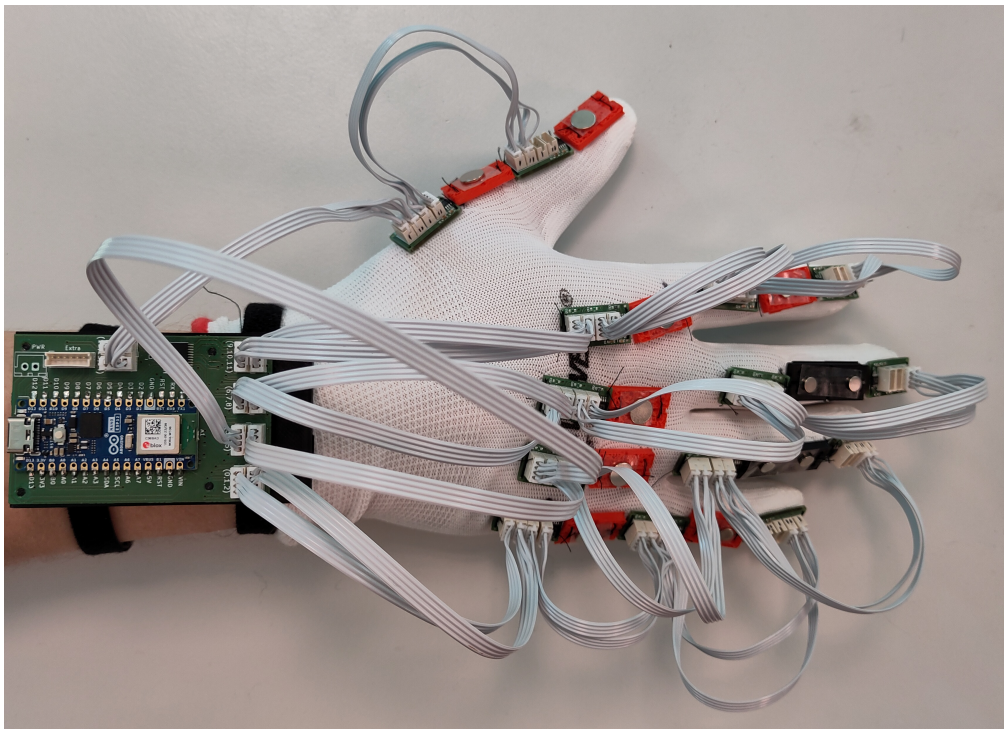
CHALMERS



Wearable sensor based system for measuring finger motions

Using Hall effect sensors to collect motion data in real time

Bachelor's thesis in Systems and Control

KARL AHLGREN, DANIEL ANDERSSON, JOEL EDLÉN, OLLE FRISK, MAX HARALDSSON, JAKOB PEDERSEN AUGSBURG

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Bachelor's thesis 2024

Wearable sensor based system for measuring finger motions

Using Hall effect sensors to collect motion data in real time

KARL AHLGREN, DANIEL ANDERSSON, JOEL EDLÉN, OLLE FRISK, MAX HARALDSSON, JAKOB PEDERSEN AUGSBURG

CHALMERS

Department of Electrical Engineering
Division of Systems and Control
EENX16-24-14
Chalmers University of Technology
Gothenburg, Sweden 2024

Wearable sensor based system for measuring finger motions
Using Hall effect sensors to collect motion data in real time
KARL AHLGREN, DANIEL ANDERSSON, JOEL EDLÉN, OLLE FRISK, MAX HARALDSSON, JAKOB PEDERSEN AUGSBURG

Supervisor: Rikard Karlsson, Department of Electrical Engineering
Examiner: Emmanuel Dean, Department of Electrical Engineering

Bachelor's Thesis 2024
Department of Electrical Engineering
Division of Systems and control
EENX16-24-14
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Image of the projects final prototype.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Wearable sensor based system for measuring finger motions
Using Hall effect sensors to collect motion data in real time
KARL AHLGREN, DANIEL ANDERSSON, JOEL EDLÉN, OLLE FRISK, MAX HARALDSSON, JAKOB PEDERSEN AUGSBURG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The hand is humanity's most used tool and an injury to this body part would be detrimental to both a persons mental and physical well being. If an injury has occurred an important step on the road to recovery is rehabilitation. With the advancements made in technology it is now possible to track the movements of the hand and fingers using different systems. The objective of this thesis was to investigate whether Hall effect sensors could be a suitable choice for such a system. This was done by creating a prototype.

The final glove is made out of 14 magnets and 3D Hall effect sensors for measuring the angles of the finger's joints. These send their data to an Arduino micro controller that packages the data and sends it to a neural network. The purpose of the network is to map the sensor data to joint angles collected from a stereo camera hand tracking system to then be able to predict angles using only the sensor data. The predicted angles are then visualized with a hand model in Unity.

The project shows promising results for the use of Hall effect sensors in a motion tracking device, as they produce unique data for different joint angles. However there is much room for improvement, especially when it comes to the camera setup and hand tracking software but also for the optimization of the neural network.

This thesis concludes that Hall effect sensors in combination with a neural network and high quality camera data is a viable choice for a finger motion tracking device.

Keywords:Hall effect, sensor, system, finger, motion tracking, magnets, neural network, camera, angles.

Sammandrag

Handen är människans mest använda verktyg och en skada på denna kroppsdel skulle drabba både en persons mentala och fysiska hälsa. Om en skada har inträffat är en viktig del av återhämtningsprocessen rehabilitering. Med de framsteg som gjorts inom tekniken är det nu möjligt att spåra handens och fingrarnas rörelser med hjälp av olika system. Syftet med denna rapport var att undersöka om Hall-effektsensorer kunde vara ett lämpligt val för ett sådant system. Detta gjordes genom att skapa en prototyp.

Den slutliga handsken är uppbyggd av 14 magneter och 3D Hall-effektsensorer för att mäta vinklarna hos fingrarnas leder. Dessa skickar sen data till en Arduino-mikrokontroller som paketerar datan och skickar den vidare till ett neuralt nätverk. Syftet med nätverket är att koppla sensordatan till ledernas vinklar som samlats in från ett handspårningssystem med kameror för att sedan kunna förutsäga vinklar med endast sensordatan. De förutsagda vinklarna visualiseras sedan med en handmodell i Unity.

Projektet visar lovande resultat för användningen av Hall-effektsensorer i ett rörelsespårningssystem, då sensorerna producerar unik data för olika vinklar. Det finns dock mycket utrymme för förbättring, särskilt när det gäller kamerakonfigurationen och handspårningsprogramvaran men också för optimering av det neurala nätverket.

Denna rapport drar slutsatsen att ett system för fingerrörelsespårning är möjligt med hjälp av Hall-effektsensorer i kombination med ett neuralt nätverk och data av hög kvalitet från kameror.

Nyckelord:Hall effekt, sensor, system, finger, rörelsespårning, magneter, neuralt nätverk, kamera, vinklar.

Acknowledgements

We, the members of project group EENX16-24-14, would like to especially thank our supervisor Rikard Karlsson and examiner Emmanuel Dean for providing continuous support and encouragement without which this project would not have been possible. Furthermore, we want to thank earlier years project groups for their work, on which we stand. Also, we thank CASE-Lab for allowing us to use their workspace during this project.

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CMC	Carpometacarpal
CS	Chip Select
DIP	Distal interphalangeal
DOF	Degree Of Freedom
GPIO	General Purpose Input Output
IMU	Inertial Measurement Unit
IP	Interphalangeal
MC	Microcontroller
MCP	Metacarpolphalangeal
MFD	Magnetic Flux Density
MSE	Mean Squared Error
PCB	Printed Circuit Board
PIP	Proximal Interphalangeal
ROS	Robotic Operating System
SPI	Serial Peripheral Interface
SPP	Serial Port Profile
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
VR	Virtual Reality

Contents

List of Figures

List of Tables

1
Introduction

Biomechanics is a field that has been around for over two millennia. This is closely related to the exploration of mechanical systems as a method to comprehend the complexity of nature. 60 years ago biomechanics started to blossom due to its applications in orthopaedics[1].

Since then, measuring biomechanical signals has evolved, with applications ranging from healthcare to military uses. Because of this, there is a need to reliably measure these signals in a non-restricted environment[2]. Recent developments in wearable technology and notably the increased performance and availability of microcontrollers (MCs) have made it easier than ever to collect data in a mobile setting[3]. This has enabled improved methods to detect patterns or control devices amongst others[4]. Biomechanics is an important field with a wide range of applications but more research is needed to commercialize and improve it.

1.1 Background

The human hand is one of the most important parts of the human body. A decrease in the hand's functionality greatly impacts a person's quality of life by complicating basic daily activities[5]. With a rising population and a higher life expectancy, the amount of people suffering from disabilities is increasing which puts pressure on healthcare systems[6]. This in turn has sparked an increase in demand for hand rehabilitation devices to improve the therapy procedure[7].

Currently, the rehabilitation progress is measured through scales given by physical therapists which takes time and is not optimal[7]. By integrating a finger motion device that can visualize the movement of a patient's fingers, therapists could more efficiently evaluate the functionality of a patient's hand and reduce their already high workload[8].

Tracking the motion of the fingers has been a topic expanded upon in over three decades[9]. With the ever-increasing advancements in human technology, several solutions for this problem have been explored. The solutions are often categorized into vision-based and hand-motion-based. The vision-based solution utilizes cameras to track the motion of the hand, which in recent years has received attention due to advancements in imaging technology and imaging processing[10]. The problem with vision-based solutions is their limitations of portability and occlusion from

hand-held objects blocking the view of the hand[11]. Moreover, hand motion devices provide a portable solution that enables more applications in everyday life. Previous works have explored IMU's (Inertial Measurement Unit) and Flex sensors' ability to measure the angle and position of the fingers[12][13][14]. The aforementioned thesis had problems with sensor reliability, in particular, tracking the thumb. On the other hand, a solution with Hall-effect sensors would potentially provide a contactless solution which could reduce the probability of sensor unreliability and improve the performance. Such a device will be attempted to be realized and analyzed in this project.

1.2 Purpose

The project's purpose has been to analyze if Hall-effect sensors are a suitable choice for measuring finger movements by collecting reliable and accurate data to visualize the fingers' movements in real-time. Especially to try and be able to measure the complex motions of the thumb.

1.3 Delimitations

The testing of the device has solely been performed in a lab environment to reduce disturbances that could potentially have affected the data. Due to the purpose of only measuring the motion of the fingers, movements of the shoulder, elbow, and wrist have not been measured.

Another limitation of the project regards the hand model in Unity, as an already existing model: XR Hands will be used. It was deemed too time consuming and not part of the goal of the project to create one ourselves even if this would have been beneficial as discussed in section 5.6.

1.4 Problem specification

The aim was to produce a device capable of measuring the movements of fingers and thumb in real-time. These movements were then mapped and displayed as a visual model in Unity. Since the chosen method of motion detection is based on sensing magnetic field densities from several permanent magnets in close proximity to the sensor, there is a substantial risk of unintended interference. Furthermore, the aim was to track finger movements specifically. Therefore, the device should not constrain any motions of the fingers while being regarded as a portable device. A requirement on the device were a total weight of under 500 grams as to not prohibit any movements.

1.5 Research questions

With a brief understanding of what the purpose of this project was, and which problems the project has faced, the following questions have been answered:

- Will Hall-effect sensors be able to measure the fingers' 21 Degrees of Freedom (DOF)?
- Is it possible for the sensors and magnets to be placed in a certain order to reduce magnetic interference while keeping the functionality for the intended purpose?
- What are the required pre- and post-processing requirements in order to acquire the fingers' motion in real-time?

1.6 Previous work

Flex sensors, in combination with potentiometers, have been used and tested in simulating hand movements in a previous bachelor's thesis[12]. A flex sensor alters its electrical resistance depending on how much the sensor is bent. Flex sensors are frequently based on a resistive element film made of a polymer or of a carbon-based element[15]. In the aforementioned thesis, a test of flex sensors was conducted with a sample size of 10 samples, which resulted in a mean accuracy difference between the calculated and real angle, with a median filter, of $\pm 5.7°$[12].

An IMU is a sensor that has also been used in a previous bachelor's thesis [13]. In this thesis, a six DOF IMU was used. This type of sensor led to complications due to gyros' inherent susceptibility to drift issues[16]. The option of nine DOF IMUs is possible but the magnetometers could disturb other sensors since they detect magnetic fields. Nine DOF IMUs are often expensive compared to six DOF IMUs or Hall-effect sensors.

Tracking finger movements in real-time with cameras has been used, showing good results with a performance rate of 95%[17]. The visual system uses region-based tracking [18]. The problem with this system is the ability to relocate the camera configuration easily, and the requirement for a correct illuminated and spacious setting.

2
Theory

The following chapter presents the underlying knowledge to understand the process and the decisions made in this project. A basic understanding of the hand's anatomy is needed to understand the placement of the sensors and what they are measuring. The Hall-effect and the communication from the sensor to the MC will be covered in this chapter. We will explain how the sensors can generate a reading and how this reading is communicated from the sensor to the computer.

2.1 The structure of the hand

There are 27 bones and 24 muscles that control the different motions of the hand which include 27 DOF[19][20]. The forearm has two major bones, the *ulna* and *radius*. These are connected to three of the hand carpus bones where the most radial is called the *scaphoid*. This connects to the *trapezoid* which makes up the base of the thumb. There are a total of eight carpus bones assorted in two rows, the proximal and the distal row. The bones in the distal row connect to the *metacarpal bone* of the fingers. The *metacarpal bones* are connected to *phalanxes* which together make up the fingers. The *phalanxes* are made up of three bones in the fingers and two in the thumb. The *phalanx* bone closest to the *metacarpal* is called the *proximal phalanx*, the middle bone is the *middle phalanx*, and the one furthest from the metacarpal is the *distal phalanx*. For the thumb, the *metacarpal* bone is connected to the *proximal phalanx* which in turn is connected to the *distal phalanx*. The hand's complex movements are made possible because of the different muscles, some originating from the elbow and forearm, and some muscles located in the hand itself. The fine motor skills of the hand come from the muscles located in between the fingers whereas larger movements originate from the forearm and elbow[21][19].

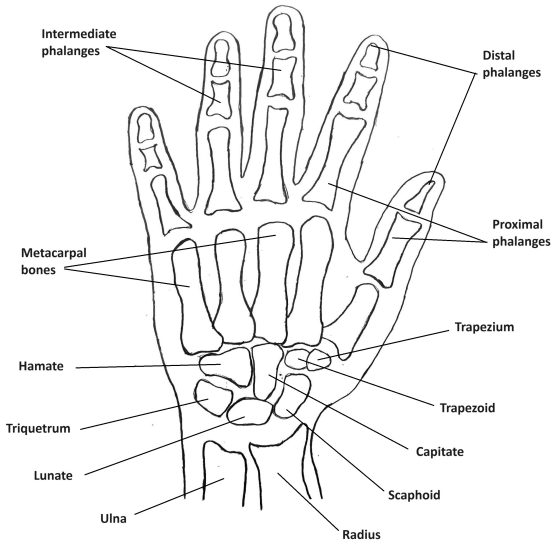


Figure 2.1: The different bones of the hand.

2.1.1 The joints of the hand

In the fingers, there are different types of joints. These make it possible for the fingers to bend. The joints between the *carpal bones* and the *metacarpal bones* are called the *carpometacarpal*(CMC) joint. Between the *metacarpal bone* and the *proximal phalanxes*, there is a joint called the *metacarpophalangeal*(MCP) joint. The remaining joints of the fingers are called *proximal interphalangeal*(PIP) and *distal interphalangeal* (DIP) joints. Since the thumb doesn't have a *middle phalanx* it only has a single *interphalangeal*(IP) joint[22].

Out of the hand's 27 DOF, the fingers make up 21[19]. The index, middle, ring, and little finger have four DOF each, where two of these come from the PIP and DIP joint which have one DOF each, and the two remaining come from the MCP joint. Since the PIP and DIP joints only have one DOF, they are only able to perform flexion or extension motion. Whereas the MCP joint with its two DOF also performs abduction and adduction. The thumb on the other hand has a more complex anatomy with five DOF.

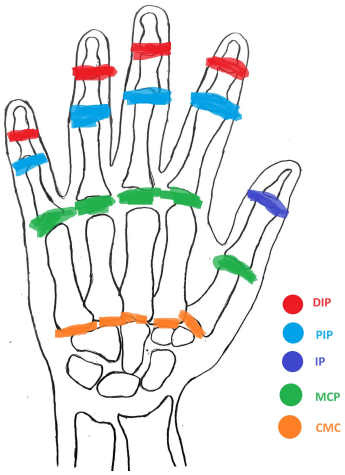


Figure 2.2: The different joints of the hand.

2.1.2 Anatomy of the thumb

The increased motion of the thumb compared to the other fingers is determined by the CMC joint. The CMC joint is a saddle joint, which gives the thumb the ability to move in what can be described as flexion, extension, abduction, and adduction. The saddle joint can be seen as two hinges, one which allows only flexion and extension and the other abduction and adduction. When working together they create two axes of rotation[23][19].

The thumb CMC joint has two DOF, which are defined by the motions. The most important movement of the thumb is the opposition, which refers to touching the other fingertips with the thumb. This movement can be done with both precision and power and is one of the most useful motions for everyday activities[21].

2.1.3 Movements of the hand

The hand's complex movements can be explained by its many muscles. The fingers' extending motions are possible thanks to tendons running in the palm of the fingers, and likewise, the flexion is made possible by tendons running from the back of the hand to the fingertips.

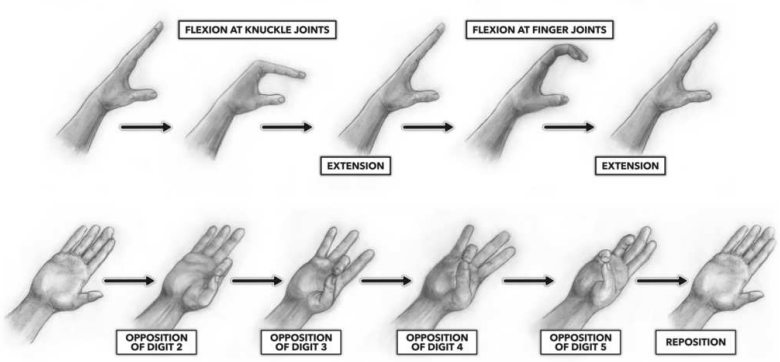


Figure 2.3: Different movements of the fingers[24]. Reprinted with permission.

Between the fingers *metacarpal bones* there are short muscles, these make the spreading of the fingers possible, or also known as adduction and abduction[25].

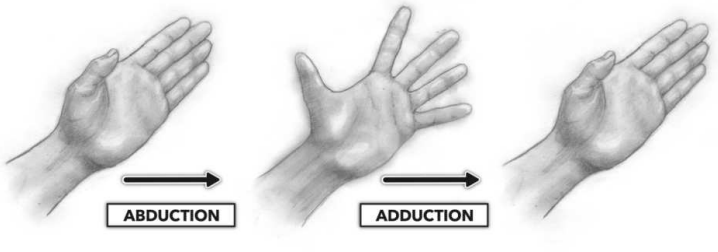


Figure 2.4: The fingers abduction and adduction[24]. Reprinted with permission.

2.2 Hall-effect

The Hall-effect, named after Edwin Hall in 1879, is what the Hall-effect sensors are named after and are integral to the project. The theory relevant to the project can be divided into theoretical and applied parts, with a focus on the usability of sensors[26].

2.2.1 Theoretical view

The Hall-effect is a result of the Lorentz force that applies a force to electrons traveling through a conductor which is perpendicular to the direction of both the

current flow and the applied magnetic field. The path of the traveling electrons through the conducting material is altered by a force equal to the Lorentz force equation[27].

$$\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \tag{2.1}$$

Where $\mathbf{F}$ is the force acting upon each particle traveling through the magnetic field, $\mathbf{E}$ is the electric field, $\mathbf{v}$ is the velocity of the charge, $\mathbf{B}$ is the magnetic field, and q is the charge of the particle. As the particles move through the transducer, the Lorentz force changes the particles' travel path which results in the particles concentrating on the side that is perpendicular to both the electric field and the applied magnetic field as derived using the right-hand rule, leading to voltage building up over the width of the transducer[28]. This voltage, referred to as the Hall voltage, can be measured with probes and is a linear function of the velocity of the charge carrier through the transducer, the strength of the applied magnetic field in the relevant axis, and the width of the transducer. The measured voltage over the transducer is given by the following equation:

$$V_H = -v_x B_z w \tag{2.2}$$

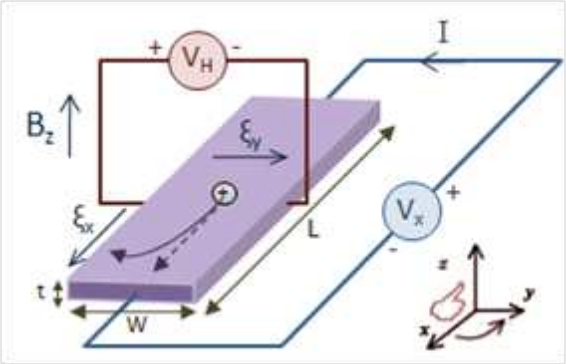


Figure 2.5: Visualisation of the Hall-effect on a particle traveling in the direction of the current, used according to public domain[29].

Where V_H is the Hall voltage, w is the width between two sides of the transducer, v_x is the drift speed of the charged particle in the x direction, and $\mathbf{B}_z$ is the vector component of $\mathbf{B}$ in the positive z direction. It is with the Hall voltage that Hall-effect sensors can be used to measure the distance to a magnet, as long as the strength of a magnet's magnetic field is known.

2.2.2 Hall-effect applied in sensor usage

The principle of the Hall-effect sensor is an integrated circuit built upon the Hall voltage measured over a semiconductor transducer. This circuit transduces magnetic fields to electrical signals using the Hall-effect[30][28]. This measured voltage decays over distance as the magnetic flux density (MFD) lessens the further away the sensor

goes from the magnet[31]. Using this fact, a distance can be calculated by solving for $\mathbf{B}_z$ in equation 2.2 above. Three-dimensional Hall-effect sensors work in the very same way as one-dimensional, but have three sensing elements in conjunction with the three vector components of $\mathbf{B}$[31].

2.2.3 Hall-effect sensor

There exist different kinds of Hall-effect sensors. There are linear Hall-effect sensors, both 2D and 3D, that give either an analog or digital signal proportional to the MFD to provide position or angular movement[30]. A Hall-effect sensor can also be used as a switch that notices when the MFD crosses a set threshold[30]. In this project, the focus is on linear Hall-effect sensors for the ability to detect angle and position.

Comparing the flex sensor combination with a Hall-effect sensor, a 3D linear Hall-effect sensor can have an accuracy of down to $\pm 2.6°$ compared to the $\pm 5.7°$ of the flex sensors[12][30]. Using Hall-effect sensors compared to the combination of flex sensors and potentiometers offers the benefit of reducing the problem of wear and tear of the sensor. The reason for this is that there is no mechanical part in the Hall-effect sensor that can get worn down.

A potential problem with using Hall-effect sensors is that different magnetic fields, if they are strong enough, can interfere and alter the reading. Thus, tests need to be conducted to see if interference between magnets close to each other occurs when multiple sensors are placed on the hand. Then, the result of these tests will determine the design of the hand motion device. Another challenge when designing the device is that the MFD falls off quickly, in the span of 10 mm around 80% of the MFD is lost. The loss of density in correlation with distance puts a limit on how far the magnets can move from the sensor and still give good readings.

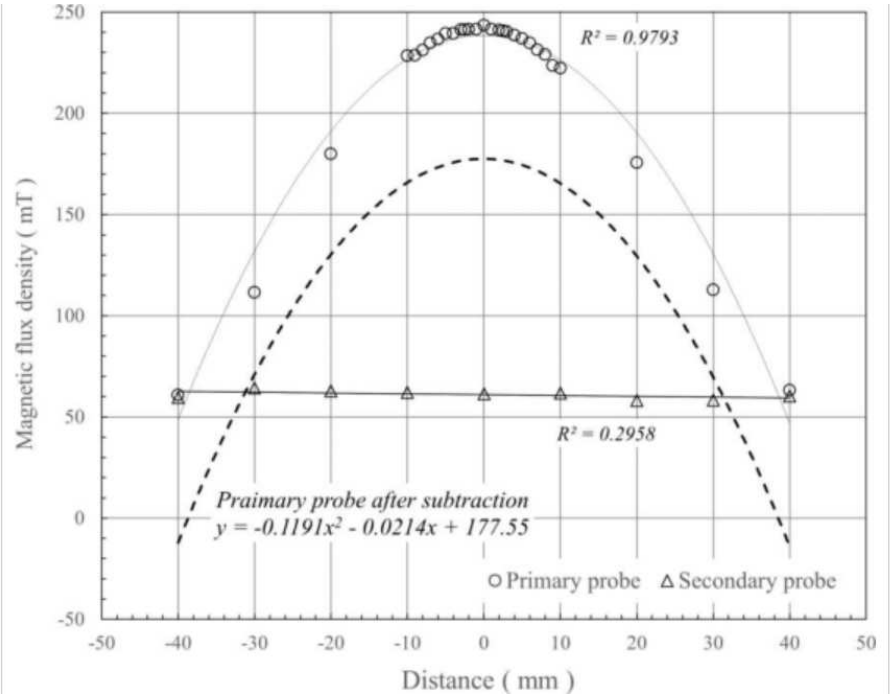


Figure 2.6: Diagram showing the MFD of the primary probe[32].

2.3 Data communication

The process of transferring the sensor data to a computer for filtering and graphical representation requires a conversion from "byte"-sized packets from each sensor on the data-bus to a format that can be sent to and interpreted by a computer. This also has to be done in consideration of our ergonomic requirements, as the physical hardware required for this purpose can vary depending on a chosen method.

2.3.1 Sensor-Microcontroller communication

The Serial Peripheral Interface (SPI) as specified by the chosen sensor is a four-wire bus that supports full duplex serial communication[33].
For a typical configuration, there is no theoretical upper limit on the number of connected sub-devices (referring to the sensors connected to the SPI-bus, from now on referred to as *subs* in this context). However, each *sub* has to be selected by a chip-select (CS) pin directly from the main processing unit (the MC connected to the SPI-bus, referred to as *main* in this context). This could make the number of available GPIO-pins (General Purpose Input Output) on the *main* a bottleneck in the design, since the required output pins scales linearly with the number of connected *subs*[34]. This is shown by the CS connections in the upper circuit diagram in figure 3.1.

A method which would enable a larger number of *subs* being connected to the bus is by controlling the CS signals using a decoder. A decoder will interpret a parallel input signal as a binary numeral and set the corresponding output port to either

high or low accordingly while keeping all other ports in the opposite state.
As in the lower example in Figure 2.7 below, two inputs enable us to control four CS signals. This means that for each available GPIO-pin on the *main*, the possible number of connected sensors will increase by a factor of two.

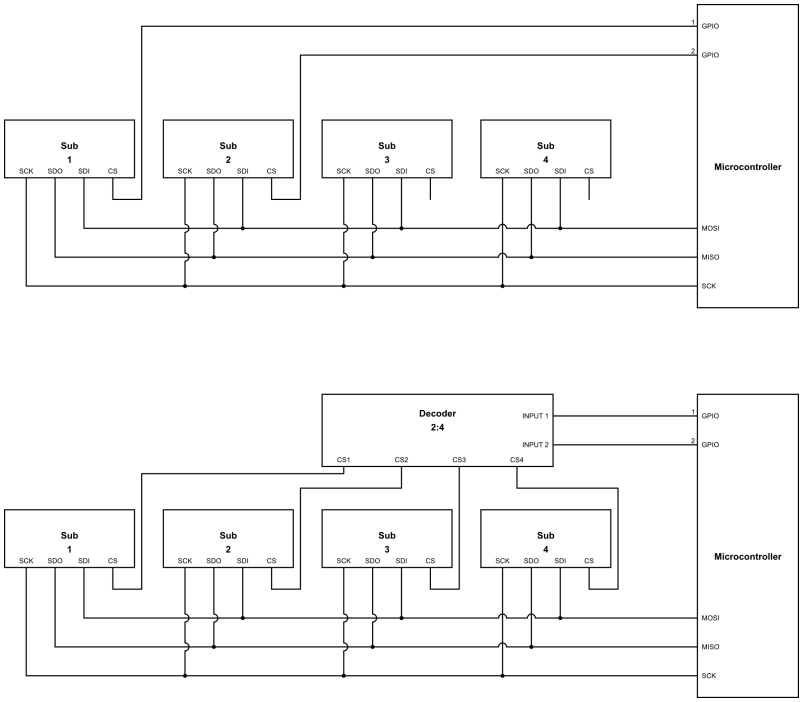


Figure 2.7: Comparison of chip selects with and without decoder.

2.3.2 Microcontroller-Computer communication

The primary way of communication between a microcontroller and computer is via a serial port, or a serial communication protocol via Universal Serial Bus (USB)[35]. In both of these cases, the protocol being used is the Universal Asynchronous Reciever-Transmitter protocol (UART).

2.4 Photogrammetry

Photogrammetry is described as the procedure of obtaining reliable information of physical objects and environment through the use of measuring and interpreting images[36]. This means that by the use of multiple cameras, and stereo vision, it is possible to estimate information about points of interest, such as X, Y, and Z coordinates. By utilizing a stereo setup, and known parameters such as the distance between the cameras b and the focal length f of the cameras, one can calculate a real-world point P, where the Z component is of special interest as it is the depth of the point. This is calculated through the disparity, which is the distance between the coordinates of the same scene point from both cameras.

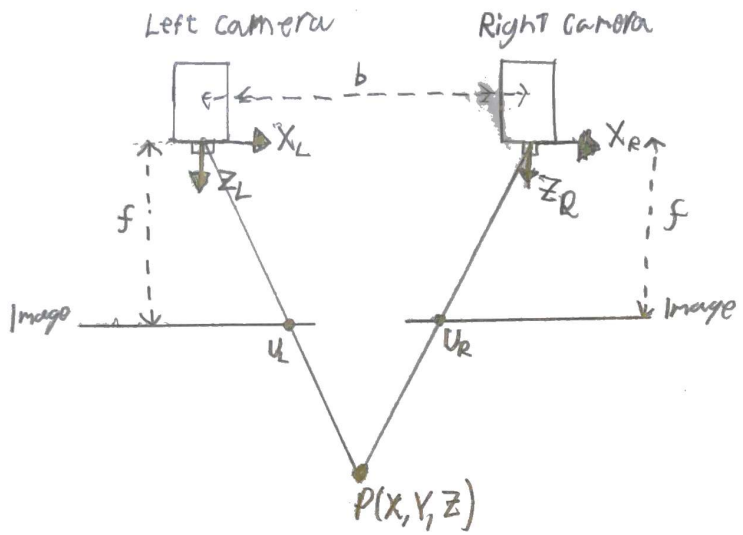


Figure 2.8: A simple drawing of a point being projected onto both cameras images.

$$u_L = \frac{f \cdot X}{Z} \tag{2.3}$$

$$u_R = \frac{f \cdot (X - b)}{Z} \tag{2.4}$$

Here, X and Z are the coordinates of P. Also, u_L and u_r is the X position of the point projected to the image. The theory behind these equations is further explained here[37].

$$\text{disparity} = u_L - u_R = \frac{f \cdot b}{Z} \tag{2.5}$$

$$\text{depth} = \frac{f \cdot b}{\text{disparity}} \tag{2.6}$$

Important to note is that when the depth increases the disparity becomes smaller, which can lead to an increased error for the estimation. The reason is that it is harder to pick up small differences in the disparity, and these small differences affect the depth greatly at large distances.

2.5 Neural network

A neural network can be thought of as a brain, where instead of neurons there are nodes that are intricately connected together to perform a task[38]. These tasks could be anything from identifying images to estimating relationships between variables. Depending on the task, the architecture of the neural network varies in order to perform as best as possible on the designated task. This includes hyperparameters which determine how the network is learning, and the structure of the network itself. One of these structures is a Feed-forward neural network. Feed-forward means that

the information only moves forward from the input nodes to the output nodes[39].

A simple network architecture typically consists of the following hyperparameters: An input layer, hidden layers, an output layer, activation functions, loss functions, and optimizers. These play an important role in predicting values based on complex data:

- The input layer is the number of inputs the network gets. These vary depending on the data set that the network will train on.

- The activation function lies between each layer and treats the data so that when passed between nodes, the output will not become a simple linear function but a more complex non-linear one. This way the network can take on more complicated tasks and map more complex data[40].

- The hidden layers lie between each activation function. In these layers are a number of nodes that each have a weight connected to it. Depending on what kind of data will be processed there can be any number of hidden layers. When the data is passed through the network each weight is considered and the output differs[41].

- The output layer is the final layer before the prediction. The size of this layer decides how many predictions will be made[41].

- Every network has a loss function and an optimizer. The network calculates a loss for every iteration it trains. When reaching the output layer an error is calculated between the prediction and actual value. Based on the number of training iterations the output will back-propagate to the hidden layers where a new loss is calculated. Depending on the loss, the weights will be optimized via the optimizer. The choice of loss function and optimizer depends on what kind of data the network is trained on and what kind of prediction it should output[41].

3
Method

In this part, the progression of the project will be described, and design solutions will be explained.

3.1 Device hardware

In this chapter, the choices of components and the reasoning behind these will be explored.

3.1.1 System overview

The final prototype consists of 14 magnet-sensor pairs, which all measure the MFD in three axes near the joints of the fingers. A reading is made over a common SPI bus from each sensor in sequence using the decoder for a rolling chip select. The most recent information is collected and stored by the MC as a complete set of measurements from each sensor. This is then sent as a packet of data via USB to the connected computer upon request.

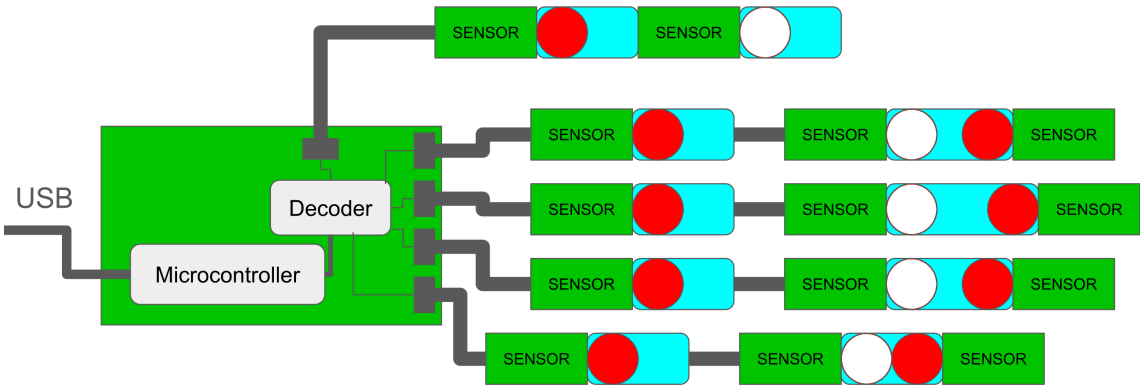


Figure 3.1: Overview of sensor setup. The position of the magnets are marked in red and white, where the whites have flipped polarity.

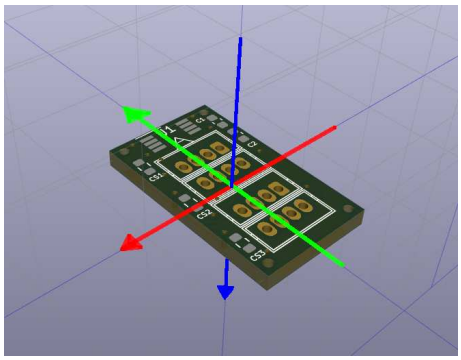


Figure 3.2: The axes of measurement. Red, green, and blue correspond to X, Y, and Z respectively. All sensors are placed such that the Y-axis aligns with the finger.

3.1.2 Sensors and sensor testing

As stated earlier in this thesis, there is a multitude of different Hall Effect sensors available on the market. Following the requirements set for this project, a sensor that could detect MFD changes in three directions was required, as this is necessary to accurately decipher the movements of the fingers. Furthermore, a sensor with high sensitivity was required as the sensor needed to detect even small changes in the angles of the fingers. Also, the sensor was required to have the correct bus interface, in this case, SPI. With all of these requirements in mind, the sensor TMAG5170 from Texas Instruments was selected[33].

The TMAG5170 is a "High-precision, linear 3D Hall-effect sensor with SPI bus interface", according to the manufacturer. It has a low linear maximum total error, as well as a low sensitivity temperature drift. Furthermore, it has a built-in temperature sensor, which helps when compensating for said drift.

3.1.3 Magnets and magnet testing

As the TMAG5170 measures the MFD, there is a need for accompanying magnets. In this case, small but strong magnets were required, which led to neodymium magnets. An assortment of different sizes and strengths were acquired and later tested before the assembly of a prototype.

In order to test the individual magnets that had been acquired for the project, two different tests had to be designed and utilized. The first test was to approximate at what distance the magnets should be in relation to the sensors, and see what settings would be best to use on the sensors in order to have the highest level of accuracy in measurements. This test was conducted by measuring the MFD at fixed distances from the sensor, with results of the measurements in the following Table 3.1. For the sake of simplicity, the different magnets are named after their individual holding force.

Magnet	Value at 5 cm	4 cm	3 cm	2 cm	1 cm	0 cm
320g	0.06mT	0.05mT	0.05mT	0.07mT	0.46mT	2.0mT
540g	0.07mT	0.06mT	0.09mT	0.2mT	1.01mT	7.5mT
970g	0.07mT	0.09mT	0.16mT	0.38mT	1.46mT	12.5mT
1.1kg	0.17mT	0.21mT	0.32mT	0.67mT	2.3mT	15.0mT
1.7kg	0.11mT	0.19mT	0.36mT	0.94mT	3.67mT	25.65mT
3.2kg	0.28mT	0.4mT	0.8mT	1.95mT	6.7mT	37.5mT

Table 3.1: Results of linear magnet testing.

This data was later visualized in Figure 3.3.

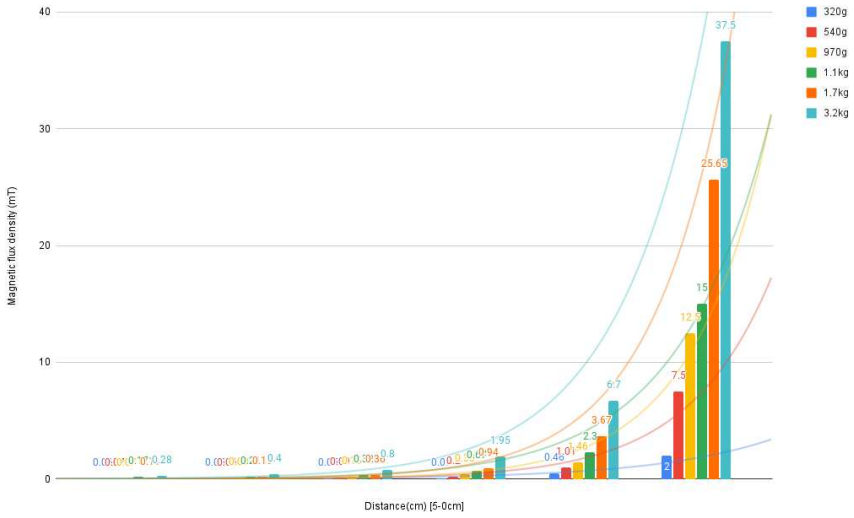
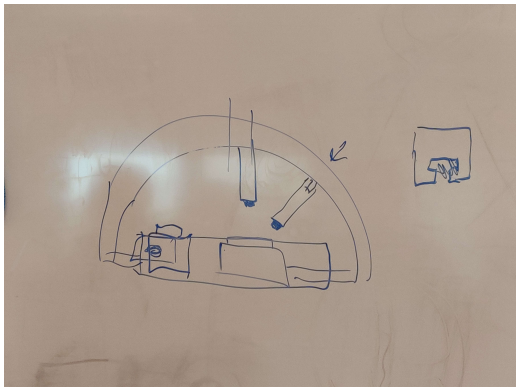


Figure 3.3: Results of linear magnet testing plotted.

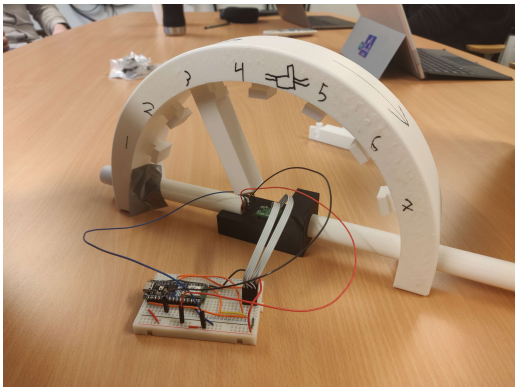
As can be seen from the graph and test results, the MFD decays exponentially with distance, resulting in a workable range of up to 3 cm between sensor and magnet as any longer distance ends up having a noise to signal ratio that is too large. Considering the maximum values of 37.5 mT for the 3.2 kg holding force magnet at close range, the chosen sensor setting, with the assumption that the magnets are set at least 1 cm apart from the sensor, was therefore set to measure up to 25 mT, the lowest limit that can be set on the TMAG5170[33].

For the secondary test, a test rig was designed and 3D printed with the requirements that it would be able to measure the MFD at several different angles in a two-axis rotation setup, at fixed distances from the sensor, and with all the considered magnets. Using an applied coordinate system, annotated later as r for the distance between the magnet and the sensor, θ for the angle along the arch, and ϕ for the measured angle in the following segments, a design allowing control over these parameters was imperative.
The design originated from the idea of measuring at least one of the angles continuously, and sweep along the axis of rotation, with the other two parameters remaining

fixed for that specific measurement.



Rough sketch of the test rig.



Finished test rig.

Figure 3.4: Test rig design process.

The design that was decided upon was an arch with several mounting points 22.5° apart along the θ axis of rotation for mounting the rods that held the magnets, which can be seen in Figure 3.4 as number 1 through 7. In the center of the arch, the sensor was mounted upon a holder, seen in the figure as the black 3D printed part with connecting wires, allowing the arch to rotate around the sensor with the magnet at fixed distances r. In order to measure the angle of ϕ, a potentiometer of type Bourns 3382H-1-103 was connected between the rotating arch and the stationary sensor holder[42].
The measurements were taken by rotating a magnet attached to one of the rods around the sensor at a fixed angle θ and saving the measured MFDs X, Y, and Z components in separate text files for each measurement that could then be processed and visualized using MATLAB.

The measured test data was later visualized using MATLAB to be able to analyze the data more intuitively.

In order to visualize the different X, Y, and Z components' measured effect on the total MFD, the data was processed to use MATLABs *scatter3()* function, which creates numerous dots. This is to plot each point according to the calculated positions using equations 3.3 through 3.5, and then coloured according to which component of the total MFD is most prevalent at each given point. The three different components of the flux density vector X, Y, and Z were mapped to a colour each, namely red, green, and blue respectively. The saturation of each colour indicate that component's prevalence at that point calculated as per equation 3.1, and the size of the point indicate the total measured MFD according to equation 3.2.

$$c_k = \left| \frac{d_k}{max(|d|)} \right| \tag{3.1}$$

$$s_k = \sqrt{X_k^2 + Y_k^2 + Z_k^2} \tag{3.2}$$

$$x_k = r \cdot \cos(\theta) \tag{3.3}$$

$$y_k = r \cdot \cos(\phi) \cdot \sin(\theta) \tag{3.4}$$

$$z_k = r \cdot \sin(\phi) \cdot \sin(\theta) \tag{3.5}$$

Where c_k is the saturation value between 0 and 1 of point k for the affected colour, d is the measured dataset, s_k is the size of point k, and x, y, and z respectively are the calculated positions in space. These measurements resulted in the visualization of the data as seen in Figure 3.5.

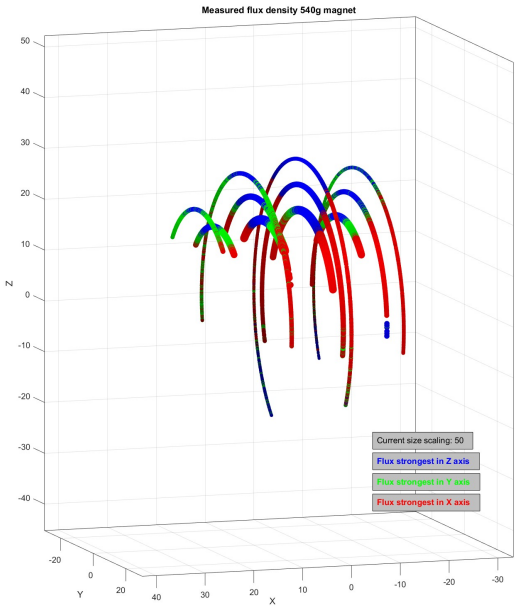


Figure 3.5: Visualization of 540g magnet measured values with sizes scaled to 50 times larger.

A few observations can be made from analyzing the figure. Firstly, the component with the highest effect on the total MFD corresponds with what would be expected, with the Z component being dominant above the sensor, and X and Y being dominant depending on where the magnet is along that plane. This acts as proof of the sensors', and thereby the system's, ability to detect a magnet's position using the three MFD components.
Another observation that can be made, is that the total MFD (indicated by the size of each dot) remains nearly identical at a set distance from the sensor. For this case the angle of the magnet does not in any meaningful way affect the total MFD, but rather only affect which component is the most dominant. The sole contributor

to changes in the total MFD is therefore the distance between the magnet and the sensor.

The chosen magnets used for the glove were the magnets with 320g holding force, and 540g holding force. The choice of these magnets are discussed in Section 5.2.4. Due to faulty wiring in the test rig, and lack of available time for repairs, no measurements were conducted with the 320g holding force magnet. Following the findings of the first linear test, a conversion ratio of the MFD measured with a 320g magnet and a 540g magnet could be calculated for each set distance from the sensor. Applying this variable conversion ratio to the measurement of the 540g magnet, a calculated set of values could be given as seen in Figure 3.6.

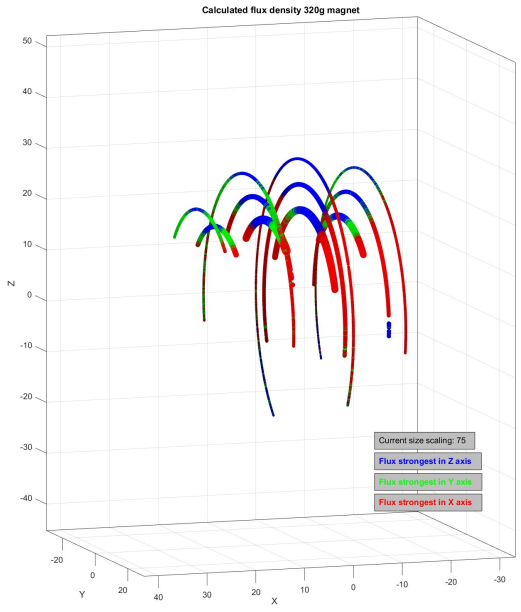


Figure 3.6: Visualization of 320g magnet calculated values with sizes scaled to 75 times larger.

Further data visualization of the stronger magnets can be found in Appendix A.

3.1.4 Microcontroller and decoder

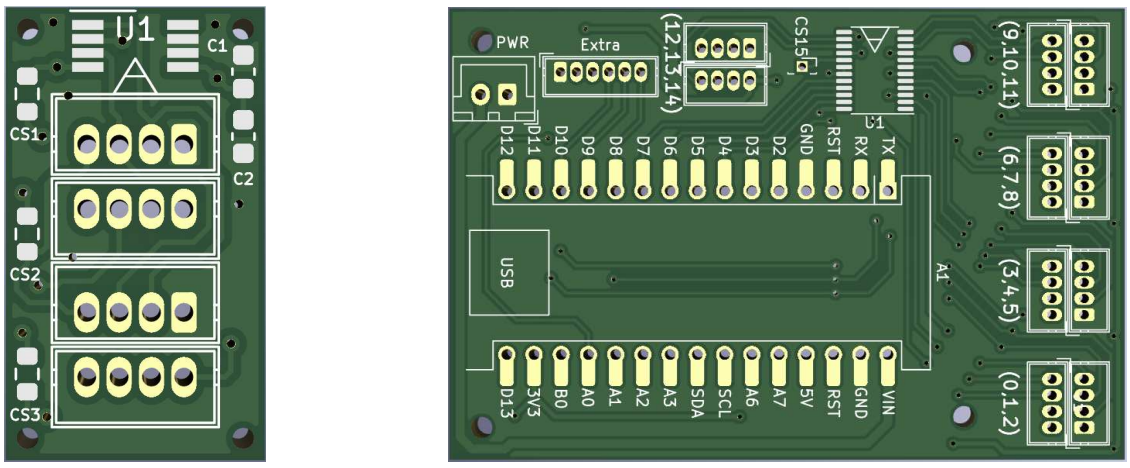
Most ordinary personal computers lack an SPI-interface, and because of this a single-board MC was chosen to read the measurements from the sensors and transmit this data via USB to a computer. Information about this was mentioned earlier in Section 2.3.1.
The Arduino® Nano ESP32 was chosen for this task because of three main qualities.

- Firstly, it has a small form factor, making it ideal for mounting on a glove, or other mounting points with a small footprint.

- Secondly, the chosen board is a well established product in the Arduino® line of products with the ESP32 MC, and is well understood amongst its user base. This comes with the benefits of having a large knowledge base amongst users and good support. It also has the benefit of being easy to acquire both monetarily and time-wise[43].
- Thirdly, the single-board MC design itself has two major additions that differentiate it from other single-board MCs of similar sizes, that being a wireless communication capabilities allowing for Wi-Fi and Bluetooth communication, and access to dual core operation, due to the ESP 32 MC. While wireless communication was not used for this project, as will be discussed later in section 5.7, the dual core operation saw great use for the project, which will be discussed later in section 3.3.1.

A trade-off this board was the number of available GPIO-pins, which is an important factor discussed in Section 2.3.1. As the sensors communicate with the MC using one common SPI bus, the number of GPIO-pins would not be enough for the number of sensors required. To solve this issue with minimum increase in weight and size of the hardware, a 74HC154 decoder was implemented into the system[44]. The chosen decoder offers a 4:16 decoding, which enables using only four of the MCs GPIO-pins to control up to 16 separate sensors by keeping all outputs high except for the selected CS which is pulled low. This guarantees that only one *sub* on the SPI-bus is active at any given time, which is critical for SPI to work.

3.1.5 Circuit and PCB design



(a) Sensor PCB **(b)** Main PCB

Figure 3.7: 3D render of the designed PCBs, made in KiCad's 3D Viewer.

The design for the Hall Effect sensor PCB was inspired by the TMAG5170 evaluation module from Texas Instruments [45]. To keep the wiring of the components from obstructing the movements of the hand excessively, the sensor PCBs were designed with double SPI bus connector pairs and three chip select jumpers (CS) as seen

in Figure 3.7a. This enables a chain of up to three sensors to be connected in a row which was deemed enough for each finger. With this in mind, the main PCB (Figure 3.7b) was designed with five sets of SPI connectors (marked by numbers 0 through 14) connecting the chip select lines through the decoder (U1) and the main bus directly to the Arduino Nano ESP32 which is surface-mounted to the largest footprint (A1). These designs was sent to a PCB manufacturer, and ordered.
See Appendix C for a more in depth view of the PCB design and layout.

3.2 Data gathering using a camera stereo setup

A stereo vision setup is used both as validation for the system and as ground truth for the neural network in the network's learning phase. The setup consisted of two Brio 300 Logitech cameras. The reasons for choosing these cameras are twofold: they lack the feature of auto-focus, which is not desirable since it can intervene with the calculations, and they have a sufficient frame rate of 30 fps while recording. A simple placement of cameras was also used where the cameras are parallel, as shown in Figure 2.8, this is to make both the calibration process and the triangulation calculations easier. The reason for choosing a stereo vision setup instead of a mono-vision setup is the gained ability to calculate depth through triangulation and also the possibility to stabilize points of interest, where one camera can lose sight of the point but the other keeps track of it.

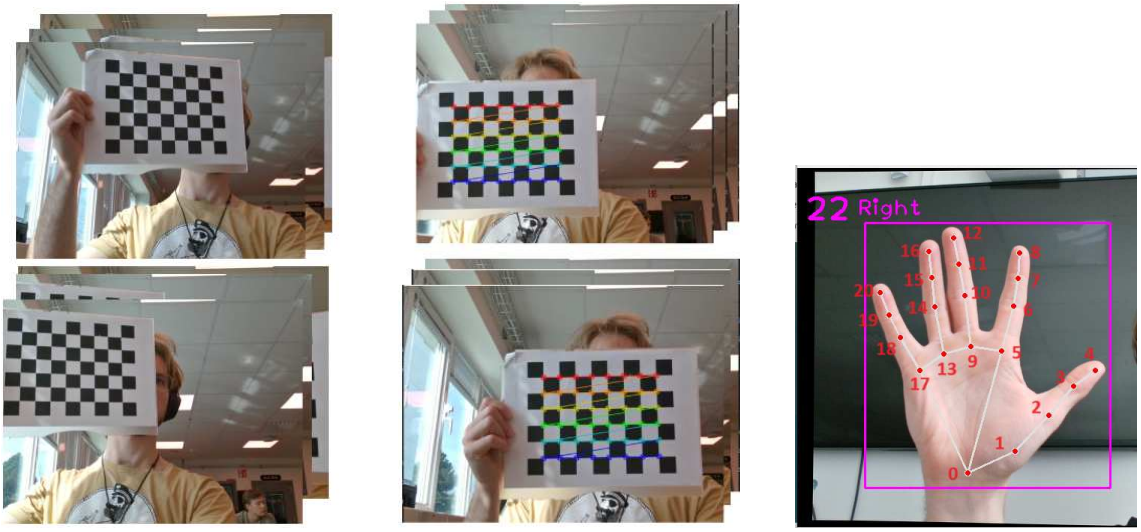
Figure 3.8: Setup of the stereo vision setup and the checkerboard for calibrating.

3.2.1 Calibrating cameras

To be able to use photogrammetry with any success, it is needed to estimate the intrinsic parameters of the cameras. Important to notice is that estimation is needed for both cameras separately, as the cameras slightly differ even if it is the same

model. The intrinsic parameters that are considered in this project are the focal lengths, optical centers, and distortion coefficients. The optical center is where the rays of light intersect in a camera and distortion coefficients account for the the distortion caused by the camera lens. The calibration process is briefly shown in Figure 3.9.

- Firstly, a collection of pictures is needed for calibrating the cameras, shown in Figure 3.9a, where an object of known size needs to be visible for both cameras. A commonly used object is a chessboard, which has a lot of support in OpenCV library[46].
- The second step is to detect the 2D coordinates of the chessboard, this is done by using OpenCVs function *findChessboardCorners()*. This process is shown in Figure 3.9b. The 2D coordinates are necessary as input arguments for the actual calibration of the cameras when using OpenCVs *calibrateCamera()*[47].
- The third step is to calibrate the cameras separately, using *calibrateCamera()*. This function outputs a camera matrix, which contains focal length and optical center, and distortion coefficients, as well as the intrinsic parameters. The function also outputs translation vector and rotational vector as extrinsic parameters, it is only the intrinsic parameters that are of interest for this application. This is because the extrinsic parameters are estimated when calibrating the cameras as a stereo setup.
- Lastly, the cameras need to be calibrated as a stereo setup. To achieve this OpenCVs function *stereoCalibrate()* can be used[47]. The function outputs both the extrinsic and intrinsic parameters, as well as a representation matrix. The representation matrix is used to represent a point from a camera's coordinate system to the other camera's coordinate system. With all these building blocks it is possible to rectify the images to a common plane using OpenCVs function *stereoRectify()*[47]. Parts of the results are shown in Figure 3.9c, where the rectification of the images is visible with the black borders in the image.



(a) Multiple images taken for calibrating the cameras.

(b) The chessboard's corners detected and drawn for all the images.

(c) Landmarks of the hand in a rectified stereo image.

Figure 3.9: Camera calibration and stereo rectification process by using functions form OpenCV. The landmarks are not added by this process but are added later by CVzone.

3.2.2 Landmark tracking and depth

For landmark detection and tracking, CVzone was used[48]. The X and Y position of the 21 points is stored in a list and the Z position of each landmark is calculated through triangulation, shown in Figure 3.8. Lastly, the angle of the bend of each joint is calculated by the use of "vg.angle()"[49]. This is a complementary package to NumPy to be able to easily calculate angles between two vectors in 3D space, thus eliminating the need to manually code the algorithms for this task.

3.2.3 Calculation of joint angles

The angles given by the hand tracking module were calculated in Python inside the hand tracking script. In this script vectors were created from the landmarks given by the hand tracking. The angles of interest for the project were the angles for the different fingers MCP, PIP and DIP joints as well as the MCP and IP joint for the thumb. Thus to create the angle for, e.g., the PIP joint of the index finger the two vector created were:

$$\mathbf{v_1} = lm_{MCP} - lm_{PIP} \tag{3.6}$$

$$\mathbf{v_2} = lm_{DIP} - lm_{PIP} \tag{3.7}$$

To calculate the angles between these two vectors the dot product of the vectors were used as in the following equation:

$$\cos\theta = \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{|\mathbf{v_1}||\mathbf{v_2}|} \tag{3.8}$$

This was simplified as stated earlier in Python using the package vg.

3.3 Code

In order to effectively communicate between the sensors and visualization software, coding is required. The following chapter will cover the different platforms used, and for which purpose.

3.3.1 Microcontroller

The chosen single-board MC, as discussed in section 3.1.4, was chosen due to its combination of low footprint, relatively powerful processing power, and multi-core capabilities. The MC runs C++ code that can be summarised with snippets of pseudocode.

The general outline of the code is to first run a setup sequence, setting the sensor registers to the settings needed and creating all the necessary internal variables to be used, including *tasks* that run continuously. Once the setup phase is complete, two *tasks* gather and then transfer data from sensor to the computer.

All Hall-effect sensors are communicated within the code through objects of the *HallSensor* class made specifically for this project, where the input arguments for all objects are: the range of measurement of between 25 to 100 mT, the CS pin for choosing the active sensor, and a setting of whether a decoder is used, as can be seen in the code snippet in listing 3.1 The *HallSensor* class contains functions for communicating with the sensors through SPI by sending the necessary data on the SPI bus while the relevant sensor CS is selected. All calculations for converting the read values that come from the sensors on the SPI bus are done in these functions, before being passed to the main loops. Code for the *HallSensor.h* can be found in the appendix in listing D.3.

Listing 3.1: Pseudocode showing objects of HallSensor class creation.

```
Create HallSensor object of Sensor Name with (chosen CS pin, measurement
    range, decoder setting);
```

Using the decoder setting, the MC sets the binary representation over a pre-defined GPIO bus high and low, instead of individual pins. This allows for the main code to treat the objects similarly without converting CS outside of the *HallSensor* class.

Utilizing the dual-core abilities of the ESP 32, two *tasks* were created to run indefinitely after the first set-up phase. The first *task*, *TaskSensorRead* as seen in listing 3.2 handles the gathering of data from the sensors through SPI, by calling upon

each individual sensor object and requesting the current measured MFD in the X, Y, and Z directions. This is then saved in each object's attributes. The second *task*, *TaskSerialSend* as seen in listing 3.3, copies the piece of memory that contains all the measured data, represented as an array of sensor objects, for mutex purposes, before sending the saved data through serial bus to the ROS 2 environment.

Listing 3.2: Pseudo code of task for gathering values from sensors.

```
TaskSensorRead{            //Task will run indefinitely.
    Take protective mutex; //Take the protection over memory
    for each sensor object num in the list of sensors{
        get MFD from sensor_num; //Acquire and save the measured data
    }
    Release protective mutex; //Release the protection over memory
}
```

Listing 3.3: Pseudo code of task for sending saved values to computer.

```
TaskSerialSend{              //Task will run indefinitely.
    Take protective mutex;   //Take the protection over memory
    Copy every sensor object to second array; //Copy the memory
    Release protective mutex; //Release the protection over memory
    for each sensor object num in the list of copied sensors{
        Send saved MFD from copied sensor_num; //Send copied data
    }
}
```

Due to the dual-core code set-up, mutex, or mutually exclusive flag, a type of memory protection that only allows code to access the memory if it has taken the mutex, was implemented to prevent race conditions of the saved values in shared memory. These are set when either the read *task* reads from the sensors and writing that to memory, or when the send *task* is copying the memory for sending. Using this guarantees that the data is safe from being corrupted due to memory access problems. The two *tasks* work in parallel with each other, and interact only when accessing the shared memory. How they interact can be seen in the following flowchart.

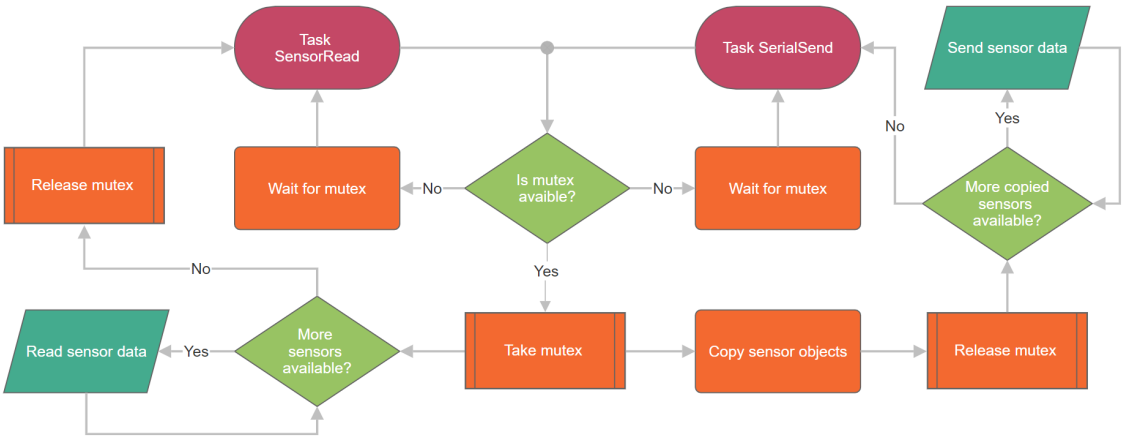


Figure 3.10: Flowchart over the two *tasks*.

As can be seen in the flowchart, the two *tasks* only interact with one another when trying to access the memory protected by the mutex. The *SensorRead task* requires the mutex during the entirety of its run, while the *SerialSend* only requires it during one step. This is to allow the *SensorRead task* to fully update all sensor objects with new measurements before *SerialSend* copies it and sends it. These parts of the code allows the system to gather data from the sensors continuously, format it, before sending it through the serial bus.

3.3.2 Neural Network

A Feed-forward neural network was made in *PyTorch*. Three different approaches were analyzed. The first approach used 42 MFD inputs, which were the x, y, and z MFD from each sensor. These values were then mapped to the 14 angles obtained from the camera setup. The output was thus 14 predicted angles, 1 for each sensor. The second approach used nine inputs with three outputs which correspond to one network per finger. One exception was the thumb which only had two sensors. This resulted in six inputs and two outputs for the thumb network. With this approach, a network was trained for each finger, and their corresponding state dictionary was saved. When the networks were used, the state dictionaries would be loaded and used on the data from the finger they were trained on. The last approach used one network per sensor that took 3 MFD inputs and outputs 1 angle.
Different hyperparameters were tested for the three different approaches where the platform "weights and biases" was used to effectivize the procedure[50]. This is an add-on that logs the performance of the network and facilitates the process of experimenting with different hyperparameters during one run. The result is then exported to their platform where it can be visualized and analyzed. The networks were trained with different hyperparameters, and the configurations that gave the best results were used with Unity.

After experimenting with hyperparameters, the following architecture was chosen. Three layers layers were used with the optimizer *Adam*, and the activation function

LeakyReLU. The learning rate was set to 0.1 with a scheduler that reduced the learning rate by a factor of 0.6 if no improvement was made for 10 iterations. The loss was calculated as the mean square error between the predicted angle, and the angle from the cameras. 10 neurons were used in each layer.

3.3.3 Visualization in Unity

Unity was used to visualize the movements of the hand. Both from the coordinates of the hand tracking script and also from the angles calculated from said coordinates. This was done to visualize that the angles were correct so that the neural network did not train on false data. Since the angles of each joint are of interest, the number of models to choose from is limited. Not all hand models are made to rotate around the joint, but instead work with either position data or vectors. As time was a constraint, the already existing hand model, XR Hands, was used[51]. This model was chosen since we found that it was the easiest to control with angle inputs. The angle data was received from the neural network as a simple string where each value represented a joint on the hand. The values were then applied on each of the joints as a rotation along one axis. The script responsible for rotating the joints simply takes the value from the string and applies it as a set rotation for the joint. As each joint depended on others it was enough to change the rotation variable and the rest of the finger adapted as the human finger would. In the Figure below can be seen the XR Hands model to the left side by side with the hand tracking skeleton to the right.

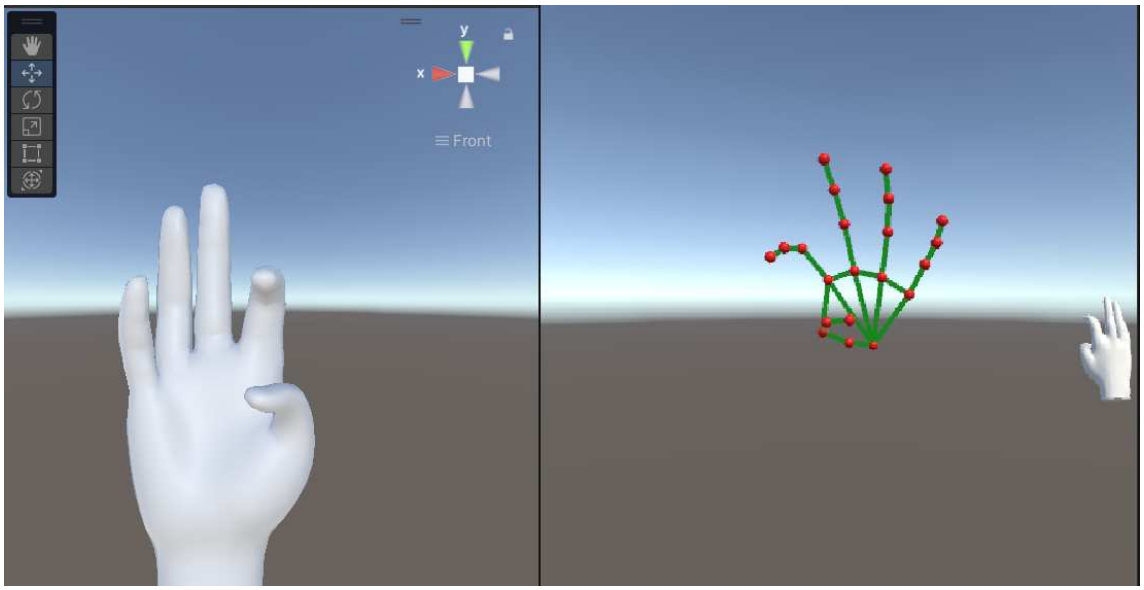


Figure 3.11: Hand models in Unity showing the bending of the thumb and index finger.

3.3.4 ROS

The *ROS*, robotic operating system, structure is divided into two main parts: one part being the training of the network and the other for continuously running sensor

data through the neural network and sending the prediction from the network to Unity.

The structure for training the network, as shown in Figure3.12, is composed of two publisher nodes that publish data to a subscriber, which gather the data used for training. In this structure, the node called *publisher node* refers to the camera stereo setup (shown in Figure3.8). The node publishes both the rectified image as an Image message and the positions of the landmarks generated by CVzones handtracker as a float64multiarray message. The node that is called *micro ros platformio node* is a node in the micro-controller, created using micro-ROS[52]. This node publishes the MFD in X,Y and Z axes for each sensor connected in a float64multiarray message. The subscriber *trainnet node*, displays the published Image message and, more importantly, reads a sensor message for each message gathered from the cameras. This ensures that the data is the correct format for training the network, but it comes with the cost of not utilizing all the data from the sensors. The *publisher node* publishes the camera data at a rate of 15 Hz, while the *micro ros platformio node* publishes at 120Hz. Consequently, a lot of sensor data goes unused when gathering training data. The reasons for this difference are that the camera operates in 30 frames per second and requires additonal calculations in python code, while the micro controller code is written in C++ which is faster. The sensor data gathering is also not hindered as much by the hardware.

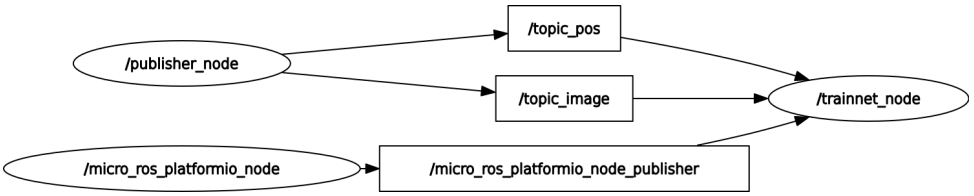


Figure 3.12: The architecture for training the network.

After training the network, the next structure continuously runs sensor data through the neural network to be able to get a prediction of the angles for each joint. In the subscriber *publish2unity node* the predicted data from the network is sent to Unity through a network socket.

Figure 3.13: The architecture for sending data to Unity.

4
Results

The final glove iteration was made up of 14 Hall effect sensors and magnets, collecting data from the 14 joints of interest. These sensors were grouped into different series. One series for each of the four fingers and one for the thumb. These are in turn connected to a MC via a PCB.

4.1 Assembly

The final hand motion system prototype was made by using a polyester glove. Each of the sensor's PCBs and magnet mounts were sewn on by hand. The MC was fitted in a 3D-printed "case" that enabled the MC to be somewhat moved along the arm depending on the person using the glove. Each sensor-series were connected by wire-bundles origin from the main MC PCB.

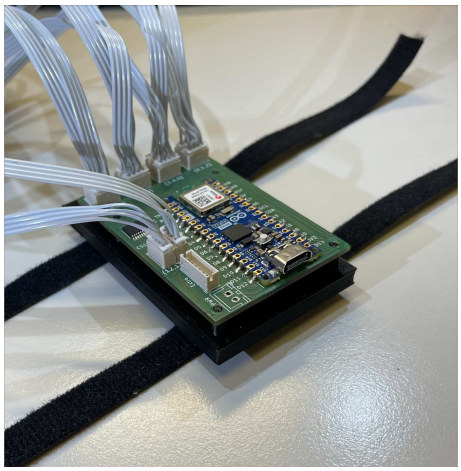


Figure 4.1: The mounting for the main PCB

As the hand offers limited space and the sensor-PCBs have to be a set size, some creative solutions were adapted. The last sensor on each finger, the one responsible for measuring the DIP joint, was rotated 180 degrees and shared its magnet mount with the PIP joint sensor. This way all the joints could be measured while every component could be fitted on the hand.

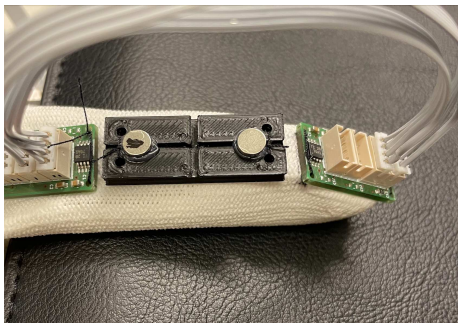


Figure 4.2: The solution for the DIP setup

In total the final glove weighs approximately 110 grams, which can be considered an acceptable weight. Each of the wires were made with a focus on mobility which is why they're extra long as to not hinder any movements.

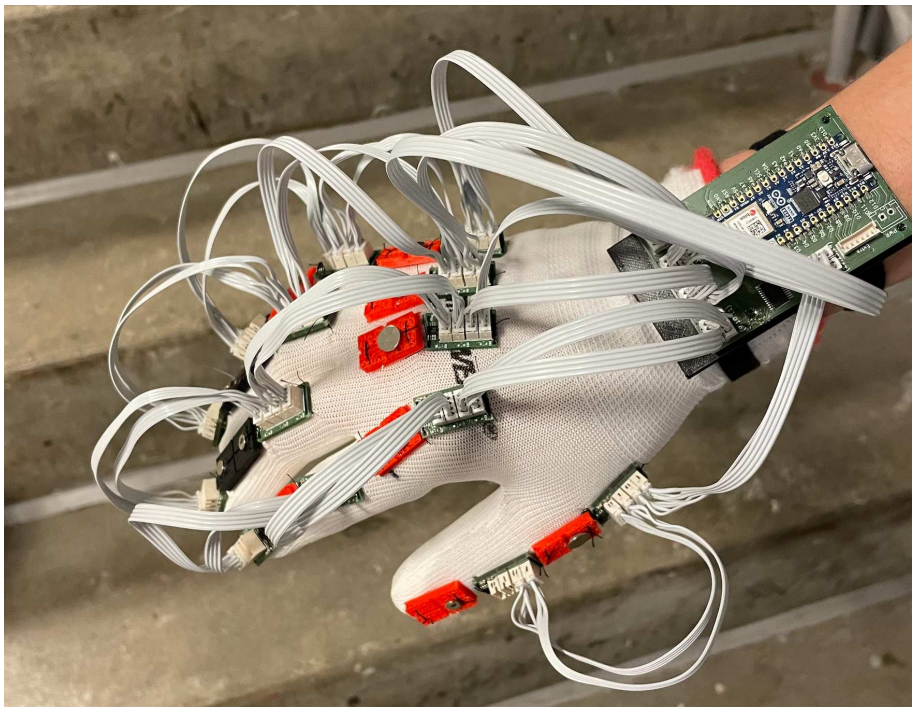


Figure 4.3: The final glove

4.2 The measured magnetic flux density

Data was sampled from the device in three fixed positions of the fingers for the purposes of evaluating the raw data itself. These can be observed below in Figure 4.5 The entire set of MFD data is in units of millitesla, and has been visualised as histogram matrices with bin widths of 0.024 in Appendix B.

Here a relevant subset of the data is explored. The distribution of the sampled values can be easily observed in the histograms of Figure 4.4.

The horizontal red lines mark the largest span between the three mean MFD values calculated for each position.
Using these visual representations and the tables of mean values as a reference, this data show distinct variations in the MFD between the different positions, for certain axes and sensors.

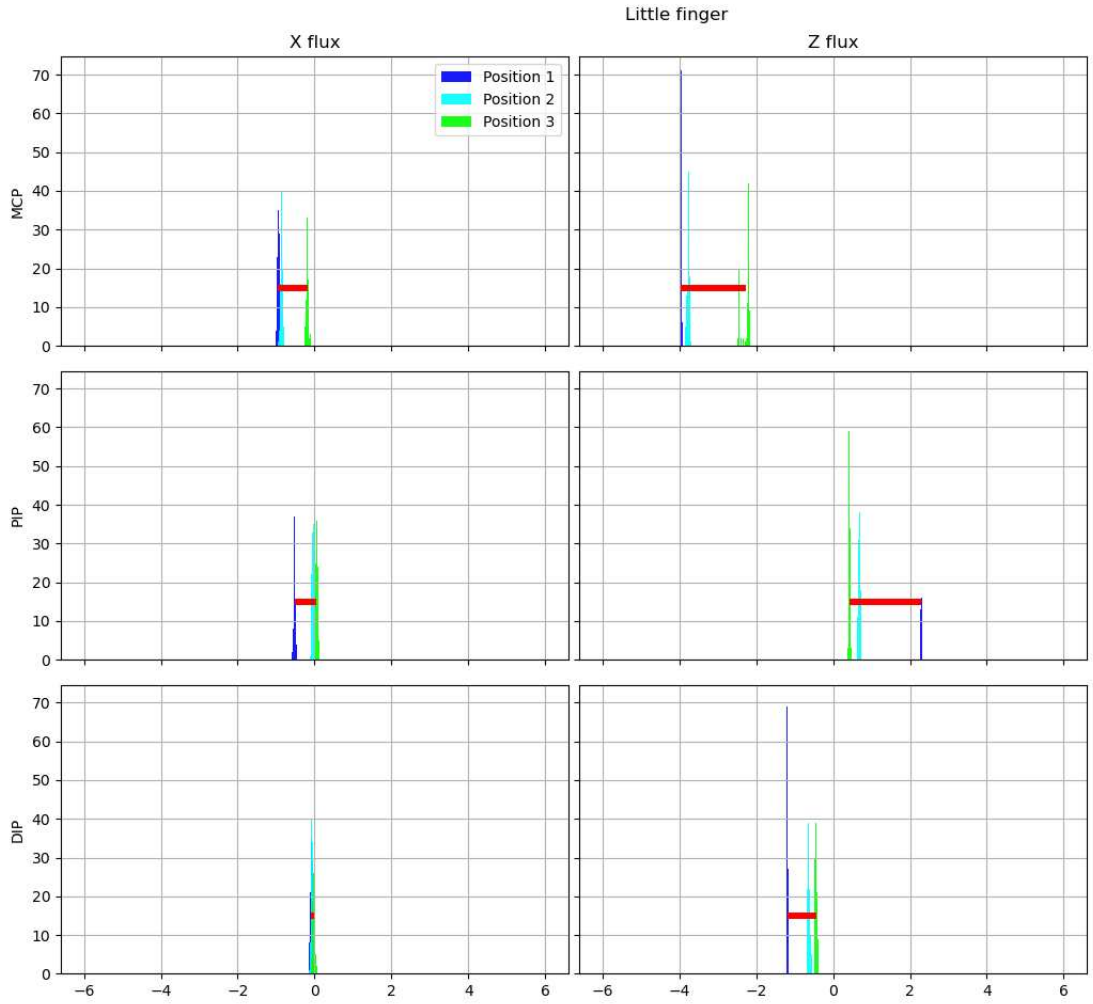
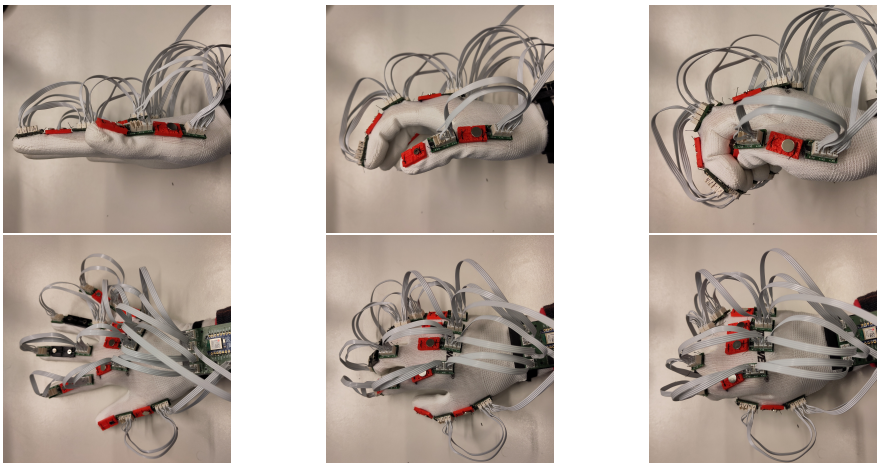


Figure 4.4: Histograms of the MFD measured on the X and Z axes from the sensors placed on the little finger



(a) Position 1, fully abducted and extended.

(b) Position 2, approximated midpoint.

(c) Position 3, fully adducted and flexed.

Figure 4.5: Finger positions for the three snapshots used for raw data analysis

Sensor	X	Z
Position 1		
1	-0.944311	-3.953676
2	-0.523369	2.292841
3	-0.083322	-1.182564
Position 2		
1	-0.858276	-3.766332
2	-0.053290	0.676049
3	-0.069790	-0.627922
Position 3		
1	-0.188716	-2.282129
2	0.057744	0.428533
3	-0.022472	-0.441395

Table 4.1: The calculated mean MFD of the X and Z axes, for the different positions.

The sensors are numbered in order from MCP-joint, PIP-joint, to DIP-joint sensor, for each finger starting from the little finger. This leads to the little finger having sensors one through three, the ring finger having sensors four through six etc.

A summary of notable patterns from the entire set of data:
See Appendix B

- **X-axis:**
  The MCP joint sensors measure a discernible difference between positions 1 and 3, with the exception of the ring finger which have a fairly static reading.

Of the DIP joint sensors, the differences between positions are not distinct and total magnitude of the MFD is close to 0.

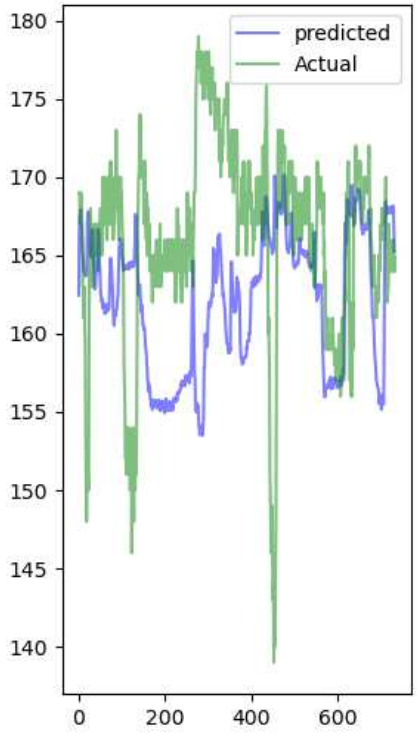
- **Y-axis:**
  Several sensors measure the most discernible difference being in regards to position 2. Shows distinct readings from the PIP and DIP joint sensors of the, as well as the thumb.

- **Z-axis:**
  The MCP sensors of the middle and ring fingers show similar differences but in opposite directions when comparing position 1 and 2, and position 1 and 3. For both the DIP and PIP joint sensors, a distinct difference in MFD for each finger between position 1 and 2, while 2 and 3 are less discernible.

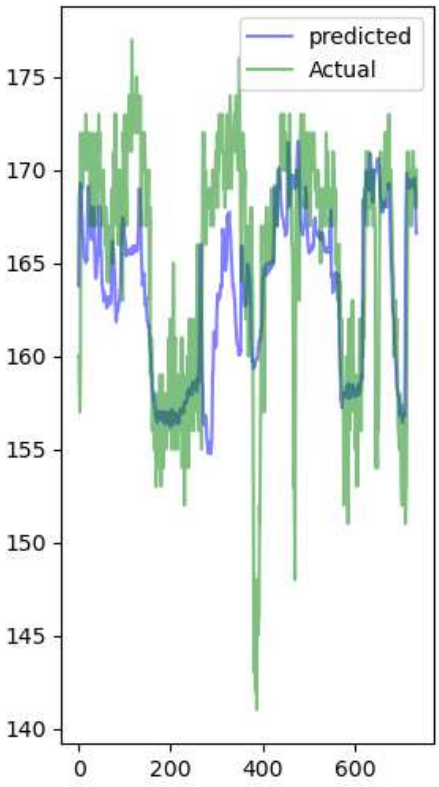
4.3 Neural network results

The results show how the predicted angles differ from the actual angles and thus how well the neural network has mapped the sensor data to the hand tracking data.

In Figure 4.6 the result from the first approach can be seen. One can see a pattern with the prediction having the same form for all 14 sensors where only the scaling has changed. This was an outcome of having the same general model for all 14 outputs. The consequences of this general model were that if one finger was moved, all the other fingers were moved as well and for some reason, the sensors on the thumb had the most impact on the movements of the other fingers. Attempts were made to decrease this behaviour by only moving one finger to give the network data to map changes for sensors to correct angles. This did not have any noticeable effects which may have been a consequence of the lack of enough and also reliable data.



Predictions for the sensor corresponding to the MCP joint in the little finger.



Model over the sensor corresponding to the MCP joint in the ring finger.

Figure 4.6: The result from the first approach with one model for all the joints. The y-axis shows the degree of bend with 180 degrees being a stretched finger. On the x-axis the number of data points is shown. The predictions for the other joints can be found in the Appendix.

The result from the second approach can be seen in Figure 4.7. It is shown that the model is unique for each finger but still has the same form within the set of sensors that embodies a finger. The problem with implementing multiple models for the hand is that the data available for the model gets divided by five leading to a much smaller data set. When modeling the angle into the unity hand model it was apparent that a change in angle in only one joint affected all the joints representing the finger which also boils down to a lack of data and the appearance of bad data.

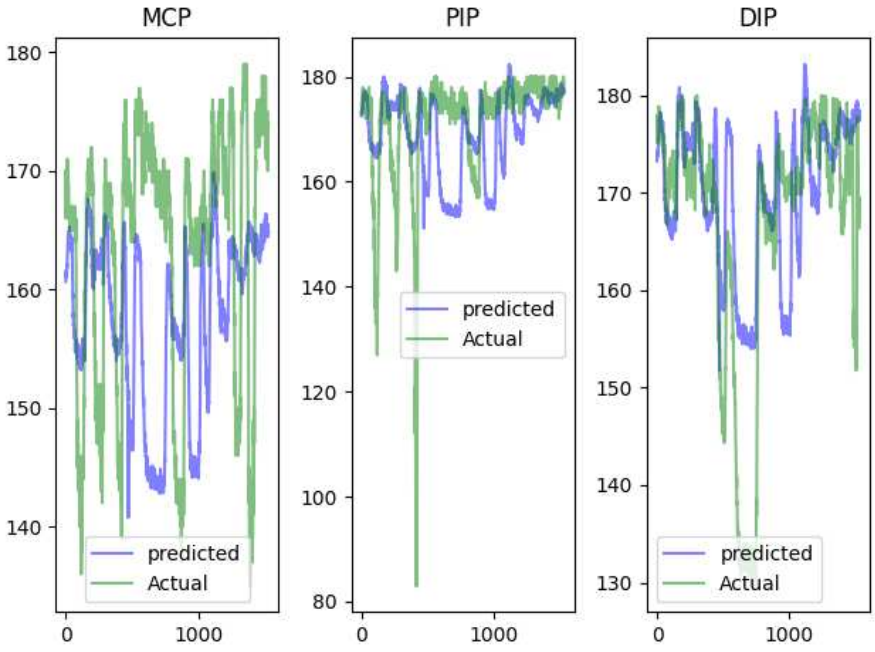
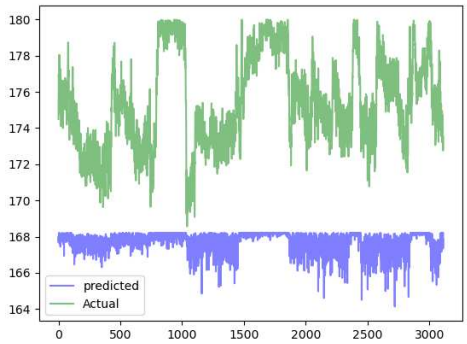
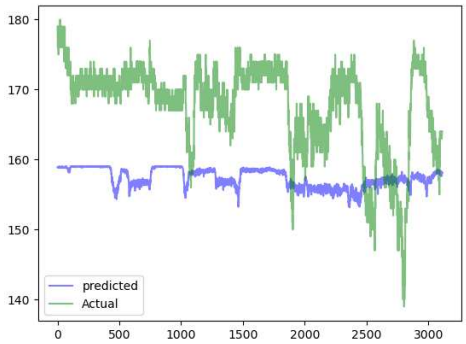


Figure 4.7: The predictions from the network trained on the data from the index finger. The predictions for the other fingers can be found in the Appendix.The y-axis shows the degree of bend with 180 degrees being a stretched finger. On the x-axis the number of data points is shown.

Lastly, as mentioned in the method, a unique network was trained for each sensor instead of dividing the fingers as a system. The data set gets divided across the 14 different networks, making the available data for each network considerably smaller. The behaviour was similar to the second approach where a movement with one finger did not move the other fingers, the difference was that this approach was a little bit more responsive.



(a) Predictions from the network trained on the sensor corresponding to the MCP joint in the index finger.



(b) Predictions from the network trained on the sensor corresponding to the PIP joint in the index finger.

Figure 4.8: Models over two of the sensors in the index finger where each sensor is model by a unique network. The y-axis shows the degree of bend with 180 degrees being a stretched finger. On the x-axis the number of data points is shown.

5
Discussion

The following sections discuss the different results, possible improvements, how limitations have affected the project, and other topics.

5.1 Ethics

When doing projects of this kind, multiple ethical aspects must be considered. During the development of the glove much data has been collected, however this data was not tagged with any information that could be linked to a person. But in the case of this hand tracking device getting released unto the market ethics quickly becomes a concern. With any data collecting tool this data could possibly be connected to the user. Further, as this type of equipment could be considered a medical device, the question of ethics surrounding the development of the project became more prevalent. Because the device is collecting data that could be connected to a certain person this information needs to be secured so that only authorized persons can access it. Who these authorized people would be lies outside the scope of the project and would be heavily regulated. Especially as this information regards a person's finger movements or maybe rather the user's limited range of motion. This information could possibly be used to discriminate, for example in a work environment.

5.2 The glove

When designing the MC PCB it was done with the intention to be able to add an extra sensor to the thumb for a total of four sensors specifically dedicated to the thumb. As of now, there is no way to map the thumbs opposition. This is because the way that the camera data was collected didn't include a way of collecting the angle between the hand's palm and thumb that would give reliable data, and therefore this feature was not usable.

The glove also suffers from being one-size and can therefore not be worn by everyone. This is a product of when sewing the glove the main requirement was that the relationship between the sensor and magnet was kept consistent and since finger sizes vary a lot depending on the person, the glove had to be custom fitted to give good results for at least one person. An improvement would be to ensure that the positioning of the sensors and magnets is consistent. A possible solution for this would for example be a better fitting glove that does not stretch when you bend

your fingers or to measure this elongation and compensate for it inside the software. This could potentially improve the learning process for the neural network.

5.2.1 Flexibility of the glove

As each of the components are sewn and the glove flexes when used it wears out the mounting and components risk falling off. The same can be said about the wire-bundles connecting each of the sensor PCBs. As they are rigid they tear when the glove flexes. This in turn has led to multiple wires getting damaged and having to be replaced, which led to more data gathering as data became invalid with broken wires.

5.2.2 Hand tracking limitations

When constructing the glove a mistake was made when a black glove was used as the base for the prototype. This was a mistake as the hand tracking software did not interpret the black glove as a hand. The detection confidence, which determines how confident the program needs to be that there is a hand in the current image frame before it gets classified as a hand, could be lowered to mitigate this problem. An effect of lowering the confidence would be that it causes some uncertainty of what a hand is , but there is another problem with using a black glove and that is that it is more difficult for the cameras to see the contours of the hand. This makes the landmarks getting incorrectly placed which makes the ground truth not trustworthy.

Some other solutions to this were either to try and repaint the glove or in some way change the software's image channels. But these solutions weren't a guaranteed fix, and as time was a constraint it was deemed easier to dismantle and re-sew the prototype on a white glove instead, which gave better results. Still, with a white glove, the hand tracking performed poorly while tracking, as the wires caused disturbances for the hand tracking software. This made the landmarks flicker to different positions that were a part of the wires instead of the actual finger. The consequence of this was an untrustworthy ground truth that made the networks perform poorly.

The poor ground truth, which results in a flawed product, is shown in Figure 4.8a as well as in the other neural network graphs. The "Actual" line should be smooth as the data was recorded with slow movements. However, it contains a lot of noise which is a product of the flickering of the landmarks. This noise led to poor predictions and made it impossible to decipher what behaviour the neural networks recreated. An improved hand tracking would be required to receive usable data for a neural network or possibly to use more low profile cables that would not obscure the camera's vision leading to a higher likelihood for the glove to be detected as a hand and stop the landmarks from flickering.

Another more technically advanced solution would be the use of rigid-flex PCBs that could remove the need for external wiring altogether[53]. These are however

normally more expensive and the flexible internal wiring is not made to endure repeated back and forth bending but to be bent to a permanent position. The nature of this glove's movements would probably lead to breakage in these PCBs.

5.2.3 Possible improvements to angles

As described in the method section, the angles were calculated using the dot product between the two vectors created between landmarks of different joints. This works well for the angles of PIP and DIP joints, as these are locked to the same axis of rotation because of their limited DOF.

However, during the project, one of the vectors for the MCP joints were calculated sub optimally. In the Python code for the angle calculation, one vector goes from the MCP landmark to the PIP landmark which is desirable. But the other vector is created using the MCP landmark and a landmark down on the wrist as seen in 3.8 as landmark 0. This works when the two vectors have a common axis of rotation but depending on hand positioning this is not always the case which leads to inaccuracies in the angle values. To fix this, a new vector could be constructed by projecting the vector between the MCP joint and wrist landmark onto the axis in question or even simpler; using a unit vector along said axis. The size of this error has not been explored but seems to be negligible as the angles given still seem reasonable in the visualization. However, any error that might have derived from the inadequate angle calculation could have been concealed by the larger problems with the hand tracking discussed earlier.

5.2.4 Magnets and magnet placement

As all the components have been hand sewn to the glove there is no guarantee that the distance between each magnet and sensor is the same. This can be one of the reasons why the neural network is having problems separating each of the joints when making its predictions and thus tends to predict a mean value instead. If instead the distance had been consistent, the magnitude of the measured MFD would possibly have been more stable, which could have led to a better model.

5.3 Choice of magnet strength

Following the results of the magnet tests in Section 3.1.3, Figure 3.3, a clear difference of MFD between the assorted strength of magnets could be seen. An argument could be made for using the stronger magnets, as they have the clearest effect on the MFD measured by the sensor and could therefore overcome the noise from measuring inaccuracies and background magnetic interference. These were not chosen for the glove in the final prototype, as potential interference in between the numerous magnet-sensor pairs on each finger would create too much conflicting data for the network to parse. Instead the two lowest, that being the 320g and the 540g holding

force magnets, were chosen as to reduce such interference. Tests using several different setups would have made the choice of magnets, but, due to time constraints, not enough magnets could be acquired in time to test the potential combinations of magnets. This leaves room for potential improvement as different sizes, shapes, and magnet grades could have worked better for this use case.

5.4 Sensor data results

To interpret the raw data, an oversimplified model without consideration of magnetic interference was considered. Abduction and adduction would be expected to result in a change of MFD most notable along the X-axis and possibly the Y-axis of the MCP joint sensors, since such a motion would align with that plane. Flexion/Extension of the fingers and thumb would be expected to only affect the Y and Z axis of the sensors as those should be aligned with the corresponding planes of movement.
More specifically the largest difference in MFD for the sensors at the MCP joints would be expected to happen between positions 2 and 3, to correlate with the physical movements of the fingers.
The largest difference in MFD for the sensors at the PIP joints should then happen during the transition from position 1 and 2, as the fingers normally bend around those joints first. This model is summarized in the table below.

Axis	**MCP**	**PIP**	**DIP**
X	Δ = Abd/Add	$\Delta \approx 0$	$\Delta \approx 0$
Y	Δ_{pos2-3} = Ext/Flex	Δ_{pos1-2} = Ext/Flex	Δ = Ext/Flex
Z	Δ_{pos2-3} = Ext/Flex	Δ_{pos1-2} = Ext/Flex	Δ = Ext/Flex

Table 5.1: Simplified model of interpretation

Inspecting the column “X flux” for the various fingers, a notable difference can be observed when comparing position 1 with the other two, with the exception of the ring finger. This could be a direct consequence of the motion of the fingers during these snapshots as they start from an abducted state and become practically parallel by position 3.
At a glance, the simplest way to interpret this data would be to focus on the values that differ by a notable amount between positions 1 and 3. This oversimplified notion does hold some water for the Y and Z axis, as these parameters would be expected to represent the only relevant plane of motion if abduction/adduction is ignored.
The ring finger and little finger show this correlation fairly well, while data from the middle finger’s MCP joint and Y flux from the PIP joint seemed to contradict this explanation.
By applying the simplified model in Table 5.1 to the collection of graphs in Appendix B the following can be shown:

- 22 sensor axes show distinct linear correlations in support of this model.

- 12 sensor axes show no clear correlation.
- The remaining 8 axes contradict this model.

Of note was also the appearance of noise for the MCP joint of the thumb, specifically in position 3. This could potentially be caused by proximity to the magnets along the index finger, and vice versa, the proximity of the index finger to the thumb could partially explain the non-linear behaviour of the Y-axis measurements of the index finger.
In order to make any concrete conclusions from this data, configurations of the fingers other than the three positions shown in Figure 4.5 would have been needed to be sampled.
This data could serve as a useful baseline for further experimentation, where the possible interference between the fingers could be studied by only changing one angle at one joint at a time.
With such data a true estimation of the complexity of the MFD readings could be made, and could serve as a practical guideline for making further adjustments and improvements to the device.

5.5 Neural network results

The performance of the networks was limited, primarily due to the data gathering. As mentioned previously, the camera had a hard time following the landmarks, especially with the cables attached. If the validation data is not accurate, then the predicted data will not be either.

The first approach which utilized 42 input arguments and output 14 different angles turned out to be problematic as motions from simply the thumb would move the whole hand. This could be because when training, flexion of the thumb was often combined with flexion of the other fingers, which may have led the network to believe these movements were interconnected. Or simply the network worked out a mean value from all the input sensors due to the problem being too complex and predicted the angles based on the thumb's movement. A similar result could be seen with the second approach where the whole finger moved jointly instead of each joint independently.

The final solution was to implement a network architecture that treated each sensor separately from the others and predicted individual angles. This led to some joints moving independently but at the cost of individual range. This can be seen in 4.8b where the predicted angles are almost a straight line compared to the actual data. As the network treats each sensor independently and trains on separate angle data, it became clear that the network had a harder time predicting some of the sensors' datasets than others. This could be either because some sensors struggled more with interference or the cameras were particularly bad at calculating the angles for some hand motions.

5.5.1 Neural network improvements

In the future, better results could potentially be achieved if one network were to be used for each sensor as the third architecture did. The problem with this approach was that the same hyperparameters were used for each sensor even though some sensor data was more difficult to interpret. If the hyperparameters had been tuned individually for each sensor, a better result may have been attained. This was never explored since time was of the essence.

The negative aspect of the aforementioned architecture would be that interference from neighbouring sensors would not be taken into account. Because interference from adjacent fingers could be used to determine motions such as abduction and adduction. If one would remove the magnets from adjacent fingers and then gather data, a dataset without interference could be attained. If then another dataset would be generated with the magnets placed back and thus include interference, the interference could be extracted and used as a means to determine the finger's position relative to adjacent fingers. A future solution could therefore be a network for each group of neighbouring joints where the interference is used as a positive factor.

The decision to try three different architectures was made late into the project. A few different configurations of hyperparameters for each architecture were tried but there was no time to optimize them individually. The hyperparameters used in the result were based on the configuration that gave a good result for all the architectures. However, since the complexity of the network varies with the three different architectures, different hyperparameters would be optimal. If one instead would have analyzed more configurations of hyperparameters for each architecture, a better result may have been achieved.

5.6 Hand model in Unity

The unity model was made up of multiple hand models, to serve multiple purposes. Firstly it displays a copy of the hand tracking model from the Python script, thus showing that the correct data is being sent to Unity. The second model, XR Hands, first served as a validation tool for the calculated angles, visualizing how the given angles would look in real life. This gave visual confirmation of how accurate these angles were. This hand model could also be used to send the neural networks predicted angles to compare these to the calculated angles. Giving a visual representation of the difference between these two angle sets.

For further development of the project, it would be highly preferable to use a self-made model. This model could be programmed to work better with the given input data. As most open access hand models in Unity are made for a Virtual Reality (VR) environment they use coordinates picked up by the VR headset's camera, and not angles. Thus they are not made to be used with rotation of the joints as they have been in the project. With a model specifically designed to be controlled by the

angles of the joints the animation of the hand could possibly have become smoother, more accurate, and without the issue of the joints getting locked in an unnatural 180-degree angle.

Time was dedicated to try and remedy this unwanted behaviour and the reason for its appearance seemed to be when the angle data sent from the hand tracking took a big leap. For example when the cameras lost sight of the landmarks and then gave incorrect values. This ended up locking the finger in an unnatural position, regardless if new correct data got sent to it or not. The solution for this was never found but it was not ruinous for the project as a whole. Furthermore, this did not seem to occur as frequently with the neural networks predicted angles, which was the most crucial point for this model.

5.7 Optimization and further development of code

Due to time constraints, there are a few noticeable improvements that could have been added to certain parts of the architecture. One such addition is data transfer from MC to computer through the use of Wi-Fi. Implementing Wi-Fi would have allowed the glove to move more freely without the constraint of having to remain as close to the computer as it has to be currently.
In addition, while the system did manage to be fast enough to be considered "real time", there are a selection of steps that could have been optimized to further decrease the amount of time taken between sensor read and visualization, and steps to increase the stability of the system as a whole. Many of these optimizations would have been possible with more time for development as well as time for research.

5.8 Validation

The results of the project were validated in multiple ways. One way this was done was by examining graphs and the difference between the angles given by the hand tracking and the angles predicted by the neural network. Another method of validation involved Unity, where visualization of both the calculated angles and the predicted angles could be shown side by side. This allowed visual comparison of the difference in angles. However, a problem with validating the results this way was the poor performance of the hand tracking, as both forms of validation involved the hand tracking software, this made it difficult to determine whether other aspects of the project were flawed or if the main issue was with the hand tracking.

5.9 Potential future solutions

The ground truth used in the thesis was inconsistent in its tracking of the landmarks. A future solution to explore could be to analyze a data set with specific degrees instead of a data set with a continuous set of angles. The neural network would be

trained on data with set angles, an example being 180, 135, and 90 degrees. This would remove the noise from the ground truth and the network may be able to predict angles in between the fixed increments due to it being a regression model and that the MFD follows a quadratic form connected to the distance shown in Figure 3.3 and Figure 2.6. This could potentially give each joint a wider range of motion by not predicting the mean value, which was the tendency for the dataset gathered in this thesis.

6
Conclusions

In this project we have concluded that Hall Effect sensors have the potential to be used for hand motion tracking. But further research and testing would be required to determine if it is a worthwhile pursuit as a technology. Furthermore the work in this thesis indicate that better gathering techniques for data is required. As shown in the results the MFD is unique for different angles, which leads to the conclusion that a neural network should be able to link the sensor data to the hand tracking angles and thus predict unique angles.

At the end the project did not manage to fully prove that the produced sensor glove could measure all of the 21 DOF of the fingers. But it clearly indicates that it would be possible with better optimization of the cameras (a reliable ground truth) and the neural network.

As for the movement of the thumb results show that the two joints measured gives unique and possible angles, however the solution to measure the opposition of the thumb was not implemented. But no results collected during the project points toward it being impossible to decode this movement using these sensors. The glove used in this project even has the option to add up to two more sensors for this purpose but were in the end never used. Again the reasoning for this was an unreliable ground truth.

The digital visualization of the hand could use a rework to reach its full potential as the open source model used in Unity was sub optimal for the task of using angles for moving the phalanges of the finger, but the model still gave us the desired information. Further, the Unity model proves that it is possible to visualize the fingers' motion in real time using the neural networks output as input for the visual model.

This project also concludes that a neural network has proven useful when dealing with this kind of challenge. It offers a great range of possibilities with it's biggest weakness being the amount of quality data it needs to do its task successfully. Had time not been a constraint, as well as more reliable data, additional work with the network could have led to great results and a more intuitive user experience.

Bibliography

[1] B. Innocenti, "Biomechanics: A fundamental tool with a long history (and even longer future!)" *Muscles, ligaments and tendons journal*, vol. 7, no. 4, p. 491, 2017.

[2] K. D. Seymore, A. C. Fain, N. J. Lobb, and T. N. Brown, "Sex and limb impact biomechanics associated with risk of injury during drop landing with body borne load," *PLoS One*, vol. 14, no. 2, 2019. DOI: 10.1371/journal.pone.0211129.

[3] K. R. Raghunathan, "History of microcontrollers: First 50 years," *IEEE Micro*, vol. 41, no. 6, pp. 97–104, 2021. DOI: 10.1109/MM.2021.3114754.

[4] B. Rivera, C. Cano, I. Luis, and D. A. Elias, "A 3d-printed knee wearable goniometer with a mobile-app interface for measuring range of motion and monitoring activities," *Sensors*, vol. 22, no. 3, p. 763, 2022. DOI: 10.3390/s22030763.

[5] Z. Yue, X. Zhang, J. Wang, *et al.*, "Hand rehabilitation robotics on poststroke motor recovery," *Behavioural neurology*, vol. 2017, 2017. DOI: 10.1155/2017/3908135. (visited on 02/03/2024).

[6] A. L. P. Fernández and S. Ortiz-Perez, "Dressing disability," in *StatPearls [Internet]*, StatPearls Publishing, 2023. (visited on 01/23/2024).

[7] R. Kabir, M. S. H. Sunny, H. U. Ahmed, and M. H. Rahman, "Hand rehabilitation devices: A comprehensive systematic review," *Micromachines*, vol. 13, no. 7, 2022. DOI: 10.3390/mi13071033. (visited on 02/03/2024).

[8] C. E. Proulx, M. Beaulac, M. David, *et al.*, "Review of the effects of soft robotic gloves for activity-based rehabilitation in individuals with reduced hand function and manual dexterity following a neurological event," *Journal of rehabilitation and assistive technologies engineering*, vol. 7, 2020. DOI: 10.1177/2055668320918130. (visited on 02/03/2024).

[9] M. Yasen and S. Jusoh, "A systematic review on hand gesture recognition techniques, challenges and applications," *PeerJ Computer Science*, vol. 5, e218, 2019. (visited on 02/03/2024).

[10] F. Al Farid, N. Hashim, J. Abdullah, *et al.*, "A structured and methodological review on vision-based hand gesture recognition system," *Journal of Imaging*, vol. 8, no. 6, p. 153, 2022. DOI: 10.3390/jimaging8060153. (visited on 01/24/2024).

[11] W. Chen, C. Yu, C. Tu, *et al.*, "A survey on hand pose estimation with wearable sensors and computer-vision-based methods," *Sensors*, vol. 20, no. 4, p. 1074, 2020. DOI: 10.3390/s20041074. (visited on 02/03/2024).

[12] A.A.Krüger, O.Dahlin, M.Dahlström, L.Janbro, O.Moberg, and P.Nässlander and L.Sedin, *"wearable sensor based system for measuring hand motions"*, B.S. Thesis, 2022. (visited on 02/01/2023).

[13] K.Funck, I.Hägeryd, M.Johansson, J.M.Göransson, J.Svensson, and L.Swala, *"wearable sensor based system for measuring hand motions"*, B.S. Thesis, 2023. (visited on 02/01/2023).

[14] B.-S. Lin, I.-J. Lee, S.-Y. Yang, Y.-C. Lo, J. Lee, and J.-L. Chen, "Design of an inertial-sensor-based data glove for hand function evaluation," *Sensors*, vol. 18, no. 5, p. 1545, 2018. DOI: https://doi.org/10.3390/s18051545. (visited on 02/02/2024).

[15] G.Saggio, *"Sensors and Actuators A: Physical"*. Elsevier, 2012, vol. Mechanical model of flex sensors used to sense finger movements. [Online]. Available: https://www.sciencedirect.com/journal/sensors-and-actuators-a-physical (visited on 02/03/2024).

[16] N. Ahmad, R. A. R. Ghazilla, N. M. Khairi, and V. Kasi, *Reviews on various inertial measurement unit (IMU) sensor applications*. EJournal Publishing, 2013, vol. 1, pp. 256–262. DOI: https://doi.org/10.12720/ijsps.1.2.256-262. (visited on 02/03/2024).

[17] L. G. M. R.List and P.Giovanoli, "Assessment of hand function during activities of daily living using motion tracking cameras: A systematic review," *Journal of Engineering in Medicine*, 2019. DOI: https://doi.org/10.1177/09544119198513. (visited on 02/02/2024).

[18] J. H. Hammer and J. Beyerer, "Robust hand tracking in realtime using a single head-mounted rgb camera," pp. 252–261, 2013. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-39330-3_27 (visited on 02/01/2024).

[19] A. Rahman and A. Al-Jumaily, "Design and development of a bilateral therapeutic hand device for stroke rehabilitation," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 405, 2013. (visited on 02/03/2024).

[20] S. Tanrıkulu, Ş. Bekmez, A. Üzümcügil, and G. Leblebicioğlu, "Anatomy and biomechanics of the wrist and hand," in *Sports Injuries: Prevention, Diagnosis, Treatment and Rehabilitation*, M. N. Doral and J. Karlsson, Eds. Springer Berlin Heidelberg, 2015, pp. 441–447. DOI: 10.1007/978-3-642-36569-0_49. [Online]. Available: https://doi.org/10.1007/978-3-642-36569-0_49.

[21] B. H.Aldskogius, *"Den friska människan"*. Liber, 2018, pp. 229–232, ISBN: 978-91-47-10569-4.

[22] I. M. Bullock, J. Borràs, and A. M. Dollar, "Assessing assumptions in kinematic hand models: A review," in *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2012, pp. 139–146. DOI: 10.1109/BioRob.2012.6290879.

[23] I. Komatsu and J. D. Lubahn, "Anatomy and biomechanics of the thumb carpometacarpal joint," *Operative Techniques in Orthopaedics*, vol. 28, no. 1, pp. 1–5, 2018.

[24] CrossFit. "Movements of the hand." (), [Online]. Available: https://www.crossfit.com/essentials/movement-about-joints-part-4-the-hand-and-fingers (visited on 05/09/2024).

[25] I. for Quality and E. in Health Care (IQWiG). “"how do hands work?"." (), [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK279362/ (visited on 05/02/2024).

[26] E. Hall, *"On a New Action of the Magnet on Electric Currents"*. 1879, vol. 2. DOI: 10.2307/2369245. (visited on 02/03/2024).

[27] R. Fitzpatrick, webpage, 2006. [Online]. Available: https://farside.ph.utexas.edu/teaching/em/lectures/node33.html (visited on 02/05/2024).

[28] E. Ramsden, *"Hall-Effect Sensors: Theory and Application"*, 2nd ed. Elsevier, 2006, ISBN: 9780750679343.

[29] public domain. [Online]. Available: https://commons.wikimedia.org/wiki/File:Hall_Effect_Measurement_Setup_for_Holes.png.

[30] T. I. Incorporated, “"introduction to hall-effect sensors",” 2023. [Online]. Available: https://www.ti.com/lit/po/slyt824a/slyt824a.pdf?ts=1706978554660&ref_url=https%253A%252F%252Fwww.google.com%252F (visited on 02/01/2024).

[31] T. I. Incorporated, “Angle measurement with multi-axis linear hall-effect sensors.,” version A, Jan. 2021. [Online]. Available: https://www.ti.com/lit/pdf/sbaa463 (visited on 02/05/2024).

[32] T. Torii, H. Sakamoto, and A. Sato, “Simplified magnetic flux density measurement for local resolution analysis of transcranial magnetic stimulation,” in *2023 IEEE International Magnetic Conference (INTERMAG)*, IEEE, 2023, pp. 1–4. DOI: https://doi.org/10.1109/INTERMAG50591.2023.10265102. (visited on 01/30/2024).

[33] *High-precision 3d linear hall-effect sensor with spi*, TMAG5170, Texas Instruments Incorporated, Sep. 2021. [Online]. Available: https://www.ti.com/lit/gpn/tmag5170 (visited on 02/03/2024).

[34] F. Leens, “An introduction to i 2 c and spi protocols,” *IEEE Instrumentation & Measurement Magazine*, vol. 12, no. 1, pp. 8–13, 2009. DOI: 10.1109/MIM.2009.4762946.

[35] H. Siebeneicher, “Universal asynchronous receiver-transmitter (uart),” [Online]. Available: https://docs.arduino.cc/learn/communication/uart/ (visited on 02/05/2024).

[36] J. JospehŁAwange, *Fundamentals of Photogrammetry. In: Environmental Geoinformatics*, M. N. Doral and J. Karlsson, Eds. Springer Berlin Heidelberg, pp. 157–174, ISBN: 978-3-642-34085-7. DOI: 10.1007/978-3-642-34085-7_11. [Online]. Available: https://doi.org/10.1007/978-3-642-34085-7_11.

[37] D. Young, University of Sussex. (1994), [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/YOUNG/vision5.html (visited on 05/08/2024).

[38] L. Hardesty, *Explained: Neural networks*, 2017. [Online]. Available: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414 (visited on 02/27/2024).

[39] N. Ketkar and J. Moolayil, *Feed-Forward neural networks*. 2021, pp. 93–131. DOI: 10.1007/978-1-4842-5364-9\{_}3. [Online]. Available: https://doi.org/10.1007/978-1-4842-5364-9_3 (visited on 02/27/2024).

[40] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

[41] H. Kukreja, N Bharath, C. Siddesh, and S Kuldeep, "An introduction to artificial neural network," *Int J Adv Res Innov Ideas Educ*, vol. 1, no. 5, pp. 27–30, 2016.

[42] *"bourns® rotary position sensors"*, 3382H-1-103, Bourns. [Online]. Available: https://docs.rs-online.com/f49a/0900766b813ed15c.pdf (visited on 04/03/2024).

[43] Arduino. "Arduino® nano esp32 - arduino official store." (), [Online]. Available: https://store.arduino.cc/collections/boards-modules/products/nano-esp32 (visited on 02/05/2024).

[44] *4-to-16 line decoder/demultiplexer*, Rev. 9, Nexperia, 2021. [Online]. Available: https://assets.nexperia.com/documents/data-sheet/74HC_HCT154.pdf.

[45] *User's guide tmag5170uevm*, SBAU388C, TI. [Online]. Available: https://www.ti.com/lit/pdf/sbau388 (visited on 05/02/2024).

[46] OpenCV. "Opencv." (), [Online]. Available: https://opencv.org/ (visited on 05/08/2024).

[47] OpenCV. "Camera calibration and 3d reconstruction." (), [Online]. Available: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a (visited on 05/08/2024).

[48] CVzone. "Cvzone." (), [Online]. Available: https://github.com/cvzone/cvzone (visited on 05/08/2024).

[49] vgpy. "Vg." (), [Online]. Available: https://vgpy.readthedocs.io/en/latest/ (visited on 05/08/2024).

[50] W. B. T. A. developer platform. "Weights biases: The ai developer platform." (), [Online]. Available: https://wandb.ai/site (visited on 05/02/2024).

[51] Unity. "Xr hands." (), [Online]. Available: https://docs.unity3d.com/Packages/com.unity.xr.hands@1.4/manual/index.html (visited on 05/10/2024).

[52] micro ROS. "Micro-ros." (), [Online]. Available: https://micro.ros.org/ (visited on 05/09/2024).

[53] multi cb. "Rigid-flex pcbs." (), [Online]. Available: https://www.multi-circuit-boards.eu/en/products/printed-circuit-boards/rigid-flex-pcb.html (visited on 05/09/2024).

[54] KiCad Developers Team, *Kicad*, version 7.0.10, 2023. [Online]. Available: https://www.kicad.org/.

[55] P. Ferreira, $Arduino_nano_pads$, https://github.com/pedronf65/KiCadShare, 2023.

A

Magnetic test results

Following are some more visualizations of test data using the test rig. These magnets were not used for the final product. Figure A.1 is for the 970g magnet. Figure A.2 is for the 1.1kg magnet. Figure A.3 is for the 1.7kg magnet. No test were concluded for the 3.2kg magnet as it was early in the project considered non viable for our uses.

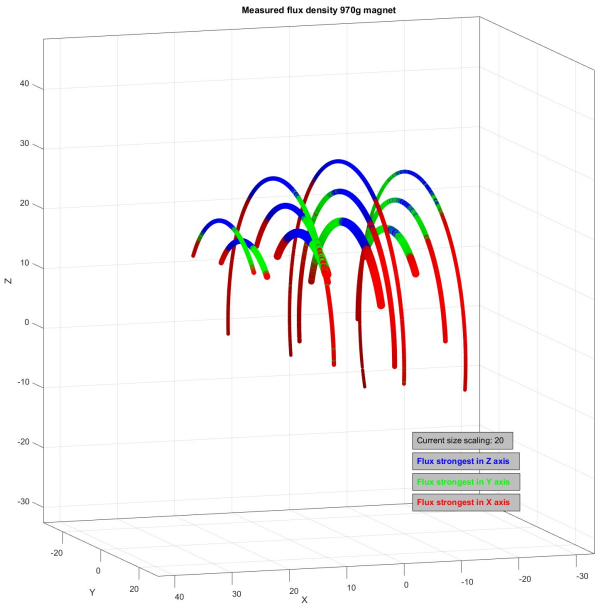


Figure A.1: Visualization of 540g magnet measured values with sizes scaled to 20 times larger.

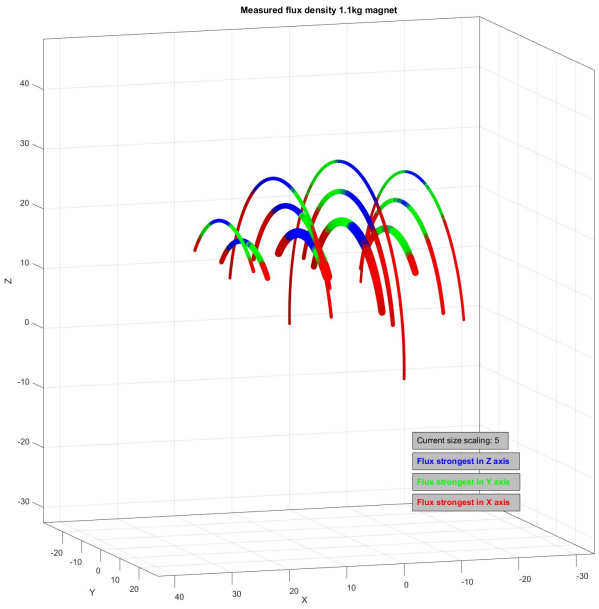


Figure A.2: Visualization of 1.1kg magnet measured values with sizes scaled to 5 times larger.

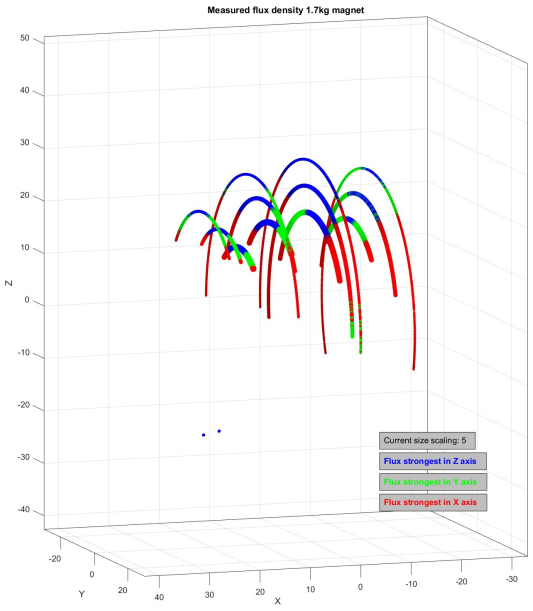
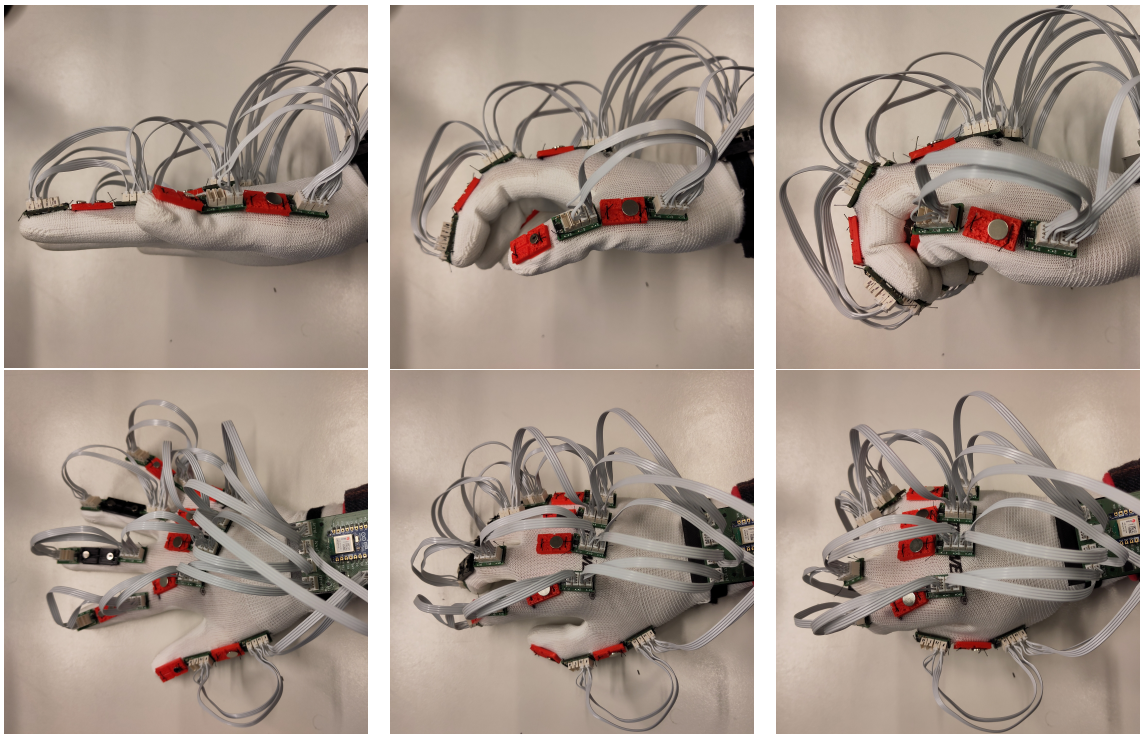


Figure A.3: Visualization of 1.7g magnet measured values with sizes scaled to 5 times larger.

B

Sensor data

Analysis of the raw sensor data was done using three different finger positions shown below in Figure B.1. In each position the sensors were sampled for a few seconds, resulting in approximately 100 MFD readings for each axis, for each sensor.
This data was visualized in histogram matrices for each finger where the first, second, and third positions are coloured as blue, cyan, and green. The rows of the matrix represent each sensor at a certain joint, with the columns representing the three axis of measurement.
In each cell, the largest difference is marked by a red line. This highlights the largest difference in the mean values for each position.



(a) Position 1 **(b)** Position 2 **(c)** Position 3

Figure B.1: Finger positions for the three snapshots used for raw data analysis

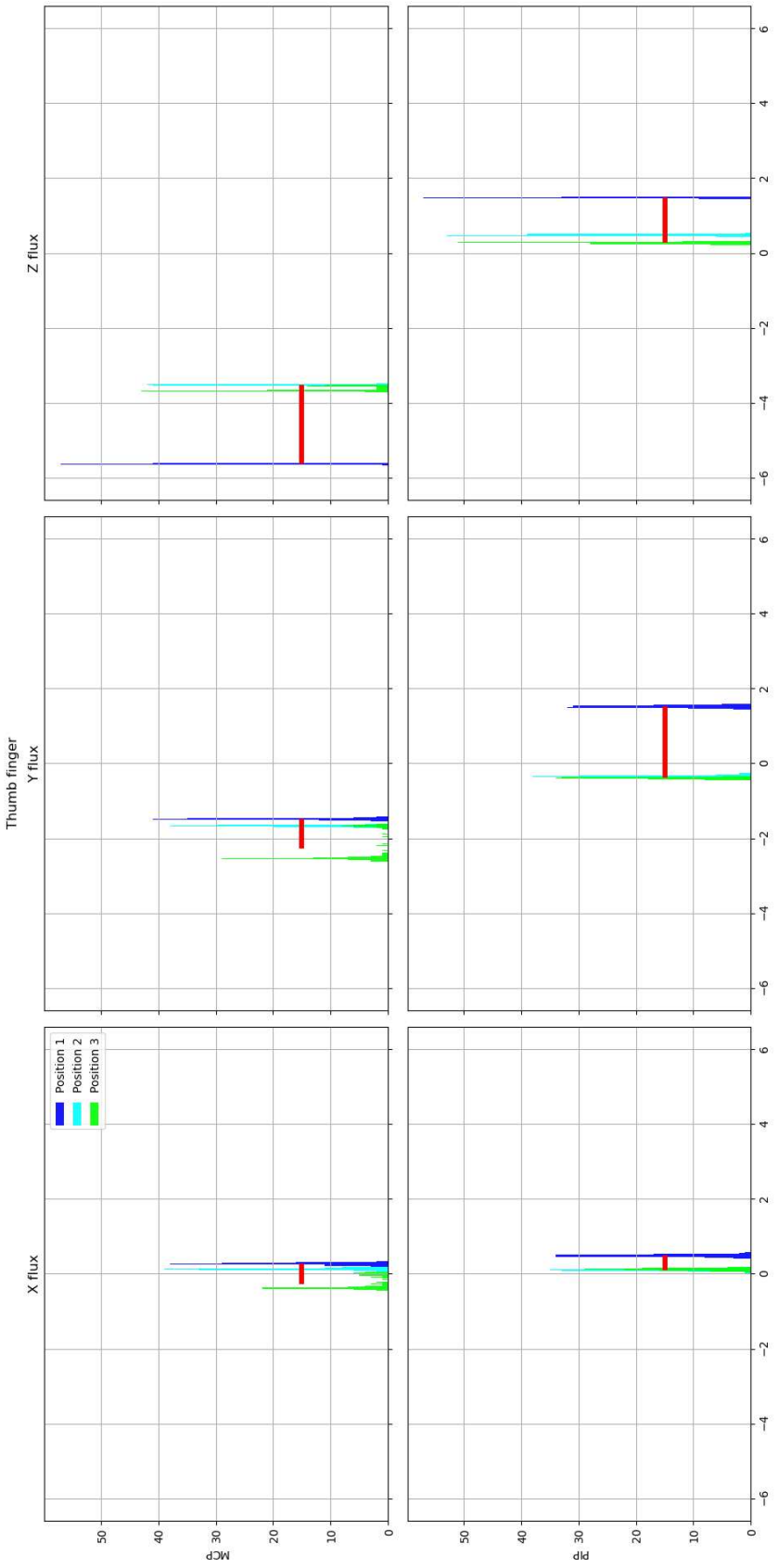


Figure B.2: Thumb

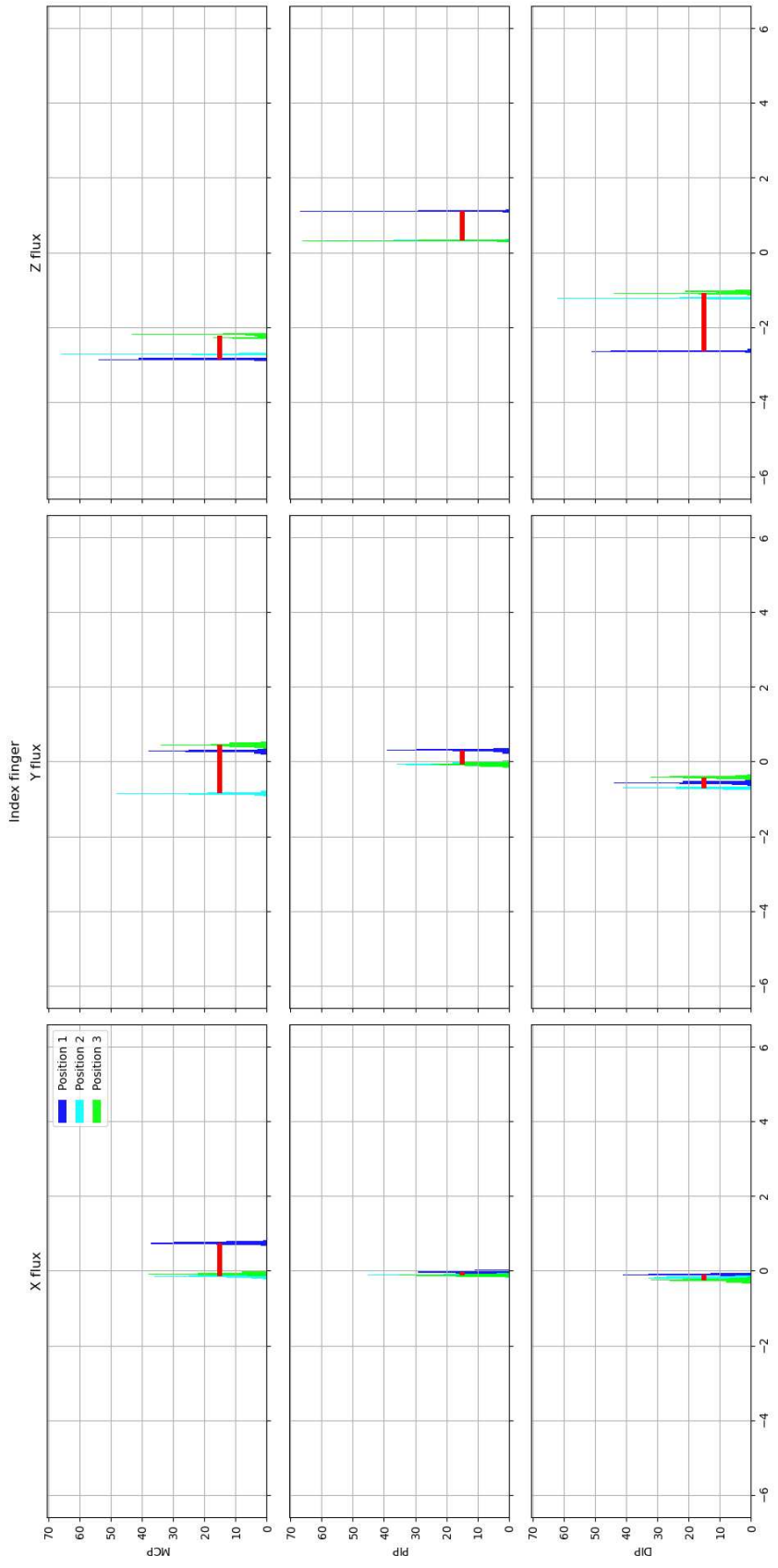


Figure B.3: Index finger

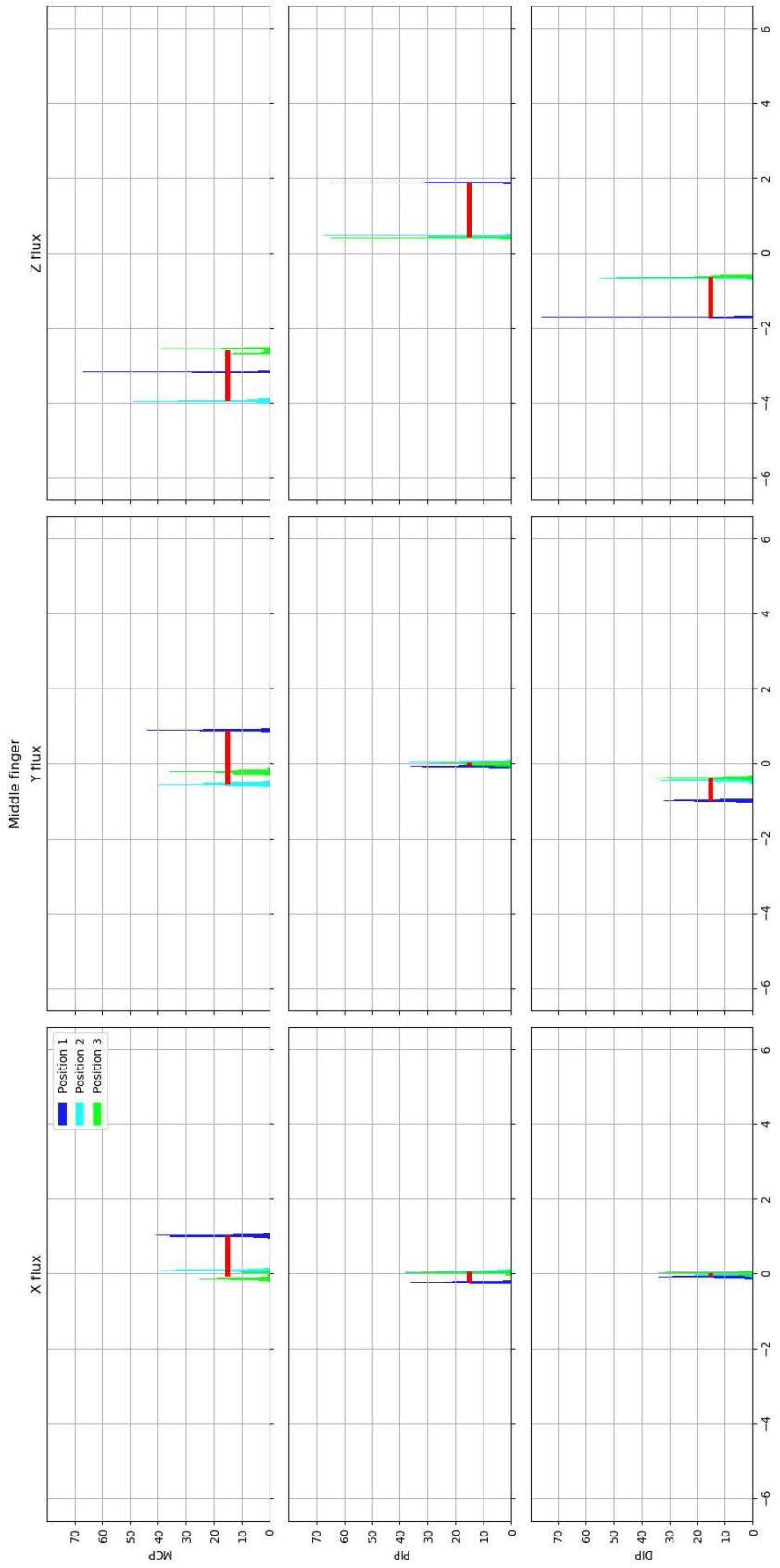


Figure B.4: Middle finger

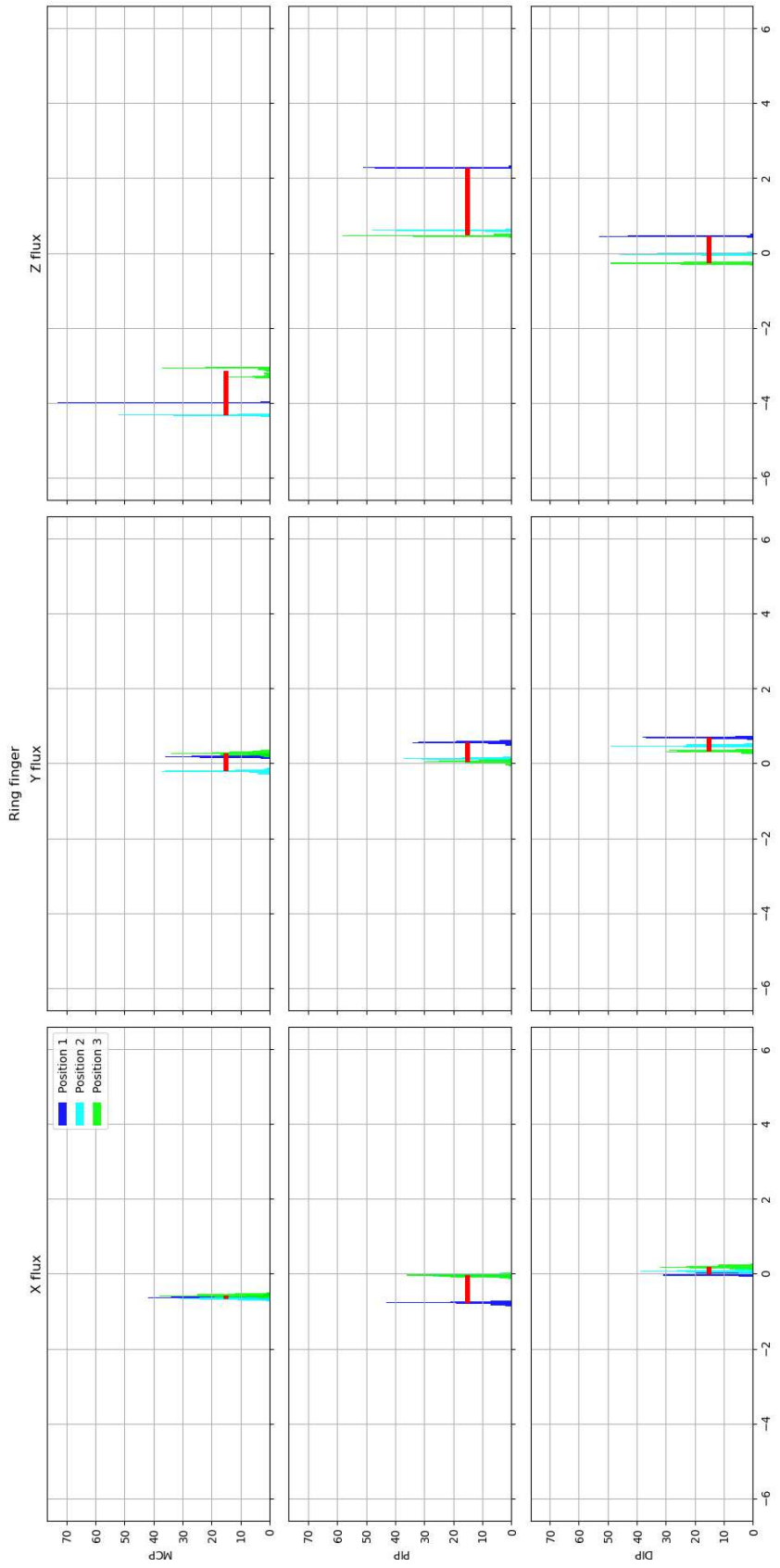


Figure B.5: Ring finger

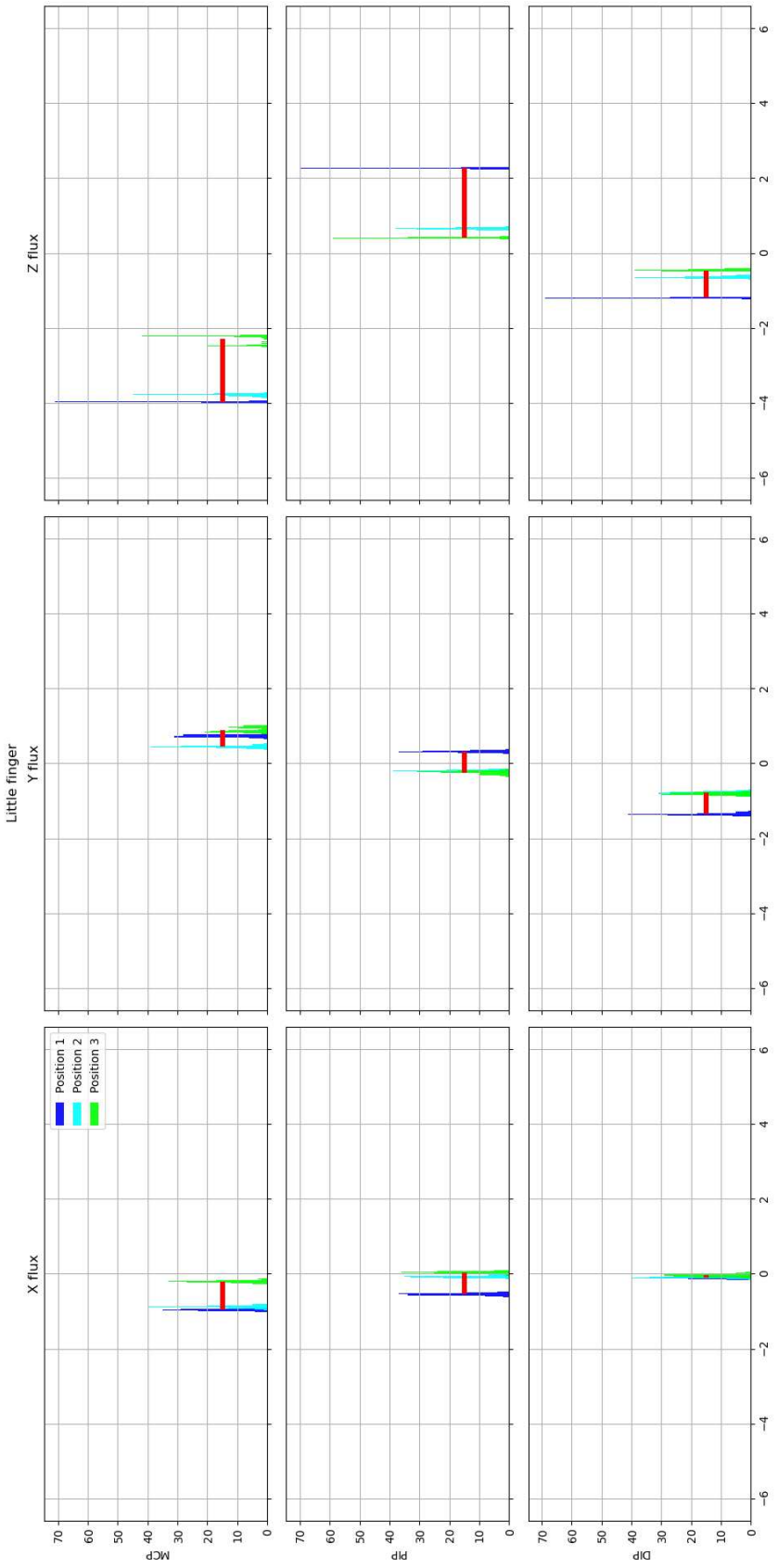


Figure B.6: Little finger

Position 1			
Sensor	**X**	**Y**	**Z**
1	-0.944311	0.752890	-3.953676
2	-0.523369	0.333852	2.292841
3	-0.083322	-1.334512	-1.182564
4	-0.600927	0.209192	-3.976695
5	-0.757075	0.581021	2.304200
6	0.007298	0.714065	0.478147
7	1.035694	0.899953	-3.137939
8	-0.210024	-0.063940	1.890202
9	-0.048358	-0.967885	-1.694774
10	0.765498	0.298302	-2.834867
11	-0.003452	0.325436	1.122446
12	-0.074074	-0.541679	-2.618300
13	0.288468	-1.468027	-5.618355
14	0.505752	1.537832	1.505133

Table B.1: Position 1, mean values

Position 2			
Sensor	X	**Y**	**Z**
1	-0.858276	0.468738	-3.766332
2	-0.053290	-0.180177	0.676049
3	-0.069790	-0.767116	-0.627922
4	-0.638627	-0.191220	-4.315625
5	-0.013956	0.149097	0.621056
6	0.085434	0.492790	-0.006974
7	0.116599	-0.529295	-3.934201
8	0.072186	0.050662	0.485021
9	0.015428	-0.433018	-0.650017
10	-0.123049	-0.830155	-2.704836
11	-0.086451	-0.043780	0.334437
12	-0.156187	-0.687008	-1.210084
13	0.151008	-1.643418	-3.508651
14	0.122332	-0.320011	0.500419

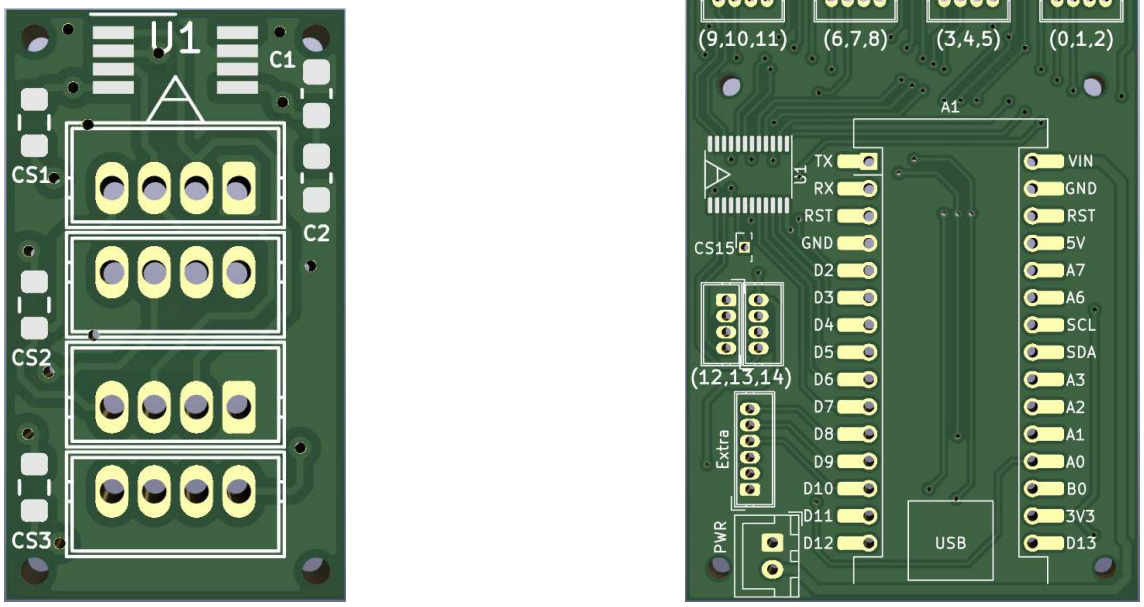
Table B.2: Position 2, mean values

Position 3			
Sensor	$\bar{X}$	**Y**	**Z**
1	-0.188716	0.909709	-2.282129
2	0.057744	-0.220890	0.428533
3	-0.022472	-0.792910	-0.441395
4	-0.557269	0.287104	-3.131188
5	-0.028614	0.058700	0.485414
6	0.205786	0.345134	-0.252294
7	-0.058191	-0.205847	-2.575599
8	0.045545	0.006574	0.426830
9	0.036806	-0.363421	-0.614182
10	-0.056773	0.466310	-2.205465
11	-0.095668	-0.045699	0.330885
12	-0.230061	-0.397553	-1.056702
13	-0.244726	-2.257476	-3.618090
14	0.142171	-0.369590	0.292735

Table B.3: Position 3, mean values

C
PCB design

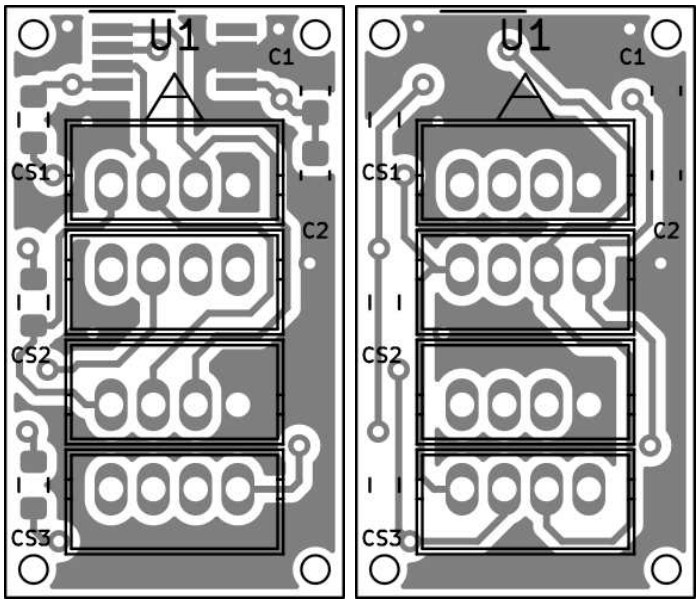
What follows is an overview of the designed PCBs used in this project.
KiCad was used for designing the PCBs[54]. In addition, CircuitLab was used for creating some initial circuit diagrams as a support during development.
As described in Section 3.1.5 the design of the sensor PCB (Figure C.1a was inspired by the TMAG5170 evaluation module from Texas Instruments[45]. On this PCB, the U1 marks the pads for the TMAG5170 Hall Effect sensor. C1 and C2 mark the location of capacitors in the 1 to 0.1 μF range for noise suppression between GND and 3.3 V. CS 1 through 3 are the different jumper pads for selecting which chip select wire to connect to the sensor.
The design for the main PCB in Figure C.1b used a footprint for Arduino Nano created by Pedro Ferreira[55]. The U1 here marks the location of the decoder.
Not used in the final design was the through hole connections marked Extra and CS15, which was added to the board to allow for additional functionality and a 16th Hall Effect sensor. The two through hole connectors marked PWR would supply the entire device with power by connecting a 9V battery, in case a wireless solution was implemented.
Figure C.3 displays the top and bottom copper layers of each circuit, with the descriptive silkscreen top layer being superimposed on both sides for clarity.



(a) Sensor PCB, with dimensions 12*21 mm

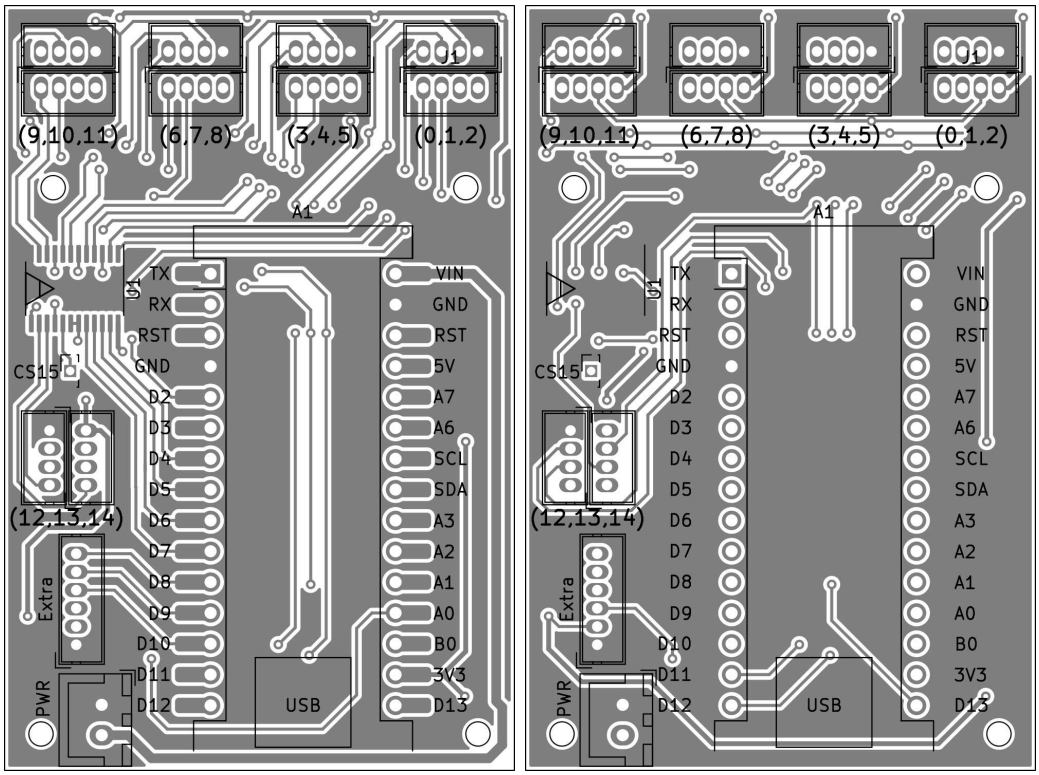
(b) Main PCB, with dimensions 42*63 mm

Figure C.1: 3D render of the designed PCBs, made in KiCad's 3D Viewer



(a) Front **(b)** Back

Figure C.2: Sensor PCB



(a) Front **(b)** Back

Figure C.3: Main PCB

D

Microcontroller code snippets

Following are a few code snippets from the code running on the Arduino.

Listing D.1: Task for gathering values from sensors.

```
void TaskSensorRead(void *pvParameters) {    // This is a task.
  for (;;) {                    // A Task shall never return or exit.
    xSemaphoreTake(sensorsMutex, portMAX_DELAY); // Take sensors mutex
    for (byte i = 0; i < number_of_sensors; i++) {// Go through all the
        sensor objects and gather data
      sensors[i].getFluxDensityInAxis('X');
      sensors[i].getFluxDensityInAxis('Y');
      sensors[i].getFluxDensityInAxis('Z');
    }
    xSemaphoreGive(sensorsMutex); // Release sensors mutex
    delay(1);                     // Delay to stop watchdog protocol
  }
}
```

Listing D.2: Task for sending saved values to computer.

```
void TaskSerialSend(void *pvParameters) {        // This is a task.
  for (;;) {
      xSemaphoreTake(sensorsMutex, portMAX_DELAY); // Take sensors mutex
      memcpy(sensorsCopy, sensors, sizeof(sensors)); // Copy all sensors
          current values to a secondary array
      xSemaphoreGive(sensorsMutex);              // Release sensors mutex
      for (byte i = 0; i < number_of_sensors; i++) { // Go through all
          the copied sensor objects and print their data
        Serial.print(sensorsCopy[i].getSavedFluxDensityX());
        Serial.print(",");
        Serial.print(sensorsCopy[i].getSavedFluxDensityY());
        Serial.print(",");
        Serial.print(sensorsCopy[i].getSavedFluxDensityZ());
        Serial.println(",");
      }
  }
}
```

Listing D.3: Structure of the HallSensor class.

```
class HallSensor {
private:
  byte chipSelectPin; //chipSelectPin: any digital pin
  byte range;        //range: 50/25/100 mT
  bool multiplexerSetting; //dictates useage of multiplexer
  float fluxDensityX;
  float fluxDensityY;
  float fluxDensityZ;
  float angledata;
  signed short tempdata;
  byte pins[4] = {4,5,6,7}; //an array that more or less acts as a port
     for multiplexer
public:
  HallSensor(byte chipSelectPin, byte range, bool multiPlexerSetting);
  void setup();
  unsigned int readRegister(byte thisRegister, int16_t thisValue, byte
     thisCommand);
  void writeRegister(byte thisRegister, int16_t thisValue, byte
     thisCommand);
  signed short getTempdata();
  float getFluxDensityInAxis(unsigned char axis);
  float getAngledata();
  byte getChipSelectPin();
  byte getRange();
  float getSavedFluxDensityX();
  float getSavedFluxDensityY();
  float getSavedFluxDensityZ();
  float getSavedAngledata();
  signed short getSavedTempdata();
  void pinSelect(bool high);
};
```

E

Network graphs

Below follows some further visualizations of the behaviour of the neural network at work. Where the green lines are the actual angle values, e.g., the output of the neural network. The blue lines represent the predicted angle values from the sensor data given by the neural network.

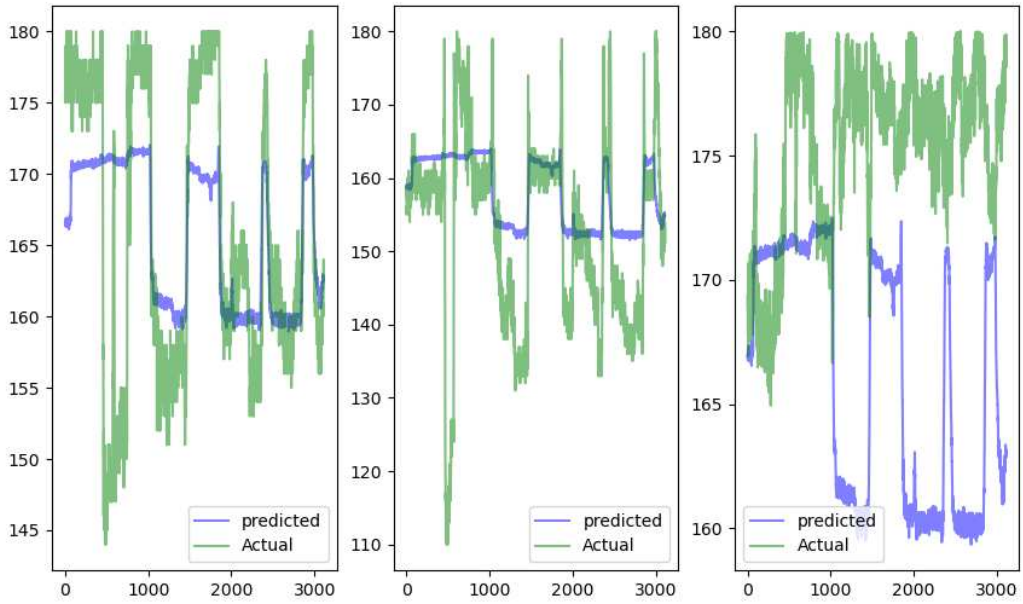


Figure E.1: Little finger

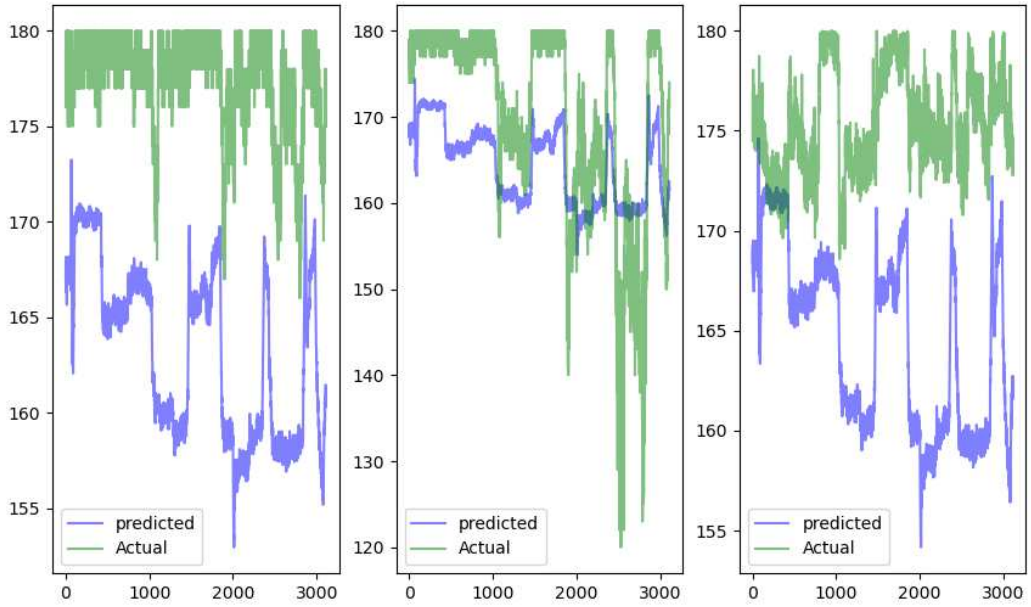


Figure E.2: Middle finger

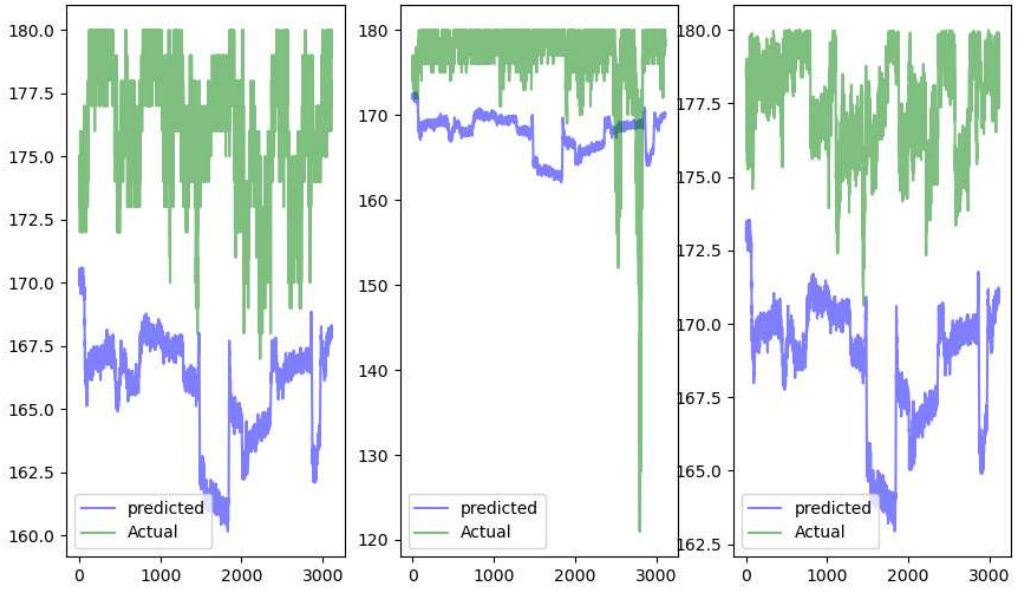


Figure E.3: Ring finger

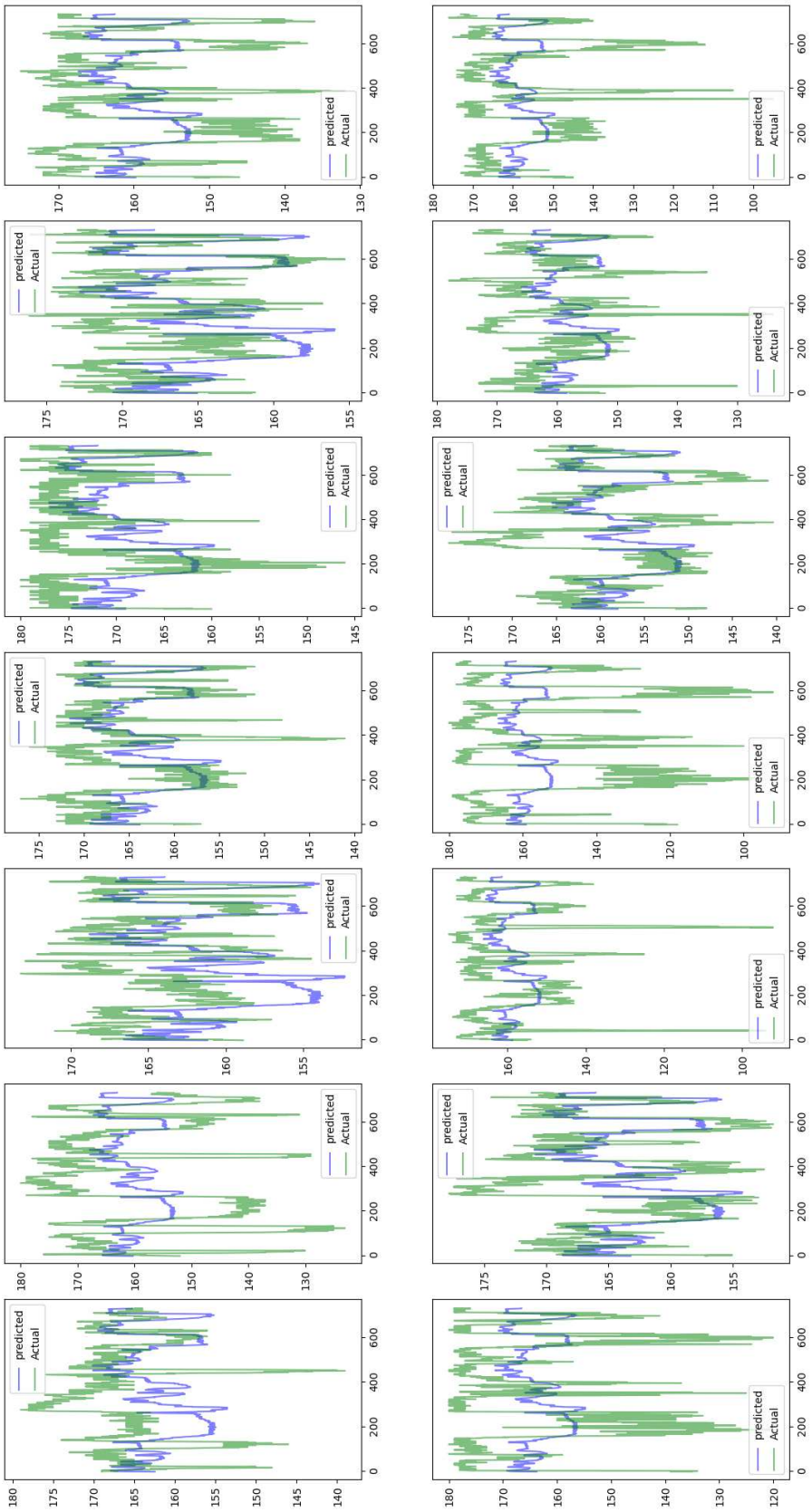


Figure E.4: All the predicted angles for each sensor when training on the same network.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS