

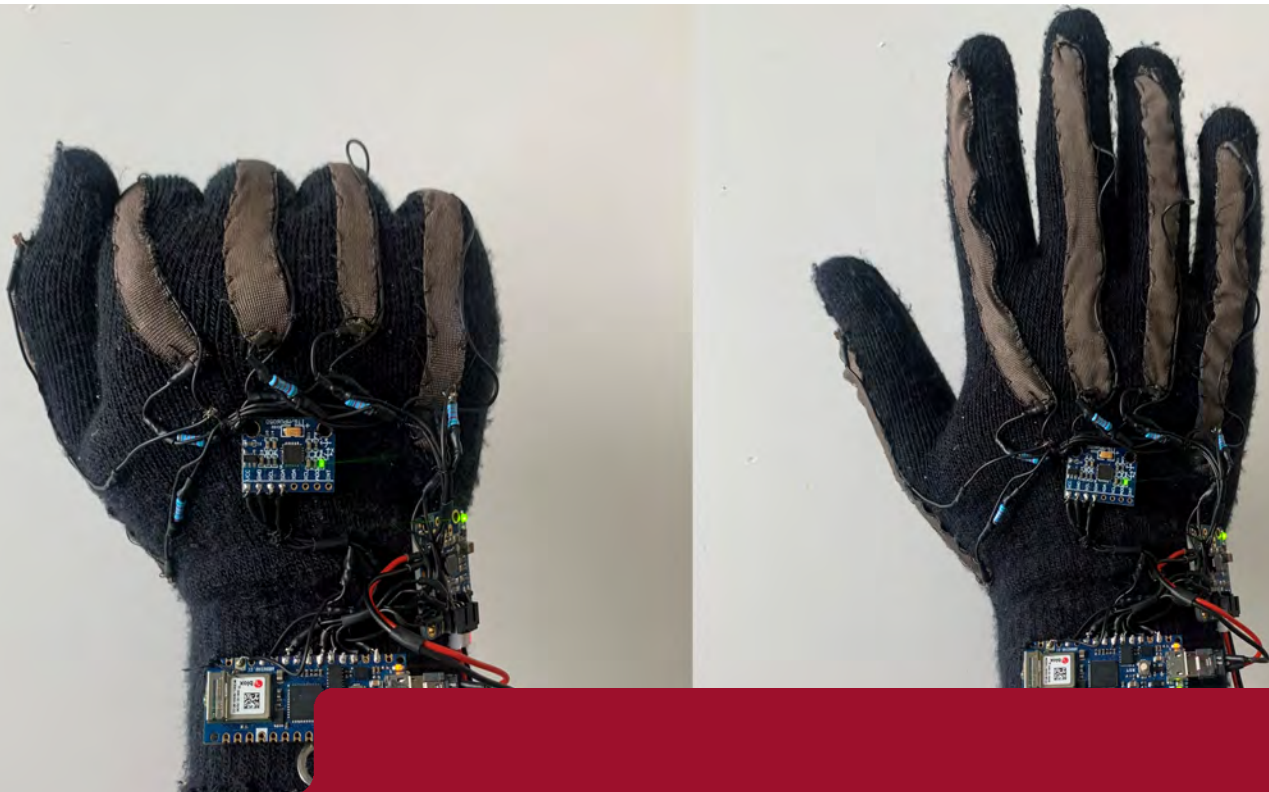


DEGREE PROJECT IN TECHNOLOGY,  
FIRST CYCLE, 15 CREDITS  
*STOCKHOLM, SWEDEN 2021*

# Sign Language Translation

**PIERRE SINANDER**

**TOMAS ISSA**



**KTH ROYAL INSTITUTE OF TECHNOLOGY  
SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT**





# **Sign Language Translation**

Pierre Sinander  
Tomas Issa

Bachelor's Thesis at ITM  
Supervisor: Nihad Subasic  
Examiner: Nihad Subasic

TRITA-ITM-EX 2021:38

# Abstract

The purpose of the thesis was to create a data glove that can translate ASL by reading the finger- and hand movements. Furthermore, the applicability of conductive fabric as stretch sensors was explored. To read the hand gestures stretch sensors constructed from conductive fabric were attached to each finger of the glove to distinguish how much they were bent. The hand movements were registered using a 3-axis accelerometer which was mounted on the glove. The sensor values were read by an Arduino Nano 33 IoT mounted to the wrist of the glove which processed the readings and translated them into the corresponding sign. The microcontroller would then wirelessly transmit the result to another device through Bluetooth Low Energy.

The glove was able to correctly translate all the signs of the ASL alphabet with an average accuracy of 93%. It was found that signs with small differences in hand gestures such as S and T were harder to distinguish between which would result in an accuracy of 70% for these specific signs.

**Keywords** – Mechatronics, data glove, American Sign Language, stretch sensors, conductive fabric.

# Sammanfattning

Syftet med uppsatsen var att skapa en datahandske som kan översätta ASL genom att läsa av finger- och handrörelser. Vidare undersöktes om ledande tyg kan användas som sträcksensorer. För att läsa av handgesterna fästes ledande tyg på varje finger på handsken för att urskilja hur mycket de böjdes. Handrörelserna registrerades med en 3-axlig accelerometer som var monterad på handsken. Sensorvärdena lästes av en Arduino Nano 33 IoT monterad på handleden som översatte till de motsvarande tecknen. Mikrokontrollern överförde sedan resultatet trådlöst till en annan enhet via Bluetooth Low Energy.

Handsken kunde korrekt översätta alla tecken på ASL-alfabetet med en genomsnittlig exakthet på 93%. Det visade sig att tecken med små skillnader i handgester som S och T var svårare att skilja mellan vilket resulterade i en noggrannhet på 70% för dessa specifika tecken.

**Nyckelord** – Mekanik, datahandske, teckenspråk, sträcksensorer, elektriskt ledande tyg.

# Acknowledgements

We would like to thank Pedro Duque and PLUX Wireless Biosignals for providing us with a sample of their conductive textile. Their contribution played a major part in the success of our thesis and for that we will always be grateful. We would also like to thank our examiner Nihad Subasic for his support and feedback during this project as well as Staffan Qvarnström for providing us with the electrical components.

# Contents

<b>1 Introduction</b> .....	1
1.1 Background.....	1
1.2 Purpose.....	1
1.3 Scope.....	1
1.4 Method.....	2
<b>2 Theoretical Background</b> .....	3
2.1 Sensors.....	3
2.1.1 Flex sensor.....	3
2.1.2 Stretch sensor.....	3
2.2 Microcontroller.....	4
2.3 MPU6050.....	4
2.4 Lithium polymer battery.....	4
2.5 DC-DC Converter.....	4
<b>3 Demonstrator</b> .....	6
3.1 Hardware.....	6
3.1.1 Stretch sensor.....	7
3.1.2 Power Supply.....	10
3.1.3 Accelerometer.....	11
3.2 Software.....	11
3.2.1 Stretch Sensor.....	12
3.2.2 Accelerometer.....	14
3.2.3 Value Interpretation.....	15
<b>4 Results</b> .....	18
4.1 Final construction.....	18
4.2 Sensor Evaluation.....	18
4.3 Application Testing.....	19
<b>5 Discussion</b> .....	21
5.1 Hardware.....	21
5.2 Software.....	21

5.3 Results.....	22
5.4 Conclusion .....	22
5.5 Future work.....	22
<b>Bibliography</b> .....	<b>24</b>
<b>Appendices</b> .....	<b>1</b>
Appendix A.....	1
Sign Language chart .....	1
Appendix B.....	1
MPU6050 datasheet.....	1
Appendix C.....	1
Nina-W10 series data sheet.....	1
Appendix D.....	1
SAM D21 Family data sheet.....	1
Appendix E.....	1
Arduino code.....	1
Appendix F .....	1
Arduino header file .....	1
Appendix G.....	1
Acumen simulation .....	1



# List of Figures

<b>Figure 3.1.</b> Electrical circuit of the glove. Drawn in Circuit Diagram [15]. .....	7
<b>Figure 3.2.</b> Theoretical maximum voltage difference as a function of resistor value. Plotted in MATLAB [13]. .....	8
<b>Figure 3.3.</b> Voltage difference plotted against current draw. Plotted in MATLAB.....	10
<b>Figure 3.4.</b> Flow chart of program logic. Drawn in Lucid Chart [17]. .....	11
<b>Figure 3.5.</b> Voltage difference over time measured by the microcontroller. Plotted in MATLAB.....	12
<b>Figure 3.6.</b> Difference between median and raw sensor readings. Plotted in MATLAB.	13
<b>Figure 3.7.</b> Boundary value compared to median value. Plotted in MATLAB. ....	14
<b>Figure 3.8.</b> Number of accepted readings plotted against the error margin for every stretch sensor. Plotted in MATLAB. ....	16
<b>Figure 4.1.</b> Image of final construction. ....	18
<b>Figure 4.2.</b> Test results from sensor evaluation. Plotted in MATLAB. ....	19
<b>Figure 4.3.</b> Test result from application testing. Plotted in MATLAB. ....	20

# List of Tables

<b>Table 3.1.</b> List of components and quantity. Drawn in Microsoft Word. ....	6
<b>Table 3.2.</b> Component and the estimated current drawn as per the corresponding datasheet. The datasheets can be found in appendices B to D. Drawn in Microsoft word. ....	9
<b>Table 3.3.</b> Comparison of sensor values between the letters S and T. Drawn in Microsoft Word. ....	16

# List of abbreviations

ASL	American Sign Language
BLE	Bluetooth Low Energy
CSD	Communication Service of the Deaf
CPU	Central Processing Unit
DC	Direct Current
DOF	Degrees of Freedom
IMU	Inertial Measurement Unit
IoT	Internet of Things
LiPo	Lithium Polymer
ML	Machine Learning
MPU	Motion Processing Unit
RAM	Random Access Memory
UCLA	The University of California, Los Angeles

# 1 Introduction

This chapter aims to introduce the reader to the project, why the project was chosen, which questions the project aims to answer and in what scope. Since the project discusses the differences between different signs in depth it is recommended that the reader familiarizes themselves with the ASL alphabet which can be found in appendix A.

## 1.1 Background

There is no doubt that lacking the ability to communicate with people on an everyday basis can lead to critical situations socially but can also lead to employment issues. As stated by the Communication Service for the Deaf (CSD), 72% of families do not sign with their deaf children and 70% of deaf people do not work or are unemployed because of their lack in communication. Approximately 1 million people in USA use American Sign Language (ASL) as their main way of communication according to CSD [1].

The idea is therefore to create a glove that can translate ASL to speech, and in that way try to increase the communication possibilities and skills for the people who uses ASL. By working on this project, our hope is to raise awareness to the current problems and try to contribute to a solution.

## 1.2 Purpose

The goal is to construct a glove that can translate ASL to speech by reading finger- and hand movements. The aim is to help people who cannot use their voice to communicate and include them even further in everyday conversations and in that way increase their participation in society.

To succeed in constructing a glove that can translate ASL by reading the finger and hand movements, the following questions at issue must be investigated.

- How can electrically conductive fabric be applied as stretch sensors?
- How can one efficiently read the finger and hand movements and translate ASL to speech with high accuracy?

## 1.3 Scope

This project will focus on translating the American Sign Language alphabet and not the entire language. By translating the alphabet, the project hopes to prove the applicability of the glove for translating the entire language.

## **1.4 Method**

The construction of the glove takes inspiration from a project conducted by students at Cornell University which developed a glove that can translate ASL alphabet using flex sensors on the fingers [2]. The flex sensors are replaced with conductive textile strips to explore the applicability of conductive textiles as stretch sensors. If these sensors outperform the flex sensors, the improvement will tackle both questions at issue since it is more efficient and leads to a smoother design of the glove. An accelerometer is mounted to the back of the glove to register the hand angle and movements which are transmitted to a microcontroller. The microcontroller will read the values from the stretch sensors and the accelerometer and translate them into the corresponding signs which then are transmitted wirelessly over Bluetooth Low Energy to an external device.

## 2 Theoretical Background

The main problem presented when translating sign language is to accurately record the users hand movements through a data glove. The data glove needs to both be accurate in its readings of the hand movements as well as non-obstructive in everyday use. This project will therefore focus on innovating upon previous designs and implementing better solutions.

### 2.1 Sensors

To register the hand movements a variety of sensors will be mounted to the glove. This chapter aims to clarify the purpose of each sensor as well as its implementation on the glove.

#### 2.1.1 Flex sensor

It is crucial to reliably be able to distinguish how much each finger is bent. This can be achieved by mounting flex sensors to each finger. The flex sensors are constructed with rubber which changes resistivity when stretched or compressed [18]. By measuring the change in resistance, it is possible to determine how much the sensor and by extension the finger is bent. Flex sensors applicability in sign language translating gloves has been proved by students at Cornell who constructed a glove using commercially available flex sensors [2]. The disadvantage with these sensors is their rigidity and stiffness which may result in fatigue over prolonged periods of use.

#### 2.1.2 Stretch sensor

An alternative to using flex sensors is stretch sensors. Stretch sensors have an increased resistivity when stretched and by measuring the change in resistivity the elongation can be determined. Therefore, by measuring the elongation of the stretch sensor, it is possible to determine how much a finger is bent. The advantage of using stretch sensors over flex sensors is their flexibility and stretchability, which makes them suitable to use over prolonged periods of time.

Using stretchable sensors in sign language translation have been proved to work in a study made by researchers at UCLA [3]. An accuracy of 98% when translating sign language was achieved, whilst drastically reducing the size of the glove and therefore demonstrated the applicability of conductive textiles as a stretch sensor. The project conducted at UCLA did not apply a commercially available textile and thus a replacement needs to be found. In a paper exploring the applicability of a conductive textile from EeonTex [12] it was concluded that the fabric was suitable for applications where human joint-movement needs to be monitored suggesting that a similar textile can be used in this project.

It should be noted though that if the EeonTex textile is exposed to sweat its characteristics and behaviour will change [11]. Considering that the sensors will be mounted to the fingers this should not be an issue.

## **2.2 Microcontroller**

A microcontroller is a form of a miniature computer. It consists of several devices such as a CPU, some RAM and input and output devices. The CPU (central processing unit) in the microcontroller is used to execute programs, the RAM (random-access memory) is used to store variables and lastly the input and output devices are used to communicate with people or other electronic devices [4].

There are many kinds of microcontrollers out in the market that are used to execute programs or work with other electronic devices in different constructions. The microcontroller which is used for the sign language glove is called Arduino Nano 33 IoT. The Arduino Nano 33 IoT has 22 pins which makes it possible to send and receive information to from different kinds of components, such as the sensors that are used on the sign language glove [5]. The Arduino Nano 33 IoT was implemented due to its smaller footprint as well its capability to connect wirelessly over Wi-Fi or Bluetooth Low Energy (BLE).

## **2.3 MPU6050**

The MPU6050 is a 6 DOF MPU consisting of a gyroscope and accelerometer, which makes it possible to measure acceleration and change in angular velocity in 3 axes [6]. The information from the accelerometer alone is enough to determine the exact orientation of the glove, which is crucial when translating ASL, since hand orientation is a big part of the language.

## **2.4 Lithium polymer battery**

Lithium Polymer batteries are rechargeable batteries. The main advantages of a LiPo battery are their weight and energy density compared to other battery variants. Furthermore, LiPo batteries are common and commercially available in many sizes making them easy to integrate into the project. LiPo batteries often have a voltage of 3.7V meaning that a DC-DC boost converter is needed to be able to power the microcontroller [7].

## **2.5 DC-DC Converter**

A dc-dc converter is an electronic device that converts voltage from a source to either higher or lower voltage, depending on what is needed. The dc-dc converter consists of inductors, transformers, and capacitors, and by using high-frequency switching, the device can store the input voltage momentarily and then release it to get the desired voltage output [8].

The boost converter used in this project is called PowerBoost 500. It is a dc-dc converter which can be used on batteries from  $1.8V$  and higher to boost it to  $5.2V$ . The voltage output is set to  $5.2V$ , where the extra  $0.2V$  is for the internal resistance in the cables and devices used. Even though the Arduino Nano 33 IoT has a maximum capacity of  $5V$ , the  $5.2V$  is safe to use [9].



# 3 Demonstrator

Chapter three aims to introduce the hardware and software implementations of each component.

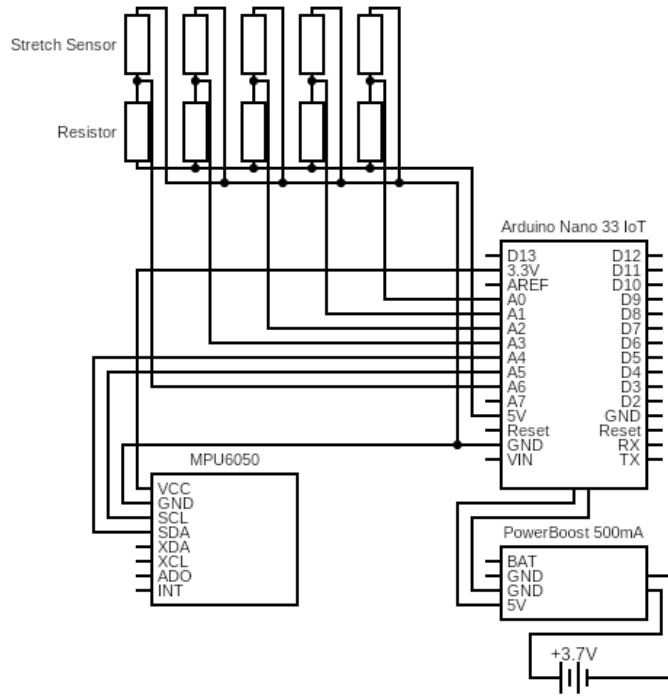
## 3.1 Hardware

Stretch sensors will be fitted to each finger and an accelerometer will be mounted to the back of the hand. The processing of data will be done by an Arduino Nano 33 IoT powered by a battery and a DC-DC converter. The used components are listed in table 3.1.

<b>Component</b>	<b>Model</b>	<b>Quantity</b>
Microcontroller	<i>Arduino Nano 33 IoT</i>	<i>1</i>
Resistor	<i>52.3Ω 0.25W Resistor</i>	<i>5</i>
Conductive Fabric strip	<i>PLUX Conductive Lycra Fabric</i>	<i>5</i>
Battery	<i>LiPo Battery 3.7V 400mAh</i>	<i>1</i>
DC-DC converter	<i>PowerBoost 500 Basic</i>	<i>1</i>
Accelerometer	<i>MPU6050</i>	<i>1</i>

*Table 3.1. List of components and quantity. Drawn in Microsoft Word.*

The stretch sensors are connected to the analog pins of the microcontroller and powered by the 5V output pin. To communicate through the I<sup>2</sup>C protocol the accelerometer is connected to pin A4 and A5 and powered by the 3.3V pin. The LiPo battery is connected to a DC-DC converter which boosts the voltage from 3.7V to 5.2V which is input to the microcontroller through mini-USB. The electrical circuit can be seen in figure 3.1.



**Figure 3.1.** Electrical circuit of the glove. Drawn in Circuit Diagram [15].

### 3.1.1 Stretch sensor

When stretched the electrical fabric changes resistance. Since the microcontroller cannot read the resistance over the sensor it will instead measure the resulting change in voltage. To accomplish this the microcontroller and sensor will be wired according to figure 3.1.

The total voltage  $U$  is the sum of the voltage over the sensor and the resistor. The voltage can be derived from the following formula.

$$U = RI \tag{3.1}$$

Since the resistor and the sensor are paired in series the current flowing through each respective component is the same. The current can be determined by dividing both sides of equation (3.1) with the resistance  $R$ .

$$I = \frac{U}{R} \tag{3.2}$$

The total resistance of the circuit is equal to the sum of the resistance in both the sensor and resistor according to

$$R = R_{Sensor} + R_{Resistor} \tag{3.3}$$

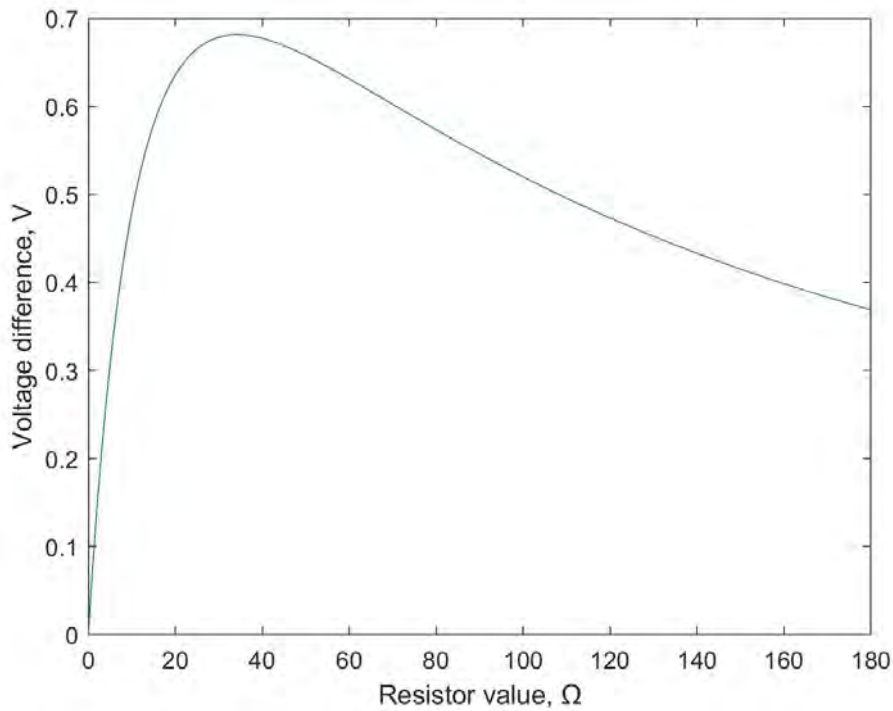
and by combining (3.1) through (3.3) the voltage over the sensor can be obtained from

$$U_{Sensor} = U \left( 1 - \frac{R_{Resistor}}{R_{Sensor} - R_{Resistor}} \right) \quad (3.4)$$

It is evident from (3.4) that the voltage over the sensor is dependent on the resistor value. Therefore, it is advantageous to choose a resistance such that the voltage difference over the sensor is as large as possible. By measuring the maximum and minimum resistance of the stretch sensor the difference in voltage can be derived from (3.4) according to

$$\Delta U = UR_{Resistance} \left( \frac{1}{R_{min} + R_{Resistor}} - \frac{1}{R_{max} + R_{Resistor}} \right) \quad (3.5)$$

The sensor resistance was obtained from a multimeter and to determine the optimal resistor value the equation is visualised in a plot with resistor values ranging from 0-180Ω.



**Figure 3.2.** Theoretical maximum voltage difference as a function of resistor value. Plotted in MATLAB [13].

The maximum voltage difference is obtained from the graph in figure 3.2 to 0.68V with a resistor value of 34Ω.

The previous calculations do not take the power draw of the sensor circuitry into consideration. The Arduino Nano 33 IoT is limited to a maximum input current of

500mA through the USB port which must be distributed between the stretch sensors as well as the microcontroller. To determine the maximum current that can be allowed to pass through the sensors the power draw from the manufacturer's datasheet is compiled for all the components in table 3.2.

<i>Component</i>	<i>Typical current draw [mA]</i>
<i>SAMD21 Cortex-M0+ 32bit low power ARM MCU</i>	<i>7</i>
<i>u-blox NINA-W102</i>	<i>130</i>
<i>MPU6050</i>	<i>3.9</i>

**Table 3.2.** *Component and the estimated current drawn as per the corresponding datasheet. The datasheets can be found in appendices B to D. Drawn in Microsoft word.*

By summing the current requirements of the respective components, the theoretical maximum power draw is obtained to roughly *141 mA* allowing a maximum current draw of *359 mA* for the stretch sensors. To ensure that the power limit is not exceeded an overhead of *10%* is added to the power draw of the stretch sensors resulting in a maximum allowed current of *323mA*.

To calculate the current through the sensor assembly the total resistance is calculated from,

$$\frac{1}{R_{tot}} = \sum_{i=1}^n \frac{1}{R_i} \quad (3.6)$$

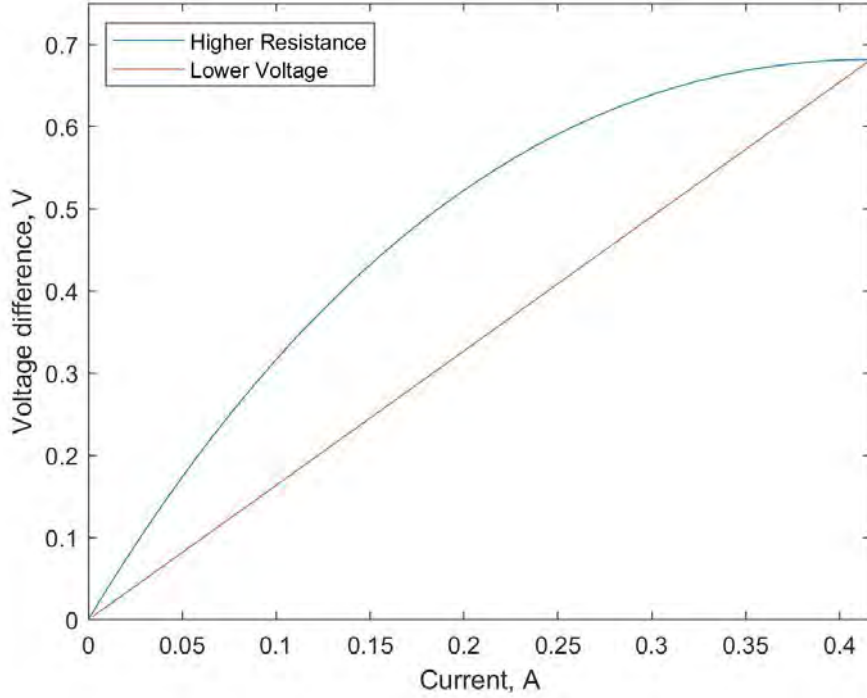
where  $R_i$  is the resistance of each sensor and resistor pairing and  $n$  is the total amount of sensors. Under the assumption that all sensors are identical the total resistance can be derived from (3.6) according to

$$R_{tot} = \frac{R_{min} + R_m}{5} \quad (3.7)$$

The current through the sensors can thus be determined by combining (3.2) with (3.7)

$$I = \frac{5U}{R_{min} + R_m} \quad (3.8)$$

With a voltage of *5V* and the ideal resistor value of *34Ω* the current is calculated to *417mA* which exceeds the aforementioned limit and therefore it needs to be reduced. From (3.8) it is evident that there are two ways to obtain a decreased current through the sensors, either the voltage can be lowered, or the resistor value can be increased. To determine the ideal approach the current draw and voltage difference of both methods is evaluated with equation (3.5) and (3.8). The equations are calculated with voltage values ranging from *0V* to *5V* as well as resistor values ranging from *34Ω* to *1MΩ*, and the resulting voltage difference is plotted against the corresponding current draw in figure 3.3.



**Figure 3.3.** Voltage difference plotted against current draw. Plotted in MATLAB.

From the graph in figure 3.3, it is clear that decreasing the current by increasing the resistance results in a higher voltage difference compared to decreasing the voltage over the sensors. Equation (3.8) is therefore derived to find the resistance value which limits the current to  $323mA$  according to

$$R_m = \frac{5U}{I} - R_{min} \quad (3.9)$$

and the resistance is found to be  $51\Omega$ .

### 3.1.2 Power Supply

Powering the glove is done by a LiPo battery integrated into the wrist of the glove. The placement of the battery presents a limitation of the physical size of the battery which is measured to be around  $40x30x5mm$ . Since the current draw of the electronics is estimated to be around  $500mA$  it is advantageous to choose a battery with the largest possible capacity given the size constraints to prolong the battery life between charging. The battery which was chosen has the dimensions  $35x25x5mm$  and a voltage of  $3.7V$ . Furthermore, the capacity of the battery is  $400mAh$  which allows the glove to be used for an estimated 48 minutes before needing to be recharged.

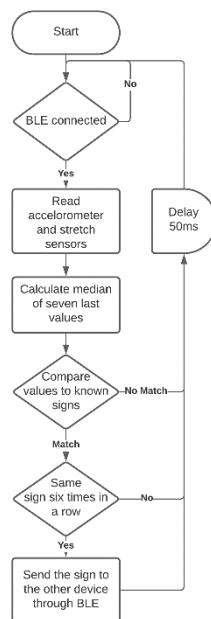
To power the microcontroller with the LiPo battery the voltage needs to be boosted from 3.7V to meet the 5V requirement of the microcontroller which is done through a DC-DC converter. The DC-DC converter was integrated in between the microcontroller and the LiPo battery. The converter connects to the battery through a JST terminal and to the Arduino through the mini-USB port.

### 3.1.3 Accelerometer

To implement the MPU6050 a model with a breakout board was chosen. The breakout board facilitates the use of the accelerometer by allowing access with through hole soldering. The MPU6050 can operate on a voltage ranging from 3-5V and has a power draw of 3.9mA. The module can therefore be powered by the Arduino through the 3.3V pin which can supply 7mA. Communication between the Arduino and MPU6050 is done through the I<sup>2</sup>C protocol. On the microcontroller pin A4 and A5 correspond to SDA and SCL respectively and are therefore connected to the MPU. The accelerometer is sewn into place on the back of the hand to be able to measure the position and gestures.

## 3.2 Software

To accurately be able to determine which gesture is signed the microcontroller must evaluate the sensor values. The logic behind the program which accomplishes this follows the flowchart depicted in figure 3.4.



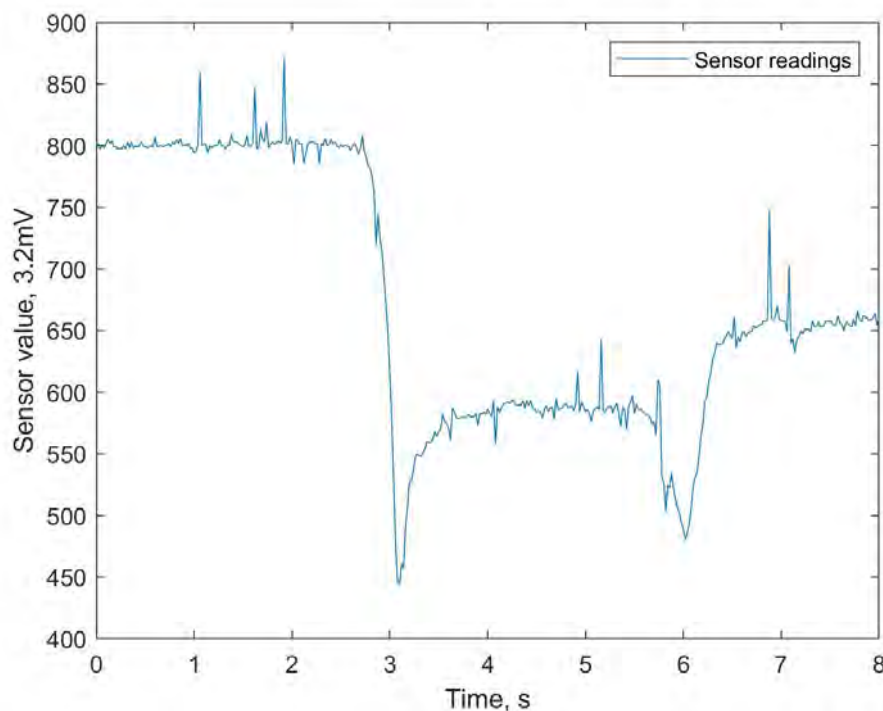
*Figure 3.4. Flow chart of program logic. Drawn in Lucid Chart [17].*

When powered on the Arduino tries to connect to another device through BLE. When a connection has been established the Arduino reads the values from the sensors and accelerometer before calculating the median of the last seven values. The medians are then compared to the known signs that are stored and imported from the header file *HandGesturesPierre.h*. If there is a match the Arduino compares the last six translated signs. If the same sign has been translated six times consecutively it is likely that the sign corresponds to the hand gesture and the translation is transmitted wirelessly over BLE to the connected device. The code and header file can be found in appendix E and F.

### 3.2.1 Stretch Sensor

To properly evaluate the applicability of the stretch sensor it is important to establish the characteristics of a desirable sensor reading. For this project it is crucial that the stretch sensor produces a repeatable sensor reading and that the sensor is precise enough to be able to differentiate between minute differences in finger placement.

By wiring up the stretch sensor to the microcontroller the change in voltage over the sensor can be measured. The result of stretching the sensor can be seen in figure 3.5.

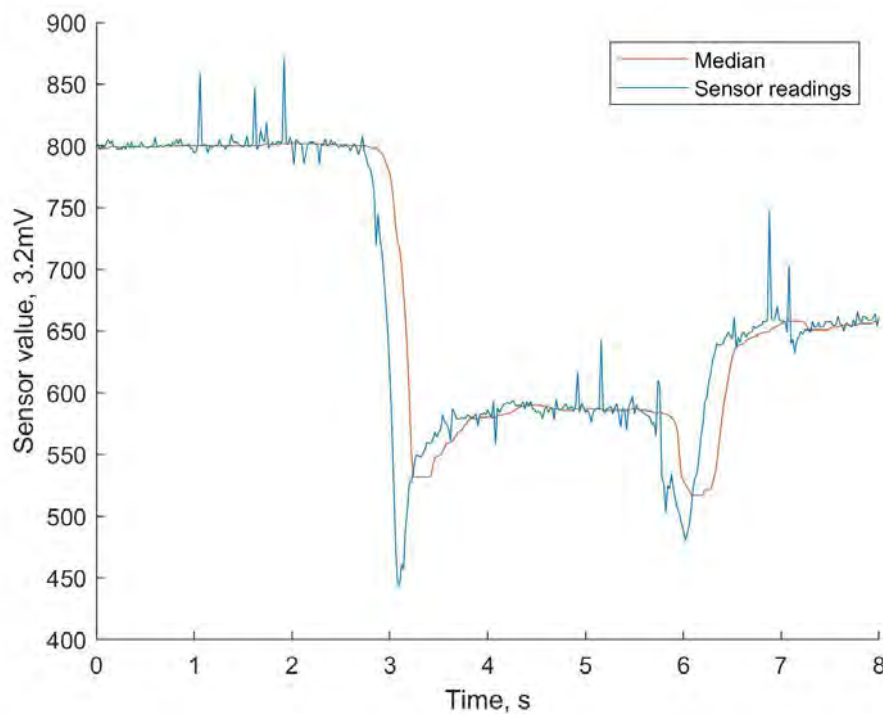


**Figure 3.5.** Voltage difference over time measured by the microcontroller. Plotted in MATLAB.

The range on the y-axis is the direct output of the microcontroller's readings. Since the Arduino Nano 33 IoT operates on 3.3V with a resolution of 10 bit each step corresponds

to a change of approximately  $3.2mV$ . It is evident from the graph in figure 3.5 that in its current implementation the conductive fabric is not applicable as a stretch sensor. The most prevalent issue at hand is the frequent spikes in readings which result in incorrect predictions of hand gestures and thus inadequate accuracy.

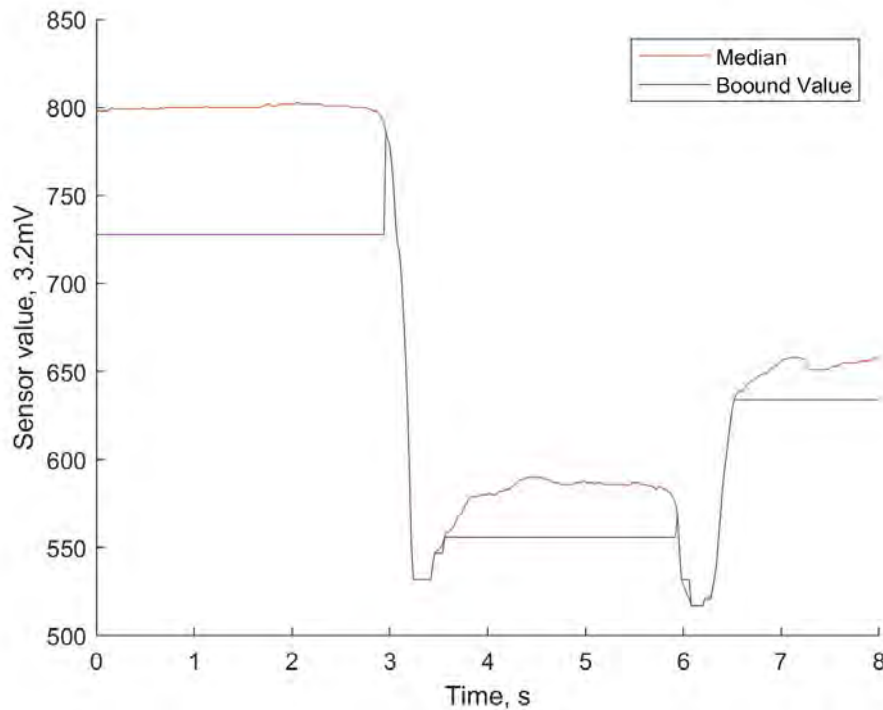
To reduce the spikes in the sensor readings the Arduino library *Running Median* [10] is implemented to store the last seven read values and calculate median. By calculating the median, the spikes even out resulting in the median graph in figure 3.6.



**Figure 3.6.** Difference between median and raw sensor readings. Plotted in MATLAB.

One downside with the calculating the median is that it results in a time delay which can be observed from the figure 3.6. The delay was measured to be around 80 milliseconds which is negligible for the scope of this project. Another issue presented when using conductive fabric as stretch sensors is that the sensor values stabilize over a prolonged period. From the timeframe  $3.5s$  to  $4.5s$  in figure 3.6 it can be observed that the values rise from  $540$  to  $600$  which corresponds to an increase of  $11\%$ . For the sensor values to stabilize faster a boundary is therefore implemented. If the rate of change is slow it is an indication that the finger has not moved, and the change can be attributed to the stabilization of the conductive fabric. The sensor value will thus only be reported if it differs more than  $3$  or roughly  $10mV$  from the previous reading resulting in the bound value graph in figure 3.7.





**Figure 3.7.** Boundary value compared to median value. Plotted in MATLAB.

The final implementation of the sensor evaluation presents a desirable behaviour where the sensor values are stable.

### 3.2.2 Accelerometer

The accelerometer is implemented to be able to determine the angle of the glove which is necessary for differentiating between different signs with the same finger placement. Furthermore, the accelerometer is used to determine if the glove is stationary or in motion which is crucial for the differentiation of the signs *I* and *J* for example. The breakout board of the MPU6050 is equipped with a processor which converts the readings from the accelerometer to  $m/s^2$  along the x,y and z-axis of the MPU which is then transmitted to the microcontroller. Since the MPU6050 communicates through the I<sup>2</sup>C-protocol the Arduino *Wire* library [16] is implemented to interpret the values from the accelerometer. When tilting the hand forward the acceleration ranges from  $3m/s^2$  to  $9.82m/s^2$ . By observing the current acceleration along the x-axis of the MPU the angle of the hand can therefore be decided.

To determine if the hand is in motion the norm of the acceleration is calculated according to equation (3.10)

$$\|a\| = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad (3.10)$$

where  $a_x$ ,  $a_y$  and  $a_z$  denote the acceleration component in each axis as given by the MPU. When stationary the only force acting on the accelerometer is the gravitational force which results in an acceleration of  $9.82m/s^2$ . Any deviation from an acceleration of  $9.82m/s^2$  therefore indicates that the hand has gone into motion. The absolute difference between the acceleration and gravitational force is therefore calculated according to (3.11)

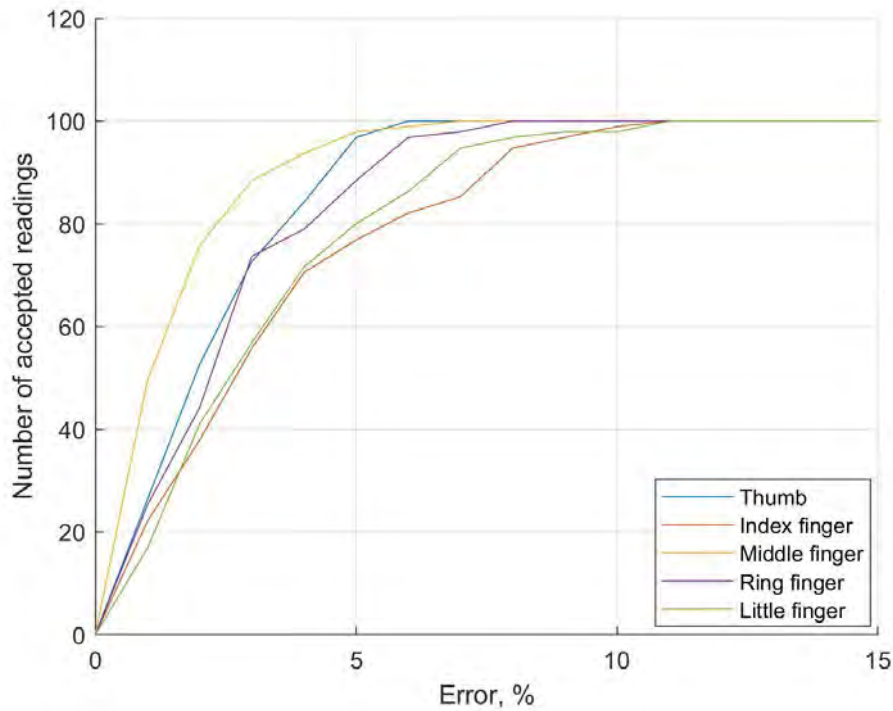
$$a_{diff} = \left| 9.82 - \sqrt{a_x^2 + a_y^2 + a_z^2} \right|. \quad (3.11)$$

### 3.2.3 Value Interpretation

For the microcontroller to evaluate the likelihood of the current gesture each sensor value is interpreted individually. For each known gesture there is a set composed of five sensor values for each individual finger. When the sensor readings are recorded the values are evaluated against the known gestures to determine the corresponding set. It is expected that there will be some degree of variance between the sensor readings since the user will not be able to replicate identical signs and an error margin is thus implemented. To evaluate the variance in sensor readings and the necessary error margin the letter A was signed 100 times repeatedly and the corresponding sensor values were recorded. The average from the readings of each respective finger was calculated from which the error was calculated according to (3.12)

$$Error = \left| 1 - \frac{Reading}{Average} \right|. \quad (3.12)$$

To determine a suitable error margin number of readings from each stretch sensor which would be accepted as the sign *A* is plotted against the error margin in figure 3.8.



**Figure 3.8.** Number of accepted readings plotted against the error margin for every stretch sensor. Plotted in MATLAB.

From figure 3.8 it can be observed that at an error margin on 10.7%, all the gestured signs would have been accepted. One issue with choosing such a large error margin is that the interval for acceptable readings between different will overlap. For example, observe the signs *S* and *T*. Calculating the average sensor value over 10 readings for each sign results in the following result presented in table 3.3.

	<i>Thumb</i>	<i>Index finger</i>	<i>Middle finger</i>	<i>Ring finger</i>	<i>Little finger</i>
<i>S</i>	689.9	651.4	701.0	572.4	665.8
<i>T</i>	697.7	704.1	683.8	603.2	714.5
<i>Difference [%]</i>	1.1	7.5	2.5	5.1	6.8

**Table 3.3.** Comparison of sensor values between the letters *S* and *T*. Drawn in Microsoft Word.

To prevent mistranslation the sign best corresponding to the sensor readings therefore needs to be calculated. The error for each respective finger is summed and the translation is determined by the sign with the least total difference. The approach is inefficient since it does require iterating through all the known signs but considering the scope of this project only encompasses the ASL alphabet which contains 26 different signs it is not deemed a problem.

To incorporate the accelerometer in the translation process the acceleration along the MPU's x-axis,  $a_x$  and the difference between the acceleration and gravitational force,  $a_{diff}$  are observed. For

translating the alphabet there are two positions of the hand that need to be differentiated between, when the hand is straight and when it is tilted forward. At these positions, the accelerometer reads  $9.82m/s^2$  and  $3m/s^2$  respectively. An arbitrary solution is therefore implemented where the position of the hand is deemed as straight when the acceleration  $a_x$  is larger than  $6m/s^2$ . If the hand is stationary or in motion is determined by  $a_{diff}$  which is implemented in a similar way to  $a_x$ . The state in which the hand is deemed to be is calculated by a threshold of  $1m/s^2$ . If the acceleration surpasses the threshold it can be concluded that the hand has begun moving.

# 4 Results

The project set out to answer two questions, how can electrically conductive fabric be applied as a stretch sensor and how can one efficiently read the finger and hand movements and translate ASL to speech. To evaluate and answer these questions a glove was fitted with conductive stretch sensors and two tests were conducted.

## 4.1 Final construction

The final construction of the glove is fitted with five strips of conductive fabric on each finger which act as stretch sensors. The sensors are powered by the Arduino Nano 33 IoT which is mounted to the wrist of the glove. Furthermore, the back of the hand is fitted with an MPU6050 which is connected and powered by the Arduino. To power the assembly a LiPo battery is sewn into the wrist and connected to a PowerBoost 500 through a JST terminal. The PowerBoost 500 converts the battery voltage of 3.7V to 5.2V which is fed to the Arduino through the mini-USB interface. The final construction is shown in figure 4.1.



*Figure 4.1. Image of final construction.*

## 4.2 Sensor Evaluation

To determine the accuracy of the glove each letter was signed 10 times consecutively. The test aims to test the repeatability of the sensor readings and in extension the gesture interpretation. By signing the same letter consecutively, it is more likely that the same gesture will be replicated more precisely and thus reducing variance from user error. The result from the test is shown in figure 4.2.

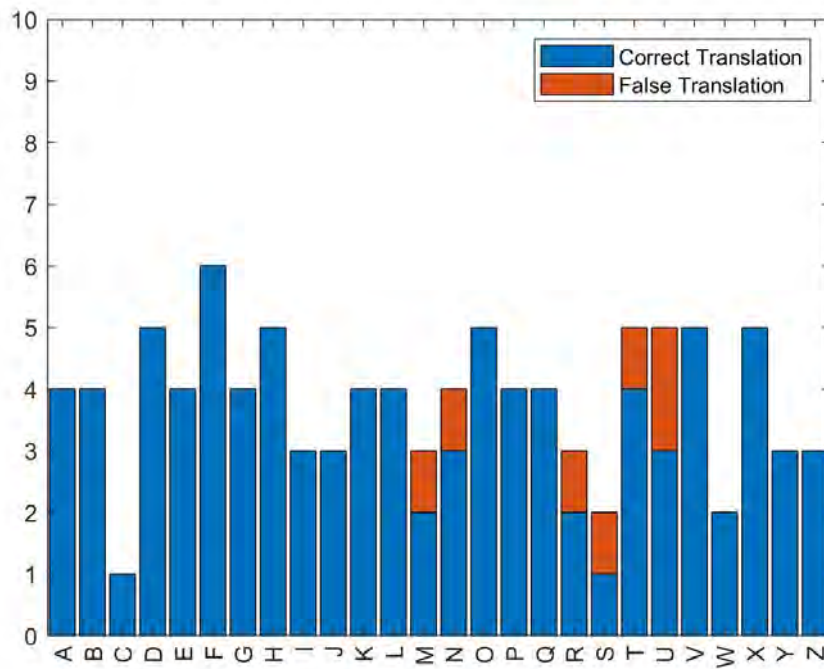


*Figure 4.2. Test results from sensor evaluation. Plotted in MATLAB.*

Out of the 260 translations 250 were correct resulting in an accuracy of 96%. In most cases a 100% accuracy was reached but for the worst cases only an accuracy of 70% could be reached.

### 4.3 Application Testing

Should the glove be used in realistic circumstances it is unlikely that the user will perform the same gesture consecutively. During this test, the user will therefore translate 100 letters generated at random to replicate a more likely scenario. The test aims to evaluate how well the translating holds up when more variance is introduced and hopes to determine the performance in a more likely use scenario. The result from the test is shown in figure 4.3.



**Figure 4.3.** Test result from application testing. Plotted in MATLAB.

Out of the 100 translations 93 were correct resulting in an accuracy of 93%.

# 5 Discussion

This chapter discusses the results of the project and changes which can be implemented to further improve the glove.

## 5.1 Hardware

One of the main drawbacks which presented itself early during the project was the power draw of the sensors. Traditional flex sensors which have been implemented to translate sign language have a resistance of roughly  $10k\Omega$  compared to the conductive fabric which has  $0.25\%$  of the resistance at  $25\Omega$ . Since power is reverse proportional to the resistance the stretch sensors draw 400 times more current comparatively. Furthermore, the current flowing through the sensors presented limitations in the effectiveness of the sensors. Early revisions of the glove were built using  $34\Omega$  resistors to achieve the optimal reading range which resulted in a current draw of approximately  $417mA$ . Once the BLE module was initiated the current draw would exceed the maximum of  $500mA$  resulting in the microcontroller failing. The final revision therefore implements  $52\Omega$  resistors which lowered the power draw to  $323mA$  but also resulted in a  $12\%$  smaller range for the sensor values.

Originally it was planned for the IMU on the Arduino Nano 33 IoT to be utilized since space on the glove is sparse but during prototyping the IMU stopped working and would no longer output any values. Since the rest of the microcontroller was still functional it was decided that an external accelerometer in the form of a MPU6050 would be implemented instead of replacing the whole microcontroller. One advantage of the MPU6050 that was discovered was that its smaller footprint made it easier and more comfortable to mount on the back of the hand compared to the microcontroller. The downside with the chosen MPU was that it only had 6 DOF which resulted in that the absolute position of the hand could not reliably be determined. For the scope of this project the signs were trivial enough for a 6 DOF MPU to suffice but should signs involving more complex movements be translated it is recommended that an MPU with 9 DOF be used instead.

In the original design hall-effect sensors would be implemented with magnets on the end of the index and middle finger. The design took inspiration from the data glove constructed by students at Cornell University who had utilized contact sensors on the end of the finger since the glove could not differentiate between  $R$ ,  $U$  and  $V$  otherwise. During testing it was found that the hall-effect sensors were redundant which led to their removal. One advantage of the stretch sensors which had not been considered was its ability to stretch in two directions compared to the traditional flex sensors which only could register flexing parallel to the finger. When signing  $R$ ,  $U$  and  $V$  the change in resistance due to stretching horizontally relative the fingers was large enough to measure and differentiate between the three letters.

## 5.2 Software

The interpretation of the accelerometer values is rudimentary and far from ideal. The glove can only recognize two states of the hand position, straight or bent as well as only being able to determine if the hand is still or moving with no differentiation between speed or direction. Furthermore, rotation around one axis is only taken into consideration since the scope of this



project does not put any further requirements on the glove. It is therefore likely that the current method would be inadequate and not provide the needed accuracy should signs with more complex hand movements be added. The implemented method of determining the position and movement of the hand should therefore only be viewed as a proof of concept rather than an actual final solution.

### **5.3 Results**

From the sensor evaluation result it is observed that the letters with the least success rate *M*, *N*, *S* and *T* all have similar gestures where the only difference is the position of the thumb. As discussed earlier in chapter 3.2.3 similar gestures were identified as a potential problem due to the sensor values being close with the average difference between *S* and *T* at 4.6%. The issue persists through both test which indicates that the issue lies with the sensor data. Since the only differentiation between the signs is the position of the thumb it is likely that the issue could be resolved by attaching another stretch sensor to it. Furthermore, in the processing of the sensor data a weighting could be attributed to the thumb values resulting in the placement of the thumb having a larger impact on the average difference.

Another interesting finding was the translation of *R* and *U*. From the sensor evaluation the accuracy was 95% but, in the application, testing the accuracy dropped to 62.5%. The glove can translate the signs with high accuracy but when the user is not reliable able to replicate the signs the translation becomes inadequate. From figure 4.2 it is evident that a high degree of accuracy is achievable with the existing hardware which would suggest that the issue could be resolved by implementing a more refined process for evaluating the sensor data.

The result show that the project has been a success. Most signs could reliably be translated and overall, the accuracy was 93% which is more than adequate as a baseline. Using the findings of this report as a foundation for future work would likely see an improve of accuracy to the high nineties which other similar projects have successfully achieved.

### **5.4 Conclusion**

This project set out to answer two questions, how can conductive fabric be applied as a stretch sensor and how can one accurately read hand gestures with a glove to translate ASL.

From the research and experiments conducted during this project it is evident that conductive fabric can be utilized as a stretch sensor by measuring the voltage change over the fabric when it is stretched. The sensor readings are precise and repeatable and can thus be applied in situations where it is crucial to accurately record differences in stretching. Furthermore, the conductive fabrics ability to stretch in several directions made so that the stretch sensors could differentiate between more signs compared to a traditional flex sensor making it superior in applications such as a translating glove.

The project also succeeded in reliably reading hand gestures to translate ASL. By implementing the conductive fabric as stretch sensors to register the finger movement and a MPU6050 to track the hand movements the ASL alphabet could be translated with an accuracy of 93%.

### **5.5 Future work**

The implemented method to process the sensor values can be refined with machine learning. By implementing machine learning into the glove, the program used will automatically improve and refine each time the glove is used, by recognizing patterns and setting reasonable margins

of error based on the values gathered. There are several Arduino libraries to implement machine learning such as *TinyML*, though it should be noted that these libraries might require an Arduino Nano 33 BLE Sense which has a more powerful CPU [14].

The findings during the project were only based on the sample of conductive fabric provided by PLUX. Whether other conductive textiles have similar or better characteristics was never explored. It is therefore recommended that work based on these findings conduct experiments and research on different conductive fabrics to evaluate the best fitting for its intended use.

# Bibliography

- [1] S. Waterfield. *ASL Day 2019: Everything You Need To Know About American Sign Language*, 2019 [Online] Available at: <https://www.newsweek.com/asl-day-2019-american-sign-language-1394695> [Accessed 2021-03-01]
- [2] M. Lin and R. Villalba. *Sign Language Glove*, Cornell University: School of Electrical and Computational Engineering, 2014. [Online] Available at: [https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2014/rdv28\\_mj1256/webpage/](https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2014/rdv28_mj1256/webpage/) [Accessed 2021-04-04]
- [3] Zhou, Z., Chen, K., Li, X. *et al.* *Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays*. *Nat Electron* 3, 571–578, 2020. [Online] Available at: <https://doi.org/10.1038/s41928-020-0428-6> [Accessed: 2021-03-30]
- [4] M. Brain. *How Microcontrollers Work*, 2000. [Online] Available at: <https://electronics.howstuffworks.com/microcontroller.htm> [Accessed: 2021-03-30]
- [5] Arduino, *Arduino Nano 33 IoT*, [Online] Available at: <https://store.arduino.cc/arduino-nano-33-iot> [Accessed: 2021-03-31]
- [6] Dejan, *Arduino and MPU6050 Accelerometer and Gyroscope Tutorial*, 2019 [Online] Available at: <https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial/> [Accessed 2021-04-10]
- [7] A&S POWER, *The advantages and disadvantages of lithium polymer battery*, 2017 [Online] Available at: <https://www.szaspower.com/industry-news/The-advantages-and-disadvantag.html> [Accessed 2021-04-25]
- [8] Digi-Key, *Introduction to DC-DC converters*, 2016 [Online] Available at: <https://www.digikey.com/en/maker/blogs/introduction-to-dc-dc-converters> [Accessed 2021-04-25]
- [9] Adafruit, *PowerBoost 500 Basic – 5V USB Boost @ 500mA from 1,8V+*. [Online] Available at: <https://www.adafruit.com/product/1903> [Accessed 2021-04-25]
- [10] Arduino, *Arduino Playground - RunningMedian*, *Playground.arduino.cc* 2014 [Online]. Available at: <https://playground.arduino.cc/Main/RunningMedian/> [Accessed 2021-04-30]
- [11] Y. Teyeme, B. Malengier, T. Tesfaye, and L. Van Langenhove, *A Fabric-Based Textile Stretch Sensor for Optimized Measurement of Strain in Clothing*, vol. 20, no. 24, p. 7323, 2020 [Online] Available at: <http://dx.doi.org/10.3390/s20247323> [Accessed: 2021-04-03]
- [12] Pal, S., Sarkar, D., S. Roy, D., Paul, A., Arora, A., *Design, development and fabrication of a conductive fabric based flexible and stretchable strain sensor*, Sayantan Pal *et al* *IOP Conf. Ser.: Mater. Sci. Eng.* vol. 912, 2020 [Online] Available at: <https://iopscience.iop.org/article/10.1088/1757-899X/912/2/022009> [Accessed: 2021-04-04]

- [13] Mathworks, *MATLAB 2019a*. [Online] Available at: <https://www.mathworks.com/products/matlab.html> [Accessed 2021-02-28]
- [14] TensorFlow, *TensorFlow Lite for Microcontrollers*, 2021 [Online] Available at: <https://www.tensorflow.org/lite/microcontrollers?fbclid=IwAR2BMHugRAJmWDQN8OmQjKGiyCZpgLjdRZ0C8zE4hqQyYZuKE1dVAr1A2Y> [Accessed 2021-05-01]
- [15] Circuit-Diagram, *Circuit-Diagram editor* [Online] Available at: <https://www.circuit-diagram.org/editor/> [Accessed 2021-05-02]
- [16] Arduino, *Wire Library*, 2019. [Online] Available at: <https://www.arduino.cc/en/reference/wire> [Accessed 2021-05-02]
- [17] Lucid, *Lucid Chart*. [Online] Available at: <https://lucid.co/product/lucidchart> [Accessed 2021-05-05]
- [18] Alapati, Sreejan & Yeole, Shivraj. *A Review on Applications of Flex Sensors*, 2017. International Journal of Emerging Technology and Advanced Engineering. Vol.7 page.97-100. [Online] Available at: [https://www.researchgate.net/publication/318850816\\_A\\_Review\\_on\\_Applications\\_of\\_Flex\\_Sensors](https://www.researchgate.net/publication/318850816_A_Review_on_Applications_of_Flex_Sensors) [Accessed 2021-04-04]

# Appendices


## Appendix A

### Sign Language chart



# Appendix B

## MPU6050 datasheet

	<b>MPU-6000/MPU-6050 Product Specification</b>	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013
---	--	---

### 5 Features

#### 5.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of  $\pm 250$ ,  $\pm 500$ ,  $\pm 1000$ , and  $\pm 2000^\circ/\text{sec}$
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5 $\mu$ A
- Factory calibrated sensitivity scale factor
- User self-test

#### 5.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of  $\pm 2g$ ,  $\pm 4g$ ,  $\pm 8g$  and  $\pm 16g$
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500 $\mu$ A
- Low power accelerometer mode current: 10 $\mu$ A at 1.25Hz, 20 $\mu$ A at 5Hz, 60 $\mu$ A at 20Hz, 110 $\mu$ A at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt
- User self-test

#### 5.3 Additional Features

The MPU-60X0 includes the following additional features:

- 9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
- Auxiliary master I<sup>2</sup>C bus for reading data from external sensors (e.g., magnetometer)
- 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled
- VDD supply voltage range of 2.375V-3.46V
- Flexible VLOGIC reference voltage supports multiple I<sup>2</sup>C interface voltages (MPU-6050 only)
- Smallest and thinnest QFN package for portable devices: 4x4x0.9mm
- Minimal cross-axis sensitivity between the accelerometer and gyroscope axes
- 1024 byte FIFO buffer reduces power consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 g shock tolerant
- 400kHz Fast Mode I<sup>2</sup>C for communicating with all registers
- 1MHz SPI serial interface for communicating with all registers (MPU-6000 only)
- 20MHz SPI serial interface for reading sensor and interrupt registers (MPU-6000 only)

# Appendix C

## Nina-W10 series data sheet



### 4.2.6 Current consumption

Typical current consumption of a NINA-W10 module is provided in Table 13. The current consumption is highly dependent on the application implementation. The consumption information listed below is taken from the Espressif ESP32 Datasheet [3].

Power mode	Activity	Typ	Max	Unit	Remarks
Wi-Fi	Wi-Fi Tx packet 16 dBm	190	320	mA	50% duty cycle
	Wi-Fi Rx and listening	100	140	mA	
Bluetooth	Bluetooth Tx Pout 0 dBm	130	230	mA	Throughput 2.1 Mbit/s
	Bluetooth Rx and listening	100		mA	Throughput 2.1 Mbit/s
Bluetooth low energy	Bluetooth Tx Pout 0 dBm	130	225	mA	Throughput 240 kbit/s
	Bluetooth Rx and listening	100		mA	Throughput 240 kbit/s
CPU idle mode	CPU speed 120 MHz	95		mA	
Modem-sleep mode	CPU speed 80 MHz. The CPU is operational. The radio is turned off.	30		mA	Immediate wake-up
Light-sleep mode	PLL and radio disabled. The CPUs are stalled. The ULP co-processor and touch controller can be periodically triggered by monitor sensors.	800		µA	Wake-up latency < 1 ms.
Deep-sleep mode	PLL and radio disabled. Digital core powered down. The CPU context is lost.	6.5		µA	Wake-up latency < 1 ms. For ultra-low-power infrequently-connected Wi-Fi/Bluetooth apps.
Hibernate mode	PLL and radio disabled. Digital core powered down. The CPU context is lost. The RTC domain is powered down.	4.5		µA	Wake-up source: RTC timer only. Wake-up latency < 1 ms. For ultra-low-power infrequently-connected Wi-Fi/Bluetooth apps.

Table 13: Current consumption during typical use cases

# Appendix D

# SAM D21 Family data sheet

## SAM D21 Family Electrical Characteristics

**Table 37-8. Current Consumption (Device Variant A)**

Mode	Conditions	T <sub>A</sub>	Min.	Typ.	Max.	Units
ACTIVE	CPU running a While(1) algorithm	25°C	3.11	3.37	3.64	mA
		85°C	3.24	3.48	3.76	
	CPU running a While(1) algorithm V <sub>DDIN</sub> =1.8V, CPU is running on Flash with 3 wait states	25°C	3.10	3.36	3.64	
		85°C	3.24	3.48	3.75	
	CPU running a While(1) algorithm, CPU is running on Flash with 3 wait states with GCLKIN as reference	25°C	60*freq + 74	60*freq + 136	62*freq + 196	μA (with freq in MHz)
		85°C	62*freq + 154	62*freq + 228	62*freq + 302	
	CPU running a Fibonacci algorithm	25°C	4.12	4.53	4.92	mA
		85°C	4.27	4.63	4.98	
	CPU running a Fibonacci algorithm V <sub>DDIN</sub> =1.8V, CPU is running on flash with 3 wait states	25°C	4.12	4.53	4.92	
		85°C	4.27	4.63	4.98	
	CPU running a Fibonacci algorithm, CPU is running on Flash with 3 wait states with GCLKIN as reference	25°C	86*freq + 76	88*freq + 136	88*freq + 196	μA (with freq in MHz)
		85°C	88*freq + 156	88*freq + 230	88*freq + 302	
	CPU running a CoreMark algorithm	25°C	5.78	6.32	6.80	mA
		85°C	5.93	6.47	7.00	
CPU running a CoreMark algorithm V <sub>DDIN</sub> =1.8V, CPU is running on flash with 3 wait states	25°C	5.17	5.60	5.96		
	85°C	5.35	5.73	6.10		
CPU running a CoreMark algorithm, CPU is running on Flash with 3 wait states with GCLKIN as reference	25°C	106*freq + 78	106*freq + 136	108*freq + 196	μA (with freq in MHz)	
	85°C	106*freq + 154	108*freq + 232	108*freq + 310		



# Appendix E

## Arduino code

```
// Authors - Pierre Sinander and Tomas Issa
// KTH - Royal Institute of Technology
// Course – MF133X – Degree Project in Mechatronics
// TRITA ITM-EX 2021:38
// Name of project – Sign Language Translation
// Date – 09-05-2021
//The program reads the values from the stretch sensors and the MPU6050 and compares them
// against the known signs stored in the file HandGesturesPierre.h. The best matching sign is the
// transmitted wirelessly over BLE.

#include "RunningMedian.h"
#include <ArduinoBLE.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
#include "HandGesturesPierre.h"
Adafruit_MPU6050 mpu;

//analog pin for each respective finger
int pin1 = A3;
int pin2 = A2;
int pin3 = A6;
int pin4 = A1;
int pin5 = A0;

int SensorReading[5] = {0,0,0,0,0}; //Array that stores the readings
int TrailingValue[5] = {0,0,0,0,0}; //Array that stores the trailing data
int Data[7] = {0,0,0,0,0,0,0};

RunningMedian s1 = RunningMedian(7); //Initiates a sample pool for each finger that stores 20
values
RunningMedian s2 = RunningMedian(7);
RunningMedian s3 = RunningMedian(7);
RunningMedian s4 = RunningMedian(7);
RunningMedian s5 = RunningMedian(7);

RunningMedian norm = RunningMedian(7);
RunningMedian x = RunningMedian(7);

int boundry = 5; //Boundry value
float BestMatch[2]; //array for storing the best match, in ex 0 is the error and 1 is the index of
the character

float error; //Variable which stores the error
```

```

int ind;
int c = 0;

BLEService SensorService("1101");
BLEUnsignedCharCharacteristic sensorLevelChar("2101", BLERead | BLENotify);

void setup() {
  //Specifies output and input pins
  pinMode(pin1, INPUT);
  pinMode(pin2, INPUT);
  pinMode(pin3, INPUT);
  pinMode(pin4, INPUT);
  pinMode(pin5, INPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);

  //Initializes the MPU6050
  if (!mpu.begin()) {
    while (1);
  }

  mpu.setAccelerometerRange(MPU6050_RANGE_4_G);
  mpu.setGyroRange(MPU6050_RANGE_250_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);

  //Initializes BLE
  if (!BLE.begin()) {
    Serial.println("starting BLE failed!");
    while (1);
  }

  BLE.setLocalName("Translating Glove");
  BLE.setDeviceName("Translating Glove");
  BLE.setAdvertisedService(SensorService);
  SensorService.addCharacteristic(sensorLevelChar);
  BLE.addService(SensorService);

  BLE.advertise();
}

void loop() {
  BLEDevice central = BLE.central();
  if (central){
    digitalWrite(LED_BUILTIN, HIGH); //Once connected the LED will be turned on

    while (central.connected()){

      BestMatch[0] = 0.12;
      BestMatch[1] = n; //Assigns an index which displays nothing

      ReadMPU();
      ReadStretchSensor();
    }
  }
}

```

```

for (int i = 0; i < n; i++){ //Iterates through the rows of the matrix
  if(Data[5] == Words[i][5] && Data[6] == Words[i][6]){
    error = Error(i);

    if(error < BestMatch[0]){
      BestMatch[0] = error;
      BestMatch[1] = i;
    }
  }

}

if(ind == BestMatch[1]){ //If it is the same reading as before we add one to c
  c++;
}

ind = BestMatch[1]; //Sets the index in the translation matrix to the best match
if(c > 4){
  sensorLevelChar.writeValue(Translation[ind]);
}

//PlotValues(); //Function to plot the readings from the stretch sensor
delay(50);
}
}
//When the Arduino isnt connected to another device through BLE the on-board LED
//will blink every second indicating that it is read for pairing
digitalWrite(LED_BUILTIN, HIGH);
delay(500);
digitalWrite(LED_BUILTIN, LOW);
delay(500);

}

float Error(int i){ //Error calculations using percent
  float max_error = 0.12;
  float error = 0;

  for(int j = 0; j < 5; j++){
    if(abs(1 - Data[j]/Words[i][j]) > max_error){
      return 1;
    }

    error = error + abs(1 - Data[j]/Words[i][j]);
  }

  return error;
}

void PlotValues(){ //Plots the values for each finger
  Serial.print(Data[0]);
  Serial.print(",");

```

```

Serial.print(Data[1]);
Serial.print(",");
Serial.print(Data[2]);
Serial.print(",");
Serial.print(Data[3]);
Serial.print(",");
Serial.println(Data[4]);
}

```

```

void ReadMPU(){
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  x.add(a.acceleration.x);
  norm.add(abs(9.82 - sqrt(sq(a.acceleration.x) + sq(a.acceleration.y) + sq(a.acceleration.z))));

  if(x.getAverage() < 6){
    Data[5] = 0; //Position of hand laying down
  }
  else{
    Data[5] = 1; //Position of hand straight up
  }

  if(norm.getAverage() < 1.1){
    Data[6] = 0; //Hand is still
  }
  else{
    Data[6] = 1; //Hand is in motion
  }
}

```

```

void ReadStretchSensor(){
  s1.add(analogRead(pin1)); //Adds the readings to each respective sample pool
  s2.add(analogRead(pin2));
  s3.add(analogRead(pin3));
  s4.add(analogRead(pin4));
  s5.add(analogRead(pin5));

  SensorReading[0] = s1.getMedian(); //Assigns the median of each respective finger
  SensorReading[1] = s2.getMedian();
  SensorReading[2] = s3.getMedian();
  SensorReading[3] = s4.getMedian();
  SensorReading[4] = s5.getMedian();

  for(int i=0; i<5; i++){ //Reads the sensor values
    if(abs(SensorReading[i] - TrailingValue[i]) > boundry){
      Data[i] = SensorReading[i];
    }
    TrailingValue[i] = SensorReading[i];
  }
}

```

# Appendix F

## Arduino header file

```
// Authors - Pierre Sinander and Tomas Issa
// KTH - Royal Institute of Technology
// Course – MF133X – Degree Project in Mechatronics
// TRITA ITM-EX 2021:38
// Name of project – Sign Language Translation
// Date – 09-05-2021
//The header file stores all the known signs as well as the corresponding values.
```

```
#define n 26 //Number of words we have in our matrix
```

```
float Words[n][7] = { //Matrix containing all the sensor values
  {725.40, 703.10, 587.90, 635.20, 726.80, 1, 0}, //A
  {527.70, 540.30, 750.80, 470.60, 544.10, 1, 0}, //B
  {757.40, 713.50, 731.90, 675.20, 671.50, 1, 0}, //C
  {758.50, 515.20, 754.90, 598.80, 750.30, 1, 0}, //D
  {695.20, 620.70, 746.50, 616.00, 637.50, 1, 0}, //E
  {557.90, 738.90, 740.20, 541.90, 530.30, 1, 0}, //F
  {734.30, 535.20, 654.50, 638.30, 709.60, 0, 0}, //G
  {676.50, 638.70, 752.10, 473.10, 700.50, 0, 0}, //H
  {725.90, 706.10, 688.20, 633.80, 557.80, 1, 0}, //I
  {725.90, 706.10, 688.20, 633.80, 557.80, 1, 1}, //J
  {681.40, 545.20, 663.50, 488.40, 662.00, 1, 0}, //K
  {693.10, 511.40, 513.70, 599.90, 704.20, 1, 0}, //L
  {688.60, 745.10, 663.10, 629.10, 656.80, 1, 0}, //M
  {651.90, 699.90, 648.40, 583.40, 601.00, 1, 0}, //N
  {775.00, 753.60, 735.70, 691.60, 743.90, 1, 0}, //O
  {723.70, 599.30, 693.50, 615.80, 695.50, 0, 0}, //P
  {700.90, 662.30, 628.90, 594.40, 697.60, 0, 0}, //Q
  {696.20, 593.20, 748.80, 455.90, 671.90, 1, 0}, //R
  {689.90, 651.40, 701.00, 572.40, 665.80, 1, 0}, //S
  {697.70, 704.10, 683.80, 603.20, 714.50, 1, 0}, //T
  {687.20, 534.30, 757.90, 426.20, 687.80, 1, 0}, //U
  {715.60, 470.70, 755.70, 417.90, 677.00, 1, 0}, //V
  {536.30, 545.00, 758.50, 463.90, 698.70, 1, 0}, //W
  {720.30, 540.10, 736.70, 518.60, 700.30, 1, 0}, //X
  {732.00, 735.40, 526.70, 640.70, 577.10, 1, 0}, //Y
  {748.40, 619.00, 752.10, 595.70, 735.50, 0, 1} //Z
};
```

```
char Translation[n+1]{'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S',
'T','U','V','W','X','Y','Z',''}
```

## Appendix G

# Acumen simulation

```
// Authors - Pierre Sinander and Tomas Issa
// KTH - Royal Institute of Technology
// Course – MF133X – Degree Project in Mechatronics
// TRITA ITM-EX 2021:38
// Name of project – Sign Language Translation
// Date – 28-03-2021
```

```
model Main(simulator) =
initially
  _3D = (),
  _3DView = ((3,-7,5) ,(0,0,0)), //Kamera positionen, kameran sitter i (3,-7,5) och är riktad mot
  Origo
```

```
alpha1 = 0, //Vinklar för respektive finger vid handen, index anger vilket finger det är
alpha2 = 0, // t.ex. alpha1 är vinkeln för tummen, alpha2 är för pekfingeret o.s.v.
alpha3 = 0,
alpha4 = 0,
alpha5 = 0,
```

```
alpha1' = 0, //Rotationshastigheten för respektive finger. Eftersom fingret består av två delar
separerade av en led
alpha2' = 0, //Så anger vi två olika vinklar för respektive del. Alpha är vinkeln för delen av
fingret innan leden
alpha3' = 0,
alpha4' = 0,
alpha5' = 0,
```

```
beta1 = 0, //Vinklar för fingrarna efter leden
beta2 = 0,
beta3 = 0,
beta4 = 0,
beta5 = 0,
```

```
beta1' = 0, //Rotationshastighet för beta vinklarna
beta2' = 0,
beta3' = 0,
beta4' = 0,
beta5' = 0,
```

```
length1 = 2, //Fingrarnas längd, undre delen innan leden
length2 = 1.5, //Längden på fingret efter leden
width = 0.25, //Fingrarnas bredd
speed = pi/10, //Hastigheten med vilken fingrarna roterar
sizeA = 0 //Storlek på texten som visar vilket tecken handsken gör
```

```

always
  _3D = (
//Skapar handen
Sphere center = (0,0,0) size = 2 color = yellow rotation = (0,0,0)

//För att bestämma positionerna av fingrarna används cylindriska koordinater. Centrum av
fingrarna beror på fyra variabler: length1, length2, alpha och beta. length1 och length2 är
konstanta medans
//alpha och beta varierar. Genom att ange vinklar på alpha och beta så kan handskens olika
tecken simuleras
//Skapar långfingret
Cylinder center = (0,0,2) size = (length1,width) color = red rotation = (pi/2 + alpha1,0,0)
Cylinder center = (0,-sin(alpha1)*length1/2 - sin(beta1)*length2/2,1.9 + cos(alpha1)*length1/2
+ cos(beta1)*length2/2) size = (length2,width) color = red rotation = (pi/2 + beta1,0,0)
//Skapar pekfinger
Cylinder center = (0.6,0,1.7) size = (length1,width) color = red rotation = (pi/2 + alpha1,0,0)
Cylinder center = (0.6,-sin(alpha2)*length1/2 - sin(beta2)*length2/2,1.6 +
cos(alpha2)*length1/2 + cos(beta2)*length2/2) size = (length2,width) color = red rotation =
(pi/2 + beta1,0,0)
//Skapar ringfinger
Cylinder center = (1.4,0,1) size = (length1,1.2*width) color = red rotation = (pi/2 + alpha1,0,0)
Cylinder center = (1.4,-sin(alpha1)*length1/2 - sin(beta1)*length2/2,0.9 +
cos(alpha1)*length1/2 + cos(beta1)*length2/2) size = (length2,1.2*width) color = red rotation =
(pi/2 + beta1,0,0)
//Skapar tummen
Cylinder center = (-0.6,0,1.7) size = (length1,width) color = red rotation = (pi/2 + alpha1,0,0)
Cylinder center = (-0.6,-sin(alpha2)*length1/2 - sin(beta2)*length2/2,1.6 +
cos(alpha2)*length1/2 + cos(beta2)*length2/2) size = (length2,width) color = red rotation =
(pi/2 + beta1,0,0)
//Skapar lillfingret
Cylinder center = (-1.2,0,1.4) size = (length1,width) color = red rotation = (pi/2 + alpha1,0,0)
Cylinder center = (-1.2,-sin(alpha1)*length1/2 - sin(beta1)*length2/2,1.3 +
cos(alpha1)*length1/2 + cos(beta1)*length2/2) size = (length2,width) color = red rotation =
(pi/2 + beta1,0,0)
Text center=(2.5,0,0) size=sizeA color=blue rotation=(0,0,0) content="A" //Visar outputen
av handsken. Storleken är 0 till början men ändras när tecknet ska visas.
),

```

```

alpha1' = speed, //Anger ekvationerna för alla vinklar, där vinklarna innan leden är samma som
speed
alpha2' = speed,
alpha3' = speed,
alpha4' = speed,
alpha5' = speed,

```

```

beta1' = 2*speed, //Delen av fingret efter leden rör sig dubbelt så snabbt
beta2' = 2*speed,
beta3' = 2*speed,
beta4' = 2*speed,

```

```
beta5' = 2*speed,
```

```
if alpha1 > pi/3.5 //Simulerar handsken när användaren visar ett A på ASL  
then speed = -pi/10,  
sizeA = 4 //När handsken nått längst ner känner den av att tecknet är ett A och ger det som  
output. Storleken på texten ändras från 0 -> 4 vilket gör den synlig  
noelse,
```

```
if alpha1 < 0 //Handen återgår till ursprungstillståndet  
then speed = 0,  
sizeA = 0 //När inget tecken längre visas blir output noll igen  
noelse
```





TRITA ITM-EX 2021:38