\theta \qquad & y_1 \cdot y_0 = cos\theta \quad z_1 \cdot y_0 = -sin\theta \\ x_1 \cdot z_0 = -sin\theta \qquad & y_1 \cdot z_0 = sin\theta \quad z_1 \cdot z_0 = cos\theta \end{aligned} \tag{2.1.1.2}$$

what is called the basic rotation around an axis can be defined. The basic rotation around the x-axis is shown in equation (2.1.1.3), around the y-axis (2.1.1.4), and z-axis (2.1.1.5).

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{bmatrix} \tag{2.1.1.3}$$

$$R_{y,\theta} = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \tag{2.1.1.4}$$

$$R_{z,\theta} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1.1.5}$$

Rotation matrixes can also be used to rotate a point with respect to the current frame or with respect to the fixed frame.

2.1.2 Rigid Motions

A pure translation, together with a pure rotation, is what defines a rigid motion. Moreover, the group of all rigid motions is called the Special Euclidean Group, SE(3), and is defined as SE(3)= $\mathbb{R}^3$xSO(3). Consider a point, p, that is rigidly attached to coordinates of frame one, with local coordinates p^1. Then to show coordinates of p with respect to frame zero and let R_1^0 be the rotation matrix representing the orientation of frame one with respect to frame zero. Moreover, let d be the vector from the origin of frame zero to the origin of frame one. Then point p with respect to zero can be expressed using:

$$p^0 = R_1^0 p^1 + d^0 \tag{2.1.2.1}$$

By adding another coordinate frame and a point p attached to frame two, this point can be expressed in either of the three coordinate frames. Then p^0 can be found as a composition of p in frame one,p^1, and frame zero, p^0, in the same way as in equation (2.1.2.2). Alternatively, since the relationship between frame zero and two is also a rigid motion, we can equally describe it as in (2.1.2.3). By comparing these two ways, we can see the following relationships in (2.1.2.4) and (2.1.2.5). These relationships show that frame zero and frame two can be expressed by a vector from the origin of frame zero to the origin of frame two where the coordinates will be the sum of the vector from the origin of zero to one with respect to frame zero, d_1^0, and the vector from the origin of one to two expressed in the orientation of the coordinate frame zero, $R_1^0 d_2^1$.

$$p^0 = R_0^1 R_2^1 + R_0^1 d_2^1 + d_1^0 \tag{2.1.2.2}$$

$$p^0 = R_0^2 p^2 + d_2^0 \tag{2.1.2.3}$$

$$R_2^0 = R_1^0 R_1^2 \tag{2.1.2.4}$$

$$d_2^0 = d_1^0 + R_1^0 d_2^1 \tag{2.1.2.5}$$

2.1.3 Homogeneous Transformation

Rigid motion can also be expressed in matrix form, which is practical when dealing with a long sequence of rigid motions; this is done by reducing the composition of rigid motions to matrix multiplications. By utilizing the relationships expressed in (2.1.2.4) and (2.1.2.5) with the matrix identity. The rigid motions can be represented by the set of matrices of the form:

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}, R \in SO(3), d \in \mathbb{R}^3 \tag{2.1.3.1}$$

where 0 is the row vector (0,0,0). These types of transformations matrices are called homogenous transformations. Moreover, since R is orthogonal the inverse transformation H^{-1} is given by:

$$H^{-1} = \begin{bmatrix} R^T & -R^T d \\ 0 & 1 \end{bmatrix} \tag{2.1.3.2}$$

To be able to perform matrix multiplication in order to represent the transformation in equation (2.1.2.1), the vectors p must be augmented by the addition of the fourth component of 1 like this:

$$P^0 = \begin{bmatrix} p^0 \\ 1 \end{bmatrix} \tag{2.1.3.3}$$

This type of vector is called homogenous representation of the vector p^0. Thus equation (2.1.2.1) becomes equivalent to the homogeneous matrix equation $P^0 = H_1^0 P^1$. And then a set of basic homogeneous transformations generating SE(3) can be given by

$$Trans_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad Rot_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) & 0 \\ 0 & sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.1.3.4}$$

$$Trans_{y,b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad Rot_{y,\beta} = \begin{bmatrix} cos(\beta) & 0 & sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(beta) & 0 & cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.1.3.5}$$

$$Trans_{z,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad Rot_{z,\gamma} = \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 & 0 \\ sin(\gamma) & cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.1.3.6}$$

for translation and rotation about the x, y, z-axes respectively.

2.2 Forward kinematics

By assuming all joints in a rigid motion have only a single degree of freedom, each joint action can be described by a single real number: the displacement case of a prismatic joint or angle of rotation in case of a revolute joint. "This assumption does not involve any real loss of generality, since such as a ball and socket joint (two degrees of freedom) or a spherical wrist (three degrees of freedom) can always be thought of as a succession of single degree-of-freedom joints with links of length zero in between" [35].

By attaching a coordinate frame rigidly to each joint and linking them together into a kinematic chain, performing a kinematic analysis is possible. Thus by attaching $o_i x_i y_i z_i$ to link i, the coordinates for each point on link i will be constant when expressed in the i^{th} coordinate frame for whatever motion executed. Consequently, when joint i is actuated, that link and attached coordinate frame will experience a resulting motion.

Consider the homogeneous transformation matrix A_i that represents the position and orientation of $o_i x_i y_i z_i$ with respect to $o_{i-1} x_{i-1} y_{i-1} z_{i-1}$. This matrix will vary as one of the configurations of the underlying succession of links in the system changes, because of the assumption that all joints are either revolute or prismatic means that A_i is a function of a single joint variable, q_i, as shown in equation (2.2.0.1). Thus the homogeneous transformation matrix in equation (2.2.0.2) will express the position and orientation of $o_i x_i y_i z_i$ with respect to $o_j x_j y_j z_j$.

$$A_i = A_i(q_i) \tag{2.2.0.1}$$

$$T_j^j = \begin{cases} A_{i+1} A_{i+2} \cdots A_{j-1} A_j & if \quad i < j \\ I & if \quad i = j \\ (T_j^j)^{-1} & if \quad i > j \end{cases} \tag{2.2.0.2}$$

To find the representation of the last object of n links of rigid objects with respect to the first can simply be found by multiplying A_n. Thus the position and orientation of the last object are given by:

$$H = T_n^0 = A_1(q_1) \cdots \cdot A_n(q_n) \tag{2.2.0.3}$$

where all homogeneous transformation A_i is of the form:

$$A_i = \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ 0 & 1 \\ & \end{bmatrix}, \tag{2.2.0.4}$$

To define A_i can be done by introducing further conventions, and in this thesis, the Denavit-Hartenberg representation of joint is used.

Denavit-Hartenberg convention

Denavit-Hartenberg convention or DH convention is a commonly used convention for selecting frames of reference in robotic applications [35]. This convention uses a product of four basic transformations to represent each homogeneous transformation A_i as shown below:

$$A_i = Rot_{z_i,\theta_i} Trans_{z_i,d_i} Trans_{x_i,a_i} Rot_{x_i,\alpha_i}$$

$$A_i = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & 0 \\ sin(\theta_i) & cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2.0.5}$$
$$* \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_i) & -sin(\alpha_i) & 0 \\ 0 & sin(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where $\boldsymbol{\theta_i}$ is the angle from x_{i-1} to x_i measured in a plane perpendicular to z_{i-1}, $\mathbf{d}_i$ is the distance from $origin_{i-1}$ to the intersection of x_i and z_{i-1}-axis measured along z_{i-1}, $\mathbf{a}_i$ is the distance between z_i and z_{i-1} measured along x_i-axis, and $\boldsymbol{\alpha_i}$ are the angle from z_{i-1} to z_i measured in a plane perpendicular to x_i.

Usually, six parameters are used to describe any homogeneous transformation matrix (three for rotation and 3 for translation). However, with a smart choice of placement for each link's coordinate frame, the description can be done with only four parameters instead of six. Just that is what the DH convention does by following two rules, and if these are fulfilled, can equation (2.2.0.5) for A_i be used. The first rule, DH1 , is the axis x_i is perpendicular to the axis z_{i-1}, and the second rule, DH2, is the axis x_i intersects the axis z_{i-1}.

2.3 Downhole sensor card

This chapter section will present the relevant theory for the design of the downhole sensor card. Will include communication protocols that were considered for the system with some of their pros and cons. And a short description of the components used (crystal oscillator, MEMS magnetometer, MEMS accelerometer)

2.3.1 USART

Universal Synchronous Asynchronous Receiver Transmitter (USART) is also called the Serial Communications Interface (SCI) [36]. The USART can be configured to be running in a full-duplex asynchronous system or as a half-duplex synchronous system. The synchronous operation uses a clock and data line; however, for asynchronous transmission, there is no separate clock accompanying the data. Therefore when an asynchronous operation, one pin is used for sending and another for receiving data. Hence both sending a receiving can be done simultaneously; this is what is known as full-duplex operation. Additionally, sending and receiving can be independently enabled. Furthermore, when using a synchronous operation, the receiver does not need to know the transmitter's baud-rate, as it is given from the clock signal. However, using the asynchronous operation, the receiver must know the transmitter's baud-rate before the inception of gathering.

2.3.2 USB

Universal Serial Bus (USB) is a standard developed in the mid-1990s that defines connectors, cables, and communications protocols used in bus connections, communication, and power supply between computers and electronic devices. USB is used to connect computer peripherals (such as keyboards, digital cameras, printers, disk drives, and network adapters) to a personal computer, but is also used for other devices such as smartphones and video game consols. USB is much more used than other varieties of earlier interfaces such as serial and parallel port.

A standard USB consists of 4 wires where one is for power (VCC), one for ground (GND), and two data wires (DATA+ and DATA-). The standard for the power supplied by the USB is 5 volts and a maximum 500mA. The max length of a USB cable is 5 meters but can be higher if the cable used is changed to a cable with lower resistance. The limit can be pushed to 10 meters because the most prominent problem is the capacitance/inductance in the line causing the signal to shift in time resulting in a timing issue.

2.3.3 SPI (Serial Peripheral Interface)

Motorola in the 1980s created the Serial Peripheral Interface (SPI) bus and is now a highly used short-distance interface bus [17], commonly used to send data between microcontrollers, sensors, and SD cards. Devises using SPI communicate in full-duplex mode using a single master. This interface supports multiple slave-devices through selection with individual slave select (SS), also called chip select (CS), lines.

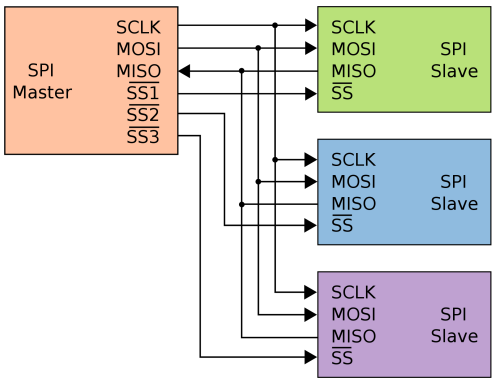


Figure 2: *This figure is taken from [30], and illustrate a single master and three slaves on a SPI bus*

SPI is a four-wire synchronous serial communication protocol, where three of the wires are common to all devices on the bus. One is the slave line for sending data to the master, called Master In Slave Out (MISO). Furthermore, the line from the master for sending data to the peripherals are called Master Out Slave In (MOSI). The last of the three common lines is the clock pulse which synchronizes data transmission generated from the master, called Serial Clock (SCK). The last of the four wires is the slave select, which is not common to all devices and will need a pin for all devices that the master can toggle to enable/disable the specific device. The slave device will only communicate with the master if the SS pin is low.

Moreover, when the SS pin is high, the slave will ignore any messages from the master. Therefore allows the system to have multiple devices connected to the same master. Consequently, the amount of SS pin on a master device will determine how many devices can be connected; this is also the only limiting factor when it comes to connectable devices.

2.3.4 I²C (Inter-Integrated Circuit)

Inter-Integrated Circuit (I²C) is a protocol that was initially developed by Philips (now NXP Semiconductors) in 1982 for use in their TV's [15]. I²C is a multi-user serial data bus that is used to connect lower speed IC's to microcontrollers and processors, over shorter distances where the wires can be up to 2-3 meters. I²C is a master-slave system that uses two lines over two wires to establish communication, called serial data (SDA) and serial clock (SCL). SDA is the wire that transmits data, and SCL is the clock line to sync the read/write operation.

According to the I²C-bus specification and user manual [16], both SDA and SCL are bidirectional lines. These lines connect to a positive supply voltage via a current-source or pull-up resistor. So when the lines are free, they are at logic HIGH. It also states that the output of the devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

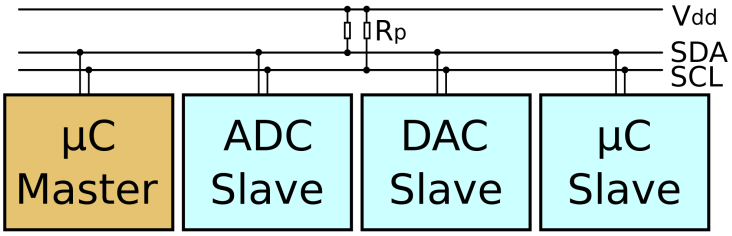


Figure 3: *This figure is taken from [31], and is an example on I²C schematic with one master (a microcontroller) and three slaves (an analog-to-digital converter (ADC), a digital-to-analog converter (DAC), and a microcontroller), and pull-up resistors*

[16]By pulling the line to ground, the device will output a logic LOW, and a logic HIGH output by letting the line float (output high impedance) so that the pull-up resistor pulls it high. Not actively raising the voltage allows multiple nodes to connect to the bus without short circuits from signal contention. The process of pulling down the level and releasing it will take some time. This time depends on bus termination, serial resistors, capacitance, cable length, and bus voltage.

In the I²C user manual [16], it is stated that the levels of the logical LOW and HIGH are not fixed and are depending on the associated level of VDD. The input reference level for LOW is at 30% of VDD and 70% of VDD for HIGH.

In the I²C user manual [16], it is stated that the levels of the logical LOW and HIGH are not fixed and are depending on the associated level of VDD. The input reference level for LOW is at 30% of VDD and 70% of VDD for HIGH. Moreover, a strong pull-up (small resistor) prevents the voltage from

being driven LOW, thus the maximum voltage level that can be read as a valid LOW (30% of VDD) determines the minimum pull-up resistor. The bus capacitance limits the maximum pull-up resistance due to the standard rise time for I^2. If the pull-up resistor value is too high, the line will not rise to a logical HIGH before it is pulled low. The maximum pull-up resistor is a function of the maximum rise time (t_r) in seconds:

$$R_p(max) = \frac{t_r}{0.8473 \cdot C_b} \tag{2.3.4.1}$$

where C_b is the capacitive load for each bus line, measured in Farhad. There is a trade of when choosing the right pull-up resistor, where A smaller resistor will give a higher bus speed because of less RC delay, and a larger resistor will give lower power consumption (EDN,2012)[18].

I^2C works by assigning a different address to each connected device on the bus. The address space consists of 10 bits, so the maximum number of devices on one bus is 1024. There is som physical restriction like bus space, bus capacitance (maximum for standard and fast mode is 400, 500 for fast mode plus), and that is designed to work within short distances. I^2C can operate at different speeds, standard mode at 100 kHz, fast mode at 400 kHz, and fast mode plus 1MHz.

2.3.5 Copper Trace Capacitance

According to Mark, in his article about I^2C design mathematics (Mark. 2018)[19], significant sources of capacitance comes from "copper trace capacitance," which is referred to as a single trace over dielectric overground. If the ICs are far from each other, can the capacitive coupling between the I^2C signal lines and the ground plane be significant. A solution suggested to reduce the coupling effects between the two signal lines is to separate them by several trace widths.

Mark also gives a way to calculate the capacitive coupling between a signal and a ground plane, shown in equation (2.3.5.1) (Mark. 2018)[19].

$$C[pf] \approx \frac{0.264\frac{[pf]}{[mm]}(4.5 + 1.41)}{Ln\left(\frac{0.598\frac{1}{[mm]}\cdot\text{layer height}}{0.08\frac{1}{[mm]}\cdot\text{trace width}+0.1\frac{1}{[mm]}\cdot\text{trace thickness}}\right)} \tag{2.3.5.1}$$

2.3.6 Crystal Oscillator

This type of oscillator circuits uses a quartz crystal as a piezoelectric resonator to provide a stable clock signal for digital IC. According to Steven Bible, (Bible, 2020)[21], quartz crystals can be modeled as an RLC circuit that has a very high-quality factor (Q factor), figure [4] shows an equivalent model of a crystal circuit. The high Q factor makes the quartz a significant component for oscillators since a high-quality factor will contribute to high-frequency stability for the oscillator circuit. [21] The ratio between reactance and operating frequency of the crystal defines the quality factor for quartz.

As written In Crystal oscillator Basics (Bible, 2020)[21], "A crystal has two resonant frequencies characterized by a zero phase shift," which is series- and parallel-resonant. Steve describes in his article (steve, 2020)[22] that at series-resonant, the crystal appears as a series resonant RLC circuit with zero phase shift. Furthermore, if to be used in an oscillator, there needs to be provided a 360-degree phase shift. He further states that at parallel-resonant, the crystal appears inductive, and with external capacitors, called load capacitance, offers 180 degrees of phase shift at the specified frequency. Moreover, typically, a microcontroller will have an inverting amplifier that provides the other 180 degrees.

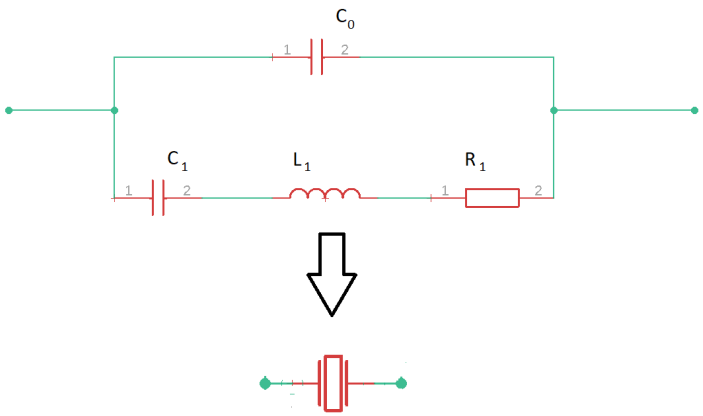


Figure 4: *The figure shows the equivalent circuit for a crystal. Where:* C_1 *represents motional arm capacitance measured in Farads.* L_1 *represents motional arm inductance measured in Henrys.* R_1 *represents resistance measured in ohms.* C_0 *represents stray/parasitic capacitance measured in Farads.*

2.3.7 MEMS Gyroscope

MEMS gyroscope, also called MEMS angular rate sensor, what separates it from other means of measuring rotation is that it does not require to have a fixed point of reference like a tachometer or potentiometer (Silicon Sensing, 2020)[8]. In 1990 did Silicon Sensing produce the first MEMS vibrating structure gyroscope, which resulted in small, widely available, and in inexpensive gyroscopes to measure rat of turning ($^0/s$). Measuring the angular rate is done utilizing the principle for centripetal force. Because of how gyroscopes work, it can only measure angular velocity and is not affected by acceleration or linear velocity.

Measuring the angular velocity is done by measuring the centripetal force on the sensor (Watson, 2020)[9]. Measuring the force applied to a mass on the radial axis on one of the axes of the sensor, the centripetal force on that axis can be measured. According to Dilip Raja (Raja, 2019)[10], this force is measured by connecting the mass to a frame with springs. This frame has capacitor plates that pare up with capacitor plates on an outer frame, as illustrated in figure [5] . When under a sudden change in direction, there will be a displacement in the inner layer.

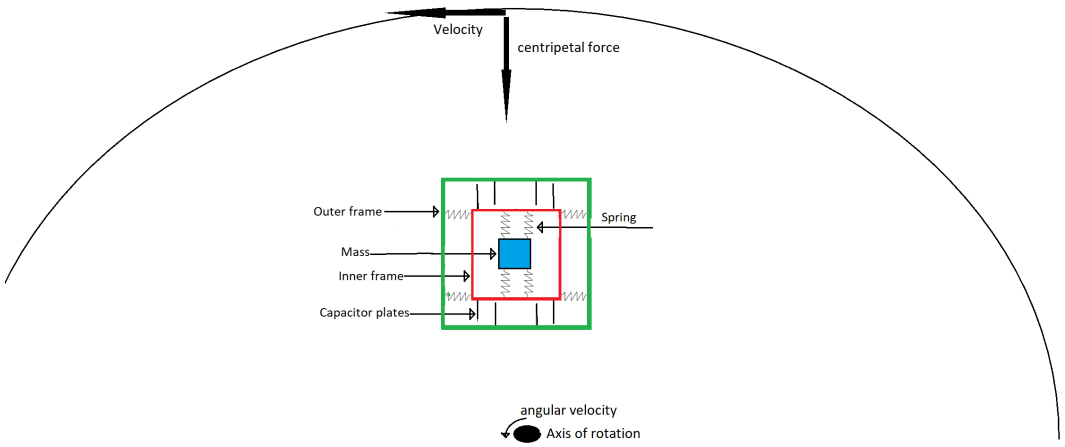


Figure 5: *This figure illustrates an example of how a gyroscope works and how it can measure the angular rate.*

Furthermore, will this displacement directly result in the distance between the capacitor plates on the bottom and the top to change. The change in distance between the capacitor plates will result in a change in the capacitance. This change will be measured, and from this measurement, the angular velocity can be found.

Since perfection is not typically available on economic manufacturing processes, an imperfection in the gyroscope will be made. This and noise will result in a zero offset in the gyroscope measurements.

2.3.8 MEMS Magnetometer

MEMS magnetometer utilizes the basis of Hall effect or Magneto Resistive effect (Raja, 2019)[10]. The sensor used in this project uses the Hall effect to measure the magnetic field strength. Figure [] illustrates the principle of a Hall effect sensor. By benefiting from two important magnetic characteristics, flux density (B) and polarity (north and south), the sensor can determine the magnetic field's strength and direction (Electronics Tutorials, 2020)[13].

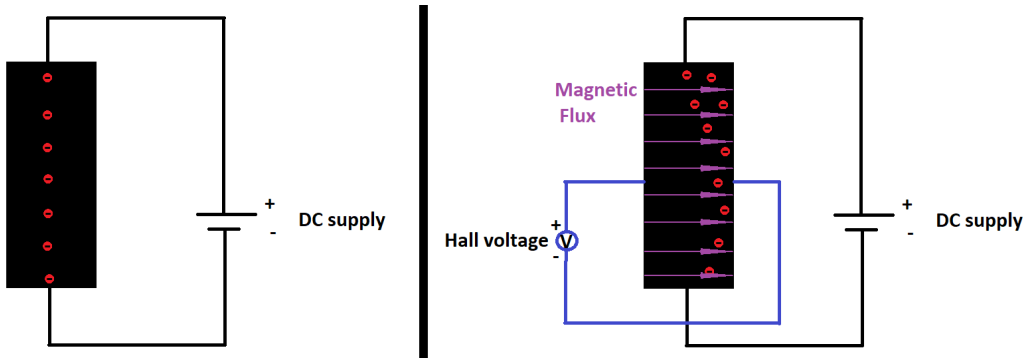


Figure 6: *Illustrates the Hall effect on a semiconductor*

Hall effect sensors contain a thin rectangular p-type semiconductor connected to a DC power supply in a closed-loop circuit [13]. Hence a low current will pass through the semiconductor. If a magnetic field goes through the semiconductor, the magnetic flux lines will put a force on the passing electrons, diverting them out. Consequently, the electron density will be higher on one side resulting in a potential difference across the semiconductor. The potential difference is the Hall voltage and determines the strength and direction of the magnetic field.

Earth's magnetic flux lines can be treated as straight lines with constant direction and magnitude. Moreover, are magnetic flux independent of the position and orientation of the sensor. Therefore magnetometers can use it as a reference to estimate the heading (north/south).

The magnetometer will be effected by neer by electrical equipment and rotating parts; this will result in a magnetic distortion. Magnetic distortion divided opp in soft-iron- and hard iron distortion. Hard-iron distortion is an offset in measurements resulting from objects that produce an additive field to the earth's magnetic field. Examples of object sources of hard-iron can be any electrical devices or electromagnets (Konvalin, 2019)[14]. The measurement offset will be constant as long as the field's orientation and position are constant relative to the sensor. Typically are hard-iron corrections determined by sampling through rotation on at least 360 degrees. Next is to identify the average of the maximum and minimum values for each axis in the circle (made

by the samples) to identify the offset. This offset will be the average distance from the samples center point to (0,0). The offset will be subtracted from the raw data, which will eliminate most of the hard-iron distortion (Konvalin, 2019)[14]. Soft-iron distortion is the result of materials that distort or influences the magnetic field but not necessarily generate a magnetic field itself. Examples of object sources of soft-iron can be iron or nickel [14]. Soft-iron distortion is dependent on the orientation of the material relative to the sensor and magnetic field. The distortion will . Therefore the distortion can not be compensated by one constant like hard-iron. Figure (7) illustrates the hard-iron- and soft-iron distorted magnetic field, and one undistorted magnetic field in the sensors x,y coordinates.

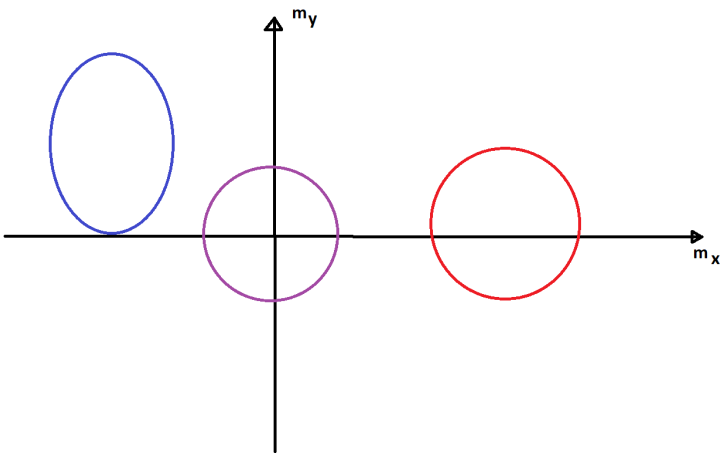


Figure 7: *Illustrates magnetic distortion where the purple circle is undistorted, read is hard-iron distortion with an offset from the origin, and green is both soft- and hard-iron distortion with an offset from the origin and an ellipsoidal stretch of the axis scale.*

The measurements can be calibrated by solving the coefficient matrix **K** and vector **b** in equation (2.3.8.2), regardless of error source (Die, et.al., 2011)[11]. Which is derived from the mathematical model for the magnetometer measurements effected by the environmental interference, sown in equation (2.3.8.1) (Die, et.al., 2011)[11].

$$B_r = CB_c + b + n \tag{2.3.8.1}$$

$$B_c = C^{-1}(B_r - b) = K(B_r - b) \tag{2.3.8.2}$$

B_r in the equations above is the raw output from the magnetometer, B_c is the ideal output, **b** is the vector containing the hard magnetic error vector. Furthermore, **C** is the matrix containing the scale factor error and the soft magnetic error.

2.3.9 MEMS Accelerometer

Figure (8) shows the working principle of a MEMS accelerometer that measures the capacitance change to measure acceleration. MEMS accelerometer works by having a mass connected to springs attached in such a way that the mass can only move along one direction (Raja, 2019)[10]; this mass has conductor plates connected to its sides [10]. These plates will pare up whit two fixed conductor plates. The result is that it will make differential capacitor pairs that will generate different capacitance when the mass is moving, as illustrated in figure (9). The change in capacitance will be measured, processed, and will correspond to a particular acceleration value.

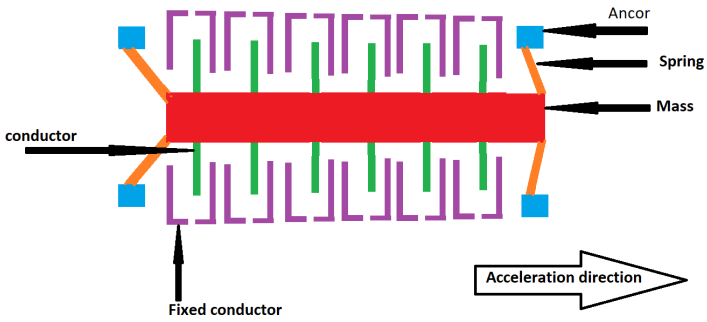


Figure 8: *This figure illustrates the working principle of a MEMS accelerometer.*

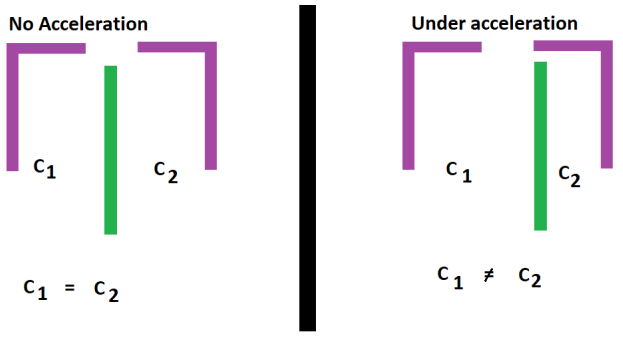


Figure 9: *This figure illustrates the differential capacitor when the accelerometer is not under acceleration and under acceleration*

Accelerometers will have a constant direction and magnitude of approximately 9.81 m/ss or 1g perpendicular to the earth's surface because of the earth's gravitational acceleration. When stationary, the sensor can use the gravitational acceleration as a reference direction for up and down with high precision.

Since MEMS accelerometers are very sensitive, noise can be expected even when the sensor is stationary. Measurement noise resulting from electrical

disturbance from other electrical equipment or electromagnetic interference can be expected. The measurements are prone to errors and can be modeled as in equation (2.3.9.1) (D'Emilia, et.al., 2011)[12].

$$\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} + \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} \tag{2.3.9.1}$$

where the diagonal in matrix S_{ij} are scaling factors along each axis and the other parameters in the matrix are scaling factors between any two axes. V_i is the measured values in x,y, and z, A_i is the true values of the measured values in the x,y, and z, and **Q** represents the zero offset array (D'Emilia, et.al., 2011)[12].

2.4 Least Squares

The least square approximation is a method of fitting an estimated regression line to the data. The best fit is determined by minimizing the sum of squared residuals. A residual is is the difference between observed and the estimated value provided by a model. This section will only explain the essential parts of the least square method. The following publications (Adams and Essex, 2014)[46], (Walpole, 2016)[47], and (Theodoridis, 2015)[48], can provide more information about the least square.

For a general linear model with independent variables $X_i[t]$ for a dependent variable Y[t] at measurement t for the fitted model parameters β_i is:

$$Y[t] = \beta_0 X_0[t] + \beta_1 X_1[t] + \cdots + \beta_{(N-1)} X_{(N-1)}[t] \tag{2.4.0.1}$$

where the error term r[t] for the fit to the model is defined as:

$$r[t] = Y[t] - \beta_0 X_0[t] - \beta_1 X_1[t] - \cdots - \beta_{(N-1)} X_{(N-1)}[t] \tag{2.4.0.2}$$

Equation (2.4.0.2) can for a series of **M** measurements be written in the form:

$$r = Y - X\beta \tag{2.4.0.3}$$

where the error residual r[t] is the column vector **r** defined as equation (2.4.0.4), Y is a column vector of M measurements on the dependent variable defined by (2.4.0.5), X is the MxN matrix of M measurements of the independent variable seen in equation (2.4.0.6), and β is a column vector of unknown coefficients to be determined.

$$r = \begin{bmatrix} r[0] \\ r[1] \\ .. \\ r[M-1] \end{bmatrix} \tag{2.4.0.4}$$

$$Y = \begin{bmatrix} Y[0] \\ Y[1] \\ .. \\ Y[M-1] \end{bmatrix} \tag{2.4.0.5}$$

$$X = \begin{bmatrix} X_0[0] & X_1[0] & .. & X_{N-1}[0] \\ X_0[1] & X_1[1] & .. & X_{N-1}[1] \\ .. & .. & .. & .. \\ X_0[M-1] & X_1[M-1] & .. & X_{N-1}[M-1] \end{bmatrix} \tag{2.4.0.6}$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ .. \\ \beta_{N-1} \end{bmatrix} \tag{2.4.0.7}$$

The equations can be solved using least squares if there are more measurements M than unknowns N by minimizing an error function E defined as the modulus squared of the vector of residuals r defined in equation (2.4.0.4):

$$E = \sum_{t=0}^{M-1} r^2[t] = ||r||^2 = ||Y - X\beta||^2 = (Y - X\beta)^T(Y - X\beta) \tag{2.4.0.8}$$

E contains dimensions of the square of elements r[t] and is proportional to the number of measurements **M** so can use the normalized error measure ϵ:

$$\epsilon = 0.5\sqrt{\frac{E}{M}} \tag{2.4.0.9}$$

Case of non-homogeneous
In case of non-homogeneous equations, also if the measurements vector dependent **Y** is not zero, the error function will be a minimum when it is stationary with respect to any perturbation about the optimal least squares solution:

$$\lim_{\delta\beta \to 0} \{E(\beta + \delta\beta)\} - \{E(\beta)\} = 0 \quad for \quad all \quad \delta\beta \tag{2.4.0.10}$$

Then the equation be comes:

$$E(\beta+\delta\beta)-E(\beta) = (Y-X(\beta+\delta\beta))^T(Y-X(\beta+\delta\beta)) - (Y-X\beta)^T(Y-X\beta)$$
$$= -Y^TX\delta\beta + X^T\beta^T\delta\beta - \delta\beta^TX^TY + \delta\beta^TX^TX\beta = 0 \tag{2.4.0.11}$$

Then using the fact that $\delta\beta^TX^TY$ and $\delta\beta^TX^TX\beta$ are scalars and their values will be unchanged by the transpose operation and the equation (2.4.0.11) can be written as :

$$\beta = (-Y^TX + X^T\beta^T)\delta\beta = 0 \Rightarrow Y^TX = \beta^TX^TX \Rightarrow \beta = (X^TX)^{-1}X^TY \tag{2.4.0.12}$$

Then this equation denotes the normal equation soulution for β in the non-homogeneous case. Thus equation (2.4.0.8) at the optimum solution when substituting β with equation (2.4.0.12) gives:

$$\begin{aligned} E &= (Y-X\beta)^T(Y-X\beta) \\ &= (Y-X((X^TX)^{-1}X^TY))^T(Y-X((X^TX)^{-1}X^TY)) \\ &= Y^TY - Y^TX((X^TX)^{-1})^TX^TY \end{aligned} \tag{2.4.0.13}$$

In the special case where number of measurements equals the number of model parameters to be fitted, the matrix $\mathbf{X}$ is square and the transpose and inverse operators are commute. The error function is in this case expected result is then zero and the fit is perfect when the number of measurements equals the number of model parameters to be fitted.

$$\begin{aligned} E &= Y^TY - Y^TX((X^TX)^{-1})^TX^TY \\ &= Y^TY - Y^TXX^{-1}(X^{-1})^TX^TY \\ &= Y^TY - Y^T(X^{-1})^TX^TY = 0 \end{aligned} \tag{2.4.0.14}$$

Case of homogeneous
When the dependent measurement vector $\mathbf{Y}$ is zero the equation is homogenous. Thus the model being fitted with least squares in now:

$$X\beta = 0 \tag{2.4.0.15}$$

Hence the error function $\mathbf{E}$ to be minimised becomes:

$$E = ||r||^2 = ||X\beta||^2 = (X\beta)^T(X\beta) \tag{2.4.0.16}$$

Then by using the normal equation solution in equation (2.4.0.12) will for a homogeneous case give a zero vector solution which is not very useful. Thus a nother method is needed to minimize the error function in (2.4.0.16) where the constraint that β has non-zero magnitude. β can be scaled to have unit magnitude since equation (2.4.0.15) is linear:

$$1 - \beta^T \beta = 0 \tag{2.4.0.17}$$

Adding multiples of (2.4.0.17) will not effect the error function and can be written as:

$$E = (X\beta)^T (X\beta) + \lambda(1 - \beta^T \beta) \tag{2.4.0.18}$$

Then the error function is equivalent to Lagrange Multipliers for constrained optimization. By using the stationary constraint $E(\beta + \delta\beta) = E(\beta)$ on the error function gives:

$$(X(\beta+\delta\beta))^T (X(\beta+\delta\beta)) + \lambda(1-(\beta+\delta\beta)^T(\beta+\delta\beta)) = (X\beta)^T(X\beta) + \lambda(1-\beta^T\beta) \tag{2.4.0.19}$$

Ignoring second order terms gives:

$$\delta\beta^T X^T X\beta + \beta^T X^T X\delta\beta - \lambda(\beta^T\delta\beta + \delta\beta^T\beta) = 0 \tag{2.4.0.20}$$

Every term in equation (2.4.0.20) is a scalar and equal to its transpose hence the optimum β that constrains the performance function is:

$$\delta\beta^T X^T X\beta - \lambda\delta\beta^T\beta = 0 \Rightarrow X^T X\beta = \lambda\beta \tag{2.4.0.21}$$

From equation 2.4.0.21 gets that the required solution vector β is an eigenvector of the product $X^T X\beta$ associated with eigenvalue λ. Thus the error function E_i associated with eigenvalue λ_i and eigenvector $X^T X\beta_i$ is:

$$E_i = (X\beta_i)^T (X\beta_i) = \beta_i^T X^T (X\beta_i) = \lambda_i \beta_i^T \beta_i = \lambda_i \tag{2.4.0.22}$$

Where E_i is the i-th eigenvector solution and equals the eigenvalue λ_i of $X^T X$, and the smallest eigenvalue equals the minimum error function. Therefore, the required solution β is the eigenvector associated with the smallest eigenvalue.

2.5 Quaternion

This section will present only some of the fundamental algebraic concepts behind the idea of quaternions, for a more in-depth description of this mathematical instrument read the following publications (John Vince,2011)[23] and (Kuipers, 1999)[24]. Quaternion is a set of mathematical operators that are used to rotate and stretch vectors. A quaternion is a combination of a scalar and a vector, where the scalar is a real number, and the vector is a set of imaginer numbers. Equation 2.5.0.1 and 2.5.0.2 shows two ways of representing a quaternion, where x,y,z is the vector part, and s is the scalar.

$$q = \begin{bmatrix} s & x & y & z \end{bmatrix} \tag{2.5.0.1}$$

$$q = s + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \tag{2.5.0.2}$$

Quaternion adding and subtracting is done in the same way as a normal vector. However, unlike vectors, two quaternions are equal only if their corresponding terms are equal. Moreover, when multiplying quaternions, the following rules about the imaginary part must be followed:

$$\mathbf{i}^2 = \text{-1}, \quad \mathbf{j}^2 = \text{-1}, \quad \mathbf{k}^2 = \text{-1}, \quad \mathbf{ijk}^2 = \text{-1}$$
$$\mathbf{ij} = \mathbf{k}, \quad \mathbf{jk} = \mathbf{i}, \quad \mathbf{ki} = \mathbf{j}, \quad \mathbf{ji} = \text{-}\mathbf{k}, \quad \mathbf{kj} = \text{-}\mathbf{i}, \quad \mathbf{ik} = \text{-}\mathbf{j}$$

By following this set of rules for the two quaternions $q_1 = x_1\mathbf{i} + y_1\mathbf{j} + z_1\mathbf{k}$ and $q_2 = x_2\mathbf{i} + y_2\mathbf{j} + z_3\mathbf{k}$, and that quaternion multiplication follows the Kronecker product ($\otimes$) of matrices then the following equation will be given for any product of two quaternions:

$$\begin{aligned} q_1 \otimes q_2 = {} & (s_1s_2 \text{ - } x_1x_2 \text{ - } y_1y_2 \text{ - } z_1z_2) + (s_1x_2 + s_2x_1 + y_1z_2 \text{ - } y_2z_1)\mathbf{i} \\ & + (s_1y_2 + s_2y_1 + z_1x_2 \text{ - } z_2x_1)\mathbf{j} + (s_1z_2 + s_2z_1 + x_1y_2 \text{ - } x_2y_1)\mathbf{k} \end{aligned} \tag{2.5.0.3}$$

The magnitude or norm of a quaternion equals:

$$|q| = \sqrt{s^2 + x^2 + y^2 + z^2} \tag{2.5.0.4}$$

furthermore, a unit quaternion has a magnitude equal to one. Furthermore, a quaternion can be normalized to a unit quaternion $\hat{q}$ by the following equation:

$$\hat{q} = \frac{q}{|q|} \tag{2.5.0.5}$$

When it comes to conjugate a quaternion, the principle of conjugation for complex numbers will be important. As in a complex number c = a + **i** when

conjugated becomes c*=a - b**i**. Which gives the equation below that can be used for any quaternion.

$$\begin{aligned} q &= s + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \\ q^* &= s \text{ - } x\mathbf{i} \text{ - } y\mathbf{j} \text{ - } z\mathbf{k} \end{aligned} \tag{2.5.0.6}$$

A quaternion rotation can be defined as a frame or a point-set rotation, as stated in the theorem below.

"For any unit quaternion

$$q = q_0 + q = cos\theta + u \cdot sin\theta \tag{2.5.0.7}$$

and for any vector $\mathbf{v} \in \mathbb{R}^3$ *the action of the operator*

$$L_{q^*}(v) = q^* \cdot v \cdot q \tag{2.5.0.8}$$

may be interpreted geometrically as

- *a rotation of the coordinate frame with respect to the vector* $\boldsymbol{v}$ *through an angle* 2θ *about* $\boldsymbol{q}$ *as the axis, or,*
- *an opposite rotation of the vector* $\boldsymbol{v}$ *with respect to the coordinate frame through an angle* 2θ *about* $\boldsymbol{q}$ *as the axis."* (Kuipers, 1999)[24]

2.6 Attitude and Heading Reference Systems

Attitude and Heading Reference Systems (AHRS) is a system that contains three-axis sensors that provides attitude information of an object, including roll, pitch, and yaw (wiki, 2020)[25]. This systems sensor pack is referred to as MARG (Magnetic, Angular Rate, and Gravity) sensor since they either consist of solid-sate or microelectromechanical systems (MEMS) magnetometer, gyroscope, and accelerometer [25]. In comparison to an inertial measurement unit (IMU) that can only measure an attitude relative to the direction of gravity, an AHRS can provide a complete measurement of orientation relative to the earth's gravity and magmatic field.

To provide the 3D orientation (roll, pitch, and yaw) of a sensor, AHRS will integrate gyroscope measurements and fusing it whit accelerometer and magnetometer measurements (xsens, 2020)[26]. Using sensor fusion will compensate for gyroscope drift resulting from an accumulated error following from the integration of errors in the measurements of the gyroscope (Madgwick, 2010)[27] (Madgwick, et.al., 2010)[28].

2.7 Madgwick sensor fusion algorithm

Sebastian O.H. Madgwick (Madgwick, 2010)[27] wanted to develop a sensor fusion algorithm that could perform as well as a high-quality commercial Kalman-based system. However, an algorithm that is less complicated to implement and can use a lower sampling rate than the demanded sampling rate for a Kalman process (Madgwick, 2010)[27]. Also, to make the algorithm less computationally heavy compared to a Kalman filter when used to linearise a state relationship describing rotational kinematics in three dimensions. Since this typically require large state vectors to linearise the problem. There have been many approaches to address this issue, e.g., Mahony developed a filter that has been an efficient and effective solution, but performance only validated for an IMU [27]. Madgwick made an orientation filter that can be used for both IMU and MARG sensors, that addresses these issues of computational load and parameter tuning associated with Kalman. The rest of this section will describe the critical points of the Madgwick filter when using a tri-axis MARG sensor for a more detailed description read (Madgwick, 2010)[27] (Madgwick, et.al., 2010)[28].

Madgwick's sensor fusion algorithm uses a quaternion representation of orientation to describe the rotational kinematics in three dimensions. This filter uses quaternions to not subject to the singularities problems associated with Euler angle representation (Madgwick, 2010)[27]. The rest of this section will explain how the filter gets orientation from angular rate, gets orientation from vector observations, fusion algorithm works, the magnetic and gyroscope distortion compensation.

Orientation from angular rate

To get the orientation from angular rate from a tri-axis gyroscope measurements about the x, y, and z axes of the sensor frame, represented as ω_x, ω_y and ω_z respectively. According to Madgwick (Madgwick, 2010)[27], equation (2.7.0.2) calculates the quaternion derivative describing the rate of change in orientation of the earth frame relative to the sensor frame ${}^{S}_{E}\dot{q}$ at time **t**. Where vector ${}^{S}\omega_t$ defined in equation (2.7.0.1) contains the angular rate in rad/s measured at time **t**. Furthermore, he says that the equation (2.7.0.3) can calculate the orientation of the earth frame relative to the sensor frame at the time **t**, ${}^{S}_{E}$ q $_{\omega,t}$, provided that the initial conditions are known. Where ${}^{S}_{E}\hat{q}_{est,t-1}$ is the privous estimation of orientation, and Δt in the equation is the sample period.

$$ {}^{S}\omega = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \end{bmatrix} \tag{2.7.0.1} $$

$$ {}^{S}_{E}\dot{q}_{\omega,t} = \frac{1}{2}\hat{q}_{est,t-1} \otimes^{S} \omega_t \tag{2.7.0.2} $$

$$ {}^{S}_{E}q_{\omega,t} = {}^{S}_{E}\hat{q}_{est,t-1} + {}^{S}_{E}\dot{q}_{\omega,t}\Delta t \tag{2.7.0.3} $$

Orientation from magnetometer and accelerometer

Madgwick states (Madgwick, 2010)[27] that the orientation from a tri-axis accelerometer and magnetometer in the context of an orientation filter will initially assume that an accelerometer would measure only gravity, and a magnetometer will only measure the earth's magnetic field. The orientation of the sensor frame relative to the earth frame can be calculated if the direction of an earth's field is known in the earth frame. He also says that there will not be a unique sensor orientation solution for any given measurements. However, a quaternion representation must find a complete solution, and this can be done by the formulation of an optimization problem and using the rotation equation (**??**), where an orientation of the sensor, ${}^{S}_{E}\hat{q}$, is aligned in the direction of the earth's field in the earth frame, ${}^{E}\hat{d}$, with the sensor frame measured in the direction of the field, ${}^{S}\hat{s}$. Therefore by using the solution to (2.7.0.6) where (2.7.0.5) defines the objective function, we find a estimated orientation ,${}^{S}_{E}\hat{q}$, that will be used in an optimization algorithm.

$$\begin{aligned} {}^{S}\hat{s} &= \begin{bmatrix} 0 & s_x & s_y & s_z \end{bmatrix} \\ {}^{E}\hat{d} &= \begin{bmatrix} 0 & d_x & d_y & d_z \end{bmatrix} \\ {}^{S}_{E}\hat{q} &= \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix} \end{aligned} \tag{2.7.0.4}$$

$$f({}^{S}_{E}\hat{q}, {}^{E}\hat{d}, {}^{S}\hat{s}) = {}^{S}_{E}\hat{q}^{*} \otimes {}^{E}\hat{d} \otimes {}^{S}_{E}\hat{q} - {}^{S}\hat{s} \tag{2.7.0.5}$$

$$\min_{{}^{S}_{E}\hat{q} \in \Re^4} f({}^{S}_{E}\hat{q}, {}^{E}\hat{d}, {}^{S}\hat{s}) \tag{2.7.0.6}$$

The optimization algorithm that Madgwick uses is the gradient descent algorithm. This algorithm that is used, is described in its general form applicable to a field predefined in any direction in equation (2.7.0.8). For n iterations that result in orientation estimate of ${}^{S}_{E}\hat{q}_{n+k}$ which bases on a set initial orientation ${}^{S}_{E}\hat{q}_0$ and step-size μ. The gradient of the solution surface computed by using equation (2.7.0.7) is defined by the object function (2.7.0.5) and its Jacobian.

$$\nabla f({}^{S}_{E}\hat{q}, {}^{E}\hat{d}, {}^{S}\hat{s}) = J^{T}({}^{S}_{E}\hat{q}_k, {}^{E}\hat{d}) * f({}^{S}_{E}\hat{q}_k, {}^{E}\hat{d}, {}^{S}\hat{s}) \tag{2.7.0.7}$$

$${}^{S}_{E}q_{n+k} = {}^{S}_{E}\hat{q}_k - \mu \frac{\nabla f({}^{S}_{E}\hat{q}, {}^{E}\hat{d}, {}^{S}\hat{s})}{||\nabla f({}^{S}_{E}\hat{q}, {}^{E}\hat{d}, {}^{S}\hat{s})||}, k = 0, 1, 2....n \tag{2.7.0.8}$$

The gradient descent algorithm is then simplified to only have components within 1 or 2 of the global coordinate frame's principal axis. For the accelerometer measurements, it is fitting to assume that the direction of gravity can define the vertical axis z, as in equation (2.7.0.9). Thus substituting ${}^{E}\hat{d}$ in (2.7.0.7) with ${}^{E}\hat{g}$(2.7.0.9) and the normalized accelerometer measurements ${}^{S}\hat{a}$ (2.7.0.10) (in g force) for ${}^{S}\hat{s}$ in (2.7.0.7) gives equation (2.7.0.11) and (2.7.0.12).

$${}^{E}\hat{g} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7.0.9}$$

$$^{S}\hat{a} = \begin{bmatrix} 0 & a_x & a_y & a_z \end{bmatrix} \tag{2.7.0.10}$$

$$f_a(^{S}_{E}\hat{q}, ^{S}\hat{a}) = \begin{bmatrix} 2(q_2q_4 - q_1q_3) - a_x \\ 2(q_1q_2 - q_3q_4) - a_y \\ 2(0.5 - q_2^2 - q_3^2) - a_z \end{bmatrix} \tag{2.7.0.11}$$

$$J_a(^{S}_{E}\hat{q}) = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \tag{2.7.0.12}$$

The same will be done for the magnetometer measurements. However, the earth's magnetic field has components in the horizontal axis and the vertical axis. Which is represented in (2.7.0.13), and this will be substituting $^{E}\hat{d}$ and normalized magnetometer measurements (2.7.0.14) (in Gauss) for $^{S}\hat{s}$ in (2.7.0.7). Consequently it gives equation (2.7.0.15) and (2.7.0.16).

$$^{E}\hat{b} = \begin{bmatrix} 0 & b_x & 0 & b_z \end{bmatrix} \tag{2.7.0.13}$$

$$^{S}\hat{m} = \begin{bmatrix} 0 & m_x & m_y & m_z \end{bmatrix} \tag{2.7.0.14}$$

$$f_m(^{S}_{E}\hat{q}, ^{E}\hat{b}, ^{S}\hat{m}) = \begin{bmatrix} 2b_x(0.5 - q_3^2 - q_4^2) + 2b_z(q_2q_4 - q_1q_3) - m_x \\ 2b_x(q_2q_3 - q_1q_4) + 2b_z(q_1q_2 + q_3q_4) - m_y \\ 2b_x(q_1q_3 + q_2q_4) + 2b_z(0.5 - q_2^2 - q_3^2) - m_z \end{bmatrix} \tag{2.7.0.15}$$

$$J_m(^{S}_{E}\hat{q}, ^{E}\hat{b}) = \begin{bmatrix} -2b_zq_3 & 2b_zq_4 & -4b_xq_3 - 2b_zq_1 & -4b_xq_4 + 2b_zq_2 \\ -2b_xq_4 + 2b_zq_2 & 2b_xq_3 + 2b_zq_1 & 2b_xq_2 + 2b_zq_4 & -2b_xq_1 + 2b_zq_3 \\ 2b_xq_3 & 2b_xq_4 - 4b_zq_2 & 2b_xq_1 - 4b_zq_3 & 2b_xq_2 \end{bmatrix} \tag{2.7.0.16}$$

To get a unique orientation , more then one field is needed, measurements and reference direction of both the earth's magnetic field and gravity are combined, as shown in (2.7.0.17) and (2.7.0.18).

$$f_{a,b}(^{S}_{E}\hat{q}, ^{S}\hat{a}, ^{E}\hat{b}, ^{S}\hat{m}) = \begin{bmatrix} f_a(^{S}_{E}\hat{q}, ^{S}\hat{a}) \\ f_m(^{S}_{E}\hat{q}, ^{E}\hat{b}, ^{S}\hat{m}) \end{bmatrix} \tag{2.7.0.17}$$

$$J_{a,b}(^{S}_{E}\hat{q}, ^{E}\hat{b}) = \begin{bmatrix} J_a^T(^{S}_{E}\hat{q}) \\ J_m^T(^{S}_{E}\hat{q}, ^{E}\hat{b}) \end{bmatrix} \tag{2.7.0.18}$$

Filter fusion algorithm

Thus to calculate the estimate orientation $^{S}_{E}\hat{q}_{\nabla,t}$ at time t by sensor measurements $^{S}\hat{a}_t$ and $^{S}\hat{m}_t$ sampled at time **t** can be computed by equation (2.7.0.20). Subscript ∇ indicates that the quaternion is calculated using the gradient descent algorithm.

$$\nabla f = J_{a,b}^T(^{S}_{E}\hat{q}_{est,t-1}, ^{E}\hat{b}) * f_{a,b}(^{S}_{E}\hat{q}_{est,t-1}, ^{S}\hat{a}, ^{E}\hat{b}, ^{S}\hat{m}) \tag{2.7.0.19}$$

$$ {}^{S}_{E}q_{\nabla,t} = {}^{S}_{E}\hat{q}_{est,t-1} - \mu\frac{\nabla f}{||\nabla f||} \tag{2.7.0.20} $$

By fusing the ${}^{S}_{E}q_{\nabla,t}$(2.7.0.20) and ${}^{S}_{E}q_{\omega,t}$(2.7.0.3) an estimate of orientation of the sensor frame reletive to the earth frame can be done. This fusion si done as described by equation (2.7.0.22) to (2.7.0.24) where β is a filter gain that can be found by equation (2.7.0.21). Where $\tilde{\omega}_\beta$ represent the estimated mean zero gyroscope measurement error of each axis.

$$ \beta = \sqrt{\frac{3}{4}}\tilde{\omega}_\beta \tag{2.7.0.21} $$

$$ {}^{S}_{E}\dot{\hat{q}}_{\epsilon,t} = \frac{\nabla f}{||\nabla f||} \tag{2.7.0.22} $$

$$ {}^{S}_{E}\dot{q}_{est,t} = {}^{S}_{E}\dot{q}_{\omega,t} - \beta {}^{S}_{E}\dot{\hat{q}}_{\epsilon,t} \tag{2.7.0.23} $$

$$ {}^{S}_{E}q_{est,t} = {}^{S}_{E}\hat{q}_{est,t-1} + {}^{S}_{E}\dot{q}_{est,t}\Delta t \tag{2.7.0.24} $$

Magnetic distortion

Madgwick suggests using the normalized estimated orientation, ${}^{S}_{E}q_{est,t}$, of the sensor to rotate the normalized magnetometer measurements that measured direction of the earth's magnetic field in the earth frame at time t, ${}^{E}\hat{h}_t$, to compensate for the magnetic distortion. Equation (2.7.0.25) show how the distortion will be compensated, and equation (2.7.0.26) shows how it will be used in the gradient descent algorithm.

$$ {}^{E}\hat{h}_t = \begin{bmatrix} 0 & h_x & h_y & h_z \end{bmatrix} = {}^{S}_{E}\hat{q}_{est,t-1} \otimes {}^{S}\hat{m}_t \otimes \hat{q}^{*}_{est,t-1} \tag{2.7.0.25} $$

$$ {}^{E}\hat{b}_t = \begin{bmatrix} 0 & \sqrt{h_x + h_y} & 0 & h_z \end{bmatrix} \tag{2.7.0.26} $$

According to research done by Madgwick, does this filter perform as well as a high-quality commercial Kalman-based system [27] [28] in terms of accuracy, when looking at the root mean square error. According to his results, the Madgwick filter outperforms the Kalman filter. According to the research done by Cirillo, De Maria, Natale, and Pirozzi, the performance for a Kalman, Mahony, and Madgwick similar in terms of accuracy when looking at both mean and standard deviation errors [29]. However, when it comes to the computational burden for estimating the orientation does Madgwick highly outperform the Kalman filter [29].

2.8 gRPC

In 2015 Google developed an open-source remote procedure call that uses HTTP/2 for transport, with features such as authentication, bidirectional streaming and flow control, cancellation, and timeouts [41]. This framework enables client and server applications to communicate transparently, and makes it easier to build connected systems [42]. gRPC framework allows making low latency, highly scalable, distributed systems whit protocols that are accurate, efficient, and language independent.

2.9 Control system Architecture

Last year Guddedal and Steinstø introduced a new architecture for the rig control system that was more robust, organized, and flexible than previously [1]. This architecture design splits parts of the control system into modules that run microservices, as illustrated in figure (10). Every module is a distributed system that follows the single-responsibility principle, making it easy to recognize the different parts of the system. Furthermore, is the structure of the control system defined into layers such that each module in the same layer has the same purpose but different function, figure (11) illustrates this. For example, the modules in the acquisition layer will be modules for data acquisition, but gathering from different sources. Thus organizing the system into a hierarchical style and can easily be navigated. There are six layers in this architecture: the data acquisition, output, filter, control, model, and the data layer. All modules serve as a gRPC server and client, making it easy to connect a new module to the system. For more detailed description can be read in Guddedal and Steinstø's thesis written in 2019 [1].

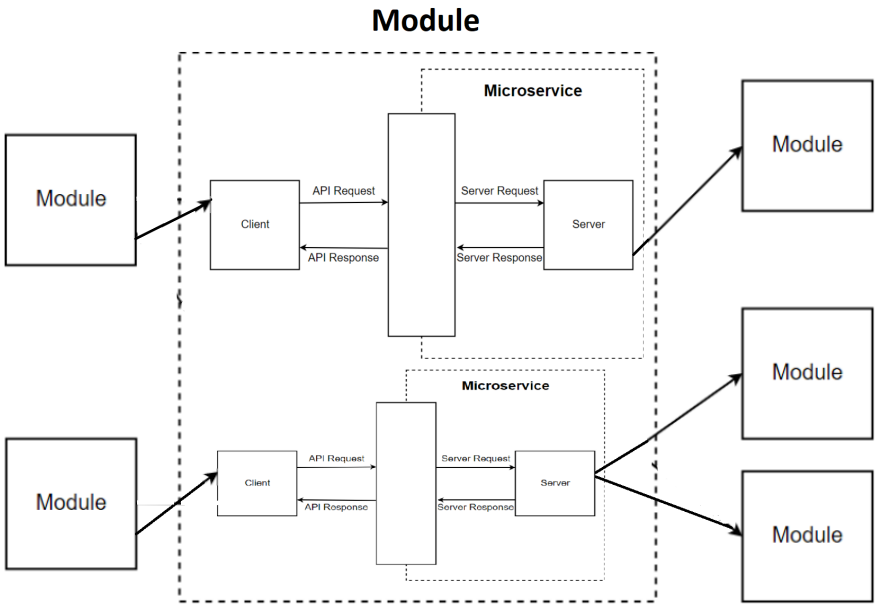


Figure 10: *Illustrates the concept of modules running microservices*

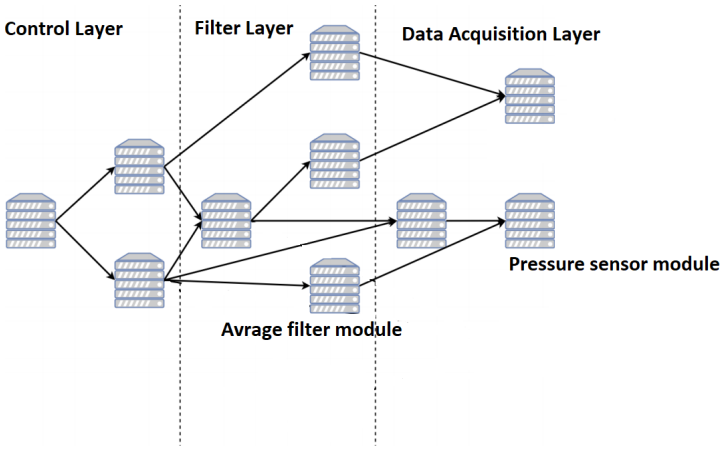


Figure 11: *Illustration of the general control system architecture concept where modules are represented as server and the dotted lines is the separetion of the different layers.*

3 Implementation

This chapter will explain how it was planned/chosen to solve the different tasks, such as designing the sensor card, calibrating, and AHRS. The main objective of this chapter is to describe all aspects of the project necessary to recreate the project/experiments at a later date.

3.1 Materials

This section of the chapter will give an overview of the equipment and devices used to generate the report's result. Thus this section will contain a short description of the components, tools, and the drilling rig used.

3.1.1 ATmega328P

According to ATmega328P datasheet [7], is this an 8-bit microcontroller device with 32-leads. This device's operating voltage is at 2.7V to 5.5V and can run at a speed of 0 to 8MHz or 16MHz. However, when running at 16MHz, the operating voltage is at 4.5 to 5.5 Volt. It has 23 programmable IØlines where 8 of them are 10-bit Analog-to-Digital-Converter. It has a programmable serial USART, Master/slave SPI serial interface, 2-wire serial interface that is compatible with I^2C. More details and features for ATmega328P can be read in [7].

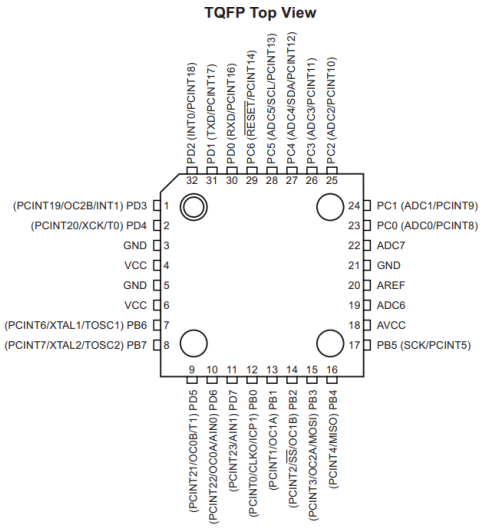


Figure 12: *The pins for the ATmega328P is illustrated in this figure taken from [7]*

Some conditions must be met to run the ATmega328P; the datasheet states that pin 4 and/or 6 (VCC) must be connected to the positive of a digital supply voltage. Moreover, that 3, 4, and/or 21 (GND) must be connected to the negative of the same digital supply voltage. Pin 29 (PC6/RESET) must be supplied with a positive voltage to run the MCU; else, if this pin gets a low level, the device will initiate a reset.

The datasheet further states that pin 7 (PB6) can be used as input to the internal oscillator amplifier, and pin 8 (PB7) as the output to the oscillator amplifier, as illustrated in figure (13). Moreover, it states that pin 18 (AVCC) is the supply voltage for the Analog-to-Digital-Converter and should be externally connected to VCC, even if the Analog-to-Digital-Converter is note used.

Pin 31 (PD1) and 30 (PD0) are TXD and RXD, respectively, for the USART serial communication. Furthermore, pin 27 (PC4) and 28 (PC5) is SDA and SCL for I^2C communication.

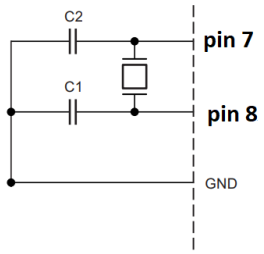


Figure 13: *Illustrates the recommended circuit for the crystal unit connected to the MCU, where C2 and C1 is the load capacitance*

3.1.2 ICM-20948

ICM-20948 is a low powered 9-axis sensor with a 3-axis gyroscope, accelerometer, and magnetometer. It supports both I^2C and SPI serial communication, supporting communication speed with 400kHz and 7MHz, respectively. The device has an operating voltage of 1.71V to 1.95V.

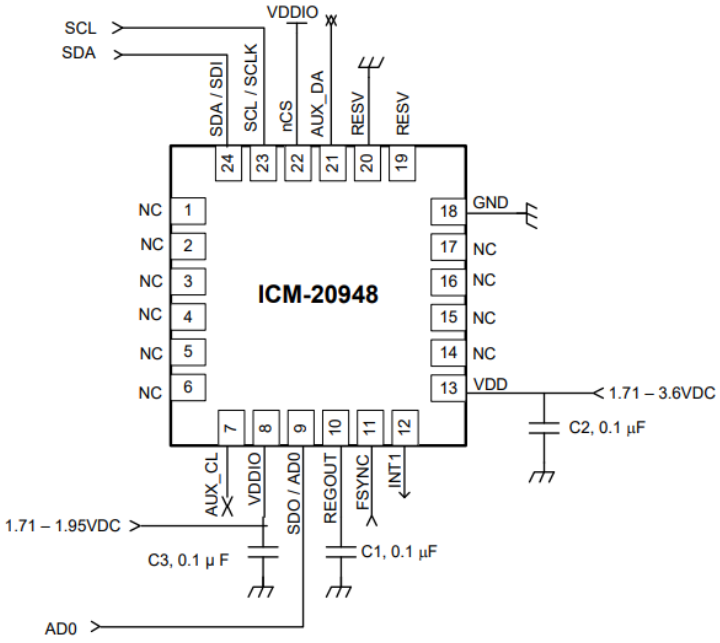


Figure 14: *Illustraits the recommended circuit for a ICM-20948 using I^2C*, taken from [32]

The gyroscope measures the angular rate and can measure either at ±250 degrees per seconds (^{o}/s), ±500 ^{o}/s, ±1000 ^{o}/s, or ±2000 ^{o}/s from 16-bit Analog-to-Digital-Converters. Accelerometer can measure in the ranges ±2 times gravity (g) , ±4g, ±8g and ±16g from 16-bit Analog-to-Digital-Converters. Magnetometer is a Hall-effect sensor with a range on ±4900 micro Tesla (μT) with a 16-bit data resolution. Moreover, the data output rate for the gyroscope and the magnetometer is 560Hz, and the magnetometer is 100Hz.

4.1 PIN OUT DIAGRAM AND SIGNAL DESCRIPTION

PIN NUMBER	PIN NAME	PIN DESCRIPTION
7	AUX_CL	I²C Master serial clock, for connecting to external sensors
8	VDDIO	Digital I/O supply voltage
9	AD0 / SDO	I²C Slave Address LSB (AD0); SPI serial data output (SDO)
10	REGOUT	Regulator filter capacitor connection
11	FSYNC	Frame synchronization digital input. Connect to GND if unused
12	INT1	Interrupt 1
13	VDD	Power supply voltage
18	GND	Power supply ground
19	RESV	Reserved. Do not connect.
20	RESV	Reserved. Connect to GND.
21	AUX_DA	I²C master serial data, for connecting to external sensors
22	nCS	Chip select (SPI mode only)
23	SCL / SCLK	I²C serial clock (SCL); SPI serial clock (SCLK)
24	SDA / SDI	I²C serial data (SDA); SPI serial data input (SDI)
1 – 6, 14 - 17	NC	Do not connect

Table 10. Signal Descriptions

Figure 15: *Pin out diagram and signal description for a ICM-20948 using I^2C*, taken from [32]

Figure 14 illustrates the recommended circuit to use when communicating over I^2C, and in figure 15 is the pinout diagram and signal description taken from the datasheet [32].

3.1.3 SparkFun Breakout ICM-20948

SparkFun has made a breakout board with the ICM-20948 sensor with connections for I^2C and SPI communication. This board includes a logic shifter making it able to run with 3.3V and 5V logic devices. SparkFun has also developed an Arduino library to be used with the ICM-20948 sensor; this library makes the sensor's communication quick and easy to setup.

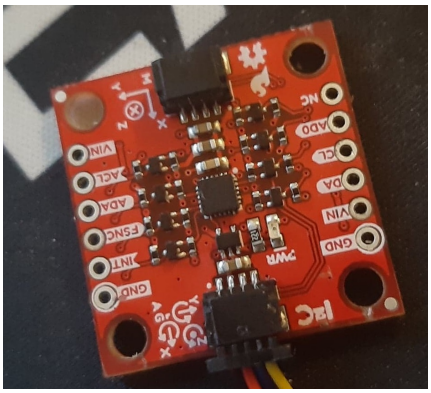


Figure 16: *Image displaying the SparkFun Breakout ICM-20948 board.*

3.1.4 XRCHA16M000F0A01R0 (crystal unit)

XRCHA16M000F0A01R0 is a surface mounted crystal unit. From the datasheet to the crystal unit [33] the following information is extracted:

- Nominal Frequency at 16MHz
- Load capacitance is 8pF (picofarad)
- Frequency Stability ±100ppm
- Frequency Tolerance ±100ppm

3.1.5 FT232RQ-REEL

FT232RQ-REEL is a 32-pin surface mounted USB to serial UART converter that operates at 1.8 to 5.25V. Furthermore, according to the datasheet [34], it has integrated USB termination resistors. From that datasheet, we get the following information about the pins seen in figure 17 (only the pins used in this project is described below)

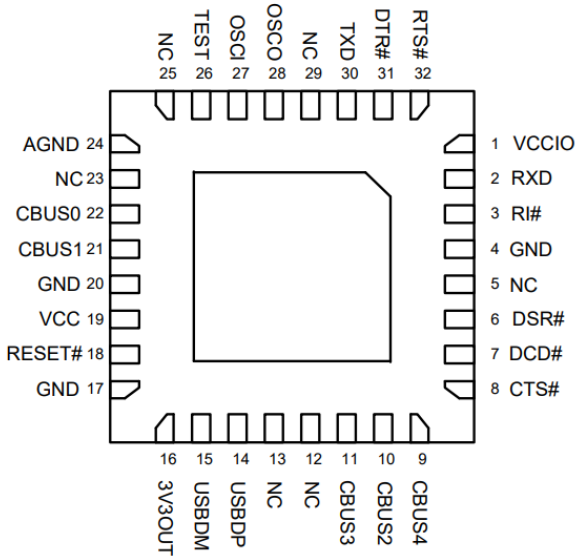


Figure 3.2 QFN-32 Package Pin Out and schematic symbol

Figure 17: *Pin out for a FT232RQ-REEL using* I^2C, taken from [34]

- Pin 14 (USBDP) is the USB data signal plus
- Pin 15 (USBDM) is the USB data signal minus

- Pin 1 (VCCIO) is the voltage supply pin and can be power from the USB bus power
- Pin 4, 17, 20 (GND) is the ground supply pin
- Pin 16 (3V3OUT) is a 3.3V output from an integrated regulator, and this pin should be decoupled to the ground using a 100nF capacitor. The purpose is to provide the internal 3.3V supply to the USB transceiver and the internal pull-up resistor.
- Pin 19 (VCC) is a voltage supply pin to the device core
- Pin 24 (AGND) is the device analog ground supply for the internal clock multiplier.
- Pin 26 (TEST) this pin will put the device into IC test mode and must be tied to ground for normal operation.
- Pin 30 (TXD) Transmit asynchronous data output
- Pin 2 (RXD) receiving asynchronous data input

3.1.6 PCB material

It was decided to use a flexible Printed Circuit board to make the sensor card as thin and flexible as possible. Thus an external manufacturer was needed to print the PCB. There was performed research to find the best producer based on price, experience. The chosen producer for the PCB was FPCway, because of their experience and their ability to print a PCB with EMI shielding.

The material used by FPCway has the following key attributes for a two layer PCB:

- Layer height 0.065mm for a 0.23mm thick PCB,
- Trace thickness 0.0347mm
- Trace width 0.0347mm

3.1.7 GDS-2062 Digital Oscilloscope

GDS-2062 is a digital oscilloscope whit two channels with a sampling rate at 1GSa/s, 24k record length, and a bandwidth of 60MHz. The 1GSa/s real-time sampling makes it possible to capture high-frequency waveforms [43]. Moreover, the 25k memory length makes it so the signal can be viewed in more detail by capturing and rebuilding the signal waveform.

3.1.8 HBM

There was already on PLC implemented in the drilling rig system when starting this project. This PLC was from the company HBM and is a QuantumX MX840B model and has eight high precision 24-bits channels for reading analog and digital inputs. The channels can use a low-pass filter to remove outliers in real-time and individually be programmed to compute and return values instead of raw 24-bit measurements through a configuration software by HBM.

3.1.9 PLC

Last year's team's criteria when purchasing the PLC's was a similar device to the HBM PLC that could be used for real-time data collection, preferably with Python support. They ended up buying one input (USB-1608GX) and one output (USB-3114) PLC from Measurement Computing.

USB-1608GX

USB-1608GX is an analog and digital input reader with 16 single-ended analog inputs that can also be med into eight differential inputs with a 16-bit resolution [38]. The measurements can be in the range of ±10V. Furthermore, it has eight digital I/O channels, two 32-bit counters, a PWM timer output, and a trigger input.

The PLC does not support synchronized measurements for all channels since each channel does not have a dedicated analog-to-digital-converter. The sampling is done in sequences with a 500Hz sampling rate, and the sampling across all channels is don in 0.032ms, which should not be a problem for our use. Connection to the control system computer is made with a USB interface.

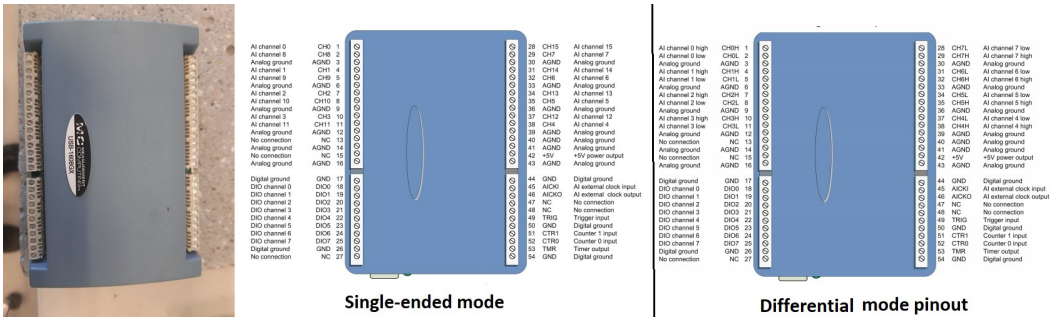


Figure 18: *A picture of USB-1608GX and images showing the pinout in single-ended and differential mode*

USB-3114

USB-3114 is an analog and digital output writer that has 16 analog output channels with 16-bit resolution. It can provide ±10V with a maximum current of 40mA per channels [37]. The throughput for single channel is 100Hz max, and for use of multiple channels is 100Hz/(number of channels). However, the hardware is designed only to accept one value per channel at one time (per write). i.e., the hardware is not designed to buffer an array of data and pace it out based on an internal/hardware clock.

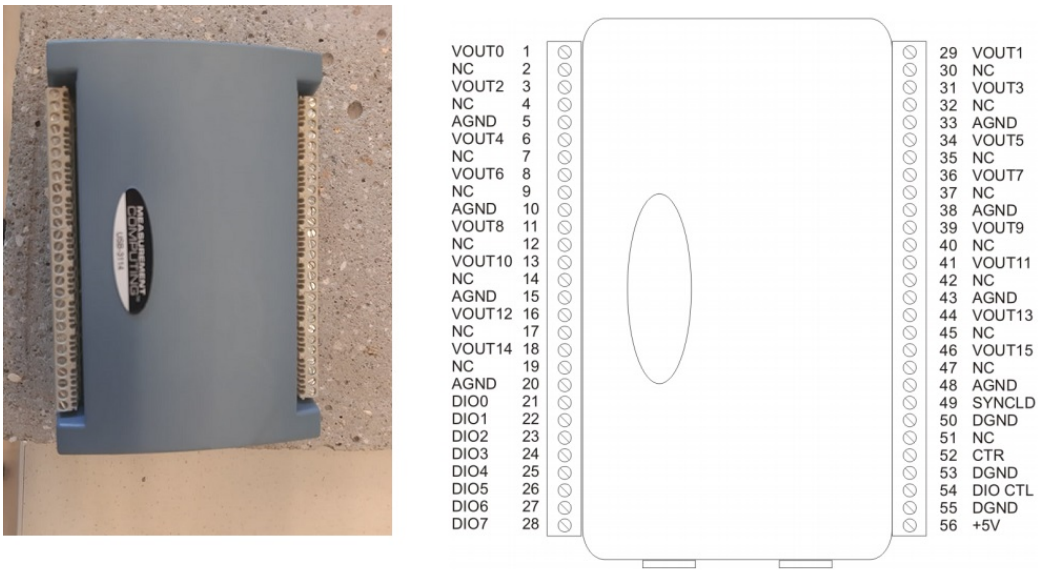


Figure 19: *A picture of USB-3114 and a image showing the pinout for the device*

3.1.10 Geckodriver G250X

G250X Geckodriver is a micro-step drive for stepper motors with rated phase current between 0 and 3.5A [39]. This driver works by taking an input as a STEP and DIRECTION puls to make a two-phase step output. The two phases will have a 90-degree phase shift relative to each other. This phase shift will determine the direction. Which is dependent on which phase comes first. For a 1.8-degree motor to complete one revolution, will the driver require 2000 pulses on the STEP pin.

The supply voltage for the driver needs to be between +15VDC and +50VDC. STEP and DIRECTION input must be either 0 to 3.3V or 0 to 5V. Moreover, STEP can have a pulse rate of 0 to 300 kHz. For the driver to read the pulses, the device sending them must also be connected to the SIGNAL GND on the Geckodriver.

3.1.11 MT-2306HS300AW stepper motor

MT-2306HS300AW is a type of Nema 23 stepper motor with a rated phase current of 3A and a step angle of 1.8-degrees [40]. Moreover, it is a four lead bipolar stepper motor.

3.1.12 Burkert 8605 Fluid Control Systems

Burkert 8605 is an electromechanical valve that can be programmed to be regulated by an input signal; this signal can be either current or voltage [44]. The supported signal ranges are 0-20 mA, 4-20mA, 0-5V or 0-10V. This device can control a proportional solenoid control valves in the power range of 40–2000 mA.

This fluid control system needs a power supply in the range of 12-24VDC and can output to the valve max 2A. Thus this device needs an external power supply and ground in addition to two signal wires (plus and negative (ground)).

3.1.13 DT35-B15251 distance sensors

DT35-B15251 is a distance sensor that uses a pair of optical axis sender and receiver to measure the distance from the sensor to an object crossing this axis. The light source is a red (have a wavelength of 658nm) laser. The measurements have a resolution of 0.1mm and with a typical accuracy of $\pm$10mm. An analog output can either send a current or voltage output with a resolution of 12bit, where the current range is 4mA to 20mA, and the voltage range is 0V to 10V.

3.1.14 Overview of 2019 UiS Drillbotics drilling rig

A miniature drilling rig was previously built at the university by former members of the UiS Drillbotics team and is designed to imitate the main functions of a typical offshore drilling rig. The rig has continuously been upgraded and changed by the new team members every year. Last year's team implemented new hardware and software upgrades in order to drill a deviated well. This section will describe how the 2019 drilling rig operates and the function of the different systems.

The miniature drilling rig, as illustrated in figure (20), has six hardware systems and a control system. The hardware systems are called: rotation, hoisting, circulation, pneumatic, whipstock positioning, power. Only the systems used in this year's competition will be described in this section.

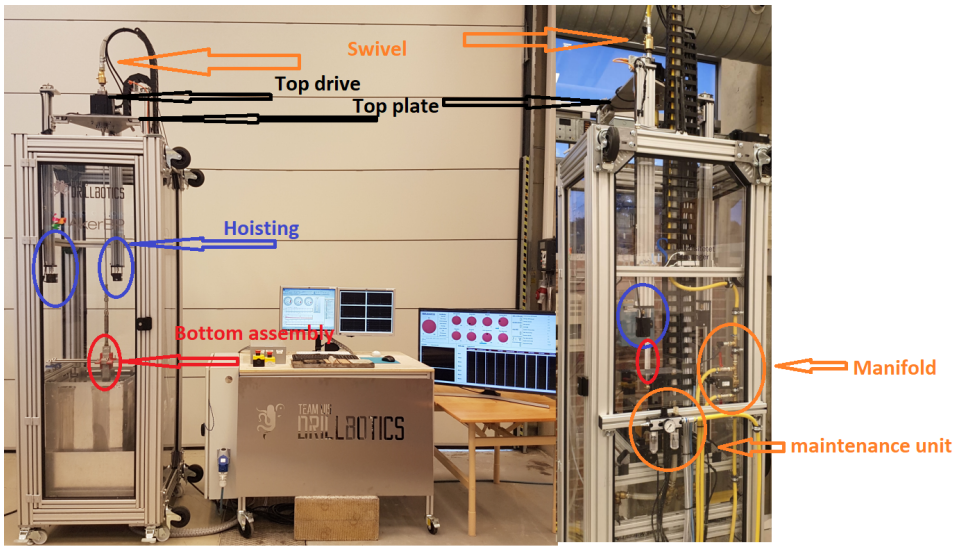


Figure 20: *Picture of the UiS drill bit with key components high highlighted [45][2]*

Rotational system

Where the rotational system is the system that performs the rotation/power transmission to the bit. Two separate rotational systems can be used, the Top Drive is one and is used for vertical drilling and a downhole motor for directional drilling. These two systems can be operated concurrently.

The downhole motor is a pneumatic motor transfers rotation directly to the drill bit, unlike the top drive that transfers torque directly to the drill string using a hollow-shaft brushless motor that rotates the assembly.

Drill string assembly

This assembly consists of an aluminum drill pipe, downhole sensor sub, knuckle joint, pneumatic downhole motor, and a drill bit. The drill string assembly is developed to drill directionally and is illustrated in figure (21), showing the placement of the different parts. By varying the weight on bit (WOB), the angle of the knuckle joint can be controlled, a spring in the joint gets either compressed (bend) or left uncompressed (no bend). By throttling the airflow coming from a compressor to the pneumatic motor, the bit RPM can be controlled.

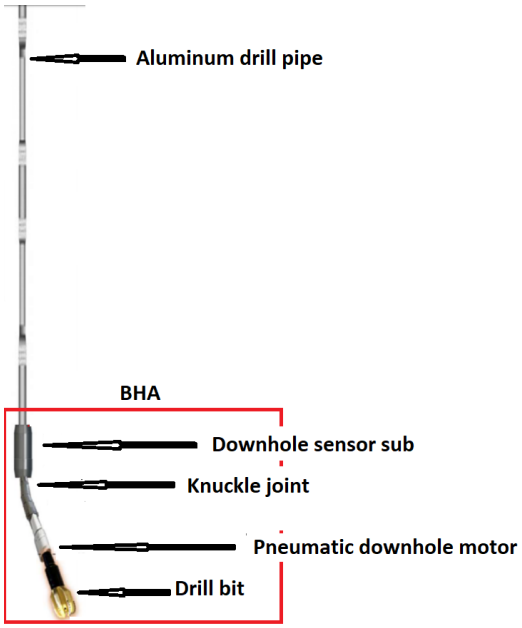


Figure 21: *illustration of the drill string assembly[2]*

Pneumatic system

This system has four separate parts: compressor, maintenance unit, manifold, and swivel with feedthrough. Where the compressor supplies compressed air through a hose into the maintenance unit outside of the rig frame. The maintenance unit lubricates the system with a mist of oil via the compressed air that goes through a pneumatic lubricator. Other parts of the maintenance unit is a water trap, regulator, and a pressure gauge.

After the maintenance unit, the air flows into the manifold by a hose. The manifold controls the airflow to the pneumatic motor, which dictates the RPM of the bit. The airflow is controlled by a series of valves mounted on the inside of the frame of the rig. Figure (22) shows the different parts of the manifold: a pressure relief valve, electronically operated solenoid valve, pressure sensor, and three manual ball valves. Two of the vales are to vent out trapped pressurized air in case of the rig shutting down, and the solenoid valve is closed—the solenoid valve restricts the airflow to control the RPM of the bit.

Next, the air flows through a hose connected to a swivel mounted on the Top Drive, then through the drill string to the pneumatic motor.

Figure 22: *Picture of the manifold mounted inside the rig*

Hoisting

The hoisting system simulates the drilling operations by operating three actuators consisting of a stepper motor and normally-closed brake. The motor raises and lowers the top plate, representing the drill floor where top drive and other components are mounted. The top plate is mounted to the motors via a tri-axial load cell that measures the hook load and free-hanging weight of approximately one-third of the drill floor weight. The distance moved by the top plate can be used to regulate the WOB.

The stepper motors are each connected to a lead screw where each revolution of the screw results in 8mm displacement of the top plate, which together whit the 2000 steps per revolution of the motor gives an elevation resolution on 4μm. A maximum WOB on 500N can be provided by combining the power of the three stepper motors. Each Stepper motor is connected to a driver that generates the pulses that control the motor.

Control System

The software architecture had a significant upgrade in 2019 and is now configured as a multithreading system, meaning that several client/server modules, referred to as threads running on multiple cores on the central processing unit (CPU). The communication between each module/thread is run in parallel and uses the gRPC API to handle it. Where this software was configured using Python and the Visual Studio Code IDE. This system is divided into six layers: data acquisition, output, filter, control, model, and the data layer and is designed as specified in chapter 2.9.

3.1.15 BHA assembly

BHA is all the parts below the drill pipe and has three sleeves that cover the inner parts of the assembly, called the top-, bend-, and bottom sleeve, figure (23) illustrates the BHA assembled. The inner parts consist off brass bushings, axel, thrust roller bearing, a pneumatic motor, sensors, and a joint connection.

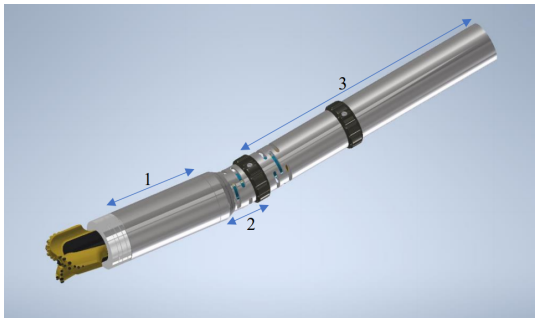


Figure 23: *An illustration of the BHA and the sleeves, where 1 is bottom, 2 is bend, and 3 is top sleeve*

3.1.16 Python

Python is an object-oriented program language used for coding the software for the AHRS. This language is a general-purpose programming language that emphasizes code readability. Python has a sizeable standard library, but many third-party libraries can easily be installed. The external libraries used in this project are:

- mcculw
- grpc
- numpy
- pySerial
- matplotlib

3.1.17 Matlab

Matlab is a program for mathematical programming that uses a scripting language based on C. In this project, Matlab used for analyzing signals, filter construction, and development of calibrating methods.

3.1.18 Arduino IDE

Arduino IDE is an application used to write and upload programs to Arduino compatible MCU. This application supports the programming languages C and C++ using special rules of code structuring. The written code only ney needs two primary functions for starting the sketch and the main program loop. This program also includes an easy way to install a new firmware to a microcontroller via a bootloader.

3.1.19 InstaCal

InstaCal is a program for configuration of Measurement Computing data acquisition hardware. This program automatically detects the hardware and makes it easy to configure the device. It can perform tests both internally and externally and calibrate devices. This program is included with the purchase of PLC from Measurement Computing. When the device is configured within this program, it can be used without further configuration; for example, a Python script can do a simple function call to one of the channels.

3.2 Methods

This chapter section of the chapter will report the development/programming/ solution-methods used to generate the result in this thesis. Such as methods for filtering, signal analysis, description of the designed software.

3.2.1 2020 Rig Design

This section will give an overview of the planned design for the drilling rig for use in the 2020 Drillbotics competition. The overview will include what was initially planned and the design that was decided to use for this year's competition. Moreover, the system decided to use and change from the last year's solution is hoisting, circulation, pneumatic, power, and the control system.

Because last year's team did not have time to change out the Arduinos that controls the actuators on the drilling rig, this task was given to this year's team. By doing this change, the goal is to make the system more reliable and centralize the control to one device instead of several. Another goal for centralizing the control of the actuators is for future teams to have an easier task of adding or removing functions from the rig. Thus the system will be designed so the operations associated with one Arduino will be decoupled from the tasks from another Arduino.

This year the plan was to use both the hollow-shaft top drive motor and the pneumatic downhole motor kin conjunction. However, the driver for the top drive hade bean damage under transport from last year's competition, even though an attempt to fix it by desoldering the damaged connector and replace it. The communication from the driver to the control system computer was yet again possible by replacing the connector. Despite the communication seemingly working correctly, it was still impossible to operate the motor in a normal fashion. Thus, the final design for this year's solution does not use the top drive motor as planned. So when it comes to changing out the Arduino used in the rotational system, only the Arduino controlling the valve for the pneumatic motor will be changed.

The hosting system will be the same, as described in chapter 3234. The only change is that the control of the stepper motors will be change to a PLC instead of the implemented Arduinos. The same applies to the pneumatic system.

The software control system from 2019 will be reused, but new modules will be introduced to the already implemented control system architecture.

3.2.2 Calculation of I²C pull-up resistor

As mentioned in chapter 2.3.4, does the serial communication I²C require a pull-up resistor to alter between HIGH and LOW logic. The minimum pull-up resistor value can be calculated by using ohm's law:

$$R_p(min) = \frac{(V_{CC} - V_{OL})}{I_{OL}} \tag{3.2.2.1}$$

where R_p(min) is the minimum pull-up resistor and V_{OL} is the LOW-level output voltage and I_{OL} is LOW-Level Output Current.

To calculate the maximum resistor value for the pull-up resistor, equations (2.3.4.1) and (2.3.5.1) is used. Where (2.3.5.1) is used to calculate the bus capacitance used in (2.3.4.1).

3.2.3 Coordinate Frames

Coordinate frames are used rigorously in this thesis, and there are three defined coordinate frames used. One is the earth frame coordinate frame that describes the orientation of the MARG sensor according to earth's compass North and East direction. The global frame is also a coordinate frame used that describes an object orientation and displacement with respect to the drill rig. Another frequently used sensor frame defines the MARG sensors orientation according to its axes of sensitivity. Figure (24), (26) and (25) illustrates the earth, global, and sensor frame, respectively.

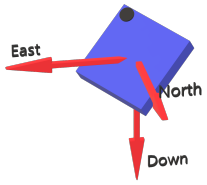


Figure 24: *Illustrates the MARG sensors earth frame.*

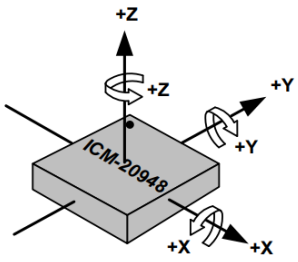


Figure 25: *Axes of sensitivity for the MARG sensor [32]*

Figure 26: *Depicting the drilling rigs global frame.*

3.2.4 Calculation of Load Capacitance for Crystal Unit

The load capacitance listed in the datasheet for a crystal unit is the capacitance; the unit needs to have an accurate and stable frequency at parallel-resonant. By using the recommended circuit for device in noisy environments given in the Atmega328P datasheet (figure 13) for the crystal unit, the load capacitance is divided into two capacitors, C1 and C2. Where the load capacitance is the capacitive load seen from the perspective of the crystal.

Moreover, in this case, it would be the sum of C1 and C2. By looking from the equivalent circuit from the perspective of the crystal, C1 and C2 would be in series. Therefore, by using the formula for the series capacitor would find the load capacitance. Furthermore, is it recommended in the datasheet for Atmega328P that C1 and C2 should be equal; thus, equation (3.2.4.2) can be used to calculate the load capacitance.

$$C_L = \frac{(C1 \cdot C2)}{(C1 + C2)} \tag{3.2.4.1}$$

$$C1 = C2 = 2 \cdot C_L \tag{3.2.4.2}$$

3.2.5 Kinematic

Kinematics is a tool chosen for analyzing the drilling rig's rigid motions and is an alternative method used to track the drill bit position. The forward kinematic method can calculate the coordinates for the drill bit in the global coordinate frame based on measured orientation and translation of the joints that the rig has. There are two revolute and one prismatic joint that defines the drilling rigs rigid motions. One of the revolute joints represents the angle of orientation of the Top Drive, while the other is the angle of the Bend sleeve. The Hoisting system defines the prismatic joint used in the kinematic method.

The Denavit-Hartenberg convention mention in chapter 2.2 is used to simplify the calculation of the forward kinematics equations defined by the rigid motions. The four parameters used in the DH convention for each joint are calculated based on the three links making up the drilling rig's kinematic chain (the first kinematic chain), which is represented symbolically in figure (27).

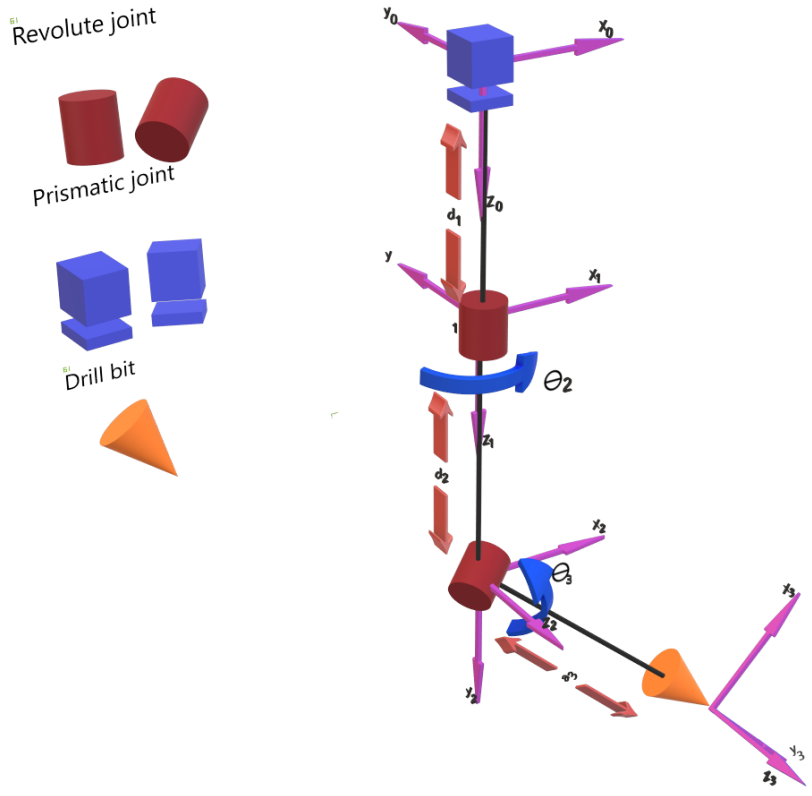


Figure 27: *A symbolic representation of the first kinematic chains for the drilling rig. Showing the coordinate frames attach to each joint and their cosponsoring links for the drilling rig*

link	a_i	α_i	d_i	θ_i
1	0	0	d_1^*	0
2	0	90	d_2	θ_2^*
3	a_3	0	0	θ_3^*

Table 1: DH parameters for the first kinematic chain, * variable

Coordinate frames in figure (27) have been placed such that they satisfy both DH1 and DH2 (2.2). Table (1) shows the resulting parameters for each link in the kinematic chain representing the drilling rig. These parameters are used to compute the homogeneous transformation matrix A_i by using the equation (2.2.0.5), resulting in the matrixes from (3.2.5.1) to (3.2.5.3).

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2.5.1}$$

$$A_2 = \begin{bmatrix} cos(\theta_2) & 0 & sin(\theta_2) & 0 \\ sin(\theta_2) & 0 & -cos(\theta_2 & 0 \\ 0 & 1 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2.5.2}$$

$$A_3 = \begin{bmatrix} cos(\theta_3) & -sin(\theta_3) & 0 & a_3 cos(\theta_3) \\ sin(\theta_3) & cos(\theta_3) & 0 & a_3 sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2.5.3}$$

The position and orientation of the drill bit in the global coordinate frame are then calculated using equation (2.2.0.3). Thus resulting in the homogeneous transformation matrix in (3.2.5.4), and from this matrix, the x, y, and z coordinate in the global frame for the drill bit can be defined by equation (3.2.5.5), (3.2.5.6) and (3.2.5.7) respectively. Where θ_2, is defined by the measured angle of the Top Drive, and θ_3 is the measured angle of the Bend sleeve at time t. Moreover, is d_1 the measured displacement of the Top plate, d_2 is the distance between frames one and two, and a_3 is the distance from Bend Sleeve to the drill bit.

$$T_3^0 = \begin{bmatrix} cos(\theta_2)cos(\theta_3) & -cos(\theta_2)sin(\theta_3) & sin(\theta_1) & a_3cos(\theta_2)cos(\theta_3) \\ sin(\theta_2)cos(\theta_3) & -sin(\theta_2)sin(\theta_3) & -cos(\theta_2) & a_3sin(\theta_2)cos(\theta_3) \\ sin(\theta_2) & cos(\theta_3) & 1 & a_3sin(\theta_3) + d_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2.5.4}$$

$$x = a_3cos(\theta_2)cos(\theta_3) \tag{3.2.5.5}$$

$$y = a_3sin(\theta_2)cos(\theta_3) \tag{3.2.5.6}$$

$$z = a_3sin(\theta_3) + d_2 + d_1 \tag{3.2.5.7}$$

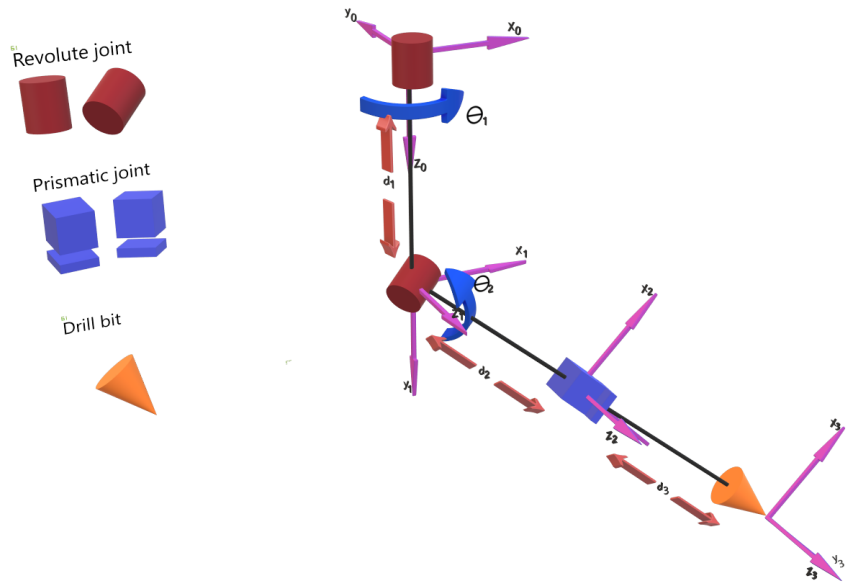


Figure 28: *A symbolic representation of the second kinematic chains for the drilling rig. Showing the coordinate frames attach to each joint and their cosponsoring links for the drilling rig*

However, these equations are only accurate for as long as the Bend sleeve coordinate frame is over the rock surface. When it is not over the rock's surface, the kinematic chain (the second kinematic chain) can be represented as they are in figure (28). The reasoning behind this representation is that from the point of time the top of the Bend sleeve passes the rock's surface, a new group of rigid motions will be created because the Top sleeve will try to follow the well path throughout the rock, as illustrated in figure (29). This group is divided into two joints, a revolute and prismatic joint, with zero distance between. The prismatic joint will represent all future displacement for the BHA; the value of displacement will equal the displacement of the Top plate. So not to calculate this displacement twice in the homogeneous transformation

matrix T_3^0, link zero in the first kinematic chain will be ignored and not used in the second kinematic chain. The new prismatic joint will be placed on the y-axis of the revolute joint since the direction of the displacement is dependent on the orientation of the new revolute joint.

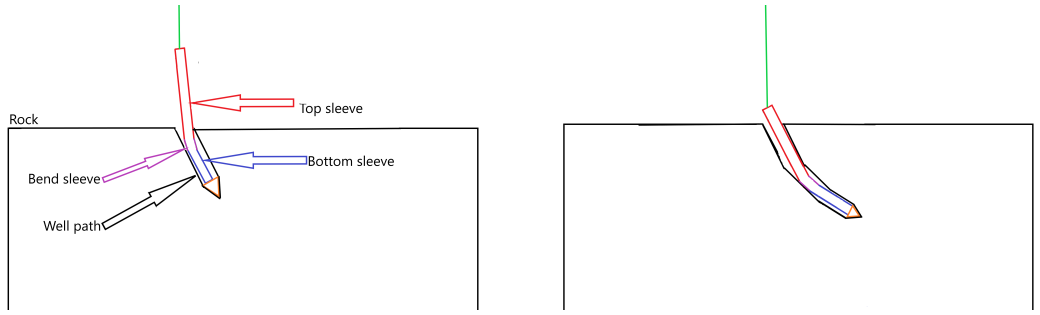


Figure 29: *An exaggerated illustration of how the BHA gets rotated by the well path*

Furthermore, the higher the distance of the well path, the more links will be created. However, the product of all these links can be represented as only one revolution and one prismatic joint. This assumption should be reasonable since this long kinematic chain can be thought of as a succession of links of length zero between. The reason is that the walls of the well path will make it so the center of rotation with respect to the global frame (frame zero) will be the same for all of the revolute joints in the kinematic chain, and the fact that all the links will effectively become static after a new one is made. Therefore, all coordinate frames for all links in the kinematic chain can have their origin placed at the same point in space. Thus it can be thought of as a revolute and prismatic joint instead of a succession of links and yet resulting in the same position for the drill bit. One thing to note is that link three between joints three and two will take the shortest path; hence will go through the rock. However, this is only a theoretical link and does not represent a part of the physical BHA.

Since the coordinate frames in figure (28) are placed using DH1 and DH2 (2.2), the following DH parameters and matrices were calculated.

link	a_i	α_i	d_i	θ_i
1	0	-90	d_1	θ_1^*
2	0	90	d_2	θ_2^*
3	0	0	d_3^*	0

Table 2: DH parameters for second kinematic chain, * variable

$$A_2 = \begin{bmatrix} cos(\theta_1) & 0 & -sin(\theta_1) & 0 \\ sin(\theta_1) & 0 & cos(\theta_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2.5.8}$$

$$A_2 = \begin{bmatrix} cos(\theta_2) & 0 & sin(\theta_2) & 0 \\ sin(\theta_2) & 0 & -cos(\theta_2 & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2.5.9}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2.5.10}$$

$$T_3^0 = \begin{bmatrix} cos(\theta_1)cos(\theta_2) & -sin(\theta_1) & cos(\theta_1)sin(\theta_2) & d_3cos(\theta_1)sin(\theta_2) - d_2sin(\theta_1) \\ sin(\theta_1)cos(\theta_2) & cos(\theta_1) & sin(\theta_1)sin(\theta_2) & d_3sin(\theta_1)sin(\theta_2) + d_2cos(\theta_1) \\ -sin(\theta_1) & 0 & cos(\theta_2) & d_3cos(\theta_2) + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2.5.11}$$

$$x = d_3cos(\theta_1)sin(\theta_2) - d_2sin(\theta_1) \tag{3.2.5.12}$$

$$y = d_3sin(\theta_1)sin(\theta_2) + d_2cos(\theta_1) \tag{3.2.5.13}$$

$$z = d_3cos(\theta_2) + d_1 \tag{3.2.5.14}$$

where θ_1, is defined by the measured angle of the Top Drive, and θ_2 defined as the sum of the initial angle and measured angle of the Bend sleeve at time t. Moreover, is d_1 a constant with the length of the Bend sleeve and the bottom sleeve added together, and d_3 is the measured displacement of Top plate minus d_1.

3.2.6 Boot-loading Arduino

The code in the microcontroller that accepts the code from the computer and places it in the memory of the MCU is called the bootloader. Furthermore, is this the part of the MCU that sets the internal fuses that control the behavior for it, e.g., to run at either 8MHz or 16MHz is decide by these fuses.

To change the bootloader code for an Arduino compatible MCU, can this be done by using Arduino IDE through another Arduino programed to be AVRISP device that is connected to the target MCU. The circuit diagram for this setup is shown in figure (30).

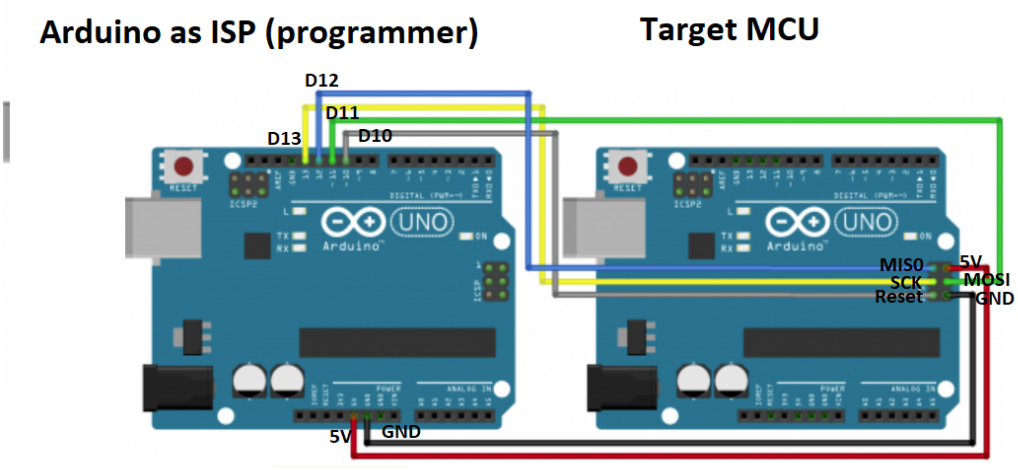


Figure 30: *Schematic used for boot-loading the MCU*

A sketch called ArduinoISP can be uploaded to the programing Arduino to turn it into an AVRISP. Then by selecting a board in the IDE while the programming Arduino is still connected, will define the board, the target MCU will be defined as, next, the programmer should be defined as "Arduino as ISP". The last step is to select Burn Bootloader, and then the bootloader will be installed on the MCU. The board selected in the IDE will determine the functions of the MCU. Figure () illustrates the process.

3.2.7 Accelerometer Calibration

The data set needed for this method mus contain measurements done while the device is static. The ideal data set will contain a large set of samples that are generated when the device is static in different orientations, the true value for these orientations must be known.

In this thesis is least square method used to estimate the measurement error for a accelerometer. The calibration method for the accelerometer used in this thesis is based on that each axis's output vector sum should be equal to the acceleration of gravity when in a static condition.

$$g = \sqrt{a_x^2 + a_y^2 + a_{,z}^2} \tag{3.2.7.1}$$

where a_i is the ideal values of measurements. Thus the error of measurements will be defined as:

$$E = g^2 - a_x^2 + a_y^2 + a_z^2 = 0 \tag{3.2.7.2}$$

By using the model for error in measurements defined in equation (2.3.9.1) the

error can be shown to be:

$$E = \begin{bmatrix} g_x & g_y & g_z \end{bmatrix}^2 - \left(\begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix}^{-1} \cdot \left(\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} - \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} \right) \right)^2 \quad (3.2.7.3)$$

The second term in equation (3.2.7.3) can be written as:

$$\begin{bmatrix} D_{xx}V_x + D_{xy}V_y + D_{xz}V_z + q_x \\ D_{yx}V_x + D_{yy}V_y + D_{yz}V_z + q_y \\ D_{zx}V_x + D_{zy}V_y + D_{zz}V_z + q_z \end{bmatrix} = \begin{bmatrix} V_x & V_y & V_z & 1 \end{bmatrix} \begin{bmatrix} D_{xx} & D_{yx} & D_{zx} \\ D_{xy} & D_{yy} & D_{zy} \\ D_{xz} & D_{yz} & D_{zz} \\ q_x & q_y & q_z \end{bmatrix} \quad (3.2.7.4)$$

where **D** is the inverse of **S**, Then by substituting (3.2.7.4) into (2.3.9.1) will define the error function **E** as:

$$E = \begin{bmatrix} g_x & g_y & g_z \end{bmatrix}^2 - \left(\begin{bmatrix} V_x & V_y & V_z & 1 \end{bmatrix} \begin{bmatrix} D_{xx} & D_{yx} & D_{zx} \\ D_{xy} & D_{yy} & D_{zy} \\ D_{xz} & D_{yz} & D_{zz} \\ q_x & q_y & q_z \end{bmatrix} \right)^2 = G^2 - (VD)^2 \quad (3.2.7.5)$$

Therefore can the residual error r be written as:

$$r^2 = G^2 - (VD)^2 \Rightarrow r = G - VD \quad (3.2.7.6)$$

The residual error $r[M]$ for the M-th measurements is then:

$$\begin{bmatrix} r[0] \\ r[]1 \\ .. \\ r[M-1] \end{bmatrix} = \begin{bmatrix} g_x[0] & g_y[0] & g_z[0] \\ g_x[1] & g_y[1] & g_z[1] \\ .. & .. & .. \\ g_x[M-1] & g_y[M-1] & g_z[M-1] \end{bmatrix} - \begin{bmatrix} V_x[0] & V_y[0] & V_z[0] & 1 \\ V_x[1] & V_y[1] & V_z[1] & 1 \\ .. & .. & .. & .. \\ V_x[M-1] & V_y[M-1] & V_z[M-1] & 1 \end{bmatrix} \begin{bmatrix} D_{xx} & D_{yx} & D_{zx} \\ D_{xy} & D_{yy} & D_{zy} \\ D_{xz} & D_{yz} & D_{zz} \\ q_x & q_y & q_z \end{bmatrix} \quad (3.2.7.7)$$

Where **G** is the vector of dependent variables, **V** is the measurement matrix, and **D** is the solution vector. From equation (3.2.7.6) shows that the module being fitted has the non-homogeneous form $r == Y - X\beta$ and can solve the normal equation for β with equation $\beta = (X^T X)^{-1} X^T Y$(2.4.0.12). Where **Y** is the known normalized Earth gravity vector, **X** is sensor raw data collected at different stationary positions, and β is the 12 calibration parameters that need to be determined (scaling factor and zero offset distortion compensation). Where the scale and offset in the accelerometer measurements can is defined as:

$$\beta = \begin{bmatrix} \beta_{11} & \beta_{21} & \beta_{31} \\ \beta_{12} & \beta_{22} & \beta_{32} \\ \beta_{13} & \beta_{23} & \beta_{33} \\ \beta_{14} & \beta_{24} & \beta_{34} \end{bmatrix} \quad (3.2.7.8)$$

$$Q = \begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} \beta_{14} \\ \beta_{24} \\ \beta_{34} \end{bmatrix} \tag{3.2.7.9}$$

$$S = \begin{bmatrix} S11 & S21 & S31 \\ S12 & S22 & S32 \\ S13 & S23 & S33 \end{bmatrix} = \begin{bmatrix} \beta_{11} & \beta_{21} & \beta_{31} \\ \beta_{12} & \beta_{22} & \beta_{32} \\ \beta_{13} & \beta_{23} & \beta_{33} \end{bmatrix} \tag{3.2.7.10}$$

3.2.8 Simple magnetometer calibration

The hard iron is typically the larges but also the easiest to correct since it can be seen as a constant zero offset error. The simplest way will be to use an extensive data set with measurements done while the device is dynamic, and orients around its axes. By using the data set, the maximum and minimum values along the magnetometer's three sensitivity axes can be known. The average can then be subtracted from the data to re-center the response surface to the origin.

$$b_x = \frac{max(mag_x) + min(mag_x)}{2} \tag{3.2.8.1}$$

$$b_y = \frac{max(mag_y) + min(mag_y)}{2} \tag{3.2.8.2}$$

$$b_z = \frac{max(mag_z) + min(mag_z)}{2} \tag{3.2.8.3}$$

The average values along the magnetometer's three sensitivity axes are also used for the soft iron compensation. However, it is used to calculate the average ratio along each axis and the average of all three axes. This value will be the diagonal values for the inverse scale factor error. Thus axis that over-measures the magnetic field will have its value reduced and vice-versa.

$$K_{xx} = \frac{\frac{b_x + b_y + b_z}{3}}{b_x} \tag{3.2.8.4}$$

$$K_{yy} = \frac{\frac{b_x + b_y + b_z}{3}}{b_y} \tag{3.2.8.5}$$

$$K_{zz} = \frac{\frac{b_x + b_y + b_z}{3}}{b_z} \tag{3.2.8.6}$$

3.2.9 Magnetometer Calibration using least square

To calibrate the magnetometer there is a need measured data to find the environmental distortion. This method is based on having a large set of samples that are generated while the device it dynamic. The best set of data would be for a device that orienting around its six principal directions $(\pm x, \pm y, \pm z)$.

In this thesis is least square method used to estimate the electromagnetic distortion for a magnetometer. The ideal values for the magnetometer measurements lies on the surface of a sphere with radius equal to the G the geomagnetic field strength. By using equation (2.3.8.2) and the G geomagnetic field strength the error function defined by the surface of a sphere can be defined as in (3.2.9.1):

$$\begin{aligned} G^2 = G_c^2 &= (K(B_r - b))^2 = (K(B_r - b))^T (K(B_r - b)) \\ &= \left(\begin{bmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{bmatrix} \left(\begin{bmatrix} B_{r,x} \\ B_{r,y} \\ B_{r,z} \end{bmatrix} - \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \right) \right)^2 \end{aligned} \tag{3.2.9.1}$$

$$\begin{aligned} E &= (K(B_r - b))^T (K(B_r - b)) - G^2 \\ &= (B_r - b)^T K^T K (B_r - b) - G^2 = 0 \end{aligned} \tag{3.2.9.2}$$

insignificant soft iron distortion
However, for a magnetometer, the scale factor will very allot depending on the environment the sensor is placed. Thus the solution for the error residual vector **r** can vary. For example, for a location with insignificant soft iron distortion, the matrix **K** can be assumed to be the identity matrix. Hence the error function can be seen as:

$$E = (B_r - b)^T (B_r - b) - G^2 = B_r^T B_r - 2B_r^T b + b^T b - G^2 = 0 \tag{3.2.9.3}$$

The residual error r[i] for the i-th observation is:

$$\begin{aligned} r[i] &= B_{r,x}[i]^2 + B_{r,y}[i]^2 + B_{r,z}[i]^2 \\ &\quad - 2B_{r,x}[i]b_x - 2B_{r,y}[i]b_y - 2B_{r,z}[i]b_z + b_x^2 + b_y^2 + b_z^2 - G^2 = 0 \end{aligned} \tag{3.2.9.4}$$

Simplifying and return to matrix form:

$$\begin{aligned} r[i] &= (B_{r,x}[i]^2 + B_{r,y}[i]^2 + B_{r,z}[i]^2) \\ &\quad - \begin{bmatrix} B_{r,x}[i] & B_{r,y}[i] & B_{r,z}[i] & 1 \end{bmatrix} \begin{bmatrix} 2b_x \\ 2b_y \\ 2b_z \\ -b_x^2 - b_y^2 - b_z^2 + G^2 \end{bmatrix} \end{aligned} \tag{3.2.9.5}$$

Then the definitions of the error residual vector **r** is:

$$r = \begin{bmatrix} r[0] \\ r[1] \\ .. \\ r[M-1] \end{bmatrix} \tag{3.2.9.6}$$

and Y the vector of dependent variables:

$$Y = \begin{bmatrix} B_{r,x}[0]^2 + B_{r,y}[0]^2 + B_{r,z}[0]^2 \\ B_{r,x}[1]^2 + B_{r,y}[1]^2 + B_{r,z}[1]^2 \\ .. \\ B_{r,x}[M-1]^2 + B_{r,y}[M-1]^2 + B_{r,z}[M-1]^2 \end{bmatrix} \tag{3.2.9.7}$$

and X the measurement matrix:

$$X = \begin{bmatrix} B_{r,x}[0] & B_{r,y}[0] & B_{r,z}[0] & 1 \\ B_{r,x}[1] & B_{r,y}[1] & B_{r,z}[1] & 1 \\ .. & .. & .. & .. \\ B_{r,x}[M-1] & B_{r,y}[M-1] & B_{r,z}[M-1] & 1 \end{bmatrix} \tag{3.2.9.8}$$

and the solution vector β as:

$$\beta = \begin{bmatrix} 2b_x \\ 2b_y \\ 2b_z \\ -b_x^2 - b_y^2 - b_z^2 + G^2 \end{bmatrix} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} \tag{3.2.9.9}$$

The error residual equation can be written as:

$$r = Y - X\beta \tag{3.2.9.10}$$

This is model being fitted has the non-homogeneous form $r == Y - X\beta$ and can solve the normal equation for β with equation $\beta = (X^T X)^{-1} X^T Y$(2.4.0.12). Where **Y** the vector of dependent variables, **X** is sensor raw data, and β is vector parameter that need to be determined. Where the soft iron is the identity matrix, The hard iron **V** is given by equation (3.2.9.9) as:

$$\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = 0.5 \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \tag{3.2.9.11}$$

The dimensionless measure of fit error ϵ used is defined to be:

$$\epsilon = \frac{1}{2G^2} \sqrt{\frac{E}{M}} \tag{3.2.9.12}$$

Significant soft iron distortion

When there are significant soft iron distortion the error function can be defined as in equation (3.2.9.2). Because the error residual r[i] is defined as the divergence of the square of the radius of the geomagnetic sphere and squared modulus of the calibrated magnetometer measurement B_c[i]. By expending equation (3.2.9.2) gives:

$$\begin{aligned} r[i] &= (K(B_r - b))^T K(B_r - b) - G^2 \\ &= (B_r - b)^T K^T K(B_r - b) - G^2 \\ &= (B_r - b)^T A(B_r - b) - G^2 \\ &= B_r^T AB_r - B_r^T Ab - b^T AB_r + V^T AV - G^2 \end{aligned} \tag{3.2.9.13}$$

Because $B_r^T Ab$ is a scalar:

$$r[i] = B_r^T AB_r - 2B_r^T Ab + V^T AV - G^2 \tag{3.2.9.14}$$

Expending (3.2.9.14), then simplifying, and returning to matrix form results in:

$$\begin{aligned} r[i] = &\begin{bmatrix} B_{r,x}[i]^2 \\ B_{r,x}[i]^2 \\ B_{r,x}[i] \\ B_{r,x}[i] \\ B_{r,x}[i] \\ B_{r,x}[i] \\ 1 \end{bmatrix}^T \begin{bmatrix} A_{xx} \\ A_{yy} \\ A_{zz} \\ -2A_{xx}V_x \\ -2A_{yy}V_y \\ -2A_{zz}V_z \\ A_{xx}V_x^2 + A_{yy}V_y^2 + A_{zz}V_z^2 - G^2 \end{bmatrix} \\ &= X\beta \end{aligned} \tag{3.2.9.15}$$

From equation (3.2.9.15) it can be seen that the model being fitted is homogeneous and there for can equation (2.4.0.22) be used to solve for β. Where β is defined as:

$$\beta = \begin{bmatrix} A_{xx} \\ A_{yy} \\ A_{zz} \\ -2A_{xx}V_x \\ -2A_{yy}V_y \\ -2A_{zz}V_z \\ A_{xx}V_x^2 + A_{yy}V_y^2 + A_{zz}V_z^2 - G^2 \end{bmatrix} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \end{bmatrix} \tag{3.2.9.16}$$

and X for M measurements is:

$$X = \begin{bmatrix} B_{r,x}[0]^2 & B_{r,x}[0]^2 & B_{r,x}[0] & B_{r,x}[0] & B_{r,x}[0] & B_{r,x}[0] & 1 \\ B_{r,x}[1]^2 & B_{r,x}[1]^2 & B_{r,x}[1] & B_{r,x}[1] & B_{r,x}[1] & B_{r,x}[1] & 1 \\ .. & .. & .. & .. & .. & .. & .. \\ B_{r,x}[M-1]^2 & B_{r,x}[M-1]^2 & B_{r,x}[M-1] & B_{r,x}[M-1] & B_{r,x}[M-1] & B_{r,x}[M-1] & 1 \end{bmatrix} \tag{3.2.9.17}$$

The ellipsoid fit matrix A is defined as:

$$A = \begin{bmatrix} A_{xx} & 0 & 0 \\ 0 & A_{yy} & 0 \\ 0 & 0 & A_{zz} \end{bmatrix} = \begin{bmatrix} \beta_0 & 0 & 0 \\ 0 & \beta_1 & 0 \\ 0 & 0 & \beta_2 \end{bmatrix} \tag{3.2.9.18}$$

and is normalized to have unit determinant before being used. The hard iron is defined as:

$$\begin{bmatrix} -2A_{xx}V_x \\ -2A_{yy}V_y \\ -2A_{zz}V_z \end{bmatrix} = \begin{bmatrix} \beta_3 \\ \beta_4 \\ \beta_5 \end{bmatrix}$$
$$\Rightarrow \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} \frac{\beta_3}{-2A_{xx}} \\ \frac{\beta_4}{-2A_{yy}} \\ \frac{\beta_5}{-2A_{zz}} \end{bmatrix} = \begin{bmatrix} \frac{-\beta_3}{2\beta_0} \\ \frac{-\beta_4}{2\beta_1} \\ \frac{-\beta_5}{2\beta_2} \end{bmatrix} \tag{3.2.9.19}$$

The inverse soft iron is can be found from the square root of the diagonal matrix A:

$$S^{-1} = \begin{bmatrix} S_{xx} & 0 & 0 \\ 0 & S_{yy} & 0 \\ 0 & 0 & S_{zz} \end{bmatrix} = \sqrt{A} = \begin{bmatrix} \sqrt{\beta_0} & 0 & 0 \\ 0 & \sqrt{\beta_1} & 0 \\ 0 & 0 & \sqrt{\beta_2} \end{bmatrix} \tag{3.2.9.20}$$

The fit error is defined as:

$$\epsilon = \frac{1}{2G^2}\sqrt{\frac{\lambda_{min}}{M}} \tag{3.2.9.21}$$

3.2.10 Simple gyroscope calibration

Since the gyroscope measurements will be integrated to find the orientation angle, the bias offset will grow linearly over time, thus leading to a drift in the estimation. The constant bias error of a gyroscope can be calculated by taking the average of the measurements over a long period while the gyro is static. This bias can then be subtracted from future measurements to eliminate this error. In this project, the bias is calculated from measurements done in six different positions. Then the average value for each axis is calculated.

There was no calibration regarding this error type because it was not a critical problem for the estimation used in this thesis. Because the gyroscope is only used to indicate that the sensor is rotating, and not to find the absolute position, only compensation for the bias offset is needed.

$$b_x = \frac{max(g_x) + min(g_x)}{2} \tag{3.2.10.1}$$

$$b_y = \frac{max(g_y) + min(g_y)}{2} \tag{3.2.10.2}$$

$$b_z = \frac{max(g_z) + min(g_z)}{2} \tag{3.2.10.3}$$

3.2.11 Matlab script

In conjunction with developing the calibration method for the MEMS sensors, there was developed a script in Matlab. This script computes and tests the calibration parameters for the different sensors.

The first section of the code loads logged data from a CSV file, then split the table into three variables, accelerometer, gyroscope, and magnetometer, which containing M samples from the three axes. The data can be loaded from one or multiple files. The variables are matrixes of an Mx3 dimension, where each axis is a column vector in this matrix. The same section is the accelerometer and magnetometer data converted from ms^2 to g's for the accelerometer, and from micro Tesla to Gauss for the magnetometer. The conversion is done since these are the units of measurements used in the AHRS.

The next section filters the data with a simple recursive moving average filter. This filtering is done to remove the additive white noise in the signals, as illustrated in figure(31). The window size is found using Matlab's signal analyzer and analyzing the signal's power spectrum in the frequency domain before and after applying the filter. This type of filter is used because it smooths, and with small window size, the data will reflect the trends in the signal accurately enough for our purpose. Furthermore, it is not computationally heavy to use, since, after the initial calculation, only a subtraction of the oldest sample and an addition of a new data is necessary to filter the future samples. The recursive filter is developed as a separate Matlab function with two parameters, filter size and signal.

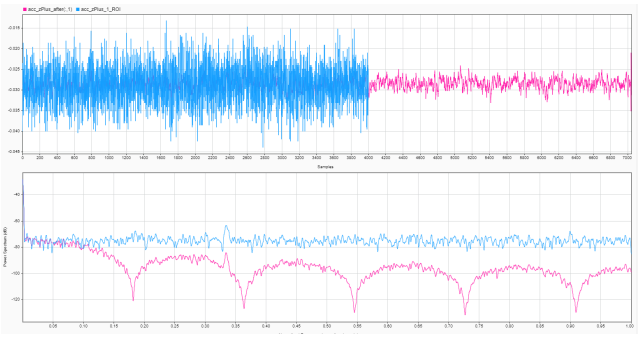


Figure 31: *Displaying in the top window the sensor signal from the accelerometers x-axis before and after filtering where the blue signal is before and pink is after. The bottom window shows the power spectrum in the frequency domain for before and after filtering.*

In the third section, the offset and scale distortion of the accelerometer measurements is computed. The distortion is calculated from six different static positions of the accelerometer. A matrix X and Y are constructed, where the X matrix will contain all the measurements, and Y will contain the corresponding

known real values. The scale and offset is then computed following the method in section 3.2.7. Then by using the calculated scale and offset to calibrate a different sampled signal.

The following section of the script calculates the electromagnetic distortion in the magnetometer signal. This scale and offset are calculated using the insignificant soft iron distortion method described in section(3.2.9). The computed values are used to calibrate a different signal then used for the least square method. The next section in the script computes the scale and offset using the significant soft iron distortion approach instead.

The sixth section of the Matlab script computes the gyroscope zero offset using six different static sampled positions. The zero mean offset is calculated from either the median or the average of the sampled values.

The penultimate section of the script is a simple method to find the scale and offset for magnetometer measurements. This method is described in (3.2.8). After finding the hard and soft iron distortion parameters, they are used to calibrate a sampled signal.

The last section of the Matlab script plots a comparison between the uncalibrated and calibrated signals. Furthermore, does it plots the raw magnetometer data in one figure and the calibrated in another, the plots are bout in 2D and 3D, where the 2D illustrates the samples in the coordinate planes xy, xz, and yz.

3.2.12 Translating Arduino software to Python

When translating the Arduino C code, it was essential to keep the inherent logic in the software code developed by previous years. Thus every function and variable in the code was changed to its equivalent in Python. However, the control structure in Python and Arduino C is not the same, and therefore not all statements in C can be used in Python. Switch-case is an example of a control structure in C that Python does not have. This structure was used frequently in the Arduino codes. However, a Switch-case can be replaced by multiple if-else statements. The difference between them is that a Switch-case evaluates only character or integer value, while if-else evaluates integer, character, pointer or floating-point type or boolean type. Furthermore, Switch-case executes one case after another until a break statement or end of the Switch-case, while if-else will execute either if or else statement. The differences are not vital for the code in question since all cases contained a break statement. So if-else could be replacing the Switch-case.

3.2.13 Software PLC

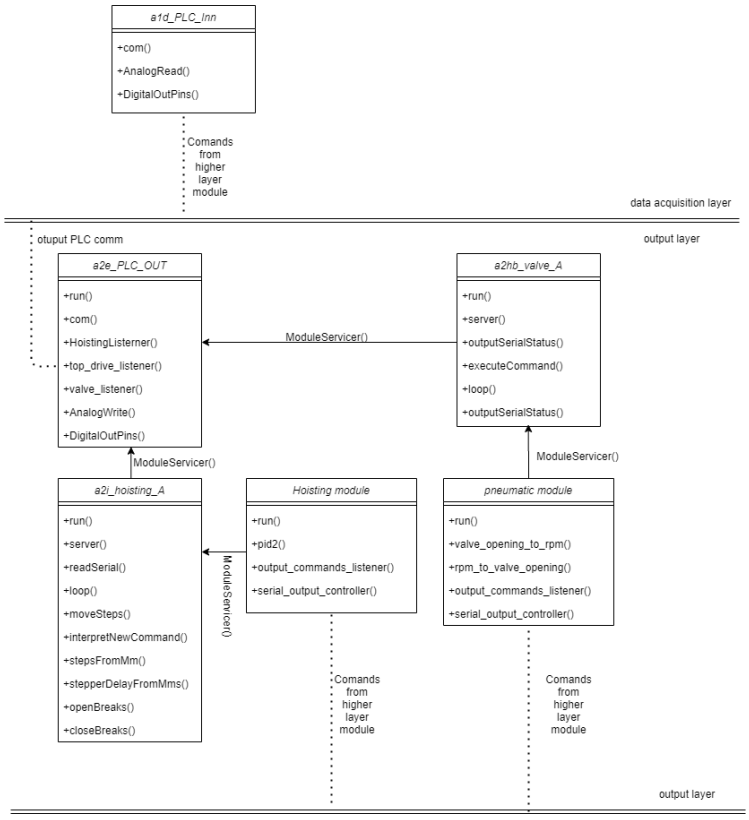


Figure 32: *Class diagram for the Software developed for PLC. Dotted line indicates incoming and out going communication beyond the layers, arrow indicate communication with in the same layer. Double lines indicate layer boundary.*

The software for the PLC mentioned in section (3.1.9) is located on the computer that serves as the central control system. This software is programmed in Python and is designed to be used with the control system architecture mention in (3.1.14). So each code that was on the Arduinos will be change into a module working in the control system at the output layer. Because of the problems with the Top drive and the time constraints associated with non-execs to the laboratory, only the Arduino codes for the valve and hoisting system got translated and tested. Some parts of the other systems were translated but not finished.

To keep the same internal logic of the software for the central control system, the old modules that communicated through serial USB needed to be changed. The only change done to them was to change the serial communication to the gRPC method. The PID-regulator for the stepper motors and other such functions already implemented in the old modules were not touched.

The new modules developed have the same communication between modules developed by last year's UiS Drillbotics team. Witch consists of servers and

clients for each module to be able to send and transmit data. The server is declared with a port number and initialized in the function server(), which uses the class ModuleServicer(). ModuleServicer() can stream different data structure that is declared by the functions in the ModuleServicer() class. The client in the module is a function call to the gRPC server with a specific port number and request.

3.2.13.1 a1d_PLC_Inn.py

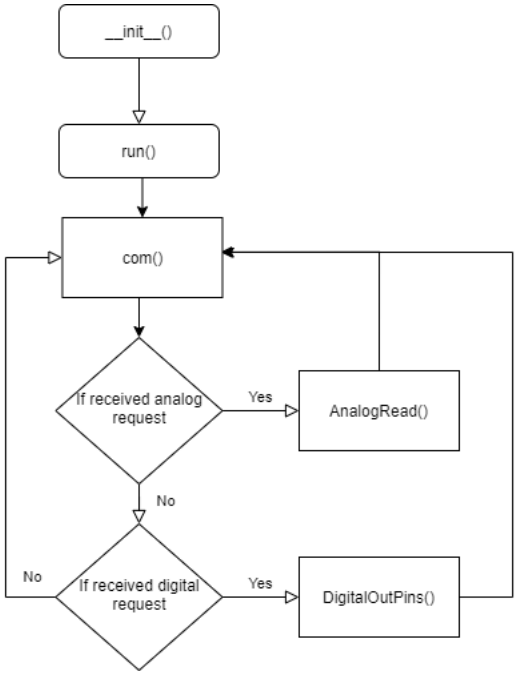


Figure 33: *Flowchart for the a1d_PLC_Inn module, wher com is server and client for the PLC inn module*

This software is located at the data acquisition layer of the system architecture for the central control system. This module consists of three functions, com(), AnalogRead(), and DigitalOutPins().

DigitalOutPins() has two parameters, pin_num and BOOL, and uses them to manipulate the different digital channels on the inn-put PLC. Where pin_num declares the channel number, and BOOL defines the channel's state. The communication to the PLC is all done through the imported mcculw library. To use this library to operate the PLC, the PLC must have been initialized and configured in the InstaCal software.

AnalogRead() function has one parameter, ch_number, and acquires data from the PLC. The parameter ch_number declares the channel to read. After reading the data to variable val, val is paced into a list together with the channel number and a time stamp.

This code is unfinished, so function com() does not work as intended, com() was supposed to work as the client for this module.

3.2.13.2 a2e_PLC_OUT.py

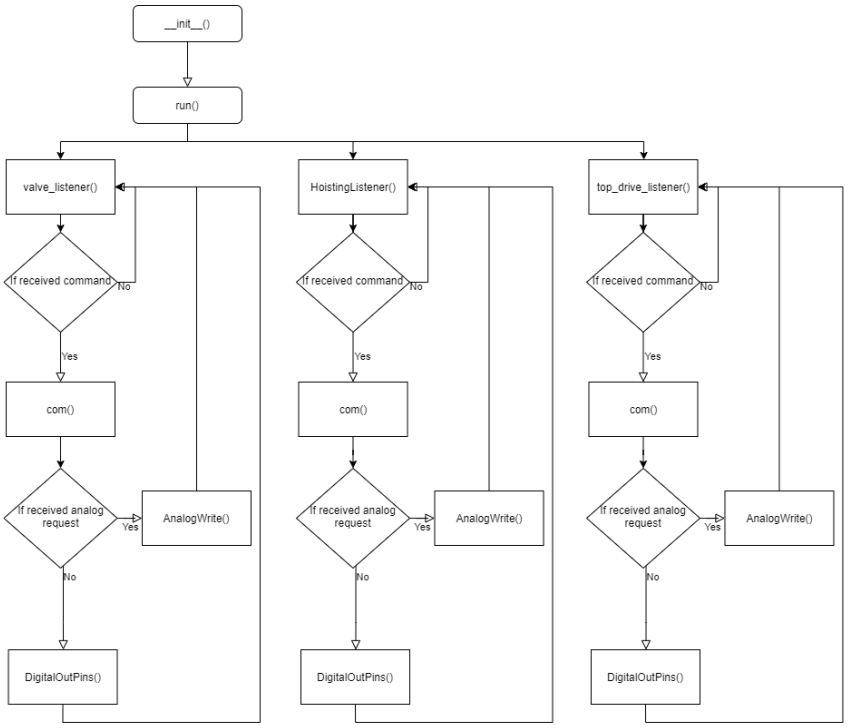


Figure 34: *Flowchart for the a2e_PLC_out module*

This software operates the out-put PLC and is a module in the output layer of the system architecture of the central control system. This code contains seven functions named, run(), com(), HoistingListener (), top_drive_listener(), valve_listener(), AnalogWrite(), and DigitalOutPins(). **HoistingListener(), top_drive_listener(), and valve_listener()** is defined as program threads and are clients in the module. These threads operate independently and are actively listening for commands from other modules. When a command has been registered, the com() function is used to decrypt de command.

com() function has three parameters, pin_num, value, and a. Where a indicates if to use analog or digital ports, pin_num is the channel number on the PLC to be used, value defines the state for the channel. com() uses either the DigitalOutPins() or AnalogWrite() function depending on the parameter a.

AnalogWrite() is the function that changes the state of the analog channels on the out-put PLC. The function has two parameters, ch_number and val. The parameter ch_number declares the channel to operate, and val defines the state of the channel. ul.v_out() function from the mcculw library is used

to write the command to the PLC. mcculw library operates the PLC, so no further program is needed on the PLC.

DigitalOutPins() has two parameters, pin_num and BOOL, and uses them to manipulate the different digital channels on the inn-put PLC. Where pin_num declares the channel number, and BOOL defines the channel's state. The communication to the PLC is all done through the imported mcculw library. To use this library to operate the PLC, the PLC must have been initialized and configured in the InstaCal software.

run() is the function that initialize the three threads HoistingListener(), top_drive_listener(), and valve_listener() and runs until the system is shut down.

3.2.13.3 a2hb_valve_A.py

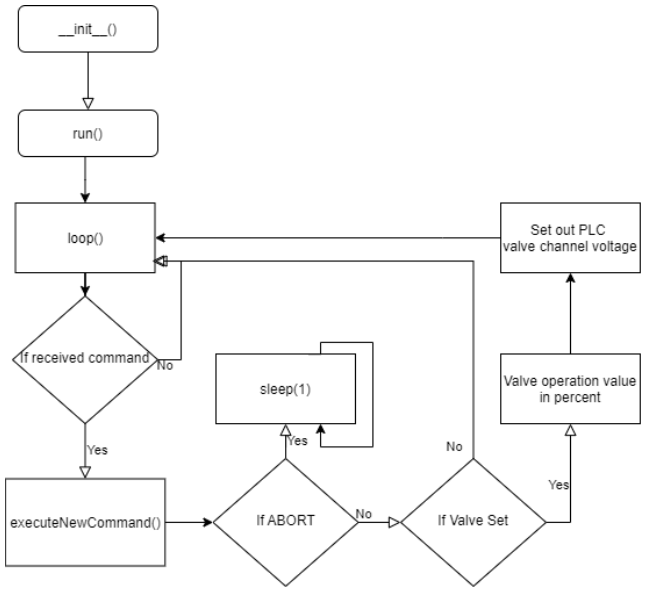


Figure 35: *Flowchart for the a2e_valve_A module*

This software is based on the Arduino code that operated the solenoid valve that restricts the airflow to control the drill bit RPM. This code operates as a module in the output layer of the control system architecture. Five functions define the purpose of the module; these functions are called server(), run(), outputSerialStatus(), executeCommand(), loop().

outputSerialStatus(), executeCommand(), and loop() are named after the functions in the Arduino valve code.

run() is the function that initialize the thread loop(). loop() program thread accts as the client for the valve module., where if a command is received, the

variable commandParameters will be declared with the command parameters operation, id, and amount. Then a call to the function executeCommand() will be performed. This function executes the internal logic thas was on the Arduino valve code. Which is first to check for what operation should be performed, it will either be the abort or valve-set operation. The abort operation will set the channel's value on the output PLC connected to the solenoid valve to zero, then use function outputSerialStatus() to send an updated status for the solenoid valve to the server for the valve module.

Valve-set operation will use the commanded amount to set the position of the valve. The amount is given in percent where zero is closed, and 100 is fully open. Since the analog channels of the output PLC have a range of zero to ten volt, the solenoid valve is programmed to read this value. Moreover, because the channels have a resolution of 16bits, the output state is computed from 0 to 65539. This value is used to set the voltage out fo the channel connected to the valve, where 0 is zero volts, and 65539 is 10 volts out. After sending the command to the PLC, an updating report is sent back to the system with outputSerialStatus().

outputSerialStatus() uses a producer-consumer method, where this function is the producer, and the module server is the consumer. What is produced is a list containing an updated rapport for the rest of the system, containing the last operation finished, the id of that operation, the current state of the valve.

3.2.13.4 a2i_hoisting_A.py

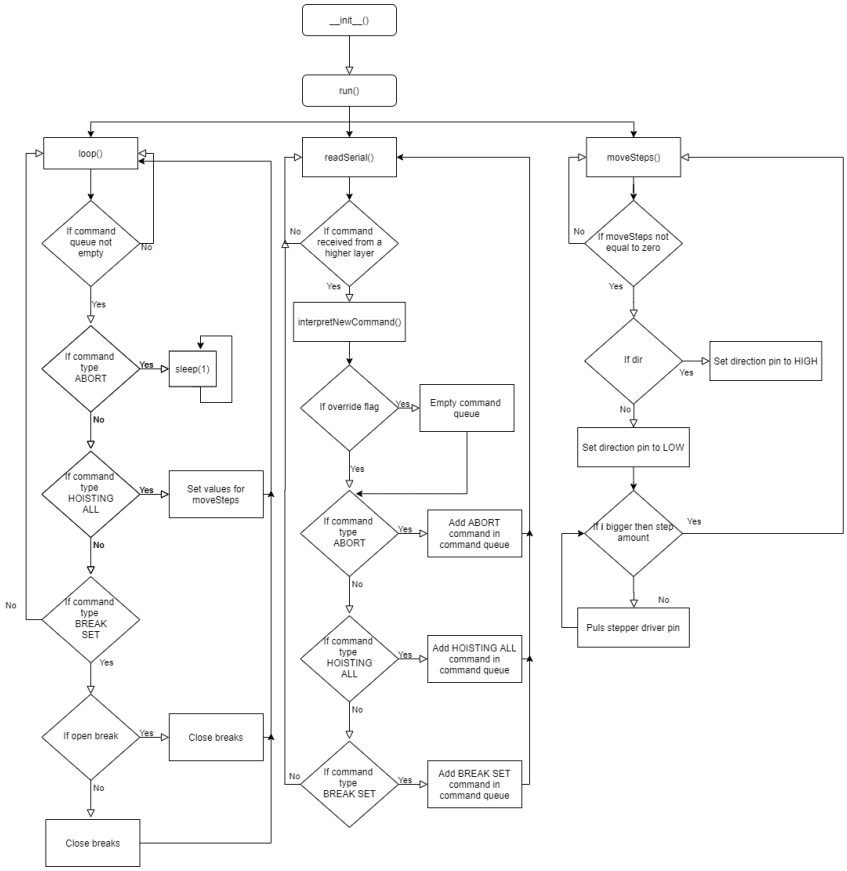


Figure 36: *Flowchart for the a2i_hoisting_A module*

This software is based on the Arduino code that operated the stepper motors, the actuators for the Top plate. This code operates as a module in the output layer of the control system architecture. This module has three main program threads called loop(), readSerial(), and moveSteps().

readSerial() is the client for the valve module and where the commands from other modules are interpreted. When a command is received, the function **interpretNewCommand()** is then executed to interpret the command.

interpretNewCommand() interprets the command by comparing the operation id received in the command message. However, the override flag will first be checked and, if true, will clear the queue for the operations to be executed. If the abort command is detected, the abort operation will be queued to be performed next. Else if hoisting all command is received, the operation for hoisting actuators one, two, and three will be queued together with the amount of displacement in millimeter, and the velocity in millimeter second, and the direction. The queue operates as a producer-consumer method in variable sharing between threads.

loop() is the main thread in the module and operates based on the commands queued by the interpretNewCommand() function. The command operation id will be used in a series of if-else-if statements, representing the switch case used in the Arduino. If command type abort, all operations are canceled, and the thread will go in an endless empty loop. If a move command type is received, move steps queue is declared with the move parameters amount of steps, stepper delay, direction, actuators to control. The amount of steps is calculated using the stepsFromMm() function that multiples the displacement in millimeter with the number of steps for one millimeter. The stepper dealy defines the speed in steps per second based on the velocity millimeter per second and is calculated using stepperDelayFromMms() function. If break type command is detected, then the openBreaks() or closeBreaks() function is used. These functions will send a direct command to the PLC channels associated with the brakes.

moveSteps() is the thread is responsible for controlling the channels the stepper-motor drivers are connected. If a movement is declared in the loop(), moveSteps() program thread will set the direct channel to high or lowe state based on the commanded direction. Nex is to check if the Top plate will not be beyond the upper limit (hoisted too high). If not, it will continue with moving the actuators by switching the channels associated with the stepper-motor commanded to move. The pulsing happens according to the stepper delay variable by using a busy-wait function. When the number of steps for each actuator is performed, a signal will be sent to loop() to indicate the movement has been performed.

3.2.14 Software MCU

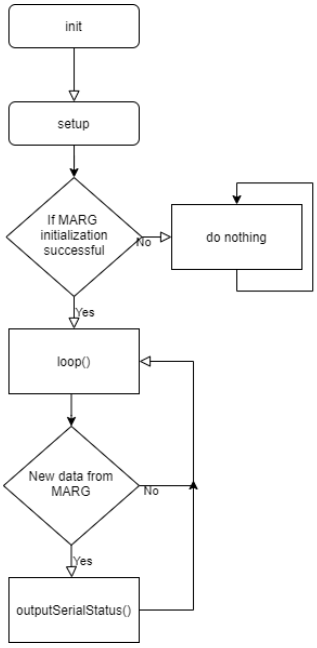


Figure 37: *Flowchart for the developed MCU software*

The software used to acquire the ICM-20948 MARG sensor measurements through the serial communication I^2C. This software utilizes the library developed by SparkFun for there sensor breakout board for the ICM-20948 (which is under an open-source license) to communicate with the ICM sensor.

The sensor card software starts with declaring the type of communication to use as I^2C. Then the setup() function is executed, here the serial USB baud rate will be specified, then checking if the sensor has successfully initialized. If successful, the code will proceed to set the sensor's sensitivity range and low-pass filter for the accelerometer and the gyroscope. The low-pass filter is a filter running on the ICM-20948 sensor. Hence the MCU will sample already filtered data.

After the initial setup for the MCU and ICM, the loop() function will be executed. This function requests the MARG sensor to send measurements for each of the three sensors and their axes. Next, the function outputSerialStatus() is executed to compose a string to be transmitted using the serial USB. The string contains a start bit "$" and then an id representing the measurement time, following with measurement data, and finishes with an end bit "@", all parameters in the string are separated by a ";".

$$\$; time; acc_x; acc_y; acc_z; gyr_x; gyr_y; gyr_z; mag_x; mag_y; mag_z; @ \qquad (3.2.14.1)$$

3.2.15 Software developed for tracing the drill bit position

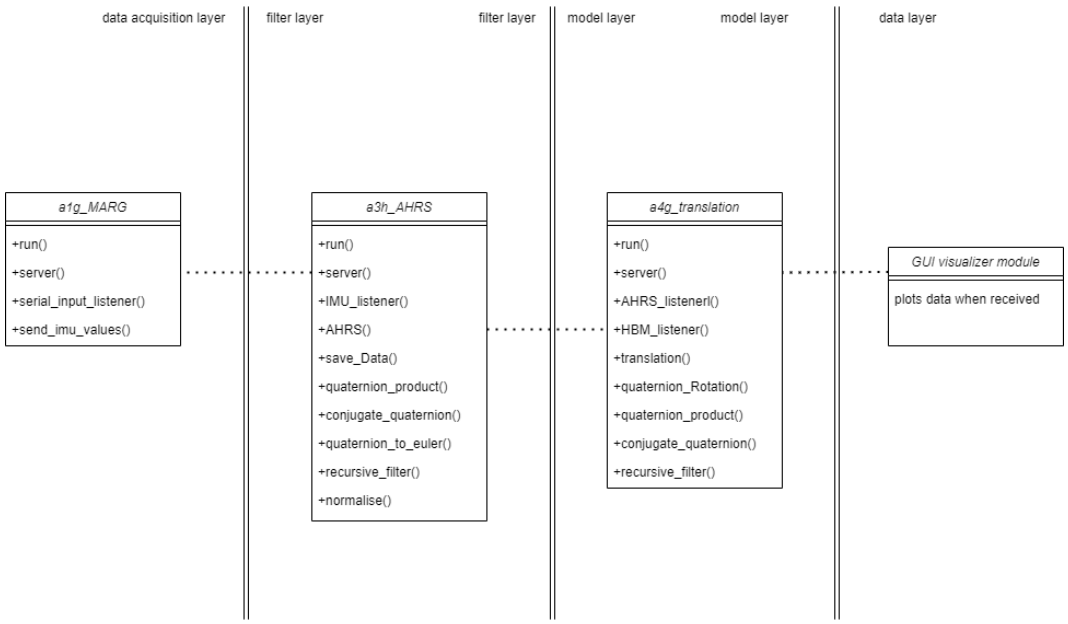


Figure 38: *Class diagram for the Software developed for tracing the drill bit position. Dotted line indicates communication beyond the layers. Double lines indicate layer boundary.*

There was developed software to estimate the drill bit position in the global coordinate frames. This software can either estimate the position based on MARG sensor measurements or a kinematic model of the drilling rig. The final estimates are done in the module called a4g_translation, and the estimation is done based on displacement and estimated orientation of the drill bit. Displacement is obtained from the HBM module that collects data from the distance sensor (3.1.13). When using kinematic sensor data from the BHA and Top drive is used to calculate the drill bit position. However, an estimated orientation for the drill bit can also be used. This estimate is based on the sensor data from a MARG sensor (3.1.2) and is computed in the a3h_AHRS module. The sensor data is transmitted from the MCU in the BHA and received in the a1g_MARG module. The a4g_translation module transmits the estimated x,y,z coordinates to the data layer.

The new modules developed have the same communication between modules developed by last year's UiS Drillbotics team. Witch consists of servers and clients for each module to be able to send and transmit data. The server is declared with a port number and initialized in the function server(), which uses the class ModuleServicer(). ModuleServicer() can stream different data structure that is declared by the functions in the ModuleServicer() class. The client in the module is a function call to the gRPC server with a specific port number and request.

3.2.15.1 a2g_MARG.py

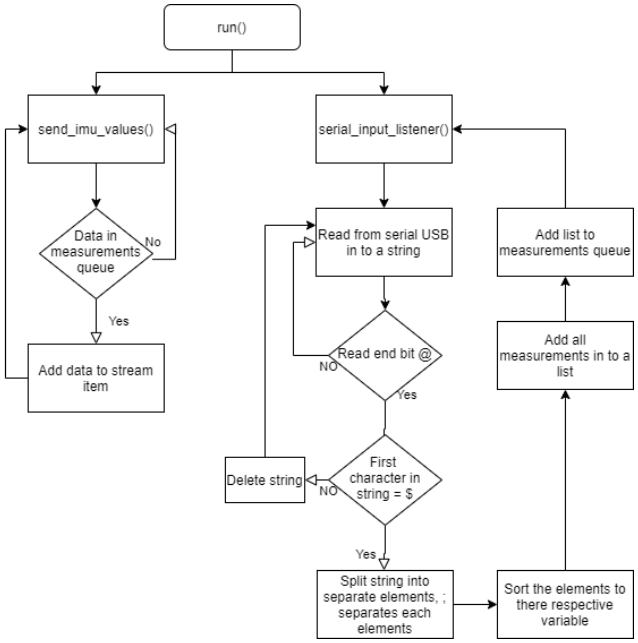


Figure 39: *Flowchart for the developed sensor fusion filter*

This code is defined as a module in the data acquisition layer of the system architecture for the control system. This module has the primary purpose of communicating with the downhole MCU to collect sensor measurements from the MARG sensor. The code for this module is divided up into four functions called server(), run(), serial_input_listener(), and send_imu_values(). Where serial_input_listener() and send_imu_values() runs as separate program threads. They run as different threads to separate the modules' two critical tasks, which is the data acquisition and transmit the data to other modules. By running them in two separate threads, one task will not take precedence. It is done in order not to obstruct the data flow between modules and serial communication with having them waiting on each other.

serial_input_listener() is the function that reads the data from the serial USB connection. The data is read until the end-bit attached to the Arduino serial string (3.2.14.1) is detected before splitting the string into separate elements. Each element in the string is divided with a ";" character to distinguish them easily. After each element is sorted to their respective variables, they are converted from character string to their original data type (int, float). The data will then be placed in a list used as the fixed-size buffer used as a queue in a producer-consumer method of sharing a variable between two program threads. Moreover, there is a security implementation related to corrupt data strings. Such as no start-bit and loss of elements; in these cases, the data string will be discarded and advance to read the next serial string in the USB buffer.

send_imu_values() will continuously run in a loop where it will wait until the fixed-size buffer from serial_input_listener() have elements to use. When there are elements in the buffer, it will add a new element into the stream queue, going to other modules.

3.2.15.2 a3h_AHRS.py

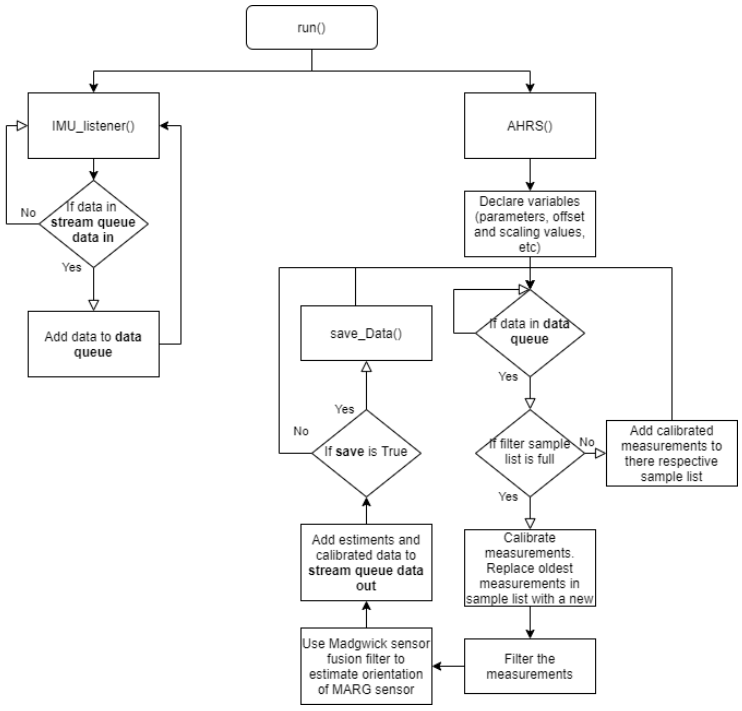


Figure 40: *Flowchart for the developed sensor fusion filter*

To estimate the drill bit position in real-time, calculations using a sensor fusion filter will be necessary for stable prediction. Since estimation from one type of sensor will, in this case, not yield a satisfactory result. Because of the need for axes of sensitivity in all coordinate axes for the drill bit, to ensure adequate predictability. Furthermore, it can not be obtained by using only one type of sensor; however, for each different type, the capabilities for the axes of sensitivity will improve. For example, a gyro can sense angular rate in any axis, but can not sense the start orientation. Moreover, this code uses the Madgwick sensor fusion filter to estimate the orientation of the MARG sensor using the measurements from a magnetometer, gyroscope, and accelerometer.

The estimation is done in the AHRS() function in the software a3h_AHRS. a3h_AHRS is placed in the filter layer of the control system architecture and has its primary functions split into two program threads, IMU_listener() and AHRS(). This module communicates with the module a2g_MARG and a4g_translation in order for the control system to track the drill bit position in the global frame.

IMU_listener() will continuously run in a loop where it will call the server of the a2g_MARG module to acquire the measurement data from the MARG sensor. Next, the data will be added to a buffer that will act as a queue for the AHRS() thread to use. This queue is operated following the producer-consumer method.

The AHRS() thread will do the calculation required to estimate the orientation of the MARG sensor. At startup, will this thread declare all the variables used in the calculation, including list size, boolean variable start state, value for the calibration parameters, and filter gain values. The next task is to fill the list containing sampled measurements used for filtering. There is used a recursive moving average filter to slightly smooth the data, in order to reduce the white noise. When the filter sample is full, the initialization of this module is finished. The measurements are collected from the queue created in IMU_listener(), and the code will not progress if this queue is empty. The magnetometer data is then calibrated with the parameters found using the method described in section (3.2.9), before being filtered. The same will be done for the accelerometer and gyroscope measurements, however, with the calibration methods specified in (3.2.7) and (3.2.10) respectively.

The next step is to use the Madgwick sensor fusion algorithm specified in section (2.7), but first, the calibrated magnetometer, gyroscope, and accelerometer measurements are placed in their respective vectors. Then the vectors for magnetometer and accelerometer are normalized before the calculation of the gradient of the solution surface computed by using equation (2.7.0.7). The penultimate step before estimating the orientation of the MARG sensor using equation (2.7.0.24) is to normalize the gradient of the solution surface and use the equation (2.7.0.23) to compute the derivative of the estimation.

After normalizing the estimated orientation of the MARG sensor, the estimate is placed in the module server's queue to be streamed to other modules.

3.2.15.3 a4g_translation.py

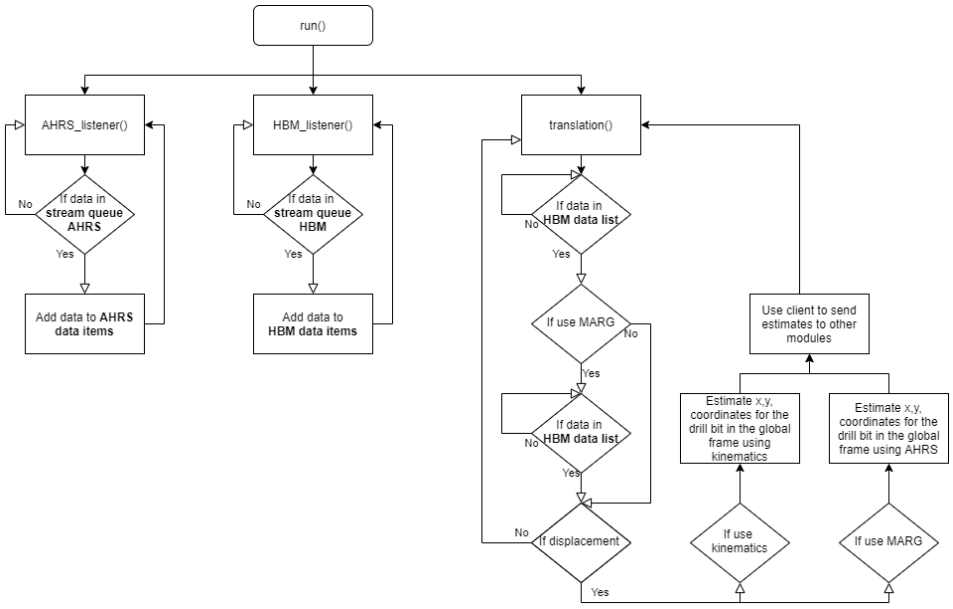


Figure 41: *Flowchart for the developed software for the final estimation resulting in the current drill bit position*

a4g_translation is the module that does the final estimation of the drill bit position and is placed in the model layer of the control system architecture. To track the drill bit position, it is not enough to just know the orientation of the bit. The translation is also essential; the translation is known by using the measurement from a distance sensor. This distance sensor measures the distance from the top of the rig to the top plate. By using the initial distance, the displacement can easily be calculated from future measurements. This module can use both the kinematics and the AHRS method to compute the drill bit's current position.

This module is divided into three program threads to collect and calculate the drill bit position. AHRS_listener(), HBM_listener(), and translation() are the functions in the code that acts as the three mentioned program threads. Where both AHRS_listener() and HBM_listener() have the same function; however, the difference is the data collected. Both functions work as a client communicating with other modules in the control system. AHRS_listener() listens to the AHRS module, and HBM_listener() listens to the HBM module.

translation() is the program thread that estimates the current drill bit position in the global coordinate frame. The first task for this thread is to wait on elements from AHRS_listener() and HBM_listener(). Next is the declaration of the translation vector to be used in the final estimation. Then a check for displacement from the last estimate is done before a new estimation can start. If it is determined to have been a new displacement, the position is estimated

using the AHRS data or the kinematic method. The orientation estimated from the AHRS module and the translation vector, is used to perform a quaternion rotation. The result from the rotation will result in a new estimated position for the drill bit in the coordinate sensor frame. This estimate will be rotated one more time with a standard rotation matrix to yield the global coordinate frame position. When using the kinematic method, the equations (3.2.5.12, 3.2.5.13, and 3.2.5.14) is used to estimate the drill bit's new position. These estimates are then shared with other modules using the server in this module.

3.3 Procedure

This section will give a detailed explanation of the procedures used to generate the results in this thesis. Description of the research problems and their suggested solutions will be presented, together with reasoning behind the choices taken and a description of any experimental setups used.

3.3.1 Changing Arduinos to PLC

3.3.1 Up until last year (2019), Arduinos have hade the main task of controlling the actuators and gathering data from sensors. Last year most of the data acquisition responsibility was given over to the measuring amplifier manufactured by HBM. This year we were going to take over the task of changing out the Arduinos to PLC to do the controlling of the actuators. The systems we are changing to Arduino is primarily the systems that are needed for our team to realize the plan design in 3.2.1. Thus the focus will be on changing out the Arduinos for the Hoisting, Rotation, and Pneumatic. There are two more Arduinos outside of these systems, which are for circulation and whipstock; these will not be of priority to change. Since they are not going to be for this year's design.

There are some crucial restrictions to follow when it comes to changing out the Arduinos for PLC. The first is to preserve the program logic of the already made software used on the Arduino. Such that after the transition, the system still operates in the same way as before the change. Therefore, the system must be tested before and after to ensure that it still runs the same. The test can be, e.g., to measure the voltage given from the Arduino and the PLC for the same operation, then to check if the measurements are equal or to measure physical (distance moved for the hoisting system, pressure for the pneumatic system) changes on the rig and compare results.

The second is to keep the same electrical setup (such as actuators, drivers, power supply, wiring) that is already implemented, to ensure that the only change to the system is the transition from Arduino to PLC. To guarantee that any fault that was not previously present in the system can only come from the software in the PLC. Thus make it easier to locate "bugs" in the code.

The transition from Arduino to PLC will be done in four steps. The first step is to translate the Arduino code into python since it is the programing language used in the central computer control system. The second step is to integrate the translated code into the existing control system architecture. The integration is done by composing a new module in the output or data acquisition layer of the control system architecture to have the responsibility of running the translated code. Next is to create modules that have the responsibility to communicate with the PLCs, which also will be in the output layer. The last

step is to test the new system with PLCs and Arduino to discover faults in program logic or bugs in the code. To ensure the rig operates equally with or without PLCs. More details of the transition will be presented later with results from testing.

3.3.2 Hole sensor card

For this year's Drillbotics competition, a new rule was added that requires the teams to have downhole measurements. Hence there will be a focus on making a sensor card to acquire these measurements and transmit them to the surface. This section will explain the reasoning behind the component choices by presenting problems to consider when designing the sensor card presented in section 4.2. Furthermore, will the procedure for designing and testing the card be described.

When designing the sensor card, the first step was to locate possible difficulties to overcome to make a reliable downhole measuring tool. Some of them are space restraint, communication, power management, environmental and induced noise, e.g.. The next step was designing the circuit for the sensor card and was done using the recommended circuit (from the datasheet) for each component and solutions to the stated problems. Thirdly was to design the circuit traces for the PCB and was done in such a way to ensure high flexibility of the card when components are soldered. The design of the PCB was done using the electronic design automation software from Autodesk called EAGLE. There were design two versions of the PCB, as shown in section 4.2.

This PCB will be installed in the bottom hole assembly (BHA) and used by the central control system. These sensors would need both power lines and signal lines to be connected to the control system. However, the number of signal wires and the distance from the BHA to the Top Drive will cause a problem with space and unnecessary exposure to noise. Proper shielding can mitigate most of the external noise, but long leads can cause induced noise. One way to reduce this noise is to reduce the length of the leads for I/O as much as is practically possible. There will also be a voltage drop caused by the resistance in the wire. Solution for this can be shorter wires or measure current instead of voltage, but not all sensors are designed to send a current signal.

On the other hand, some sensors use a communication protocol instead of an analog signal to convey the sensor data. Furthermore, not all of these communication protocols are as robust or can be used over a long distance. Thus, it was decided to have a sensor card in the BHA where its responsibility will be data acquisition and eco the data to the control system. The data will be conveyed from the sensor card to the central computer using the USB protocol since it is robust and stable over a relatively long distance and can

be used as a power supply for electronic devices.

Furthermore, are there some space constraints for the size of the sensor sub, making it essential to have a sensor card as small as possible. Thus a flexible circuit board was decided to use since these circuit boards are thinner and can easily be formed to align with the curve. Hence the sensor sub will not need to have a square form that will take up more space and harder to place then a curve shape that matches the walls of the sleeves in the BHA.

The Atmega328P microcontroller in section 3.1.1 was chosen for the data acquisition—the reason being that this microcontroller is small but can have a high clock speed. The default speed for this microcontroller is 8MHz but can be programmed to run at 16MHz. Running at a higher speed will result in a lower program loop time, therefore executing the program script faster. Furthermore, will the frequency of the acquisition be higher and can communicate with a higher baud rate. According to the datasheet for the Atmega328P, there must be a crystal unit connected to pin 7 and 8 to be able to run at 16MHz.

Therefor a crystal unit will be needed on the sensor card. Since Atmega328P datasheet specific, the crystal unit will be used in the parallel-resonant state when connected to pin 7 and 8. Therefore a load capacitance will be required, and using the method described in 3.2.4 to coumpute its value.

To be able to communicate using USB with the Atmega328P, conversion between USART to USB will be needed. Since the baud rate will be too high to be used over a long wire, requirements suggest to have a serial USART to USB converter on the sensor card. Hence FT232RQ-REEL () chosen to be implemented in the circuit board. This component is small and have a lower power consumption at 5V, which is adequate since the USB can deliver maximum 500mA.

This sensor card's primary purpose was in the start planed to do sensor measurements and communication with the sensors. However, because of changes in the design of the BHA and time constraints related to none access to the laboratory, most of the sensors were removed from the design. The MCU's primary purpose is communication with the MARG sensor over I^2C and eco it to the control system computer.

ICM-20948 was chosen as the MARG sensor because of its small dimensions, accuracy, and cost. I^2C was chosen as the communication protocol based on speed and essentially because it only needs two wires, effectively resulting in a smaller circuit board. A drawback of this device is that it operates at 1.8V, effectively having a logic signal voltage different from the MCU's 5V. Consequently, a level shifter was implemented into the circuit board to shift the 5V logic to 1.8V logic. The 1.8V will be provided by one of the MCU analog outputs, but the best solution would be to have an external power source.

3.3.3 Tracing of drill bit position

A part of this project is to find a suitable solution for tracing the drill bit position in real-time. To be able to trace the position of the drill bit, the software described in section (3.2.15) was developed. This system is developed to benefit from the mandatory downhole sensors, which was a new addition to this year's Drillbotics competition. Hopefully, this use of the sensors would give the team a higher score at the end of the competition.

This system was divided up into three main parts: the data acquisition module, the Attitude and Heading Reference System (AHRS) module, and the estimation module. The data acquisition module was developed to sample data measurements from a MARG sensor, then distribute it to the other modules. Next, the AHRS module was developed to estimate the orientation of the downhole sensor, consisting of a MARG sensor. The estimation of the sensor in the earth frame is done using the Madgwick sensor fusion algorithm, based on calibrated measurements. The calibration of the different sensors in the MARG was done using the methods in section (3.2.7), (3.2.10), and (3.2.8,3.2.9), for the accelerometer, gyroscope, and magnetometer, respectively. These methods were programmed in the Matlab script in section (3.2.11), to calculate the calibration parameters used in the AHRS module.

The last module was developed to estimate the drill bit position based on the sensor readings and the estimated sensor orientation. One of the methods tested to track the drill bit was the use of kinematic. The plan was to use the forward kinematic to calculate the position based on sensor readings in the bottom hole assembly (BHA). Moreover, the estimation using the AHRS data is done using quaternion rotation on the top plate displacement to calculate the drill bit's current position.

This system was tested using the measurements from the MARG sensor and using simulated data for the displacement and measured angle. The simulated data was used since the drilling rig was no longer operational after braking down under a test drilling. The simulated data is made based on the ideal well path, where the displacement and angle consist of points from a start value to an end value. For example, the displacement data would go from 0 to 100 millimeter with 500 samples. These simulated data would be read from a text file to simulate the readings from a sensor.

For testing the estimation based on top plate displacement and AHRS estimates, only the displacement used simulated values. In order to test the orientation estimation done by the AHRS. The testing of the AHRS was done with a static orientation at a specified angle and a pure translation along the z-axis. The standard deviation of the estimation errors was found by the Root Mean Square Error (RMSE) of the x,z vector angle for the estimation points in space. The expected values were defined as a point on a curve with $x = d \cdot sin(\theta)$, and

$y = d \cdot cos(\theta)$ positions for a dynamic test and a straight line with a specific inclination, figure (42) illustrates an example of this to curves. THE RMSE value would then determine the error of the estimated orientation angle, which would describe the performance for the estimation with and without calibrated sensor values. Furthermore, it determines if the AHRS estimates performance would be significant enough to track the drill bit position in real-time.

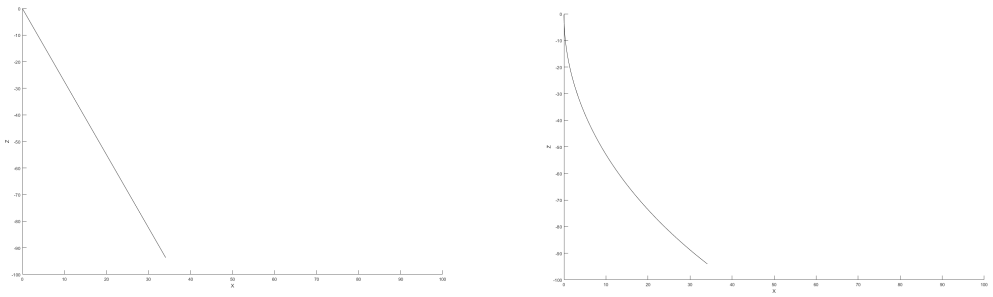


Figure 42: *Illustrating the simulated well path used in testing the estimation of drill bit position using AHRS*

Since the drilling rig was out of commission, the forward kinematic method described in section 3.2.5 was tested using simulated sensor reading of the angle of the bend sleeve and the top plate displacement, was done in order to test the kinematic model. The simulation was done by reading from a vector containing simulated measurements. These measurements were made by mathematically calculate an ideal well path for drilling through rock, figure (43) illustrates the path used. Then use these measurements in the developed software instead of the sensor reading to compute an estimated position. To calculate the error of the drill bit position estimations, the Root Mean Square Error (RMSE) is used to compute the standard deviation of the estimation errors. This error is then used to find fault and draw any conclusion from the result.

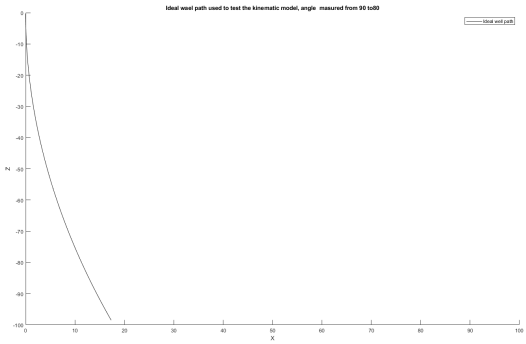


Figure 43: *Illustrating the simulated well path used in testing the kinematic model*

The tests was conducted at static 20, 30, and 60 degrees of orientation around sensors y-axis, and a dynamic rotation from 0 to 10 degrees for the AHRS method. For the kinematic method, dynamic rotation from 0 to 10, 30, and 60 degrees was conducted.

Calibration methods were also tested by comparing a sampled signal be for and after calibration, where the calibration parameters were found from a different sampled signal.

4 Results and Discussion

In this section, the result from calibration, changing from Arduino to PLC, and testing for the estimation system will be presented. Such as calibration of gyroscope, magnetometer, and accelerometer, position tracing, sensor card design. The individual results will be discussed within each subsection.

4.1 Change from Arduino to PLC

The changing of Arduino to PLC, as described in section (3.3.1), was started on but did not finish for reasons that will be described later. The first step was to translate the Arduino code to Python. The Arduino code for the pneumatic and hoisting system was translated in full, will the top-drive only partly. The reason for not finishing the top-drive code was because of the damage to the driver for the motor. Under transport home from the competition last year, the connector used to communicate with the driver was damaged. Thus time was used to try to fix the driver. After desoldering the connector and replacing it, the communication with the motor seemed to be working. However, even when there was communication, it appeared something else was still wrong with the driver. According to the program for the driver, the motor is trying to start; however, it gets a torq passing the limit, and an emergency shut down is issued. An attempt to find the cause of the problem yielded no results, so the top-drive could not be used. Thus the program could never be tested, and the work with translating the rest of the code wound be meaningless.

Moreover, the code for the hoisting and the pneumatic system was finished and tested. The pneumatic system was tested by measuring the voltage from the Arduino and PLC to check if they would give the same value for the same operation. This test the pneumatic system passed. With measuring the signal over with the solenoid valve, there was some voltage drop. The voltage drop was measured, and the average for the Arduino was 0.15 and 0.8 volt for the PLC. This difference will cause some changes to the system's reliability since the RPM of the drill bit is based on the airflow.

Furthermore, with less voltage drop, more of the signal range can be used. Moreover, the best solution to remove the voltage drop would be to change to a current signal, since the Burkert 8605 can read current signals from 0-20mA or 4-20mA only a new signal source is needed. Furthermore, a current signal would not experience this type of signal loss and is more resilient to noise. However, this type of signal source is the output PLC, not capable of producing. Therefore, this system is useable and would function as well as the old system; however, considering another output device would increase the reliability of the system.

The pneumatic and the hoisting system circuit diagram can be seen in figure (44), this circuit is the same as previous years only the Arduinos are changed out with the PLC. The translation of the hoisting system and changing out the Arduino was not a success. There was a problem with the hardware design if the USB-3114 made it unsuitable for the stepper-motor. One problem is that it can only one value per channel at one time (per write). i.e., the hardware was not designed to buffer an array of data and pace it out based on an internal/hardware clock.

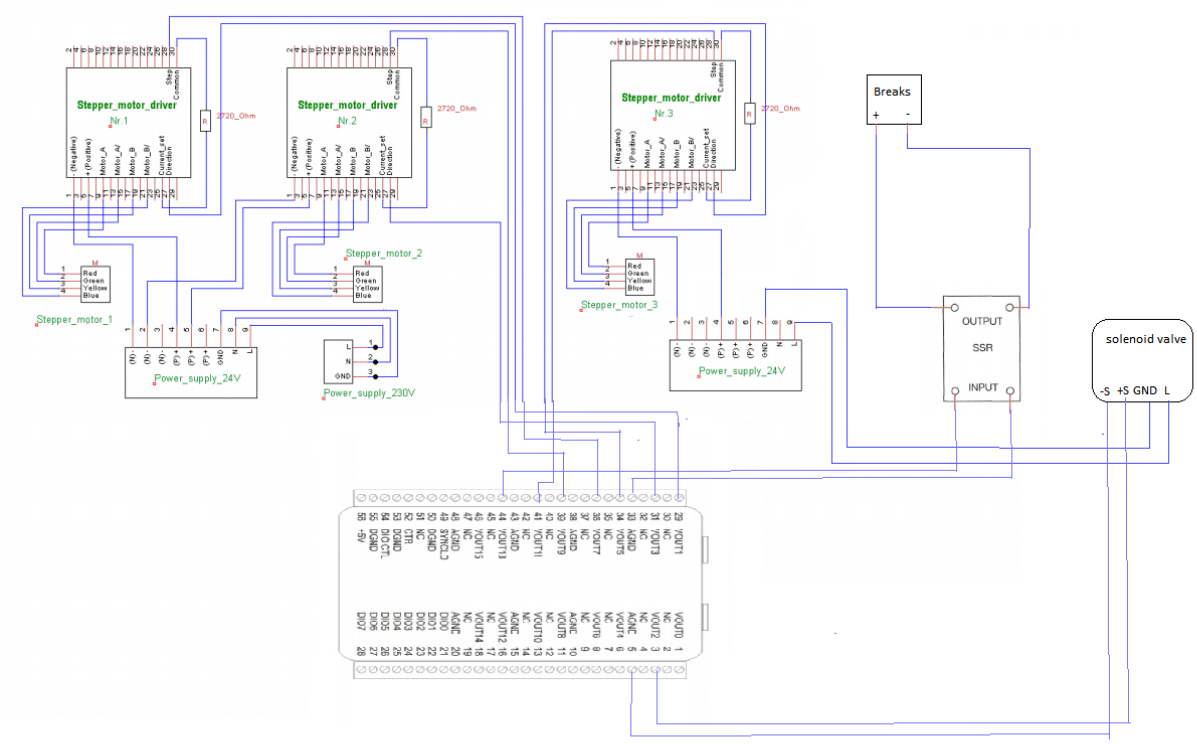


Figure 44: *Schematic for the output PLC*

The second problem is that the USB-3114 can only use the max throughput when using one channel, e.i., with the use of multiple will divide the throughput over all the channels used. For example, with the use of two channels, the throughput for these channels would be 50Hz and not 100Hz. Therefore the USB-3114 is unsuitable for our system that will use nine of the channels simultaneously. There were conducted research for a new alternative and concluded with trying som industrial PLCs. However, the investigation for the

new industrial PLCs was not finished because of the difficulty getting the order in time, as a result of the current situation of spring 2020.

The work done with changing out the Arduinos to PLC was not successful. The reason being the PLC chosen by the last year's UiS Drillbotics team was not suitable for controlling all of the actuators. That and the uncertainty about the time frame of delivery and access to the laboratory because of the 2020 crisis. However, overall, there was much time spent on change out the Arduino; however, not much has changed from the old system. What could have saved much time was to have read the datasheets for the PLC more carefully before starting. The idea of using PLC is excellent but needs PLCs more suitable for the tasks. The use of an industrial PLC instead of DAQ type PLC should be much more suitable for the system. Such as a PLC from Bachoff, which has many pros, such as a big community, eas of expending the system with there detachable modules. Cons is that the control of the actuator will not be centralized in the central control system computer. However, there are much flexibility in communication types and the ability to operate the IØindividually. What must be said that I may have a bias toward such a system because of earlier experience with them.

4.2 Sensor card design

This section will present the two different design versions for the sensor card. There were two designs since there was some fault with the first one, and changes to the sensor card's final goal. Moreover, at the end of the section, report the result of the finished product.

Design
Displayed in figure (45) is the complete circuit diagram for version one. The problem with this design is the incapability to reset using USB; this makes it hard to upload scripts and change the bootloader. Also, the MARG sensor is supplied with an incorrect voltage. Furthermore, are components not strangely placed to make the board more flexible, apparent in figure (46). These faults are fixed in the second version, and some redundant components were removed. The wire to board connector, device J2 in the circuit diagram, was removed to make the board more flexible and save space, resulting in smaller dimensions for the sensor card. This connector was meant to be used for the external sensor in the BHA. However, it is unnecessary to use such a connector, only needs copper pads to solder the wires to the PCB.

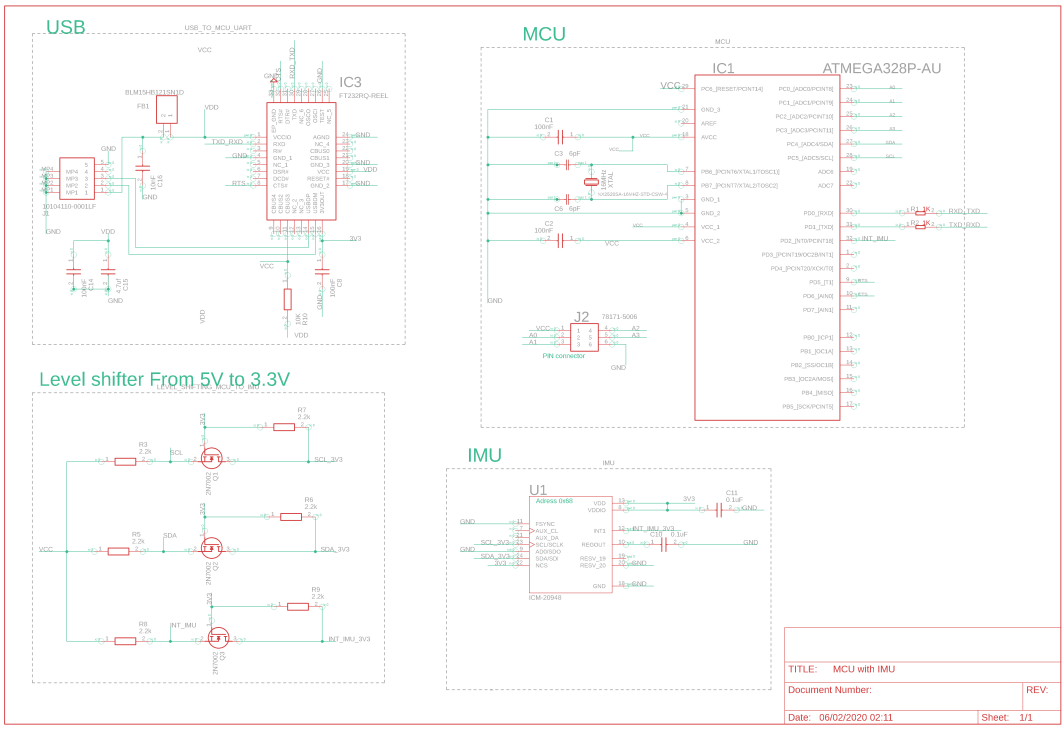


Figure 45: *First version circuit diagram for downhole sensor card*

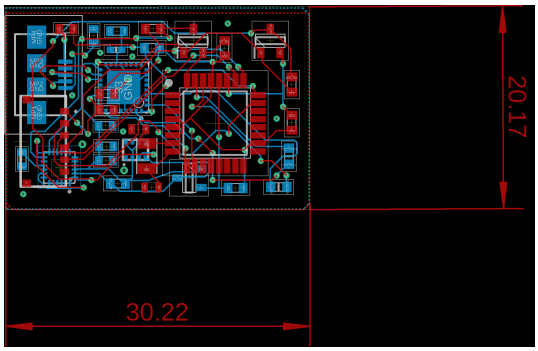


Figure 46: *First version trace schematic for PCB, with out ground traces*

The schematic in figure (47), is the final design of the electrical circuit, and figure (49) shows the trace and component placement on top of the PCB, and figure (48) shows the bottom. The diagram is divided into four groups, USB, MCU, Level shifter, and MARG. The rest of the section will be a more in-depth description of the different groups.

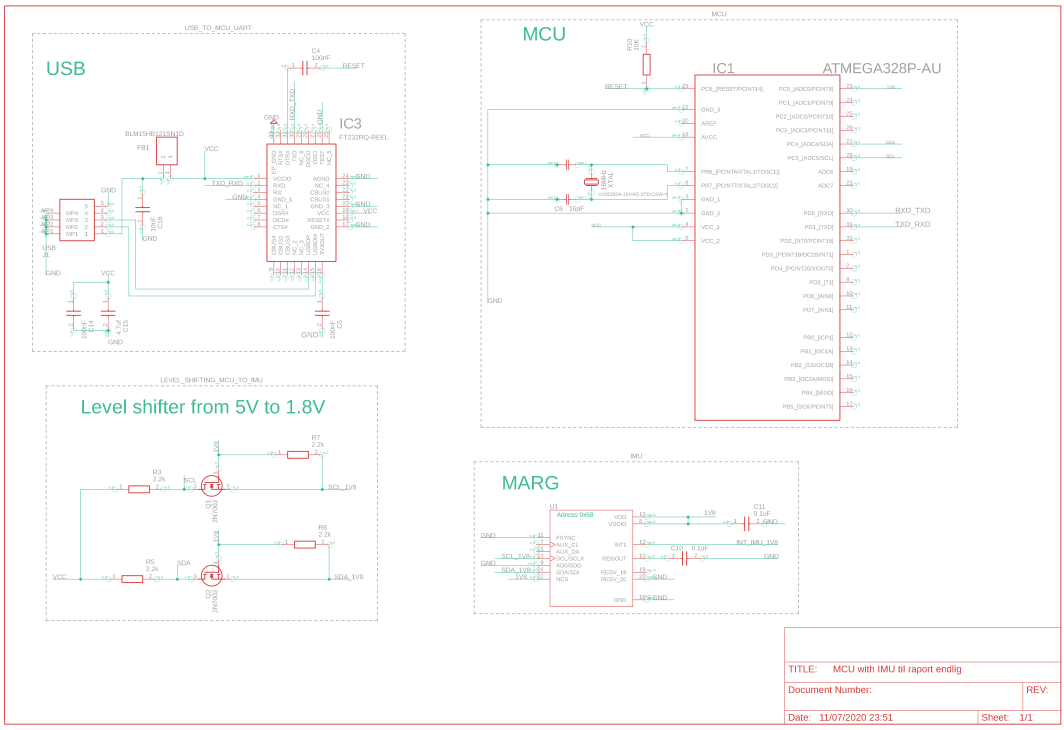


Figure 47: *Schematic for the second version of the sensor card electrical circuit.*

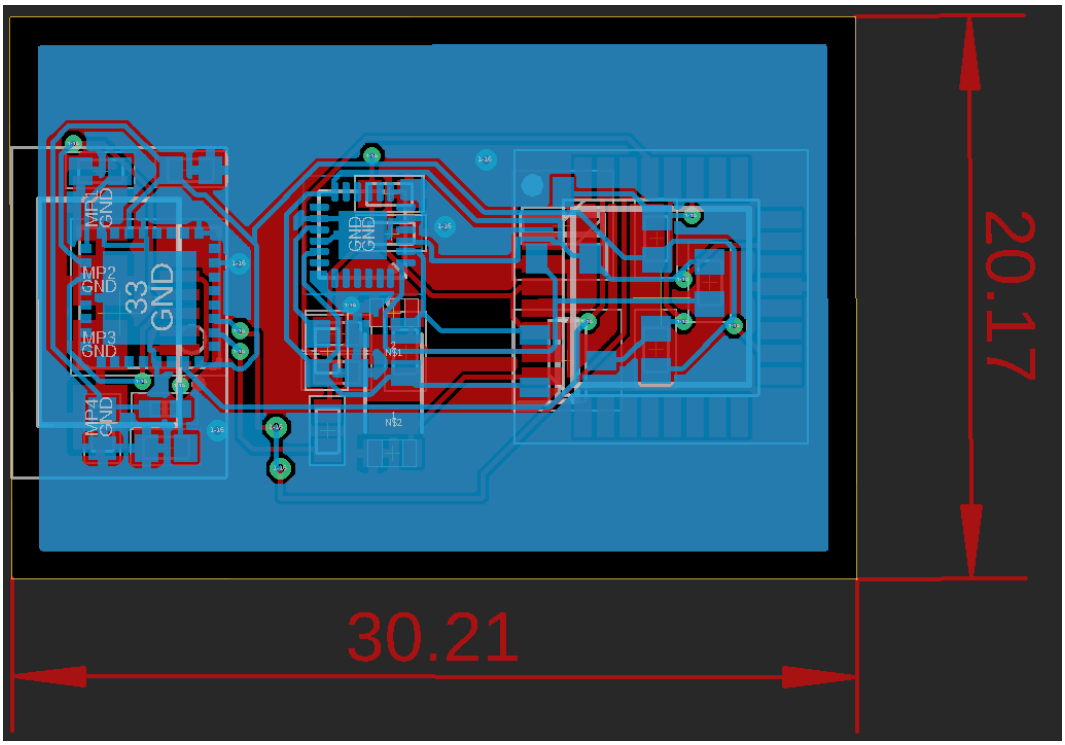


Figure 48: *Illustrating the traces and components on top of the sensor card, for the second design of the PCB*

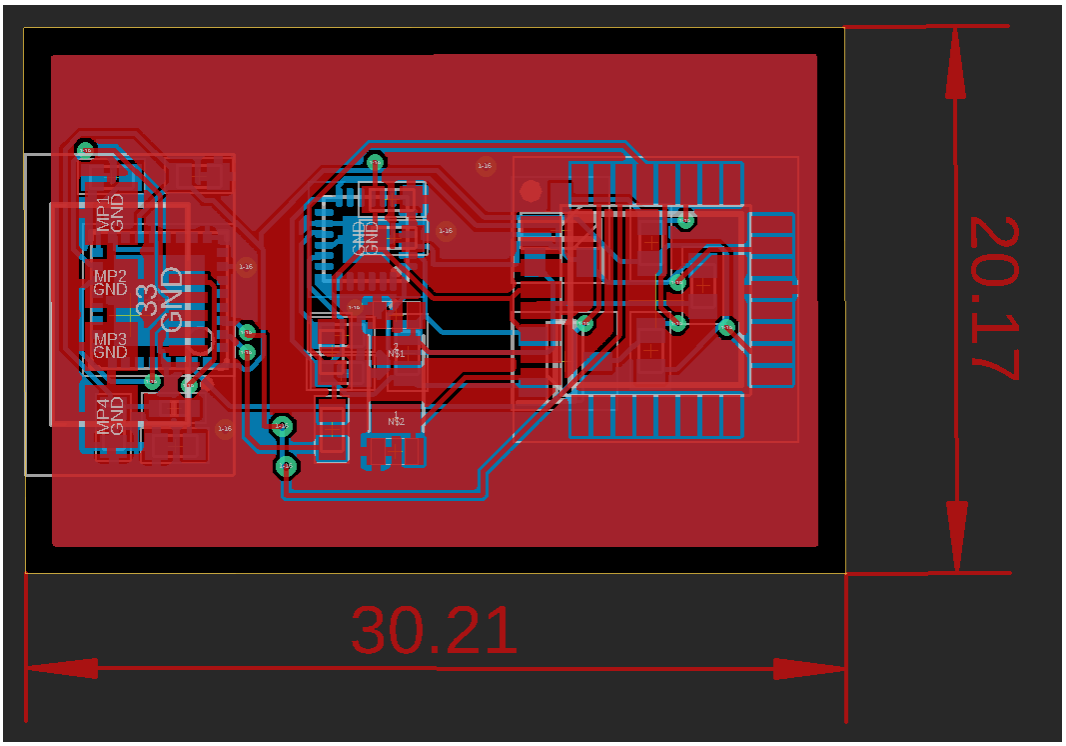


Figure 49: *Illustrating traces and components on bottom of the sensor card, for the second design of the PCB*

Figure (50) illustrates the group responsible for the USB to USART conversion. This circuit is designed by following the specifications and recommendations in the datasheet for FT232RQ-REEL [34]. It is recommended to use a ferrite bead in series with the USB power supply and pin 1 (power pin) on the FT232RQ-REEL. Furthermore, it is a 100nF capacitor coupled to the ground right before the ferrite bead. These two components are used in order to suppress high-frequency noise on the USB power line. The communication to the MCU is done through TXD (pin 30) and RXD (pin 2), and the reset by USB is done using DTR# (pin 31).

The USB cable will be connected to the USB connector U1, where pin 1 is the power supply for the sensor card with the ground on pin 5. USB DATA+ and DATA- is on pin 3 and 2, respectively, where these are connected to USBDP and USBDN on FT232RQ-REEL. In this setup, FT232RQ-REEL will have an operating current of 15mA according to the specification in the datasheet.

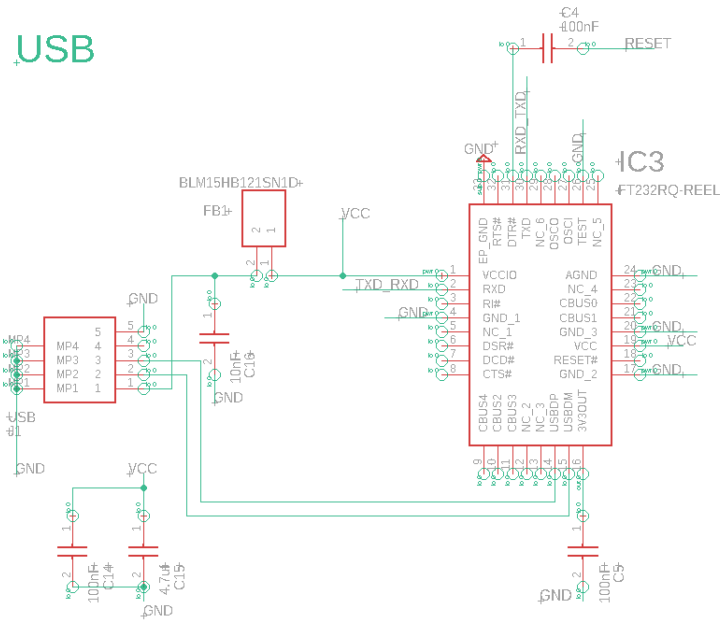


Figure 50: *Circuit diagram for the USB to USART*

The MCU group consists of one ATmega328P-AU configured to a clock speed of 16MHz, figure (51) is the schematic representing the MCU group of components. Wher A 10kΩ resistor is coupled to the positive voltage (VCC) from the reset pin to ensure a well-defined voltage when DTR# pin on FT232RQ-REEL is open. Pin 30 and 31 are the pins for USART communication and are therefore used to communicate with the FT232RQ-REEL. I^{2}C communication can be done with pin 27 and 28, which is connected to the level-shifter for communication with MARG sensor. Analog output on pin 23 is used to supply 1.8 volts to the MARG sensor and the low voltage level for the level shifter.

To run the ATmega328P-AU with a clock speed of 16MHz, the crystal unit in section (3.1.4) is connected to ATmega328P-AU's internal oscillator circuit trough pin 7 and 8. Where using equation (3.2.4.2) for the crystal unit gives 16pF for the two load capacitors C3 and C6. When the ATmega328P-AU runes at 16MHz, it uses a maximum of 14mA.

$$C1 = C2 = 2 \cdot 8 = 16 \tag{4.2.0.1}$$

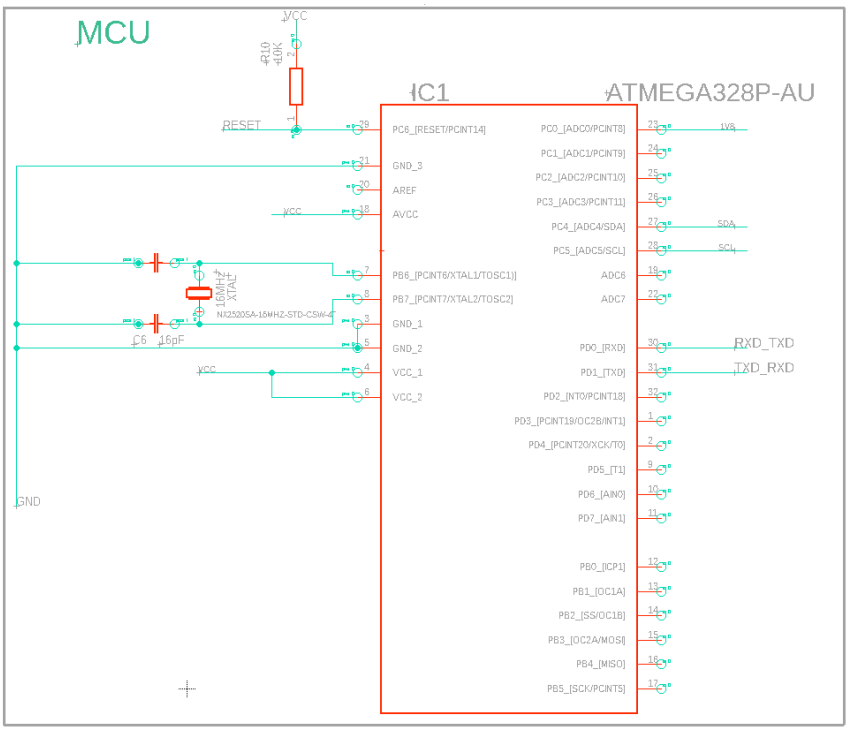


Figure 51: *Circuit diagram for the MCU*

The MARG and MCU's voltage logic level is dissimilar because of the different operating voltages, where MCU operates at 5V logic and MARG at 1.8V. Thus the I^{2}C signals from the MCU most be shifted to a lower level and vice-versa. Therefore is a bi-directional level shifter on the sensor card used to solve this problem. The level shifter is composed of an N-channel MOS-FET and pull-up resistors, as shown in figure (52). Where the source pin on the MOS-FET is connected to the signal line from the MCU, and the drain is connected to the MARG signal line. The gate of the MOS-FET is connected to the 1.8 voltage source.

The maximum and minimum pull-up resistors are calculated by using equation (3.2.2.1) and (2.3.4.1). Where $t_r = 300ns$ as stated in MARG datasheet [32] to be the i2c rise time, and then the bus capacitive load is defined by the product of the copper trace capacitance and the average length of the signal traces. The copper trace capacitance is found by calculating the capacitance per millimeter ($\frac{pF}{mm}$) of trace used on the PCB, by using equation (2.3.5.1), with the specification listed by the manufacturer of the PCB, where width layer height 0.065mm, trace width 0.15mm and trace thickness 0.0347mm. Thus the copper trace capacitance is:

$$C[pf] \approx \frac{0.264(4.5+1.41)}{Ln\left(\frac{0.598 \cdot 0.065}{0.08 \cdot 0.15 + 0.1 \cdot 0.0347}\right)} \approx 1.69 \frac{pF}{mm} \tag{4.2.0.2}$$

Then using the trace length for the I^{2}C signals to compute the bus capacitive

load with the calculated trace capacitance. The average trace length was measured to be 17.568mm. Therefore the total bus capacitive load is computed to be:

$$C_b = \frac{pF}{mm} \cdot tracelength = 1.69 \cdot 17.568 = 29.69 \tag{4.2.0.3}$$

By using the calculated capacitive load gives a maximum pull-up resistor to be :

$$R_p(max) = \frac{300 \cdot 10^{-}9}{0.8473 \cdot 29.69 \cdot 10^{-}12} = 11925\Omega \tag{4.2.0.4}$$

To calculate the minimum pull-up resistor value the equation (3.2.2.1) is used for a VDD of 1.8V, the minimum resistor value is then computed to be:

$$R_p(min) = \frac{1.8 - (1.8 * 0.3)}{0.006} = 210\Omega \tag{4.2.0.5}$$

The pull-up resistor can be chosen to be between 210Ω and 11925Ω, for this design was a 5.5kΩ resistor chosen based on speed and power trade-off. Furthermore, is 5.5kΩ the closes standard value resistor to the median of the minimum and maximum pull-up resistor values calculated.

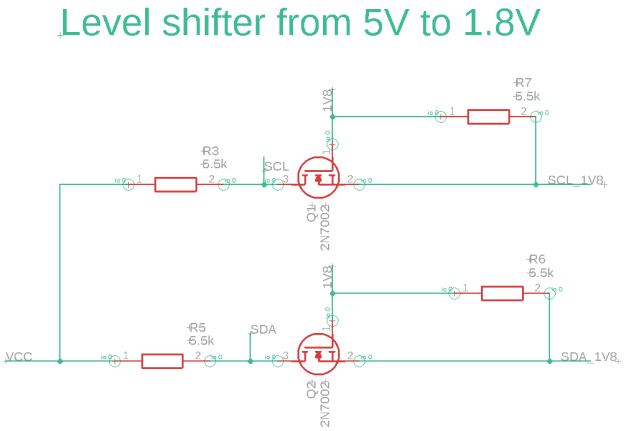


Figure 52: *Schematic for level shifter from 5 volt to 1.8 volt*

The MARG sensor used is an ICM-20948, and in its datasheet specifies a recommended operating circuit for bout communication over ISP and I^2C. The circuit recommended for I^2C is used for this sensor card and is illustrated in the circuit diagram in figure (53). When all three sensors in the ICM-20948 are in use, it uses 3mA. The recommended operating circuit is used to ensure the device runs as specified in the datasheet.

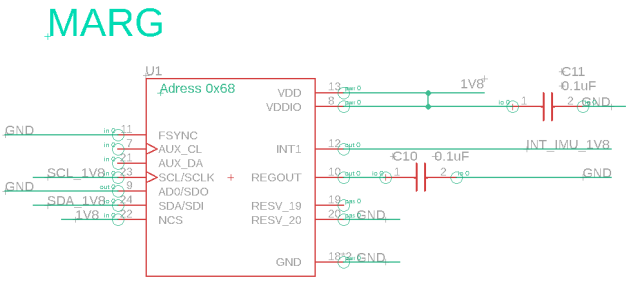


Figure 53: *Schematic for level delen krets kortet til sensoren*

With all of the current devices on the sensor card operating at full capacity, the current consumption is approximately 32mA. Therefore the USB should not have a problem supplying the sensor card with enough power.

Results

A PCB print of the first design was ordered in February and arrived in the middle of March; the resulting print is shown in figure (54). After arrival, the MCU and the USB group of the design were soldered to the board. Then, the ATmega328P-AU communication was tested through the USB to USART device; there was a connection but was unable to upload a new sketch. The reason being the problem with resetting the ATmega328P-AU through the USB communication. Resetting the device is essential to set the MCU in the correct state to upload a new sketch. The solution was to make a temporary bridge connection between the necessary inputs and output of the MCU and the converter. Thus a sketch could be uploaded, and a new firmware could be boot-loaded to the ATmega328P-AU. The boot-loading was necessary to make the MCU run at 16MHZ clock speed instead of the manufacturing default 8MHz. The method in section (3.2.6) was then performed to change the firmware settings to that of an Arduino UNO running at 16MHz. The Arduino got booted, and a sketch was uploaded to test the MCU. Since the Arduino worked as intended, the values for the load capacitance for the crystal unit was correctly calculated.

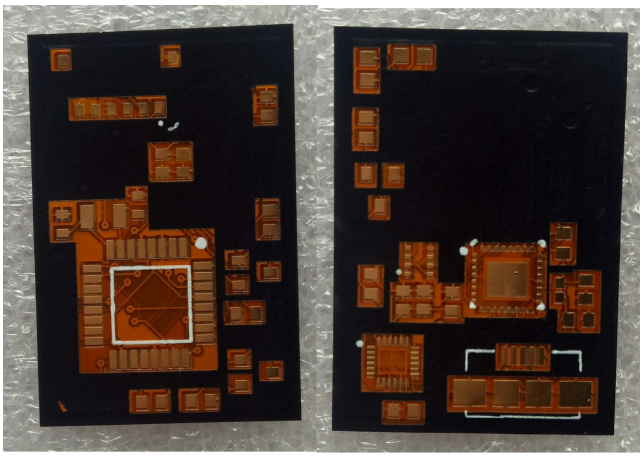


Figure 54: *Image of the printed PCB design version one, the top of the board is on the left and bottom on the right.*

The next step in the process was to solder the MARG and level shifter group to the board. Then the mistake with incorrect supply voltage for the MARG sensor was detected; version one design used 3.3 voltage from the USB to USART instead of a 1.8 voltage from the MCU. However, the level shifter was tested by bringing a connection from one of the analog pins of the ATmega. When tested, the pull-up resistors were changed to the one chosen for version two design. Furthermore, the I^2C signal on both ends of the level shifter reached dere respective LOW and HIGH logic levels, as evident from the figure (55). The blue signal is the 1.8 logic, and the yellow is the 5-volt logic, where the red dotted line is the HIGH (1.26V) logic level, and the solid line is the LOW (0.54V) for the 1.8 logic. Hence the pull-up resistors are chosen to be the correct value.

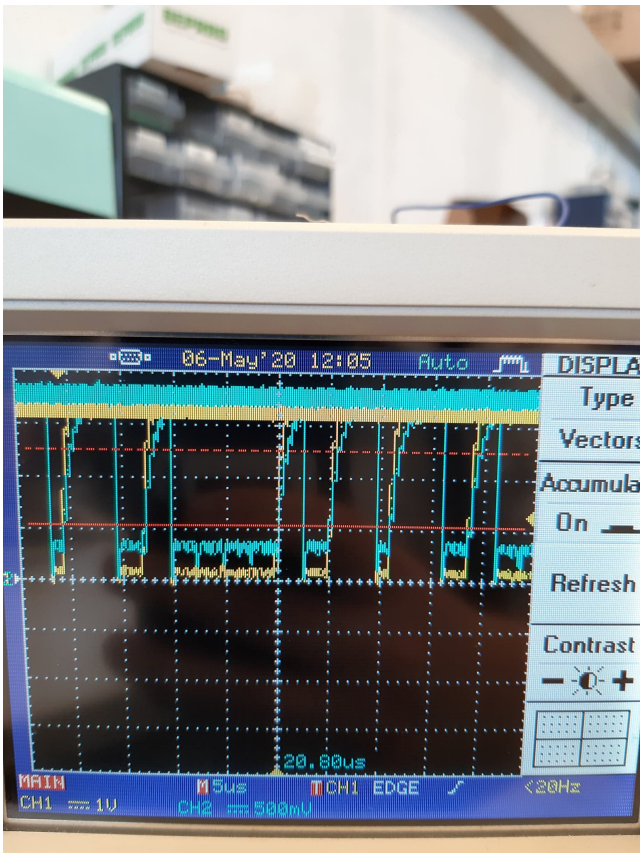


Figure 55: *Image of the printed PCB design version one, the top of the board is on the left and bottom on the right.*

Because of the uncertainty about delivery and access to the laboratory associated with the social distancing, the second version was not ordered. Hence, the AHRS testing was done using the breakout board of the ICM-20948 mentioned in section (3.1.3), instead of the sensor card. Moreover, because the rig was no longer operational after a drilling test, some components were overheated and too late to order new ones. Thus the breakout board could then easily replace the sensor card in the test since it need not be inside the BHA.

Moreover, the flexibility of the ordered sensor card made it evident that the components' placement should be more thoroughly thought out. The card was quite flexible around its length shown in figure(), but not so around its width when the components were soldered. Thus the use of a flexible circuit board should take less space than a rigid. The design process of the sensor card should have been done differently. Instead of starting with designing the circuit in a schematic, then order a printed PCB, a mockup circuit should have first been tested, e.g., used a breadboard. The components on the sensor card that were tested are the Atmega, level shifter, and the USB to USART and their associated calculated elements. These systems worked as attended when some individual faults with the first version were corrected. What is left is to test the new design with the ICM-20948 sensor connected.

4.3 Result of sensor calibration

The sensors in the ICM-20948 is factory calibrated; however, after soldering on the PCB, additional calibration can be necessary. The testing of the calibration is done by comparing the uncalibrated to the calibrated. The comparison is made both visually through plots, and for accelerometer and gyroscope is the RMSE used to find the residual error between expected values and predicted values. Furthermore, the fit error will be used for the magnetometer comparison to express the efficiency of the calibration.

4.3.1 Accelerometer calibration

The accelerometer is one of two vectors used to describe the ICM-20948 orientation relative to the earth frame. Thus the accelerometer measurements were calibrated using the method in section (3.2.7), where the least square method is used. The sampled data used to calculate the calibration parameters were measured when the sensor was static with one of its sensitivity axis pointing along the +z axis of the earth coordinate frame. There was done six static measurements with the +x, -x, +y, -y, +z, -z was pointing along the +z axis of the earth coordinate frame, as illustrated in figure (56). Thus since the gravitational acceleration equals 1, the expected value would be 0, 1, or -1 depending on the sensor axis orientation. Figure (57) shows uncalibrated, expected, and calibrated values where the sensor z-axis pointed along the +z axis of the earth coordinate frame.

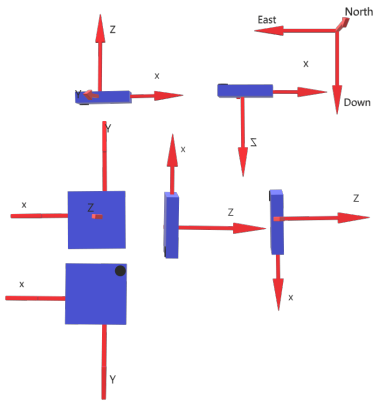


Figure 56: *Example plot of the uncalibrated, calibrated, and expected value for the accelerometer measurements, when the accelerometer axis is pointing along the +z axis of the earth coordinate frame.*

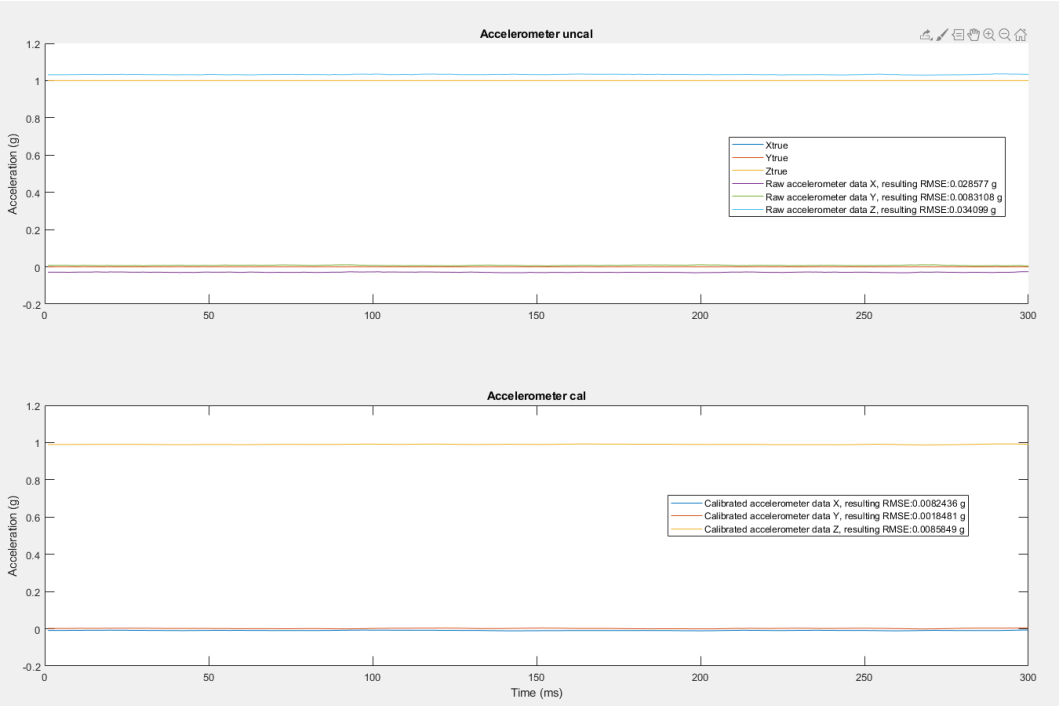


Figure 57: *Example plot of the uncalibrated, calibrated, and expected value for the accelerometer measurements, when the accelerometer axis is pointing along the +z axis of the earth coordinate frame.*

By calculating the RMSE from data samples logged from the six different orientations and then averaging the uncalibrated and calibrated value, table (3) is generated.

Data	X	Y	Z
raw data	0.015986 g	0.013826 g	0.034953 g
calibrated data	0.003051 g	0.010573 g	0.011057 g

Table 3: RMSE values calculated from six five-minute-long accelerometer samples, with a sampling rate of 100Hz.

From this table, the x-axis has a 0.012935g less deviation form the expected value than the raw data, for y-axis 0.010775g, and the z-axis 0.023896g. Thus the calibrated signal is closer to the expected value than the uncalibrated, e.i., the calibration improves the accuracy of the reading.

4.3.2 Gyroscope calibration

The gyroscope is not used to describe the orientation of the sensor in the earth frame; however, it is used to detect rotation. Thus the gyroscope was calibrated to remove the zero offset in the data; this is done by using the method in section (3.2.10).

There are used six sets of sampled data to find the calibration parameters. The six singles are measured when the sensor is static in different orientations e.i., the expected value is zero.; thus, the same data set for the accelerometer is used.

The RMSE will be used on the sampled signal to test how much the calibration improves the static measurements. When calculated for uncalibrated and calibrated measurements, the RMSE will tell how close the data is to zero rad/s. Figure (58) show an example of raw, calibrated, and expected values.

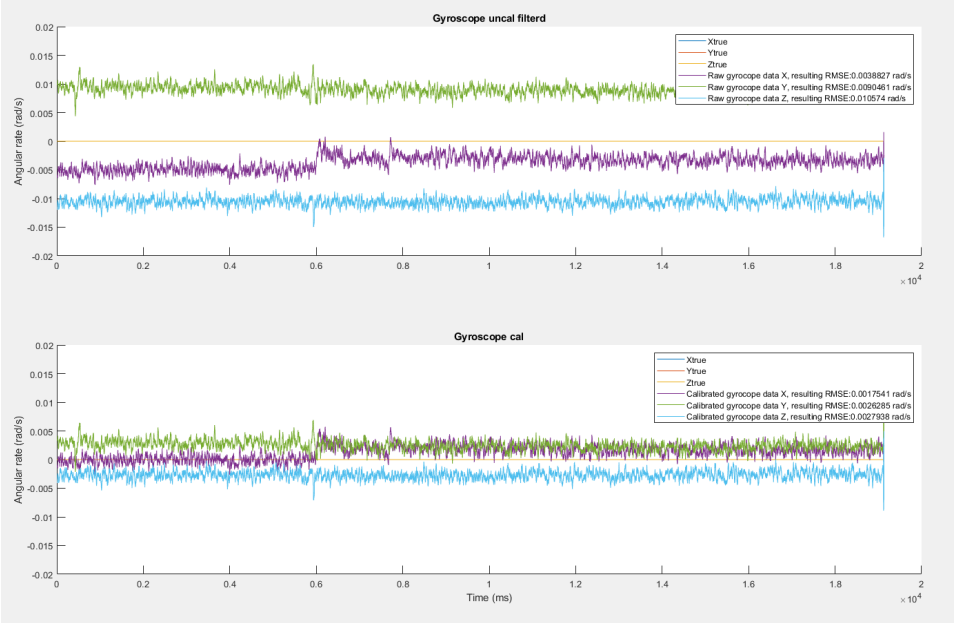


Figure 58: *Example plot of the uncalibrated, calibrated, and expected value for the gyroscope measurements, when the sensor is static.*

The RMSE was calculated from a sampled signal over a time of ten minutes. The resulting RMSE for the raw and the calibrated measurements is shown in the table (4).

Gyroscope	X	Y	Z
raw data	0.00691 rad/s	0.019583 rad/s	0.039739 rad/s
calibrated data	0.002604 rad/s	0.003358 rad/s	0.002192 rad/s

Table 4: RMSE value computed from a sampled signal of length 10 minuts, witha 100Hz sampling rate

Results indicate that the calibration is valid and achieve a more accurate static signal. Since the RMSE value is lower for the calibrated values than the raw sampled data, indicating that the calibration method has removed elements of the bias offset in the gyroscope data.

4.3.3 Magnetometer calibration

The magnetometer is one of the two vectors used to describe the ICM-20948 orientation relative to the earth frame. Furthermore, it is the most sensitive to environmental noise. This noise comes in the form of soft and hard iron distortion of the measured magnetic field. Thus the three methods of calibrating the magnetometer sensor were tested, one called simple, insignificant, and significant method; these methods are described in section (3.2.8), (3.2.9), and (3.2.9) respectively.

Insignificant soft iron distortion
In this test, the sampled signal hade an insignificant soft iron distortion. For the simple method, the performance will be determined visually from the plot. Furthermore, for the significant and insignificant method, their respective fit error will be used to determine the performance.

The sampled signal to be calibrated is a signal that is rotated around each sensitivity axis for the magnetometer sensor. Figure (59) illustrates the raw samples in a 3D perspective, while figure (60) shows the 2D representation of the measurements.

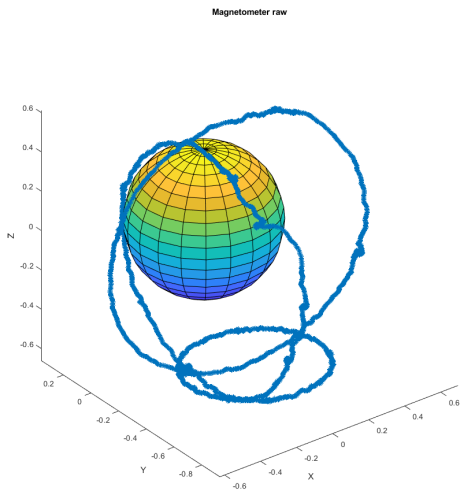


Figure 59: *3D representation of the raw magnetometer measurements when the sensor is rotated around each sensor sensitivity axis.*

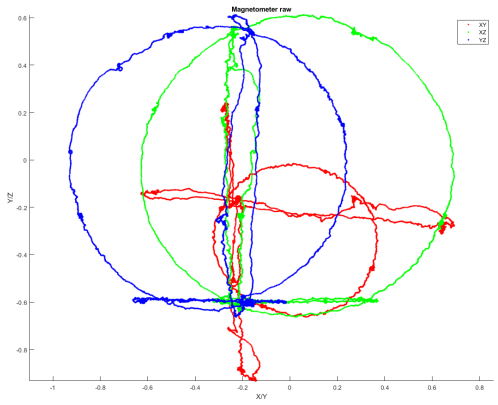


Figure 60: *2D representation of the raw magnetometer measurements when the sensor is rotated around each sensor sensitivity axis.*

Evident from the figures that the sample signal has hard iron distortion that most be offset to center the signal. After using the simple method, the data have been centered, as shown in figure(61). However, the signal has also been significantly scaled up.

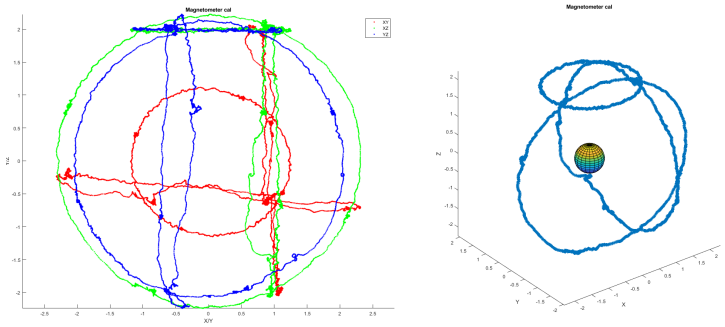


Figure 61: *2D and 3D representation of the calibrated magnetometer measurements with use of the simple method*

When using the insignificant method, the measurements have been centered, and the scale has been preserved, as illustrated in figure (62). Furthermore, by using the local geomagnetic field strength in Stavanger, which is approximately 0.509 Gauss, according to NOAA ([49]), the fit error can be calculated using equation (3.2.9.12). The resulting fit error for this signal after calibrate is 0.3253 percent.

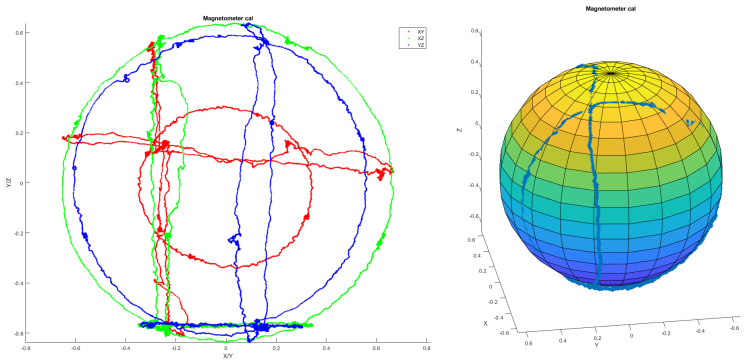


Figure 62: *2D and 3D representation of the calibrated magnetometer measurements with use of the insignificant method*

Figure (63) depicts the calibrated signal using the significant method. Evident by comparing the significant and the insignificant method that both methods center the measurements, the significant method also scales the data to make a more desirable fit for the geometric field. This fit is also reflected in the resulting fit error that is 0.1317 percent.

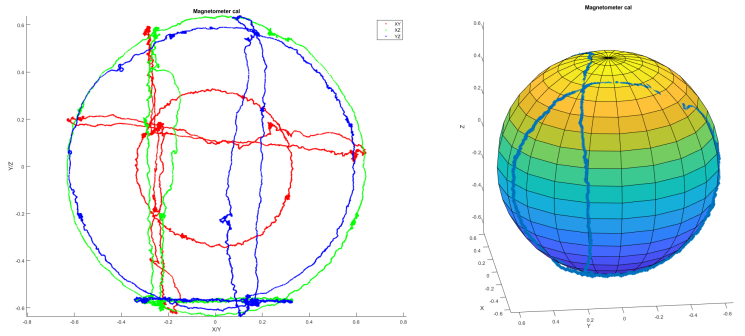


Figure 63: *2D and 3D representation of the calibrated magnetometer measurements with use of the insignificant method*

By comparing the results from the different methods, it is apparent that the simple method did not perform as well as the other two methods. The simple method can be used; however, scaling the data will also result in scaling any errors. By comparing the fit error, the significant method is the best; this is also visible by comparing the plots for the insignificant and the significant method.

Magnetometer calibration (significant soft iron distortion)

This test will test the methods using a sampled signal that has significant hard and soft iron distortion. The signal is generated while placed inside the BHA hence the distortion. The assembly is rotated so that the samples will cover the surface of the sphere defined by the geometric field. Figure (64) illustrates the raw signal before calibration, where it is evident that the signal is heavily distorted.

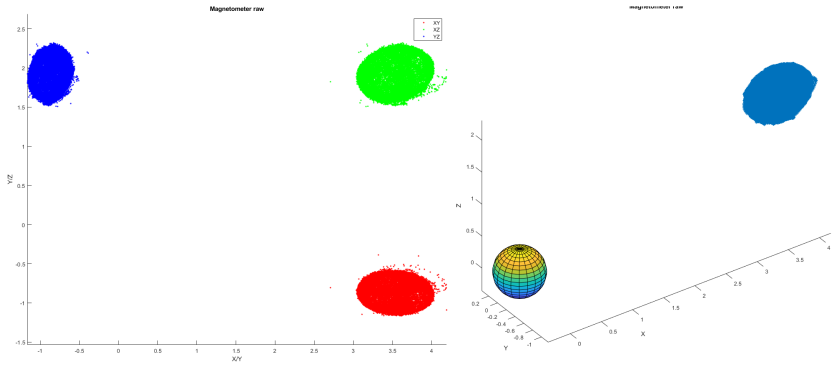


Figure 64: *2D and 3D representation of the raw magnetometer measurements with significant hard and soft iron distortion, measured while sensor is mounted inside the BHA*

After testing the simple method, it is apparent that there is too noticeable distortion in the signal for the simple method. It succeeds in gathering the signal; however, the samples are not center around 0,0, and no noticeable change in the soft iron distortion. Wich can be seen in figure (65), which illustrates the calibrated signal using the simple method.

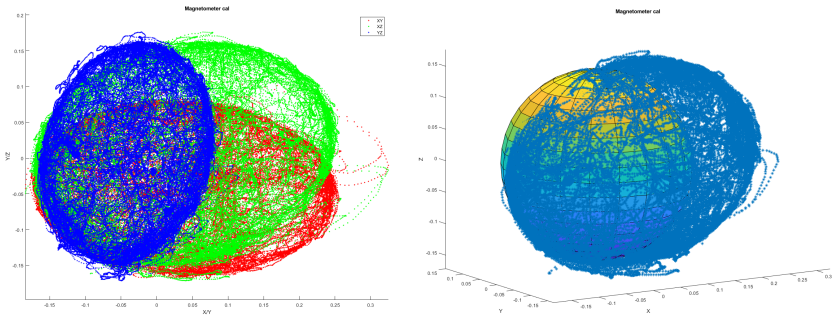


Figure 65: *2D and 3D representation of the calibrated magnetometer measurements with use of the simple method for a heavily distorted signal*

The insignificant method has centered the samples around 0,0, and have managed to compensate for the hard iron distortion, as depicted in figure(**??**). Still, the soft iron distortion is as prominent as with the use of the simple method. The fit error is now calculated to have a 10.63 percent error in the measurements.

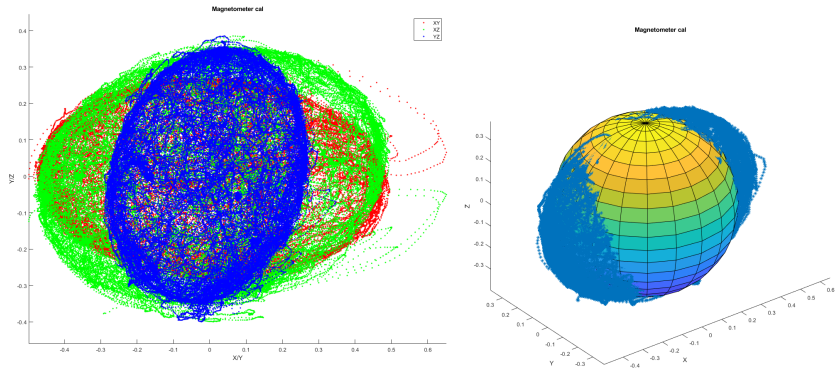


Figure 66: *2D and 3D representation of the calibrated magnetometer measurements with use of the insignificant method for a heavily distorted signal*

Figure (67) shows the calibrated measurements when using the significant method on the sampled signal. Furthermore, this method of compensation has removed a large portion of the hard and soft iron distortion. Even Though the signal still seems to have some soft iron distortion, the fit error is 0.1060 percent error in the measurements.

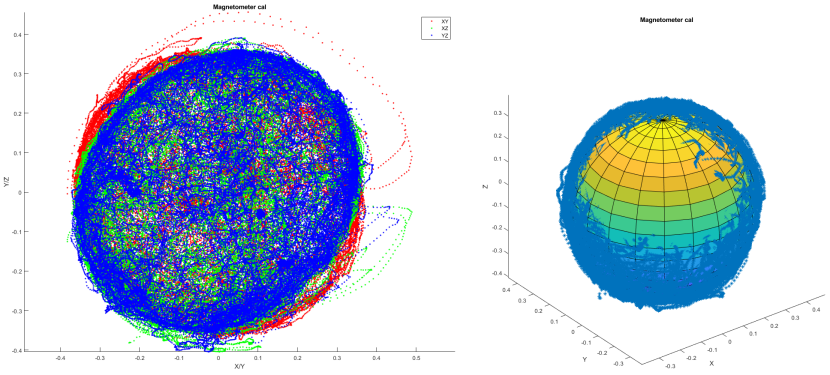


Figure 67: *2D and 3D representation of the calibrated magnetometer measurements with use of the significant method for a heavily distorted signal*

The simple method was adequate when the signal had an insignificant amount of soft iron distortion and low amounts of hard iron. However, it is not as effective as the two other methods., on the other side, it is less computational heavy. Nonetheless, the computational load associated with calculating the calibration parameters has no significance in the developed system. Moreover, when the signal is affected by significant distortion, the simple method is not sufficient. Thus it is unsuitable for the estimation system since the sensor card will be placed in an environment with much distortion.

The insignificant method is effective in correcting the hard iron distortion, even in an environment with a high degree of distortion. However, it is inefficient to remove the soft iron distortion; thus, it is not suitable for calibrating the signal measured in the BHA.

The objectively most suitable method for calibrating the magnetometer sensor placed in the BHA is the significant method. This method scored the lowest RMSE and visually can see the correction for the hard and soft iron distortion. However, it does not fully compensate for the soft iron distortion; thus, a fourth method may be looked into. Nevertheless, the significant method is sufficient enough to be used to calibrate the signal from the BHA.

4.4 Testing of the estimation software

For testing the estimation, a simulated sample of the angle and distance is used, since the rig could not be used. The sample of the distance sensor contains samples with an incrementing value of 0.2cm from zero to 100cm. For testing the kinematic model, The angle starts at 90 degrees, which according to the kinematic model in (28), will the drill bit be pointing along the z-axis global frame and the x-axis fo the sensitivity axis for the sensor points along the -z-axis global frame. Then incrementally move to an end angle with a fixed angular rate. For testing the estimates computed using the AHRS, the angle has been static; angle 70, 60, and 30 degrees have been used in the examples (where 90 is along the z-axis global frame). There was also conducted some tests with the dynamic orientation of the sensor going from 90 degrees to 80 degrees. The angles are measured with a protractor, and the sensor is moved to 90 degrees then to the desired angle for every test, to minimize error associated with sensor placement. RMSE will be used to measure the differences between the expected values and the observed estimate where the values would be the angle of the vector point in x,z coordinate. Thus the deviation error will be expressed in the difference in degrees.

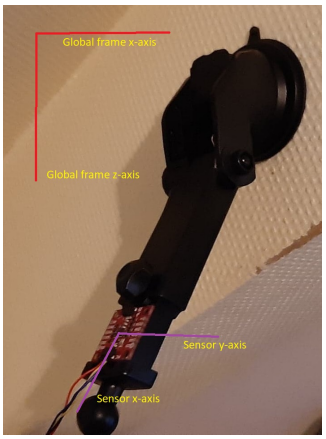


Figure 68: *Image illustrating sensor setup, and the orientation of the sensor axis*

The first experiment was with the estimates produced while using the AHRS with a static orientation. Figure(69) shows when the static orientation angle is 70 degrees, where the sensor signal was not calibrated. The average RMSE is calculated to 1.7609 degrees. Moreover, figure(70) displays the estimation for the same scenario but with calibrated sensor signals. The resulting RMSE is 0.1034 degrees.

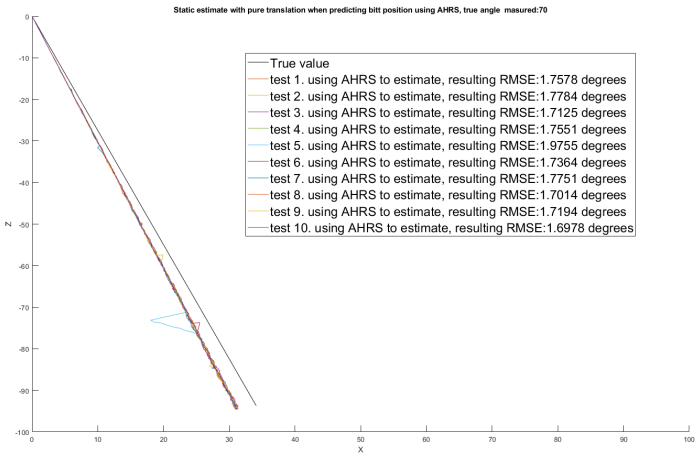


Figure 69: *Illustrating the estimated position of the drill bit based on the AHRS values, with uncalibrated sensors*

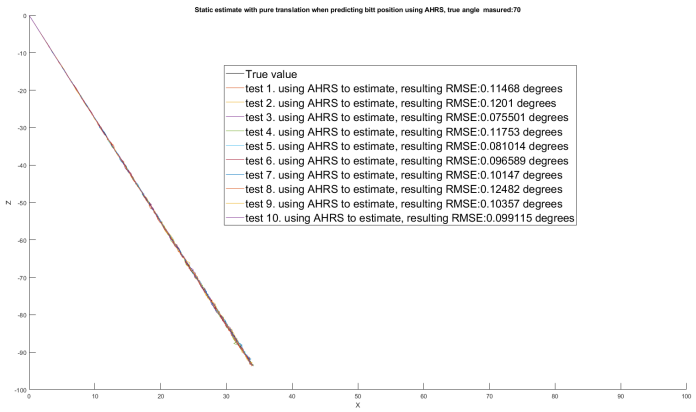


Figure 70: *Illustrating the estimated position of the drill bit based on the AHRS values, with calibrated sensors*

The test has been conducted for the angles 60 and 30 degrees in the same way. Thus generating table (5) expressing the average RMSE for both with the use of uncalibrated and calibrated signals. From the table, it is evident that the system works for static angles, where the best result is with calibrated sensors. Which has less than 0.17^0 degrees in deviation from the expected values.

Global sensor angle	RMES uncalibrated signal	RMES calibrated signal
70^o	1.7609^o	0.1034^o
60^o	1.8335^o	0.1684^o
30^o	1.3081^o	0.1576^o

Table 5: The RMSE of the estimated position of the drill bit with static angle.

Nex test is using a dynamic angle of rotation for the sensor when estimating the drill bitt position. However, it is not possible to get a proper fixed angular rate when manually rotating the sensor. Thus, it is difficult to predict the expected values for the sampled data. The aim was to have a fixed angular rate from 90 to 80 degrees over a 100cm displacement, as illustrated in figure (71). The RMSE was computed from the ideal path for his action. The average RMSE was calculated to be 10.5782 degrees of deviation from the ideal path. However, the ideal path assumes a fixed angular velocity, which the test did not achieve.

Moreover, the only information that can be taken from the test is evidence for the working principle of tracing the drill bit. This test should be done again with a fixed angular rate. Figure (71), also shows the results for the estimated drill bit position using the kinematic model: The deviation error from the ideal path was calculated to be 0.4580 degrees.

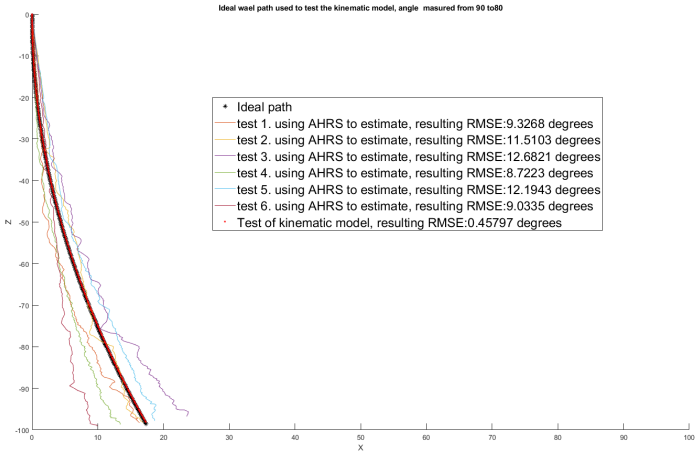


Figure 71: *2D and 3D representation of the calibrated magnetometer measurements with use of the significant method for a heavily distorted signal*

The same test was conducted for an angular displacement of 30 and 60 degrees with a fixed angular rate. Form figure (72) and (73) the performance for these to test was 1.1069 and 2.017, respectively. For the kinematic model, it is evident that with a higher angular displacement from 90 degrees, the estimation is less accurate.

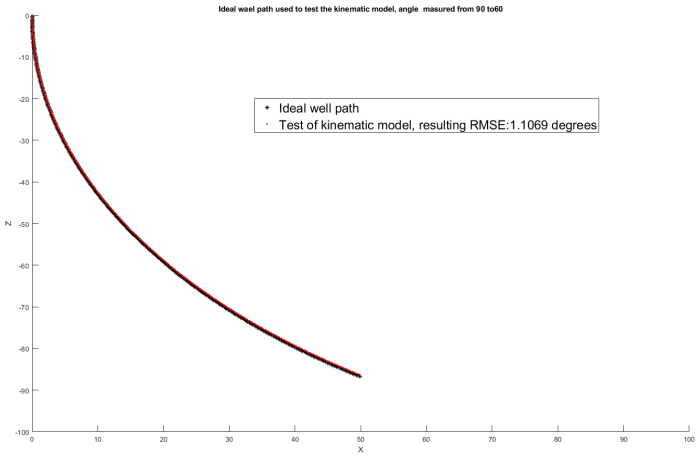


Figure 72: *2D and 3D representation of the calibrated magnetometer measurements with use of the significant method for a heavily distorted signal*

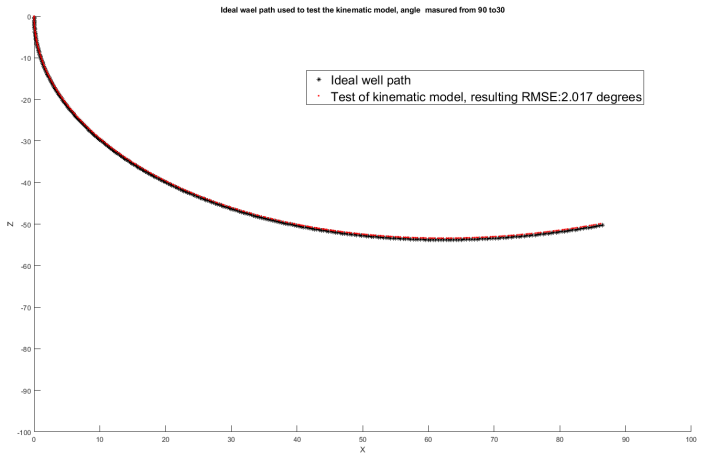


Figure 73: *2D and 3D representation of the calibrated magnetometer measurements with use of the significant method for a heavily distorted signal*

It is evident that the calibration has hade an impact on the estimation accuracy when comparing the RMSE, where the calibration on average shows an accuracy of one decimal higher than the uncalibrated. The test with a static angle shows that the estimation system manages to estimate the drill bit position within a 0.1 degree based on the position vector. However, the dynamic test was not successfully performed, and this was only a simulated test, which does not accurately represent the drilling behavior. A more accurate test should have been made; the ideal test that was planned for was to test the estimation under a drilling operation. Unfortunately, this was impossible because of technical issues with the drilling rig throughout the semester.

The same can be said about the test for the kinematic model. However, the working principle for the kinematic model has ben showed. Moreover, the assumptions and calculations of the kinematic equations seem to be valid based on the RMSE values. Therefore, it is a principle that can be explored more for further UiS Drillbotic's teams, as an alternative to track the drill bit position.

5 Conclusion

This thesis has reported the work done in relation to the design and implementation of new sensors and software for position tracing for the UiS Drillbotic's contribution to the 2020 Drillbotics competition. The project done in association with Drillbotic has proven to be challenging and educational, where practical, theoretical, and teamwork have been put to the test. The interdisciplinary teamwork for problem-solving and multidisciplinary work done within mechanical-, petroleum- and electrical engineering has been highly awarding. According to the objective of this thesis, the following work has been done:

Change from Arduino to PLC
The attempt to change the Arduinos that controls the actuators of the drilling ring with DAQ type PLCs have not been successful. The intended PLC was not suitable for controlling the actuators, and a replacement was not found in time to be able to incorporate it into the system. However, the old Arduino code was translated to be used within the central control system; however, it was not adequately tested.

Downhole sensor card
A downhole sensor card containing a MARG sensor and an MCU for data acquisition was designed; however, not built and implemented in the drilling rig. Parts of the sensor card was tested and worked as intended. However, the sensor card was not adequately designed before ordering, which made it hard later to have the sensor card to be implemented.

Sensor calibration
Calibration methods for the MARG sensor were developed and tested for an environment with a high degree of electromagnetic distortion. The results show a solution method to compute the calibration parameters for compensating for heavy distortion. A solution that should be sufficient for the control system to use.

Drill bit position tracking
A drill bit position tracking system was developed and implemented in the control system of the drilling rig. This system will estimate the position based on either, using the estimated orientation of a MARG senor computed by a HARS, or use a kinematic module for tracing the drill bit position. The system is developed to use the downhole sensor data for estimation. The system has not been adequately tested together with the rig, only used simulated downhole measurements and measurements from the MARG. Further testing is needed to confirm the performance of the system.

6 References

[1] Carsten Berge Guggedal, Magnus Steinstø, 2019, *"Control system architecture and API integration"*

[2] Jens Løkkevik, Erik Andreas Løken, 2019, *"Optimization of an Intelligent Autonomous Drilling Rig"*

[3] Sander Skjørestad, 2019, *"Directional drilling capabilities of a rebuilt and optimized autonomous laboratory scale drilling rig"*

[4] Drillbotics, *"About Drillbotics"*, https://drillbotics.com/about-drillbotics/

[5] Drillbotics, *"2020 Guidelines/Application Form – Revised"*, https://drillbotics.com/guidelines/

[6] DSATA, *"ABOUT DSATS"*, https://connect.spe.org/dsats/home

[7] Atmel, *"ATmega328P datasheet"*,2011, http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[8] Silicon Sensing, *"Find out more about our MEMS gyroscope technology"* , https://www.siliconsensing.com/technology/mems-gyroscopes/.

[9] Jeff Watson, *"MEMS Gyroscope Provides Precision Inertial Sensing in Harsh, High Temperature Environments"*, https://www.analog.com/en/technical-articles/mems-gyroscope-provides-precision-inertial-sensing.html.

[10] Dilip Raja, 2019, *"An Overview of Various MEMS Devices and their Applications"*, https://circuitdigest.com/tutorial/what-is-mems-various-mems-devices-and-applications.

[11] R. Yan, F. Zhang and H. Chen, "A MEMS-based Magnetometer Calibration Approach in AUV Navigation System," OCEANS 2019 - Marseille, Marseille, France, 2019, pp. 1-6, doi: 10.1109/OCEANSE.2019.8867368.

[12] Giulio D'Emilia , Antonella Gaspari , Fabrizio Mazzoleni , Emanuela Natale , and Alessandro Schiavi, "Calibration of tri-axial MEMS accelerometers in the low-frequency range – Part 1: comparison among methods ",2018,https://jsss.copernicus.org/articles/7/245/2018/jsss-7-245-2018.pdf.

[13] Electronics Tutorials, *"Hall Effect Sensor"*, https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html.

[14] Christopher Konvalin, 2009, *"Compensating for Tilt, Hard-Iron, and Soft-Iron Effects"*, https://www.fierceelectronics.com/components/compensating-for-tilt-hard-iron-and-soft-iron-effects.

[15] Wikipedia, *"I^2"*, https://en.wikipedia.org/wiki/I%C2%B2C.

[16] NXP Semiconductors (2014), *"I^2C-bus specification and user manual"*, https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[17] Wikipedia, *"Serial Peripheral Interface"*, https://en.wikipedia.org/wiki/I%C2%B2C.

[18] EDN, *"Design calculations for robust I2C communications", 2012*, https://www.edn.com/design-calculations-for-robust-i2c-communications/.

[19] Mark Hughes, *"I2C Design Mathematics: Capacitance and Resistance"*,2018 https://www.allaboutcircuits.com/technical-articles/i2c-design-mathematics-capacitance-and-resistance/

[20] Wikipedia, *"Serial Peripheral Interface"*, https://en.wikipedia.org/wiki/Serial_Peripheral_Interface.

[21] Steven Bible, *"Crystal Oscillator Basics and Crystal Selection for rfPIC™ and PICmicro® Devices"*, http://ww1.microchip.com/downloads/en/appnotes/00826a.pdf.

[22] Steve TLA Microsystems, *"Microcontroller crystal oscillators"*, http://www.tla.co.nz/xtal1.html.

[23] John Vince, *"Rotation Transforms for Computer Graphics"*, 2011, Springer, ISBN 978-0-85729-153-0, e-ISBN 978-0-85729-154-7.

[24] Jack B. Kuipers, 1999, *"Quaternions and rotation sequences"*, https://www.emis.de/proceedings/Varna/vol1/GEOM09.pdf.

[25] Wikipedia, *"Attitude and heading reference system"*, https://en.wikipedia.org/wiki/Attitude_and_heading_referenc_system.

[26] xsens, *"Attitude Heading Reference System (AHRS)"*, https://www.xsens.com/inertial-sensor-modules.

[27] Sebastian O.H. Madgwick, *"An efficient orientation filter for inertial and inertial/magnetic sensor arrays"*,2010, https://www.researchgate.net/publication/221775760_Estimation_of_IMU_and_MARG_orientation_using_a_gradient_descent_algorithm.

[28] Sebastian O.H. Madgwick, Andrew J.L. Harrison, Ravi Vaidyanathan *"Estimation of IMU and MARG orientation using a gradient descent algorithm"*,2011, https://ieeexplore.ieee.org/document/5975346.

[29] A. Cirillo, P. Cirillo, G. De Maria, C. Natale, S. Pirozzi, *"A comparison of multisensor attitude estimation algorithms"*,2010, https://www.researchgate.net/profile/Pasquale_Cirillo/publication/303738116_A_comparison_of_multisensor_attitude_estimation_algorithms/links/5750181208aeb753e7b4a0c0/A-comparison-of-multisensor-attitude-estimation-algorithms.pdf.

[30] Wikimedia Commons, uploaded 19.12.2006 *"SPI three slaves.svg"*,2011, https://commons.wikimedia.org/wiki/File:SPI_three_slaves.svg.

[31] Wikimedia Commons, uploaded 17.12.2006 *"I2C.svg"*,2011, https://commons.wikimedia.org/wiki/File:I2C.svg.

[32] TDK InvenSense, *"ICM-20948 datasheet"*, https://invensense.tdk.com/wp-content/uploads/2016/06/DS-000189-ICM-20948-v1.3.pdf?ref_disty=digikey

[33] MURATA, *"Crystal, 16 MHz, SMD, datasheet"*, http://www.farnell.com/datasheets/2631966.pdf?_ga=2.89803803.18697197.1593136653-1084935214.1585830096

[34] FTDI Chip, *"FT232R USB UART IC datasheet"*, https://docs.rs-online.com/f1f3/0900766b81451bdd.pdf

[35] Mark W. Spong, Seth Hutchinson, M. Vidyasagar, *"Robot modeling and control"*, 2006, Willy, ISBN-10: 0-471-64990-2, ISBN-13: 978-0-471-664990-8 https://docs.rs-online.com/f1f3/0900766b81451bdd.pdf

[36] Microchip Technology Inc, *"PICmicro MID-RANGE MCU FAMILY"*, 2006, Willy, ISBN-10: 0-471-64990-2, ISBN-13: 978-0-471-664990-8 http://ww1.microchip.com/downloads/en/devicedoc/31018a.pdf

[37] Measurement computing, *"USB-3114 datasheet"*, https://www.mccdaq.com/pdfs/manuals/USB-3114.pdf

[38] Measurement computing, *"USB-1608G datasheet"*, https://www.mccdaq.com/PDFs/manuals/USB-1608GX.pdf

[39] Geckodrive motor controls, *"G250X MANUAL STEP MOTOR DRIVE"*, https://www.geckodrive.com/amfile/file/download/file_id/14/product_id/15/

[40] MoTech Motor, *"Nema 23 Stepper Motor"*, http://motechmotor.com/productDetail-0105-30.html

[41] Wikipedia, *"gRPC"*, https://en.wikipedia.org/wiki/GRPC

[42] gRPC, *"qRPC FAQ"*, https://grpc.io/faq/

[43] Gw INSTEK, *"Digital Storage Oscilloscope GDS-2000A Series user manual"*, https://www.gwinstek.com/en-global/products/downloadSeriesDownNew/904/94

[44] Burkert, *"datasheet for burkert 8605"*, https://www.burkert.com/en/Media/plm/MAN/MA/MA8605-Manual-EU-EN.pdf?id=MAN0000000000000001000329112ENC

[45] Magomed Khadisov, Andreas Jakobsen, Håkon Hagen, *"Laboratory drilling rig construction, testing and modeling for optimization and problem management"*, 2018

[46] Robert A. Adams, Christopher Essex, *"Calculus", 8th ed.*, 2013, Pearson, ISBN 978-0-321-78107-9

[47] Ronald E. Walpole, Raymond H. Myers, Sharon L. Myers, Keying Ye, *"Probability and Statistics for Engineers and Scientists", 9th ed.*, 2016, Pearson, ISBN10 1292161361

[48] Sergios Theodoridis, *"Machine Learning: A Bayesian and Optimization Perspective"*, 2015, Academic press, ISBN 978-0-12-801522-3

[49] National Centers for Environmental Information, *"Magnetic Field Estimated Values"*, https://www.ngdc.noaa.gov/geomag/calculators/magcalc.shtml?#igrfwmm