



Development of a low cost and easy to assemble glove for hand movement acquisition as an open-source project

By Thomas AUJAS

Supervised by Dr Alison Griffiths

A thesis submitted to the Faculty of Computing, Engineering and Science, Staffordshire University in fulfilment of the requirement for the MSc Robotics and Smart Technologies

MSc Robotics and Smart Technologies

20th of August 2021

School of Creative Arts and Engineering

Staffordshire University

Student number: 20015669

a015669k@student.staffs.ac.uk

Abstract

Over the past few years, Virtual Reality (VR) has managed to accurately replicate the reality to immerse the user in a simulated world. One of VR's best aspect is the interaction with the environment like grabbing or touching virtual objects. Moreover, many companies developed gloves to be able to sense hand movements and reproduce them in game for even better immersion. Therefore, the gloves can be also used for mining, military, transports, and construction simulators. Sensor gloves are useful in other domains such as medicine. Indeed, the sensor gloves can be used to measure finger joint angles to control Arthritis (a disabling and painful disease) easily and with precision. It is also used to control an anthropomorphic robotic hand and exo-gloves (for rehabilitation purposes) that mimics movements. It could benefit a wide range of research areas such as in robotics using telemanipulation, or in the medical field (tissue engineering) for limb regeneration. Finally, it can be used to translate sign language into a speech or text. The requirements change depending on the application of course, for VR mainstream games, the accuracy and the number of joints measured by the gloves is low. For simulations, medicine, accuracy is really important as well as measuring as many joint as possible. Of course, cost changes depending on the application, whereas for VR games, the cost must be accessible to the general public, in medicine, the equipment is expensive. Developing an accessible cost glove while accurately measuring all the hand joints is the goal of the project. To do so, a 3D printer is involved to print all the necessary parts of the glove. The glove is also composed of potentiometers that measures the angles. The angles are obtained in different ways, with a linear relationship or with a second-degree relationship, which requires a calibration. The calibration helps to have referential position and values, as well as estimating the polynomial relation between the potentiometers and the joints' angles. Some testing has been performed to determine the accuracy and repeatability. The results were acceptable with a good accuracy overall, but a bad repeatability. The latter could be improved by attaching the device better to the hand. Another test included a game design software to simulate a hand model (Unity) was also performed. The hand model moved accordingly to the angles from the glove values. From the user point of view, the hand is synchronised with its hand, which is already convenient for VR games.

Table des matières

Abstract	2
List of figures	4
List of Tables.....	6
List of Acronyms.....	6
1 Introduction.....	8
1.1 Aim	8
1.2 Objectives	8
1.3 Justification of hand acquisition system implementation.....	8
1.4 Dissertation organisation	10
2 Literature review: Methods for hand movement acquisition.....	10
2.1 Sensors usually used for hand movement data acquisition	10
2.1.1 Flex sensor:	10
2.1.2 IMU (Inertial Measurement Unit):.....	10
2.1.3 Potentiometers	11
2.2 The designs, sensor placement and data acquisition	11
2.2.1 Virtual Reality acquisition system	11
2.2.2 Measuring more joints with higher accuracy	12
2.3 Data processing.....	15
2.3.1 Calibration.....	15
2.3.2 Filtering the noise and signal drift.....	15
2.4 Testing methods.....	16
2.5 Accuracy	16
2.5.1 Repeatability.....	17
2.6 Conclusion	17
3 Hand Movement Acquisition: Design and System Implementation.....	18
3.1 Design considerations and technical specifications.....	18
3.2 Mechanical design (kinematic diagram).....	20
3.2.1 Kinetic diagram	20
3.2.2 Link movements and angles relations	22
3.2.3 Alternative design	24
3.3 Electronic requirements	25
3.3.1 Circuit diagram.....	25
3.3.2 Power consumption	26
3.3.3 Connection within the assembly	27
3.4 Software requirements	27
3.5 3D printed Glove	28
3.5.1 The different parts	28
3.5.2 The assembly	30
3.5.3 The printer and the settings	33
3.6 Cost (bill of materials).....	35
3.7 Summary.....	37
4 Coding.....	37
4.1 Languages and software justification	37

4.2	Arduino code	37
4.3	Python	38
4.3.1	Receiving the data while interfacing	38
4.3.2	Calibration	40
4.3.3	Saving the calibration with JSON	45
4.3.4	Processing the data	47
4.4	Summary	48
5	Testing the glove	49
5.1	Testing with Unity for a fast visual verification	49
5.1.1	Communicating with Python	49
5.1.2	Moving the virtual hand	49
5.1.3	Visual feedback	51
5.1.4	DIP joint issues	51
5.2	Accuracy test (and precision test)	52
5.2.1	Visual test with an overlay for MCP _x , PIP and wrist _x	52
5.2.2	Comparison with an IMU	53
5.2.3	MCP _z accuracy	54
5.3	Repeatability	55
5.4	Testing the device on participant's hand	56
5.5	Response time	56
5.6	Summary	56
6	Conclusion	57
6.1	Comparison to objectives	57
6.2	Critical appraisal	57
6.2.1	Performance	57
6.2.2	Calibration	57
6.2.3	Electrical components	58
6.3	Future work	58
6.3.1	Adding the second hand	58
6.3.2	Adding a wireless communication	58
6.3.3	Adding a haptic feedback	58
6.3.4	PCB (Printed circuit board)	58
	References	59
	Appendices	62
	A. GitHub link	62
	B. Unity program	62

List of figures

Figure 1: Price range market research results	9
Figure 2: Spectra Symbol flex sensor 3.75 inches	10
Figure 3: IMU GY-521 [6]	11

Figure 4: Potentiometers a) Rotary [7], b) Slider [8]	11
Figure 5: Flex sensor glove: a) for VR [9], b) for robot hand with additional sensor for the wrist [10]	12
Figure 6: Rotary sensor glove [11]: a) Kinematics diagram, b) Final design by Ivan Petrov..	12
Figure 7: 5DT glove sensor 14 Ultra [12]	13
Figure 8: Modification to Petrov design [11] to measure multiple joints separately	13
Figure 9: Slider sensor glove [13]: joint measurement	14
Figure 10: Very similar design using IMUs: a) IGES glove [14], b) Salchow-Hömmen's IMU glove [15]	15
Figure 11: Comparing the IMU and the Vicon system [15]	16
Figure 12: Testing conditions for slider potentiometers with IMUs [13]	17
Figure 13: Joints and phalanges of the hand	18
Figure 14: Movements of the thumb [17]	19
Figure 15: Kinematic diagram for every finger and the thumb.....	21
Figure 16: Glove assembled performing the same hand position for comparison.....	21
Figure 17: Simulation of the PIP joint with the two links in blue.....	23
Figure 18: Fitted curves for the finger vs potentiometer.....	24
Figure 19: Kinematic diagram and design of an alternative solution	24
Figure 20: Circuit diagram of the glove	25
Figure 21: MCPx, MCPz and PIP sub-assembly for all fingers.....	31
Figure 22: CMCx and CMCy for the thumb movements.....	31
Figure 23: Arduino and multiplexer box.....	31
Figure 24: Assembly where all the finger parts are connected to	32
Figure 25: Complete assembly in the CAD software.....	32
Figure 26: Picture of the glove fully assembled.....	33
Figure 27: CURA file for MCPx, MCPz and PIP parts	34
Figure 28: CURA file for CMCx and CMCy parts	34
Figure 29: CURA file for the Arduino and the multiplexer boxes.....	35
Figure 30: CURA file for the hand bracelet	35
Figure 31: Flowchart of the Arduino code	38
Figure 32: Arduino software indicating the Serial port.....	39
Figure 33: First pages asking for the Serial port	39
Figure 34: Comparison between polynomial regression with 11 and 3 values.....	41
Figure 35: First step of calibration with possibility of showing webcam	42

Figure 36: Tkinter window, for Hand flat, finger close and thumb far away.	42
Figure 37: Tkinter window, for Hand halfway closed	43
Figure 38: Tkinter window, for Hand closed	43
Figure 39: Tkinter window, for Wrist folds at -45°	44
Figure 40: Tkinter window, for Wrist folds at 45°	44
Figure 41: Plot of the regression using the Python polyfit function	45
Figure 42: Tkinter window, asking for the name of the calibration	46
Figure 43: JSON file storing all the data of calibration	46
Figure 44: Initial position in simulation	50
Figure 45: Testing different hand pose in Unity without using the wrist	51
Figure 46: Test with the overlay at 30° and 60° for both joints	52
Figure 47: Accuracy test with an IMU connected to a Raspberry Pi	53
Figure 48: Comparison of measurements of the MPCx from the device and the IMU	54
Figure 49: MPCz test with major and Index closed, then separated using a 20° 3D printed spacer	55
Figure 50: Repeatability test on MCPx	55

List of Tables

Table 1: Enumeration of the Degrees Of Freedom of the hand and fingers	19
Table 2: “Potentiometer” angles collected from the Finger joint angles	23
Table 3: Glove 3D parts explanations	28
Table 4: Bill of material for one glove	36
Table 5: Final organisation of the glove data	48

List of Acronyms

Acronym	Meaning	Definition
CAD	Computer Aided Design	Set of software and geometric modelling techniques to design, test virtually - using a computer and digital simulation techniques - and produce manufactured products and the tools to manufacture them.
CMC	Carpometacarpal	Joint where the metacarpal bone of the thumb attaches to the trapezium (carpal) bone of the wrist

DIP	Distal interphalangeal	Joint situated between the proximal bones and the distal phalanges of the fingers
DOF	Degrees Of Freedom	The Degrees Of Freedom of a rigid body is defined as the number of independent movements it has.
IMU	Inertial Measurement Unit	Sensor used in navigation, capable of integrating the movements (acceleration and angular speed) to estimate the orientation (roll, pitch and heading angles), its linear speed and its position.
MCP	Metacarpophalangeal	Joint situated between the metacarpal bones and the proximal phalanges of the fingers
PCB	Printed Circuit Board	is a support, making it possible to maintain and electrically connect a set of electronic components to one another
PIP	Proximal interphalangeal	Joint situated between the intermediate bones and the proximal phalanges of the fingers
VR	Virtual Reality	Computer technology that simulates the physical presence of a user in an environment artificially generated by software

1 Introduction

1.1 Aim

Sensor gloves for hand and finger movements acquisition are often very expensive due to the difficulty required to sense the fingers joints. The price depends on the field in which the glove will be used. For instance, the gloves used in mainstream Virtual Reality (VR) games only measure “how much the finger have bent” since accuracy is not their aim, but the time response is. However, in the field of medicine, all the joints must be measured with accuracy and precision.

The aim of the project is to develop one sensor glove able to measure the all the angle of the fingers’ joints with accuracy and precision. It has to be low-cost and accessible for the general public to be assembled at home requiring only a 3D printer (an open-source project).

1.2 Objectives

- Using Microsoft Forms, writing a questionnaire to start a market research on the project area. The market research will define what people want/need and in which domain the glove will contribute the most.
- Literature reviewing some article describing the sensors and the methods to measure the human hand movements.
- Inspired by the many articles, designing the system with a suitable implementation. Of course, before testing the whole device, intermediate design ideas must be tested; one finger, then the thumb and so on.
- Assembling and programming the device, an accuracy and a repeatability test will be performed.
- Displaying a user interface to simplify the connection and the calibration process.
- Finishing the dissertation with the suggestions for improvements based on the test and the market research.
- Creating GitHub project to have a link to share the codes (and printing files).
- Writing the dissertation document and preparing the oral presentation.

1.3 Justification of hand acquisition system implementation

Using the results of the market research, a justification for the glove be found. The market research was a form with multiple question about interest potential users have in this artifact and why. First, a question asking what such a glove could be used for. The most common answers were:

- In the industry to manipulate hazardous or contaminated materials.
- For Virtual reality (VR) games or VR training simulation for dangerous operations.
- Replace game controllers and keyboards.
- Rehabilitation of the hand.
- Controlling a surgical robot at distance.
- Motion capture for the film or animation industry.

Nowadays, this type of glove is mostly known to be used for VR games, for motion capture and for hand rehabilitation [1]-[2]. However, nobody mentioned the use of the sensor glove for a sign language training system [3]. To know the interest of people in this technology, the following questions were about the interest in buying a sensor glove. 71 people answered the questionnaire, 40 would like to buy one. Next, independently from the previous answer, the question was: “*What price ranges would you be willing to pay in pounds? Or which price seems reasonable?*”. The graph bellow shows the result of the question. The chosen ranges are between 20 to 500 pounds with a peak for 50-100, and there are very few people that are willing to pay over 500 pounds.

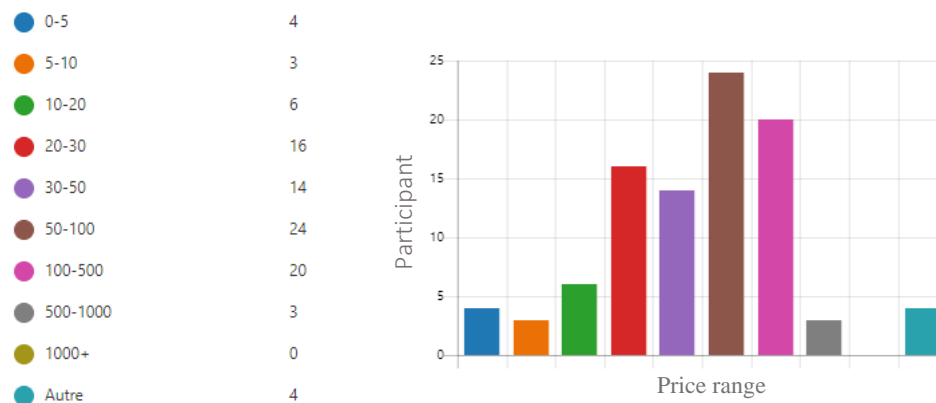


Figure 1: Price range market research results

Concerning the last question, it was asked to give a justification about the price ranges choice. Most people said that it all depends on the quality of the product (accuracy, speed, precision, material) and the use of it. For instance, for a domestic use (VR games) a small price is sufficient, but for a medical use, the price should be higher since the quality should be better. Many other said that they will not have any personal use for such a glove, which implies that they do not want to spend too much.

Lastly, since the project is open source, knowing if someone is willing to build the glove is important. In the questionnaire, it is asked if they had a 3D printer and there is only four people

has one at home. Then, the same people were asked if they were interested in printing, buying the electrical components, and assembling the glove. To this question, three said that they would try this project.

From the answers given in the market research, it seems that people have an idea of what this device is capable of. However, less than the half is not interested because they do not see a personal use. Finally, very few people own a 3D printer and only young people that sees the potential in VR games are interested in the project. Indeed, the project promises something inexpensive compared to what is proposed currently, and as seen in the result for the price ranges, from 100-500 pounds it starts to be expensive.

1.4 Dissertation organisation

The first step for any project is to search which sensor suits the most the project. Therefore, the first section will be about the many glove sensor articles which will be analysed to conclude on which sensor is appropriate for the project. Then, the dissertation will continue with the design choices and the system implementation. Next, the glove will be subject to multiple tests and the results will be discussed in this part. Finally, the dissertation will reach the conclusion, where a critical appraisal and further works will be discussed.

2 Literature review: Methods for hand movement acquisition

2.1 Sensors usually used for hand movement data acquisition

2.1.1 Flex sensor:

The flex sensor is composed of a flexible conductive ink printed on thin flexible base forming a resistor. The work principle of those sensors is similar to a potentiometer: when they are folded, their resistances change. The value of the resistance corresponds to “how much the sensor has been bent”. The unit price for this sensor is between £8-15 [4]-[5] (Figure 1), which is the highest cost sensor of all.



Figure 2: Spectra Symbol flex sensor 3.75 inches

2.1.2 IMU (Inertial Measurement Unit):

This sensor measures a body's specific force and angular rate. Therefore, they are composed of 6-axes: a 3-axis accelerometer and 3-axis gyroscope (sometimes 9-axis with a magnetometer).

To build a glove sensor, the values of accelerometer (for orientation) as well as the gyroscope (for angular speed) are required. The unit price of the IMU is around £0.5 [6] (Figure 3).



Figure 3: IMU GY-521 [6]

2.1.3 Potentiometers

Potentiometers are a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. Both slider and rotational potentiometer can be used to measure the joint angles, however, they are used in different ways. For the rotary sensor, the shaft rotates in the body of the potentiometer, whereas for the slider, the shaft slides inside the body. They are the cheapest sensor, often sold by group of 10 for less than £3 [7]-[8] (Figure 4).

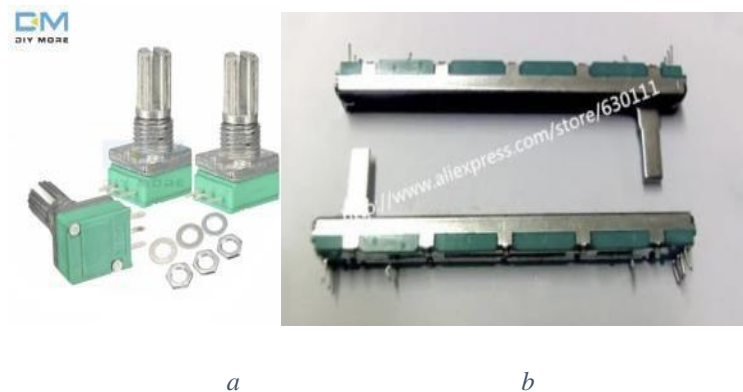


Figure 4: Potentiometers a) Rotary [7], b) Slider [8]

2.2 The designs, sensor placement and data acquisition

2.2.1 Virtual Reality acquisition system

The flex sensor is widely known for this type of glove, the sensors were used in the first glove sensor created for video games: The Power Glove by Nintendo. Therefore, the design is similar with the sensors glued or stitched to a stretchable glove at each finger from the start to the end of the finger (Figure 5).

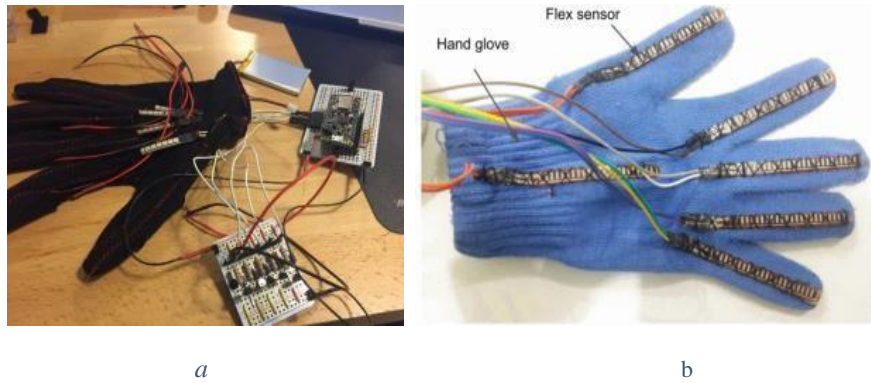


Figure 5: Flex sensor glove: a) for VR [9], b) for robot hand with additional sensor for the wrist [10]

This type of glove is most likely used for VR because it uses only 5 flex sensors which only measure how much the fingers have been bent. The rotary potentiometer can also be used to control the bent angle as well, by 3D printing two links linked by a revolute joint [11] (Figure 6) with:

- the first link fixed to the potentiometer's shaft and the potentiometer's body is embedded to the proximal phalange.
- the second link forms a revolute joint with a part embedded to the distal phalange.

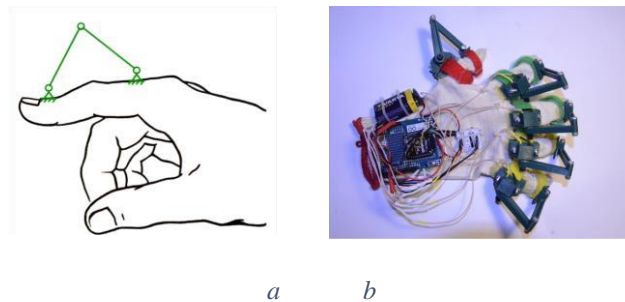


Figure 6: Rotary sensor glove [11]: a) Kinematics diagram, b) Final design by Ivan Petrov

2.2.2 Measuring more joints with higher accuracy

Now, if the glove must be used for data analysis with higher accuracy and with more joints measured, increasing the number of sensors is required. Moreover, they need to be placed in a different configuration with a specific placement on the hand. For instance, the company 5DT created the Data Glove ultra 14 in Figure 7 [12] with two sensors for each finger (positioned on the joints), which allows the glove to measure the joints separately. There are also 4 additional flex sensors between every finger to detect the fingers movements.



Figure 7: 5DT glove sensor 14 Ultra [12]

Therefore, by applying the same reasoning, adding 2 sensors for each finger to Ivan Petrov's glove design improves the number of movements detected by the glove (see Figure 8).

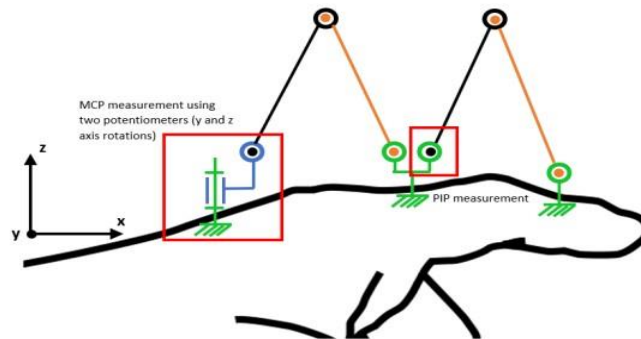


Figure 8: Modification to Petrov design [11] to measure multiple joints separately

Therefore, by adding more sensor, more joints can be measured, but the glove will be more expensive of course. Positioning the sensor on the phalanges is the chosen method for the IMU and the slider potentiometer. An article [13] shows a project that uses slider potentiometers, springs and flexible wires (fishing lines) to measure only the flexion/extension of the 5 fingers. The proposed system is showed in Fig. 9 where only one finger is explained for clarity. One side of the wire is attached to the pole of the slider potentiometer whereas the other end is attached to the middle of the phalange.

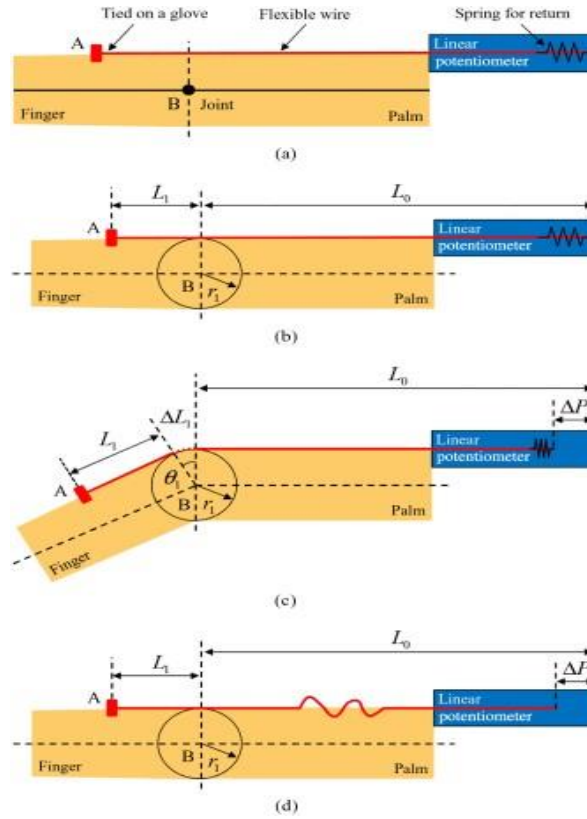


Figure 9: Slider sensor glove [13]: joint measurement

The angle is computed by measuring the potentiometer value difference which is equivalent to the value of displacement and knowing the radius of the finger joint which is equal to half the thickness of the finger, the joint angle is deduced. To compute (or deduce) the angle of the joint, the sensor displacement of the previous joint must be considered, as the string will pass through both joints. However, the fingers can move to the right and left but also folding.

Using IMU, measuring all the movement of the joints can be measured more easily. The iSEG-Glove [14] shown in Fig. 10(a) and the glove developed by Salchow-Hömmen *et al* [15] in Fig. 10(b) have the same approach: 15 IMUs are positioned on the phalanges of each finger to measure their orientation and an additional IMU is placed on the palm of the hand to get the relative orientation and velocity of the phalanges compared to the hand. By measuring the angle of each IMU and computing the difference in angle between two sensors next to each other, the orientation of the joints is obtained.



Figure 10: Very similar design using IMUs: a) IGES glove [14], b) Salchow-Hömmen's IMU glove [15]

2.3 Data processing

Now that the values are obtained from the sensors, the next step is to process the values. To do so, a calibration is often used to improve the glove accuracy and define some initial positions. Furthermore, some sensors require filtering since they are sensitive to noise.

2.3.1 Calibration

A calibration phase must be performed before using the gloves for its purpose. A calibration routine has been created for every glove to improve the global accuracy of the sensors or simply make it work properly. The glove wearer must position its fingers in specific poses, with each pose placing a finger joint group and relevant data glove sensors at their minimum and maximum boundaries (opened hand and closed hand).

A calibration is necessary if the glove is worn by someone else because the boundaries will be different. To avoid calibrating every time, a “calibration profile” for each user can be saved. However, the IMUs require a calibration before usage (by placing the sensor on a levelled surface) to reduce the errors.

2.3.2 Filtering the noise and signal drift

The data acquired must be treated because many sensors are subject to noise:

- Resistive elements can have signal drift issues.
- IMU's gyroscope suffers of Angle Random Walk which is a high frequency noise due to thermoelectrical reactions. Whereas the mechanical noise of the accelerometer sensor

comes from thermomechanical noise. Finally, the two IMU's sensors can have a bias corresponding to an offset at its output.

By implementing filters before interpreting values, the data can be smoothed and corrected. To remove the high frequency sensor noise, they must use a low-pass filter, and a high-pass filter to block low frequency noise. To solve signal drift issues, they removed an estimated error calculated by getting the average error beforehand.

2.4 Testing methods

Accuracy and Precision (repeatability) tests must be performed to find the performances of the sensors.

2.5 Accuracy

Measuring the accuracy is difficult because the data from the device must be compared to the real data. One solution is to compare the sensors between them. The gloves can be compared with a motion capture device such as the Vicon MX motion. Or the Humanware Humanglove (UG: Universal Goniometric) with Hall effect sensors, which is one of the most precise glove sensors for a long time. However, the glove requires a mould of the user's hand, and it also requires a tedious amount of calibration. This is the reason this sensor is not consider in the paper, but it can be used as a reference for testing.

The Salchow-Hömmen *et al* [15] show the experiment performed in Figure 11: they fixed the forearm to prevent any disturbance in the motion capture and captured the Vicon and the IMU glove at the same time. In this experiment, the reference is the motion capture system.

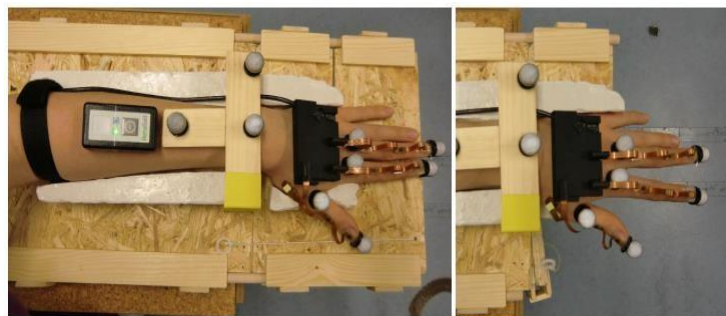


Figure 11: Comparing the IMU and the Vicon system [15]

A similar approach is used for the slider potentiometer [13] in Figure 12. In that case, the IMU is considered as a reference. The test will consist of folding the fingers multiple times, with the hand elevated to be able to fold the finger.

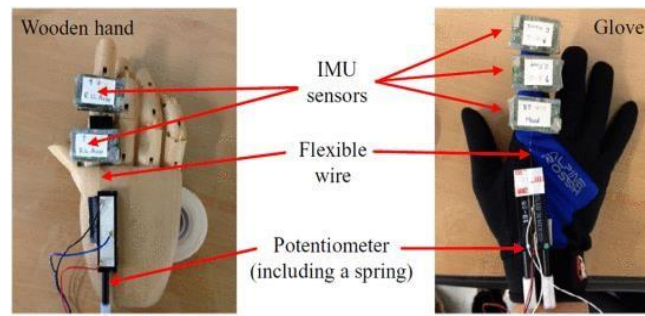


Figure 12: Testing conditions for slider potentiometers with IMUs [13]

To obtain the best results, the experiment must be performed with multiple devices at the same time on the same hand. Otherwise, it is really complicated to reproduce the same scenarios or finger movement for comparison.

2.5.1 Repeatability

Repeatability is the variation in measurements obtained when one person measures the same unit with the same measuring equipment. For a glove sensor, the values should not shift as the time goes. Two tests can be performed to control the precision of the glove sensors by [14]: “The ‘flat hand’ test which examines each data glove’s ability to maintain a minimum repeatable value after full stretch of each data glove sensor. “and “The ‘plaster mould’ test examines the ability of each data glove to reproduce angular readings when positioned in a repeatable position.”.

2.6 Conclusion

To conclude, all the sensors can be used to build a sensor glove, but the goal was to find the lower cost sensor that can provide an acceptable accuracy and that offers the capability of measuring all the joints with all their possible movements. The largely used flex sensor is too expensive to be considered as low cost, exceeding 100£ just for a VR glove whilst being the easier to implement. The IMU remains the most precise and reliable sensor for any application (VR or medicine) and the sensors are slightly more expensive than the potentiometers. Although, it must be calibrated frequently to keep their accuracy and the postprocessing of the data is more tedious. Finally, the potentiometers are the cheapest sensors, but existing projects are often prototypes and very few tests/ results are provided to conclude on their performances. Therefore, using potentiometers with Petrov’s method can be the best way to find an inexpensive solution to complete the objectives and discover the results. Some part of the design can be similar, though, the calibration and the post-processing should be entirely modified to fit the aim and the objectives.

3 Hand Movement Acquisition: Design and System Implementation

After choosing the potentiometer to measure the hand and finger movements, the design of the glove will be found. The glove will be composed of many 3D parts develop in a Computer Aided printed (CAD) software. The part will then be 3D printed and tested. To test, an adequate program will be developed, as well as the circuitry. Finally, a bill of material containing all the necessary component to assemble the glove will be established to get the cost of the project.

3.1 Design considerations and technical specifications

The goal is to measure angles the finger joint angle with the lowest cost sensor: the potentiometers. A potentiometer is a three-terminal resistor with a rotating contact that forms an adjustable voltage divider. The potentiometer's shaft can rotate around itself in the potentiometer's body, therefore, the potentiometer has only one Degree of Freedom. The Degrees of freedom (DOF) of a rigid body is defined as the number of independent movements it has.

Each finger has 3 phalanges: proximal, intermediate and the distal phalanx which move with two joints called proximal inter phalange (PIP) and distal inter phalange (DIP). However, moving the finger individually is more complex as they move with a freer joint from the hand called the metacarpal phalange (MCP). The thumb has an additional bone, the Trapezium, which allows it to move more freely than the others, the joints is called the carpometacarpal (CMC) joint. The DIP and the PIP are the only joints with one degree of freedom, whereas the MPC and the CMC have more than one DOF (MPC has 2 DOF and CMC has 3 DOF). All the joints and phalanges are represented in Figure 13.

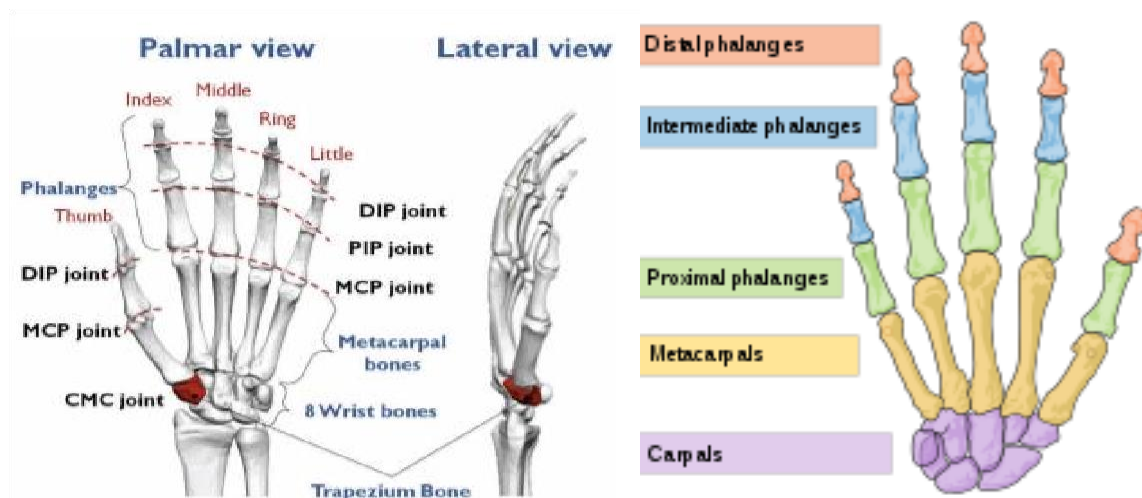


Figure 13: Joints and phalanges of the hand

Due to the small area that represents the hand and the fingers, measuring accurately the hand movement becomes a real challenge. And unfortunately, that makes the devices often very expensive for the general public. The potentiometers can measure only one degree of freedom, so a clever design to sense movements on multiple axis must be found. There is a potentiometer for every DOF in the hand:

Table 1: Enumeration of the Degrees Of Freedom of the hand and fingers

Joints	DOF	Location	Total DOF
DIP	1	Every finger	$5 \times 1 = 5$
PIP	1	Every finger except the thumb	$4 \times 1 = 4$
MCP	2	Every finger	$5 \times 2 = 10$
Wrist	2	Only the wrist	2
CMC	3	Only the thumb	$1 \times 3 = 3$

The total number of DOF for the fingers is 22, and 25 with the wrist, which is enormous. The objective would be to reduce the number of potentiometers used to reduce the cost and the weight of the device to be more comfortable for the user. Hrabia *et al* [16] conducted an experiment to find a relationship between the DIP and the PIP. The movement of the distal phalange is indeed related on the movement of the intermediate because the two are moved by the same muscle. Therefore, after experimenting and multiple people and by using a normal distribution to verify the results, they found an approximated equation relating the two joints for every finger (except the thumb):

$$\Delta DIP = 0.88 \Delta PIP \text{ for all four fingers (with SD=0.10, R2=0.77)}$$

From this estimation, the DIP is ignored which reduce the number measurements on the four fingers. Another joint where a DOF that can be ignored is the CMC of the thumb. The CMC, with 3 DOF, allows the thumb to perform a wide variety of movements.



Figure 14: Movements of the thumb [17]

This will be the most tricky and difficult joint to measure. Therefore, a relation between the CMC and MPC of the thumb concerning the abduction and adduction is found. The movement of the CMC and MPC on the right and left are performed on the z axis, as shown in the Figure 15. The following relationship has been tested in the Part 5.1 of this paper:

$$\Delta CMC_z = \frac{2}{3} \Delta MPC_z$$

Finally, concerning the wrist, even though the hand has 3 DOF compared to the arm, the wrist can be reduced to 2 DOF. Indeed, from the arm the hand can rotate around 3 different axes, but the wrist offers only 2 rotations. The last rotation is operated by the arm which rotates around itself. Therefore, this rotation is ignored for our actuation system. **Therefore, the total number of DOF that can be measured is reduced to 19 DOF.**

Lastly, the potentiometers are not the smallest (10x11x22 cm) but looking at the size of the fingers, the size of these potentiometers will not be causing problems. However, it is important to remind that the aim is to have a comfortable glove for a long use and adaptable to every size of hand.

Electronically speaking, the **Arduino Nano** is the microcontroller used to collect the data from the potentiometers because of its small size. However, as mentioned before, the Arduino must read 19 potentiometers and there are only 8 analogue pins available on the Arduino. Therefore, the use of a multiplexer is required, which is an electric component with 16 analogue inputs/outputs. The multiplexer pins can be read separately by sending a binary number indicating which pin to read. Finally, the data will be collected by the Arduino and send to the computer.

3.2 Mechanical design (kinematic diagram)

3.2.1 Kinetic diagram

A kinematic diagram is a drawing of the joints which connect the links together. It is used to help to understand how a mechanical system moves. Previously in part 3.1, the terminology “number of DOF” was used, but for the kinematic diagram, it is better to use mechanical joint. A joint -or mechanical joint- is a section of a machine that connects two a link/part to another, the name of the joint depends on the number of Degree of Freedom it provides. For instance, the potentiometer has one DOF, it is therefore called considered as a **Revolute joint**. The MPC and wrist have 2 DOF which represent a **Universal joint**. The CMC has 3 DOF which

correspond to a **Ball joint**. In Figure 15, the kinematic diagram of the solution is shown and Figure 16 shows the result glove for comparison.

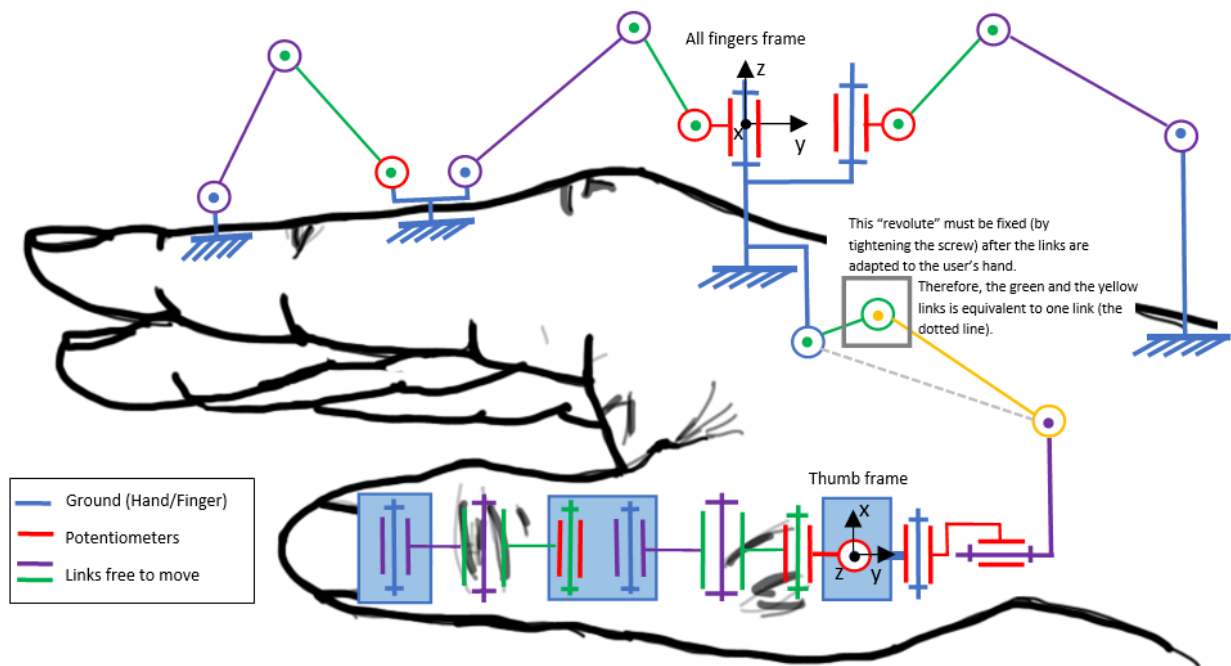


Figure 15: Kinematic diagram for every finger and the thumb

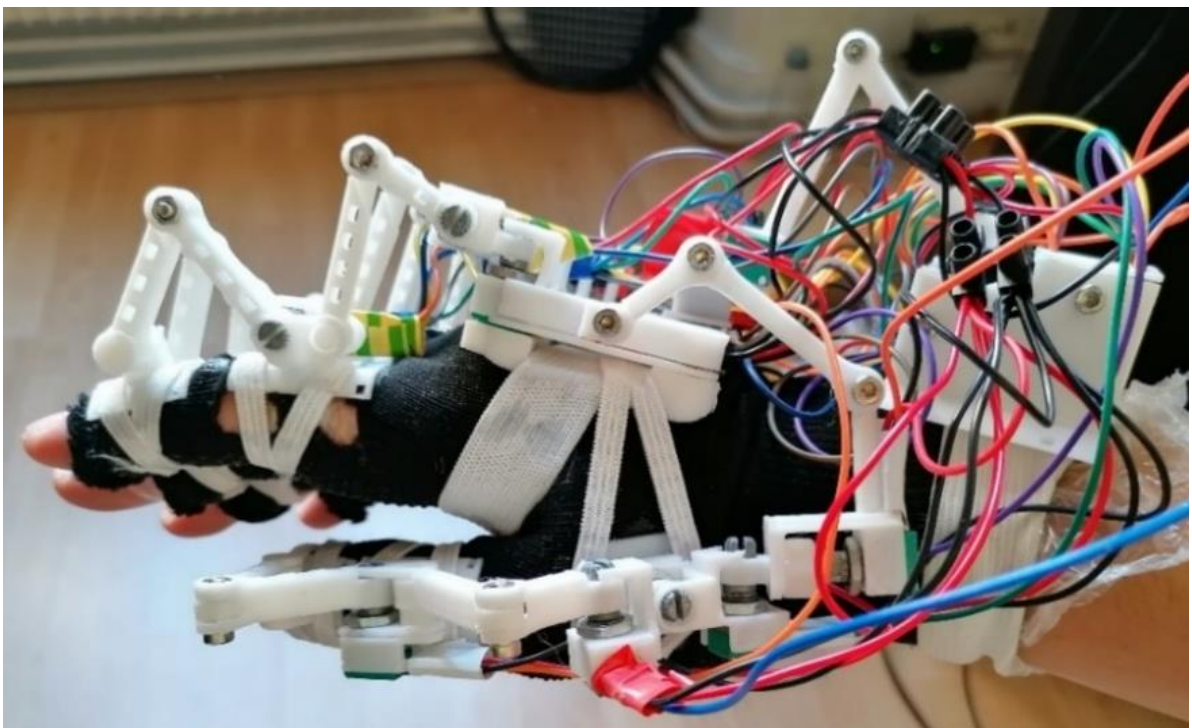


Figure 16: Glove assembled performing the same hand position for comparison

The diagram is represented in 2D with two different planes and frames, one frame for the fingers and one for the thumb. It is composed of:

- **mechanical joints:** In our case, the drawing is exclusively composed of revolute, that can be represented as a circle or by a cylinder trapped between two lines (because a revolute joint can only rotate around the axis).
- **the body or ground:** the ground is the part of a system that does not move, it is fixed. In our case the grounds are the hand and the fingers (the phalanges in particular).
- **links:** they are straight lines that link two joints together.

3.2.2 Link movements and angles relations

As stated in the Design considerations, the device's sensor for the fingers' joints are potentiometers which implies that only revolute joints can be utilize. Since the DIP and PIP joints are revolute, in theory, the measurement of these joint should be easier. In reality, to measure the angle between two phalanges, the potentiometer's axis should be situated at the same location where the finger folds. Unfortunately, this solution is possible only on the side of the finger, which is only possible for the index and the pinkie. The only location where the movement of the user will not be contained is on the top of the phalanges. In addition, the glove must be comfortable and adaptable to every hand size. After testing multiple design and solutions, a similar design to Ivan Petrov's work has been found (at least for the joints in the x axis). Therefore, the final chosen design should have been exploitable. However, in the article, he measured the bend of the fingers by calibrating a start and end position, whereas the goal of the project is to measure the finger joints' angle values. From this design, two types of relation between the angles of the joints and the sensor values can be found:

- **The linear relations:** The value obtained by the potentiometer will be the angle at which the joint moved, which is basically an equality. This happens only when the potentiometer is the only revolute rotating around an axis. From the Figure 15, only the MCP in the z axis (reduced to MCP_z), the CMC in the x and y axis (reduced to CMC_x and CMC_y respectively) and wrist in the z axis (called $wrist_z$) are in this case.
- **The second-degree relations:** to measure the PIP and MCP_x , $wrist_x$ and CMC_x which all rotate around x in their frame, adding a revolute joint was necessary. Indeed, unlike the MCP_z , the when the finger folds, the link will hit the finger. Moreover, the distance between two phalanges will change as the finger folds.

The final design is two revolute joints fixed the phalanges near to the joint. One of the phalanges' revolute is a potentiometer with a link attached to it. Another link is connected to the previous link and to the phalanges as shown in Figure 15. Since there are two revolute joints

moving together, a linear relationship does not exist anymore. Instead, there is a polynomial relation, a second degree in particular. This assumption can be verified using Fusion 360 by simulating the joints and links attached to the finger as shown in Figure 17.

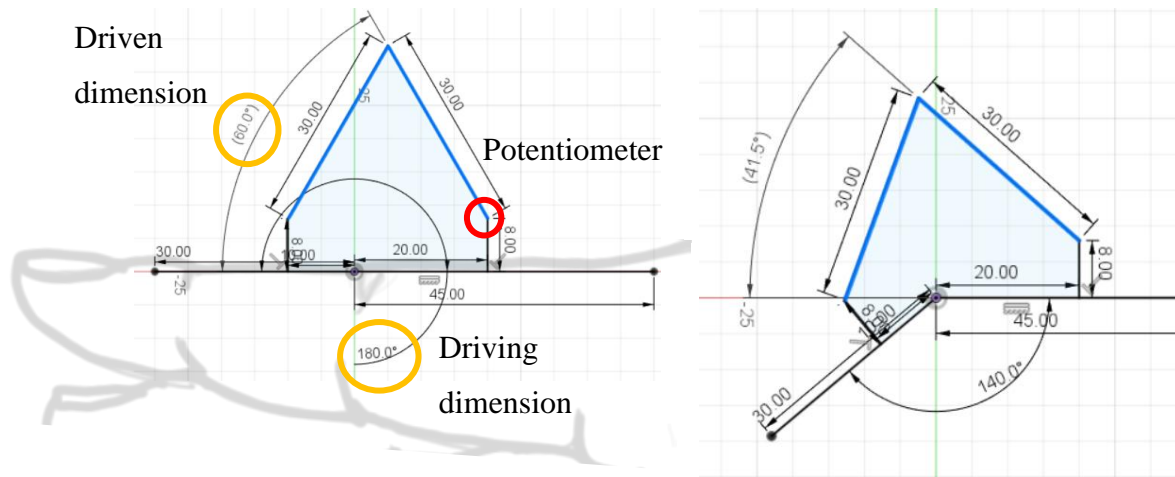


Figure 17: Simulation of the PIP joint with the two links in blue

Thanks to the simulation, by changing the driving dimension, the finger folding is simulated. The potentiometer values can be collected with a driven dimension in FUSION 360. By bending the finger at different angles and by collecting the values obtained by the “potentiometer”, a table comparing the two values has been created in Table 2. Of course, this is just a simulation, in reality, the data obtained from the potentiometer is between 0 and 1023.

Table 2: “Potentiometer” angles collected from the Finger joint angles

Finger joint angles	190	180	170	160	150	140	130	120	110	100	90	80
Potentiometer angle	64.9	60	55.2	50.4	45.8	41.5	37.3	33.4	29.9	26.6	23.6	20.9

The table values are used to plot a scatter plot in Excel. From the scatter plot, an option to estimate an equation of that would go through every point is proposed, this is called a **regression**. The choice concerning the type of regression is wide: linear, polynomial, exponential, logarithm and so on. The best result was obtained with the polynomial regression with a degree 2, as shown in the Figure 18.

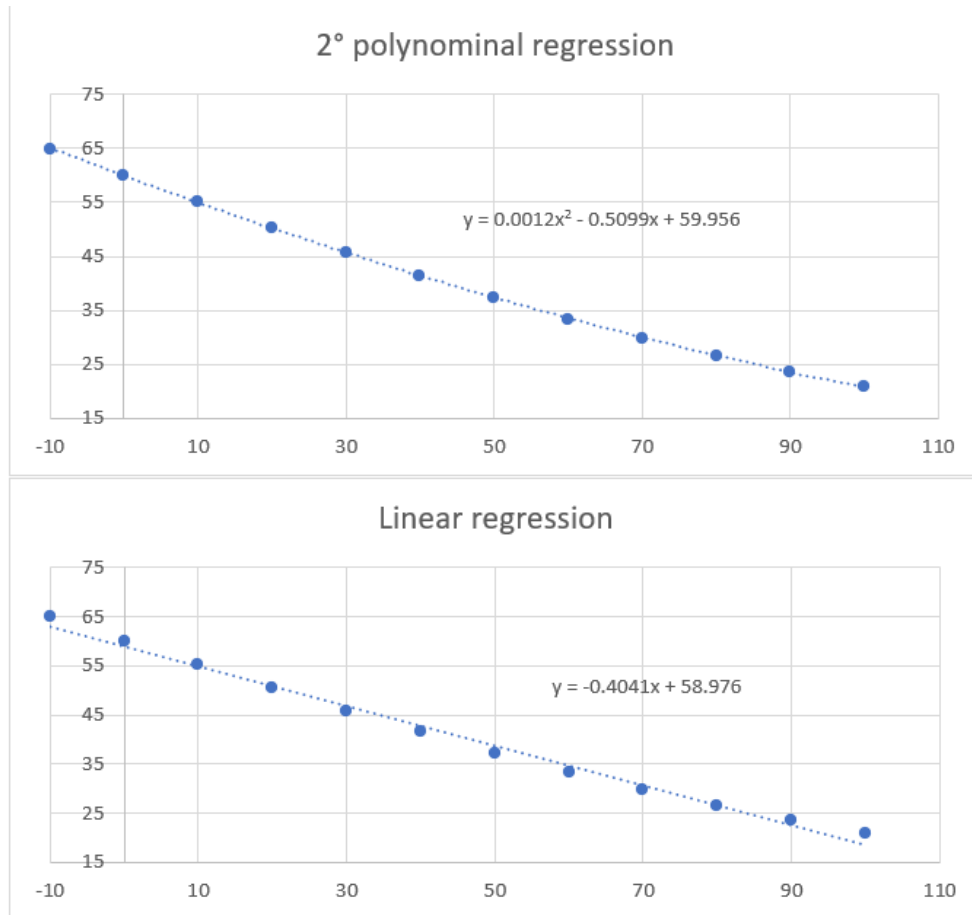


Figure 18: Fitted curves for the finger vs potentiometer

3.2.3 Alternative design

There was one possible design to keep a real linear relationship using another 1 DOF mechanical joint widely used in robotic: the prismatic joint. Compared to the revolute, the prismatic slides without rotating, whereas the revolute rotates without sliding, but both move around one axis. The prismatic was useful to create a linear movement in one direction.

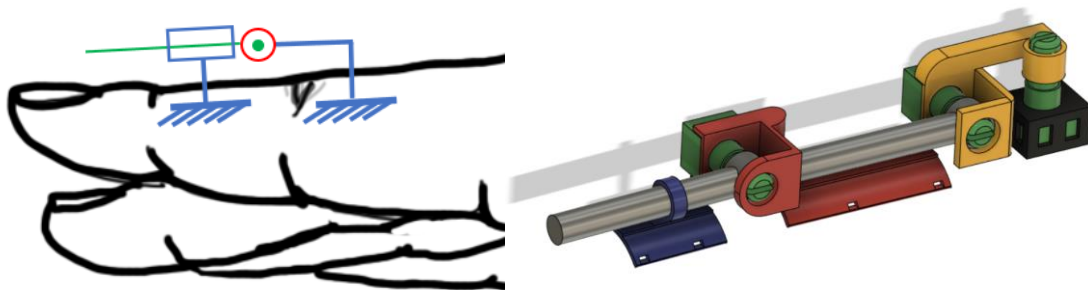


Figure 19: Kinematic diagram and design of an alternative solution

However, after testing the solution in Figure 19, the parts were moving too much on the finger and the design would not fit the finger nor fit every hand size.

3.3 Electronic requirements

Now that the measurement design has been decided, the circuit diagram must be drawn before wiring anything. Once the circuit is ready, wiring the components together is easy, and before powering the system, a verification of the circuit must be done.

3.3.1 Circuit diagram

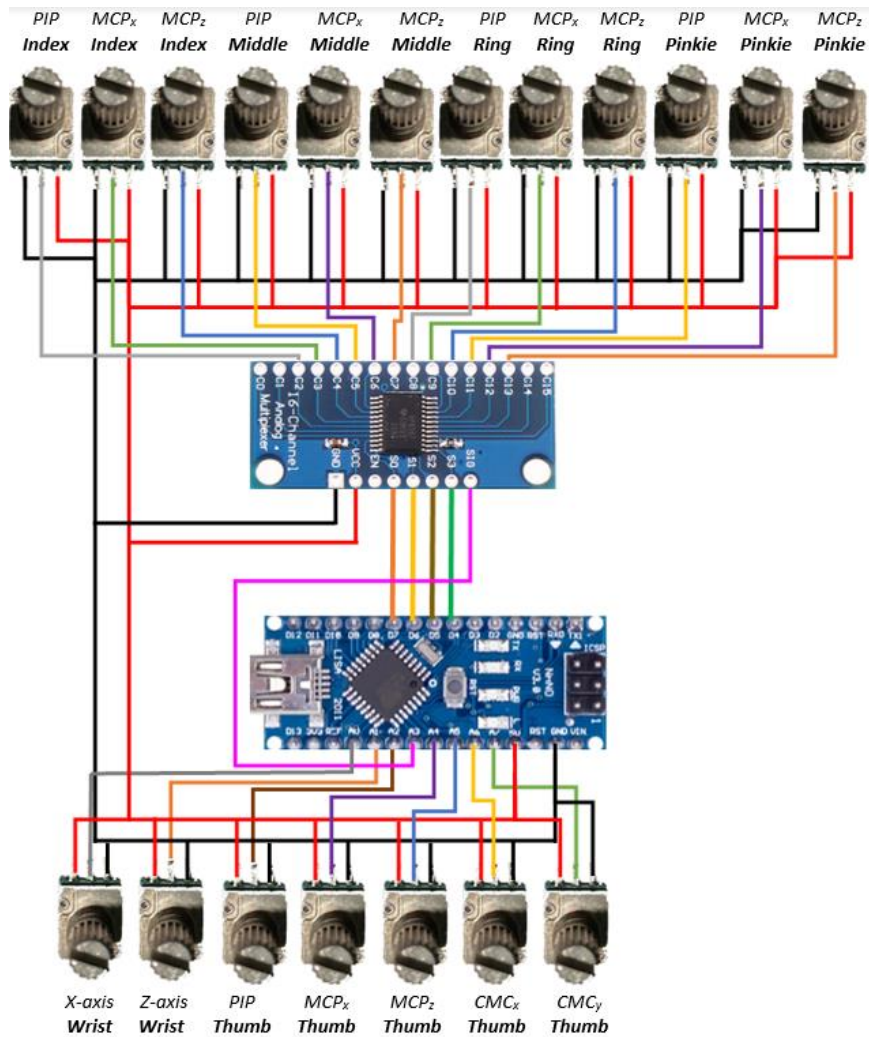


Figure 20: Circuit diagram of the glove

The diagram looks messy, but at least in theory, the circuit is simple. Indeed, every electric component in this circuit are powered to the Arduino in **parallel**. It implies that every component will be supplied with the same voltage, but the current is variable. Therefore, the current passing through the variable resistances can be measured as:

$$U = RI \leftrightarrow I = \frac{U}{R}$$

with U the tension/voltage in Volts, R the resistance in Ohms and I the intensity/current in Amper.

The current is measured using the analogue pins of the Arduino that can read numerical data or with the Multiplexer which has the same purpose.

3.3.2 Power consumption

The circuit diagram is finished, however, before wiring, the power consumption must be found to ensure that the Arduino can supply all the components. Every component in the circuit is in parallel that means that all the components are supply with a tension of 5V. However, the current drawn by the components must be determined. The components that draw current are:

- **19 potentiometers:**

The potentiometers' resistance is $R_{1\ pot} = 10\ k\Omega$, since there are 19 potentiometers:

$$I_{19\ pot} = n \times I = n \times \frac{U}{R} = 19 \times \frac{5}{10000} = 0.0095A = \mathbf{9.5mA}$$

- **Multiplexer:** The 74HC4067 is a CMOS logic chip which contains CMOS logic. Having a very low static current is a property of this kind of logic (inputs fixed at one or zero) as shown in the see datasheet [18], page 4: $I_{cc} < 160\ \mu A$

$$I_{MUX} = \mathbf{160\ \mu A}$$

From the current draw by every component, the total current of the circuit can be deducted:

$$I_{Tot} = I_{19\ pot} + I_{MUX} + 9.5mA + 160\ \mu A = \mathbf{10.16\ mA}$$

And the power consumption:

$$P = I_{Tot} \times 1\ \text{hour} = 0.01016A \times 3600s = \mathbf{36.576A.h}$$

The current provided by the Arduino pin 5V/Ground when powered through USB is approximately **400mA** (according to the documentation of the Arduino nano), whilst all the components consume only 10.16mA, which represent only 2 percent. Therefore, the Arduino will largely be able to power all the components. Furthermore, if the device would have been wireless and supplied by a battery, 35.576 A.h would a small consumption.

3.3.3 Connection within the assembly

All the potentiometers' terminals are wired before assembling, but the pins of the Arduino or the multiplexer are not connected yet. The assembling does not required soldering, although it is recommended; indeed, the wires are Dupont connectors (Jumpers wires). However, with the small size of the potentiometer, the terminal pins are very small, and the wires have not a good connection. Therefore, their plastic housings are removed and to fix the wire to the terminals, the head of the wire is crimped to the potentiometer's terminal. To avoid shortening the circuit, electrical tape separates the wires. The other head of the wire still has the housing because they will connect to the Arduino and multiplexer which have standard pins. Once all the potentiometers are fixed to the glove, the data terminals of the potentiometers can be wired to the Arduino and Mux. Moreover, the alimentation terminals can be attached together using small terminal strips, joining at some point the 5V and GND pins on the Arduino as shown in Figure 26. Of course, solder the pins and the wires is the best solution to ensure a more reliable connection, but this requires a good experience in soldering. Furthermore, some components such as the multiplexer and the IMU cannot be bought with the pins soldered. It was not particularly difficult to solder the pins, but for the rest of the project, continuing without soldering was important. Indeed, since the project is open source, it should have multiple options depending on the person's equipment. With some research, and depending on the country, those parts can be bought on other sites with the pins soldered for a reasonable price.

3.4 Software requirements

To realise this artifact, three software were required: **Arduino** to collect the data, **Python** to process the data and design an interface, and finally **Unity** to display a simulation of a hand replication see Part 5.1 Testing with Unity . However, since Unity is a cross-platform game engine, a game can be build and run as an application *.exe*. Therefore, except if the user wants to modify or create a game from Unity, Unity should not be required. Arduino and Python are required to the good functioning of the glove. Moreover, Python needs some libraries that are not initially installed, this is the list of libraries:

- **Tkinter**: used to show a window with interactive widgets such as button, entries, and images.
- **Socket**: used to communicate between two devices wirelessly or between two languages (C# and Python for instance)
- **NumPy**: a well-known library with many mathematics functions such as *polyfit* which will be used to perform polynomial regressions.

- **OpenCV**: library specialised in real-time image processing.

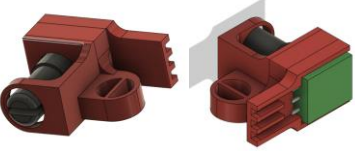
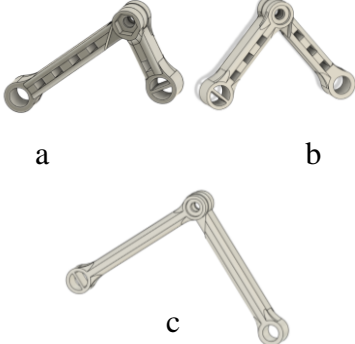
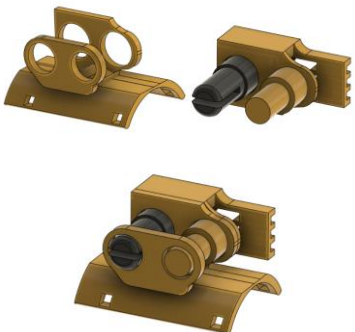
3.5 3D printed Glove


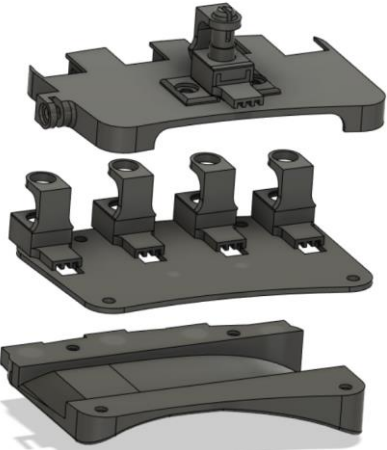
The objective of the 3D modelling process was to facilitate the assembly as much possible. The glove is entirely composed of 3D printed parts attached with strip band, with or it can be sewed. Basically, the goal is to print the parts, remove the eventual supports and assembling it without any post processing.


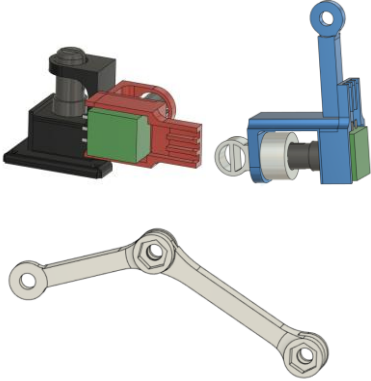
3.5.1 The different parts

The 3D models were generated with the educational version of Fusion 360, a Computer Aided Design (CAD) software. All the parts presented here will be 3D printed, and some parts such as the potentiometers are here to visualise better how the system works. All the potentiometers are fixed to the part using the nut provided with them. The following Table 3 explains all the different parts of the glove.

Table 3: Glove 3D parts explanations

	<p>This part holds the potentiometer that is measuring the angles of the MPC on the X-axis. The circular shape with the bar in the middle is attached to the potentiometer that measures MCP on the Z-axis.</p>
 <p>a</p> <p>b</p> <p>c</p>	<p>Those links attached to the potentiometers on one side and to the other phalange on the other. They are be assembled using a screw and a nut, slightly tighten to allow the rotation.</p> <p>a. links are attached to the MCPx potentiometer</p> <p>b. links are attached to the PIP potentiometer</p> <p>c. links are attached to the wristx potentiometer</p>
	<p>With its semi-cylindrical shape, this part is fixed to the proximal phalange. There are holes so that it can be sewed to the glove. Or it can be glued on the glove.</p> <p>The upper section of this part holds the potentiometer that will measure the angles of the PIP. The shaft near the potentiometer is here to attach the links above.</p>

	<p>With its semi-cylindrical shape, this part will be fixed to the proximal phalange. There are holes so that it can be sewed to the glove. Or it can be glued on the glove.</p> <p>The second part of the links (without the bar in the middle) will be attached to the pin. To embed the pin, it needs to be forced into the hole.</p>
	<p>The main body of the device is this part. It is separated in three parts:</p> <ol style="list-style-type: none"> 1. The first part is the cover which is screwed to the other pieces using screws and nuts. There are holes with hexagonal shapes to embed the nuts. Moreover, there are two screw holes' size on the middle of the cover to attach the potentiometer's holder of the wristz. Lastly, a huge hole is here to let the wires through. 2. The second part holds the potentiometers' holders of the MCPz. There are four of them for each finger. For the thumb, on the left of the cover, a screw hole is here to attach the CMC parts. 3. The last part must be attached to the hand which explains its round shape. A rounded extrusions is present in the centre of the part to allow an elastic strap to be stuck into.

	<p>This is the box which contains the electronic components: the Arduino and the multiplexer.</p> <p>The parts on top are for the Multiplexer, the wires need to be connected, then the lid can be screwed, and the wires are trapped within the box.</p> <p>The part on the bottom is for the Arduino but the Multiplexer's box as well. The Multiplexer box is fixed to the bigger box and the Arduino fits in the back. The lids (middle) have a bigger hole to allow the wires to go in the slot.</p>
	<p>The parts of for the CMC joint are similar to the others but with some modifications.</p> <ul style="list-style-type: none"> • The black part is attached to the thumb metacarpal phalange. • The links are meant to be one, since the joint in the middle is supposed to be an embedded joint (0 DOF). To do so, the screw and the nut should be tightened as much as possible. The aim of this joint is the possibility to adapt to the user hand. • The CMCy (blue and white part) is the only part measuring on the y axis, thus the part is different from the others.

3.5.2 The assembly

The assembly consists in assembly every parts together. The assembly is composed of 4 sub-assembly:

1. The MCP_x , MCP_z and PIP sub-assembly for all fingers (Figure 21). The thumb also has the same parts but reversed and the MCP_z has a modification. This is the first prototype of the whole glove because it can measure one finger already. Therefore, once this assembly have been validated by testing with Unity (see part 5.1), the glove has been designed around the same concept:

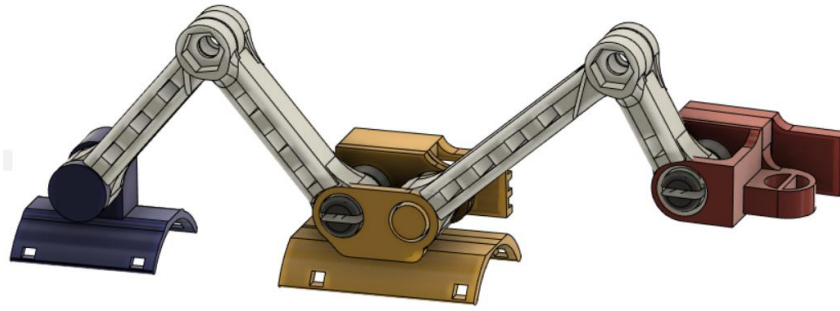


Figure 21: MCP_x, MCP_z and PIP sub-assembly for all fingers

2. The CMC_x and CMC_y for the thumb movements (Figure 22):

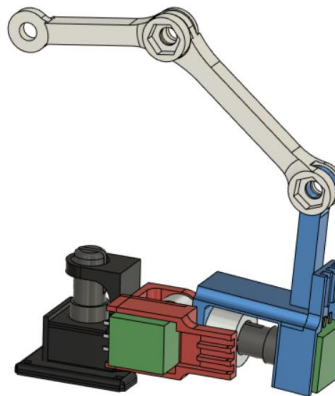


Figure 22: CMC_x and CMC_y for the thumb movements

3. The Arduino and multiplexer box. The pin that holds the link can be printed with the multiplexer box or it can be glued to the box afterwards (Figure 23):

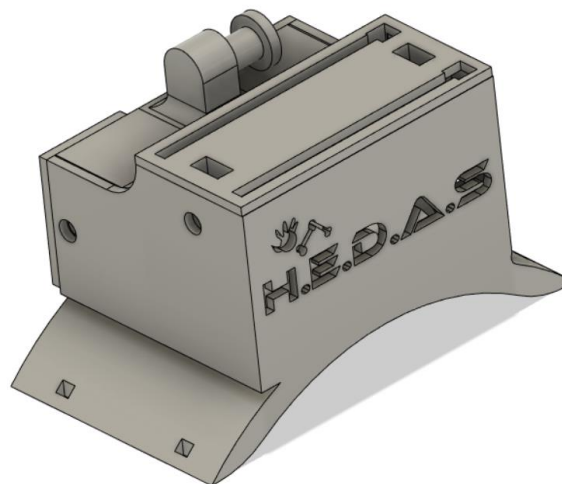


Figure 23: Arduino and multiplexer box

4. The body of the glove, where all the previous sub assembly are connected to (Figure 24):

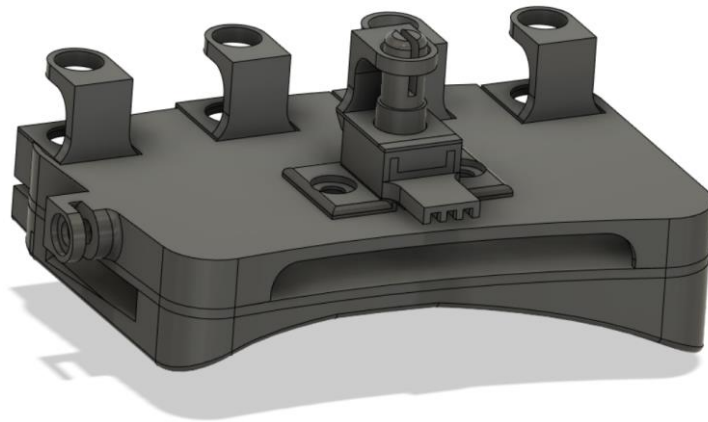


Figure 24: Assembly where all the finger parts are connected to

Finally, the assembly (Figure 25) without the fabric glove should look like this (the PIP and MCP parts are normally reversed):

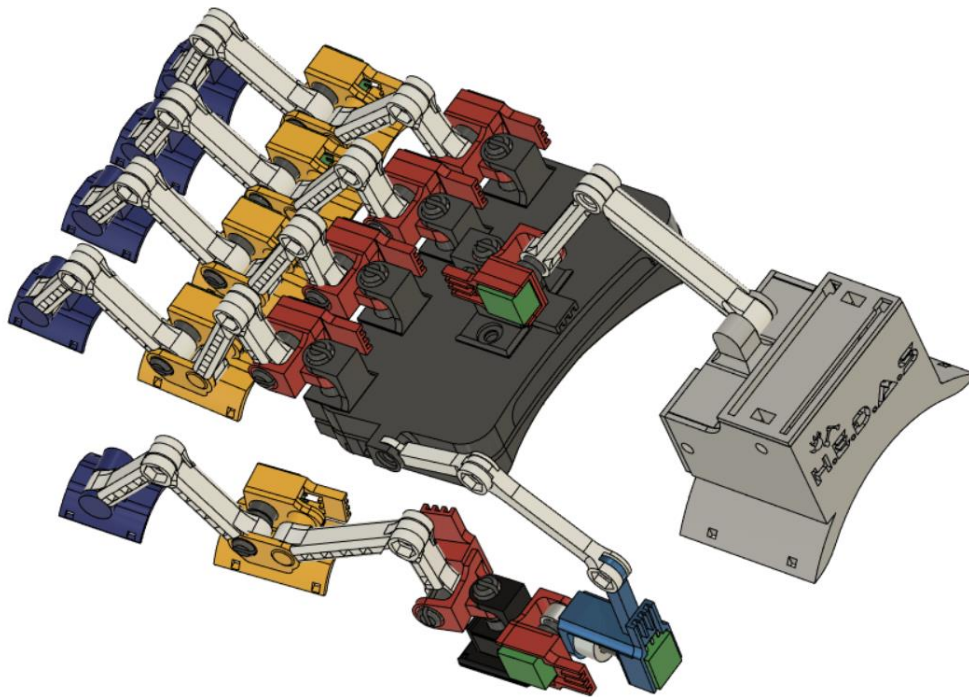


Figure 25: Complete assembly in the CAD software

The picture bellow is the real glove assembled on the fabric glove using glue and elastic bands. The parts can also be fixed to the glove by sewing them (notice the holes in every part attached to the glove). However, the glove is too elastic therefore, the sewing method is not appropriate. With a stiffer glove, this method could work:

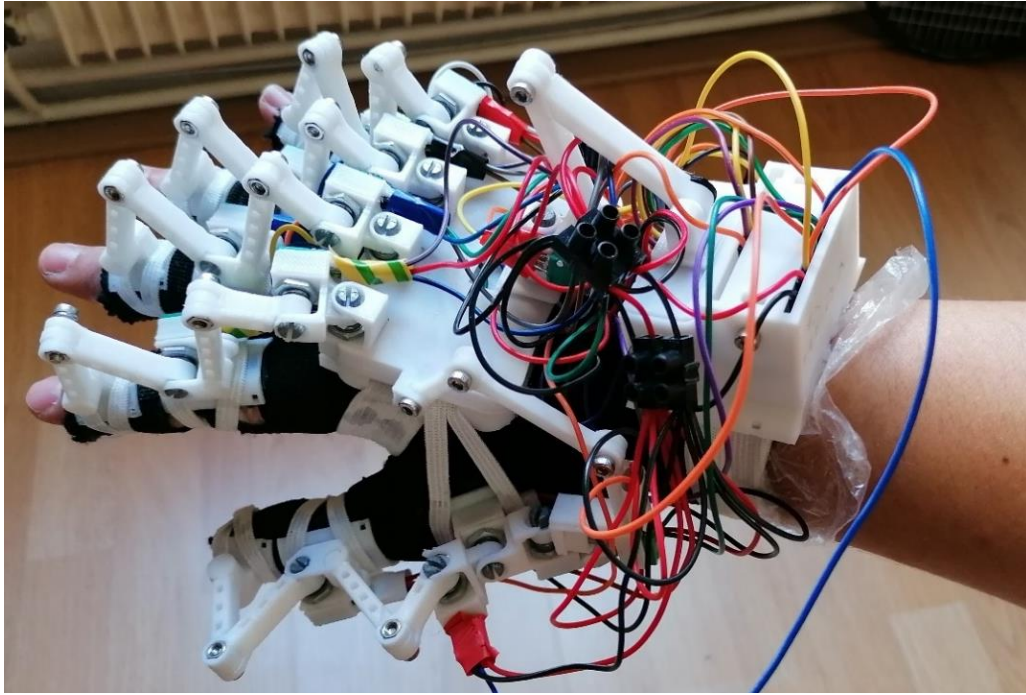


Figure 26: Picture of the glove fully assembled

3.5.3 The printer and the settings

The printer used to print all the part was the **Crealty Ender 3**, a low-cost printer with an acceptable accuracy. The software used to slice the 3D files is Ultimaker Cura. This software is called a slicer because it will divide the part in layers. The software offers a variety of options to enhance the printing experience such as the layer height, wall thickness, support and so on... All the part can be printed with a layer height of 0.28 (the lowest resolution) which is sufficient but the circular part of the potentiometers' holder containing the links can break more easily since they are very slim. However, it is not important since a small area stopping the link from moving is required. Even if the parts are relatively small, the printing time was quite long, therefore, printing with a high layer, lowers the printing time. But the prints often lose accuracy and their strength in some area as well. The parts were printed with a PLA filament, they had a great strength despite the thin shell. Of course, the parts could have been printed in another material with a higher yield strength such as ABS or PETG, but they are often more difficult to print. The potentiometers' nuts were tightened directly on the part, and the parts never broke. Furthermore, when the finger folds, the links are subjected to some forces, but they are strong and never broke.

The printing can be divided in 4 parts. The following screenshot are from the Cura, and they show the different sub-assembly printed together:

1. The MCP_x, MCP_z and PIP parts (Figure 27):

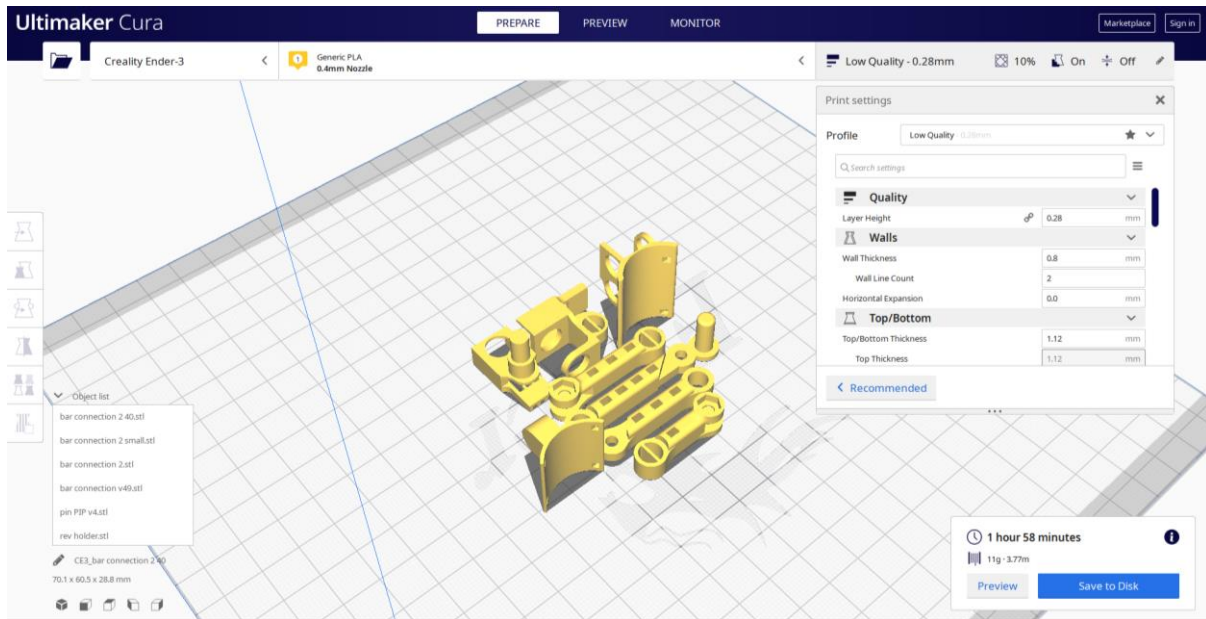


Figure 27: CURA file for MCP_x, MCP_z and PIP parts

2. The CMC_x and CMC_y (Figure 28):

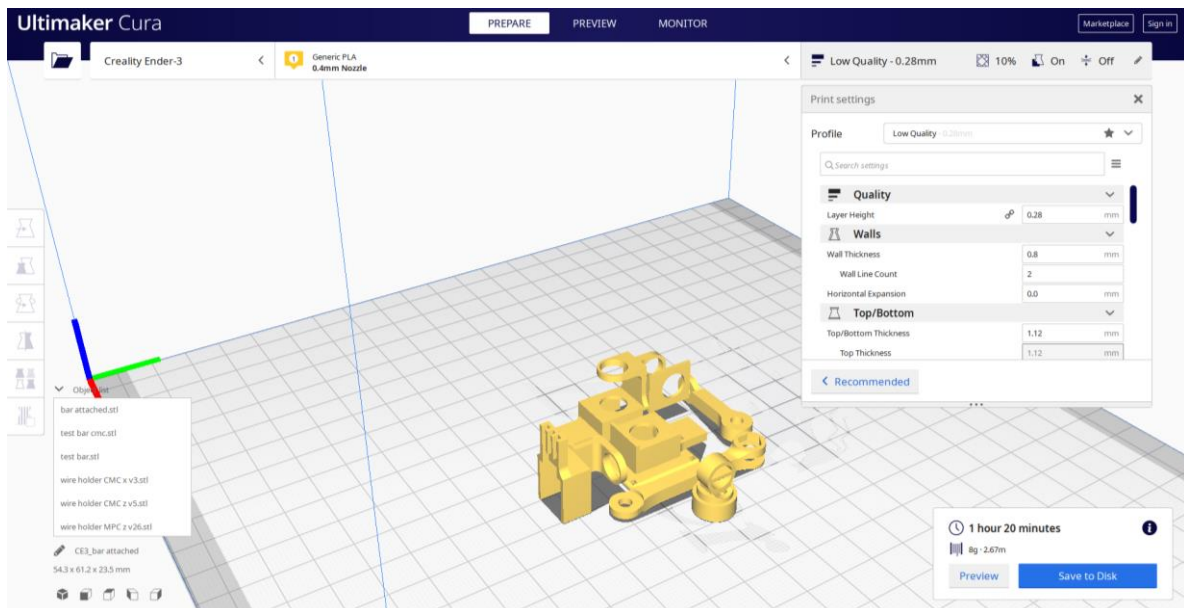


Figure 28: CURA file for CMC_x and CMC_y parts

3. The box for the Arduino and the multiplexer and the links for the wrists (Figure 29):

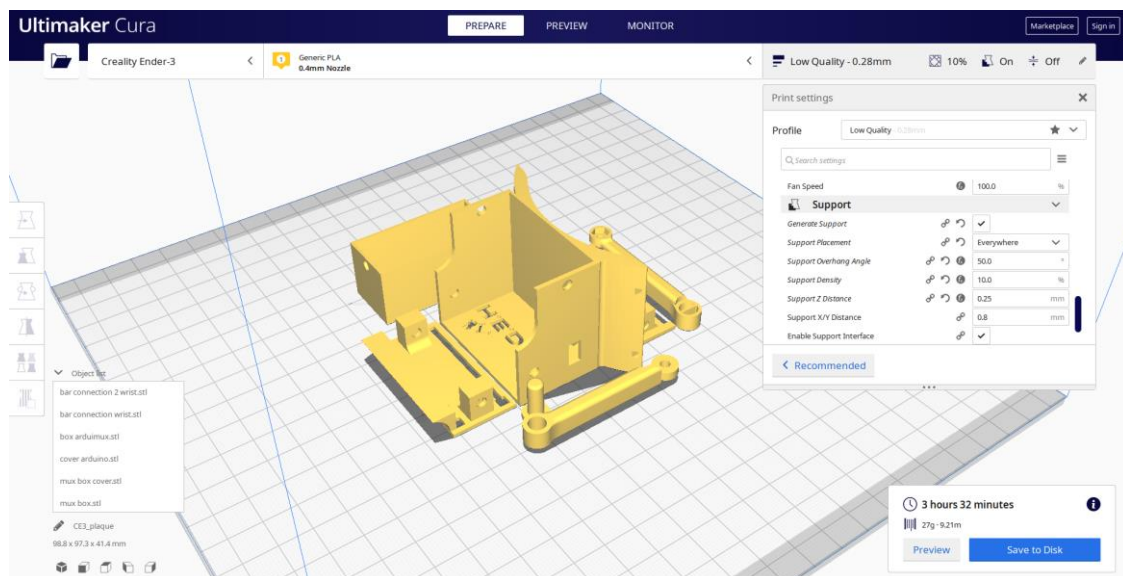


Figure 29: CURA file for the Arduino and the multiplexer boxes

4. The body/hand bracelet:

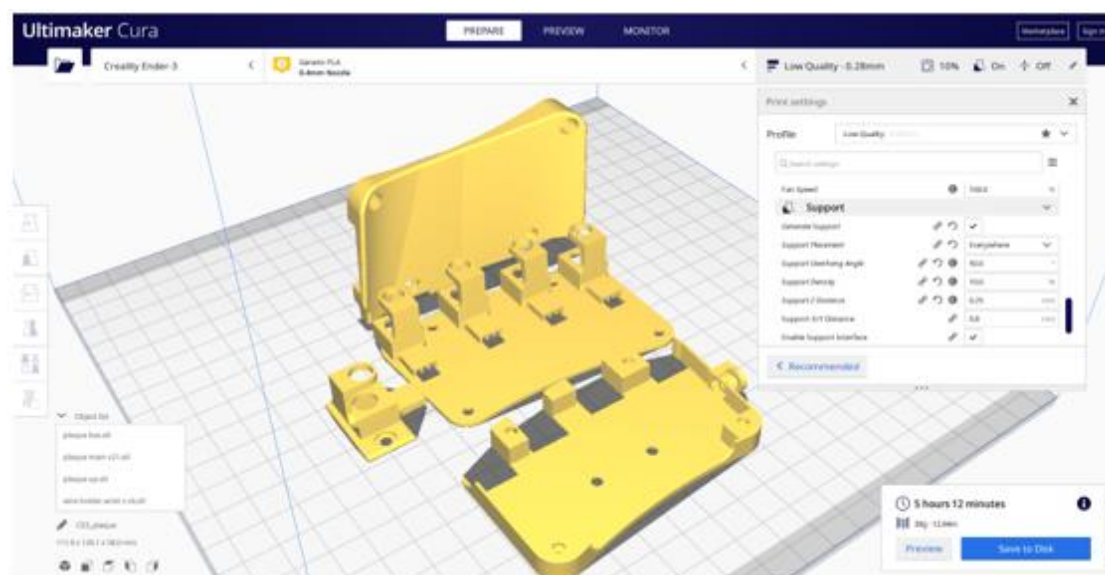


Figure 30: CURA file for the hand bracelet

Then, the parts sliced are simply stored in a SD card and then, the SD card is plugged into the 3D printer. The file containing the **.gcode** of the print can be chosen and the print starts. By summing all the duration of printing, the time for printing the glove can be determined. The time to print all the parts necessary to assemble the glove is 17h and 56 min.

3.6 Cost (bill of materials)

The Table 4 is a bill of material where all the parts required for this artifact, as well as their prices and the links to the shop are included.

Table 4: Bill of material for one glove

Part	Price (€)	Link	Product descriptions
19 rotary potentiometers	3.98	Aliexpress	2x10 rotary potentiometers
Arduino Nano	3.9	Aliexpress	One Arduino nano + 30cm cable
Multiplexer 16 Pins	1.4	Aliexpress	One multiplexer (no pin soldered)
IMU (GY-521)	1.9	Aliexpress	One IMU (no pin soldered)
Jumpers F to F of 20/30 cm	3.87	Aliexpress	40 Female to Female Jumper of 20/30 cm
Wire-nut connector	0.65	Aliexpress	12 pieces of Wire-nut connector
Mini USB cable of 1.5m or 3m	3.90	Aliexpress	Mini USB cable of 1.5m or 3m
Elastics	1.21	Aliexpress	Elastic band: 3mm large and 5m long
Nut + Screw m3x8: 23 pieces	2.26	Aliexpress	3x10 screw M3x8 and 25 M3 nuts

Total	20 €
--------------	-------------

All the component for this glove can be found on the website Aliexpress for a reasonable price of 20€ or £17,20. Concerning the 3D printer, since it is required to own one to build the glove, it will be not counted as a cost. However, the plastic filament is a consumable, so it is considered. First, the amount of filament must be determined:

- MCP_x and PIP parts: 5x11 = 55g
- Arduino and Multiplexer box/bracelet: 24g
- CMC parts: 8g
- Hand bracelet (body): 36g

In total, the filament used is 130g. Depending on the type of filament and the brand, the cost varies. The filament used for the device presented in this artifact costs 20 pounds for 1kg. Therefore, the cost of the filament consumed is $\frac{123}{1000} \times 20 = £2.46$.

The cost of one glove is estimated to about 23,20€ or £19.66. From the market research, people were considering £50-100 as a reasonable price range, therefore, the aim to develop a low-cost sensor is a success.

3.7 Summary

Overall, the design has been chosen to be easy to assemble once the parts are 3D printed. However, the values obtained from some potentiometers needs a calibration and post-processing. Furthermore, the wiring, simple in theory, is much more complicated in reality, and it requires to be vigilant. In return, the glove is very cheap, and it cost less than 20 pounds.

4 Coding

The programming is divided in 3 parts:

- **The Arduino code:** the role of the Arduino program is to send the data through the Serial port.
- **The python code:** Python can read the data sent by Arduino. Python is used to display an interface that the user will use to select the serial port and the calibration process.
- **Unity code (in C#):** the code in Unity is in C sharp, the goal of Unity is to move the hand and finger when it receives data from Python using “socket” to quickly test the device.

In this section, only the Arduino and Python code will be treated because Unity is used to test. Therefore, the following discussion will be about collecting data and processing it.

4.1 Languages and software justification

Using python was necessary to display an interface to facilitate the calibration process with images, explanation, and interactive buttons. Moreover, Python can be used to write and to read files, or to connect a Bluetooth device easily, and overall, Python is compatible with most software and application. Lastly, for further work, Python is the ideal coding language used for machine learning, NLP, and neural network connections. Unfortunately, coding an Arduino with Python is not possible. Instead, the Arduino library Numpy allows a Python code to send command to the Arduino, for instance reading an analogue value or toggling a digital pin. However, this method is not ideal since sending the command and waiting for the values is time consuming. Therefore, instead of sending command with Numpy, the Arduino is sending the value to the computer through the serial port (USB) and received by Python to be processed.

4.2 Arduino code

The Arduino code's role in the software part is the reader and sender of the value to the computer through a Serial communication. The following flowchart in Figure 31 shows the code operation.

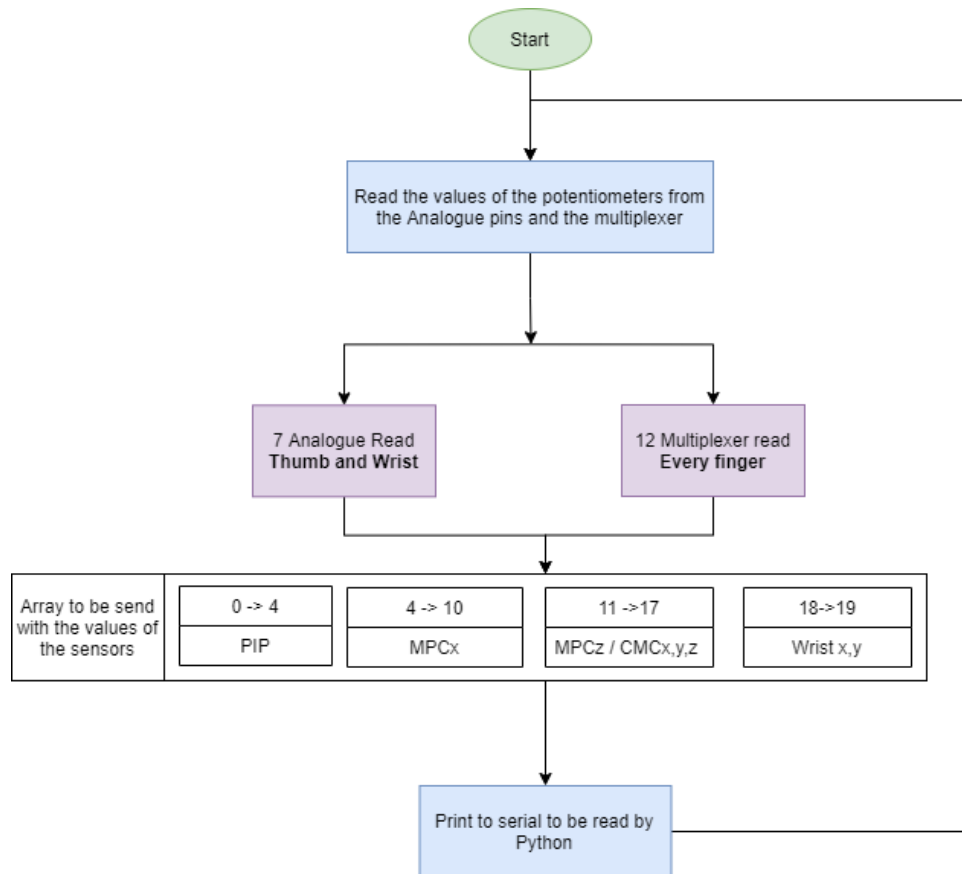


Figure 31: Flowchart of the Arduino code

The data are simply collected one by one in a specific order to simplify the process in Python. Once the data is collected, it organised the data in a specific order and then it is transmitted to Python through the Serial port. To do so, the values are printed in the serial port with ***Serial.print*** and they are separated by a comma. The Arduino code is stored in the Arduino, therefore, as soon as the controller is powered, it starts printing the data. The data is sent but not yet received by Python.

4.3 Python

Once the data is sent by the Arduino, Python must read the Serial Port in order to receive the data. A calibration must be performed before using the glove, but the user also has the choice to use a previous calibration.

4.3.1 Receiving the data while interfacing

It is important to know to which port the Arduino is connected. With the help of the library `serial.tools.list_ports`, every port (USB, Bluetooth...) used by the computer can be listed. The user is then required to select the good port. The port can be determined when compiling the Arduino code, the user have to choose which port to use to compile. Once the serial port is chosen the data be read through the serial port.

Arduino Nano, ATmega328P (Old Bootloader) sur COM3

Figure 32: Arduino software indicating the Serial port

If a device is connected, the message “Choose the part to which the device is connected” followed by the list of port is displayed on the screen as shown in the first picture of Figure 33. Otherwise, this message will show up as shown in the picture bellow.

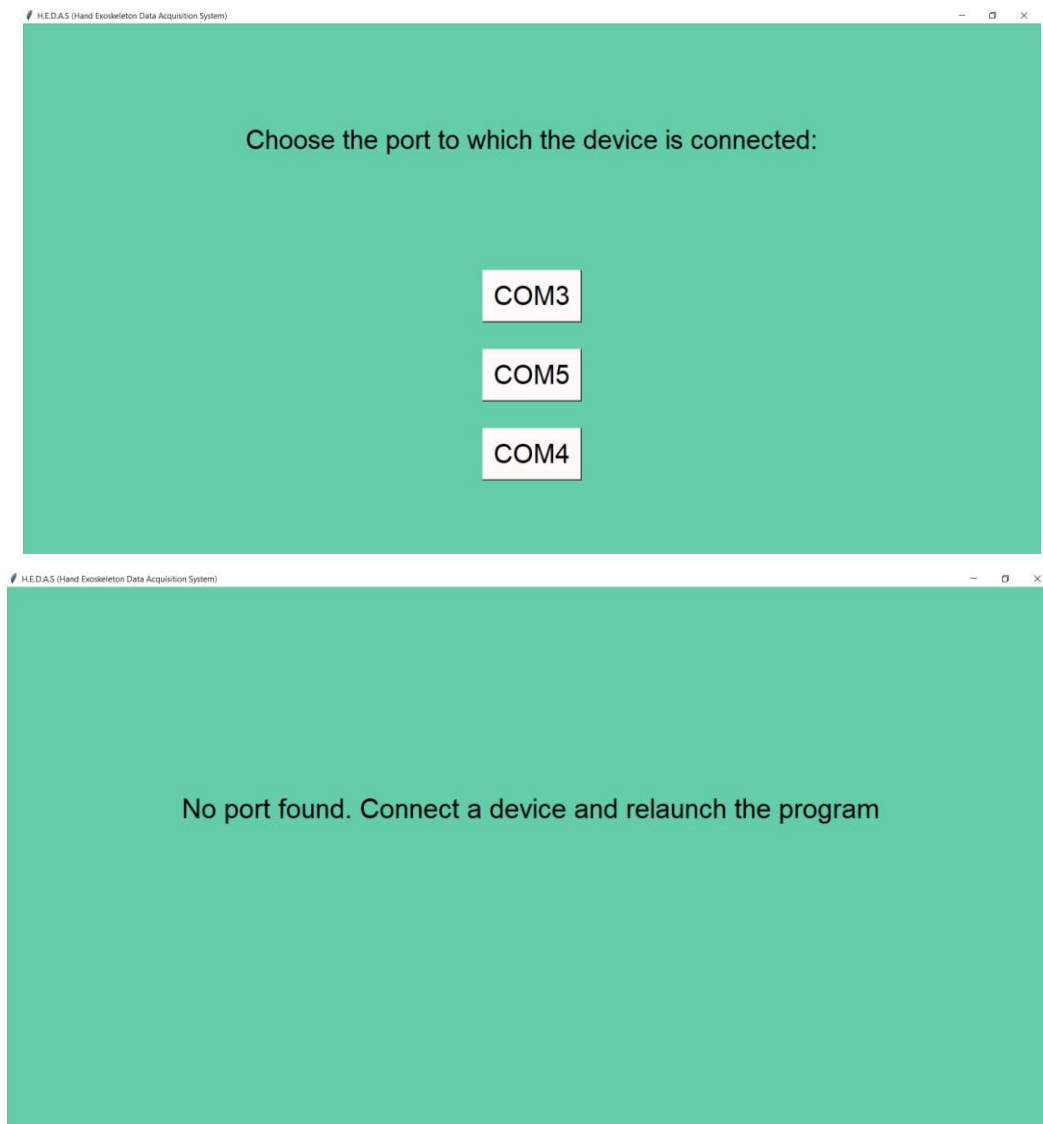


Figure 33: First pages asking for the Serial port

The windows show in Figure 33 are opened when the Python code is launched. They are created with the library “Tkinter”. Tkinter allows to open a window with text, button, entries, and it is compatible with OpenCV. The button is linked to a function; therefore, each function corresponds to a specific action.

Once the Serial port is chosen, the data should be received to calibrate afterwards. However, the interface runs in its specific loop, and the code is paused as long as there is no action.

Therefore, while the interface is waiting, the code is stopped, and the data cannot be received from the Arduino. To rectify this issue while keeping the code in one file, the use of Threads is required. Python threads are used to launch multiple lightweight processes (tasks, function calls) at the same time but that doesn't necessarily mean on multiple processors. Python threads will therefore not speed up a program that already uses 100% of a CPU's time which is different from parallel programming.

In total, three threads will run in the code:

- displaying the interface while waiting for an input (pushing buttons or entering values)
- collecting the data from the Arduino and processing it, and finally, sending it. Of course, the data is not processed nor send until the calibration is done.
- showing the webcam feedback of the computer to help calibrating the glove (see next part “Calibration”)

4.3.2 Calibration

The calibration is an important part of the process. As seen in the part 3.2 Mechanical design (kinematic diagram), since there is a second-degree relation between the angle joint of the finger and the potentiometer, multiple measurement must be done to perform a polynomial regression. Moreover, even if the relation between the fingers' angle and the potentiometers is linear, a measurement at a known angle is taken to obtain the “Zero position”.

In part 3.2, there was a measure every 10 degrees to calibrate the glove, which is not considerable for the user since this is time consuming and tiring for the hand. The goal is to reduce the number of measurement while keeping the same accuracy. By measuring 3 values, evenly distributed, the fitted curves have similar equations. The Figure 34 shows the polynomial regression for with only 3 measurements in comparison to 11 measurements.

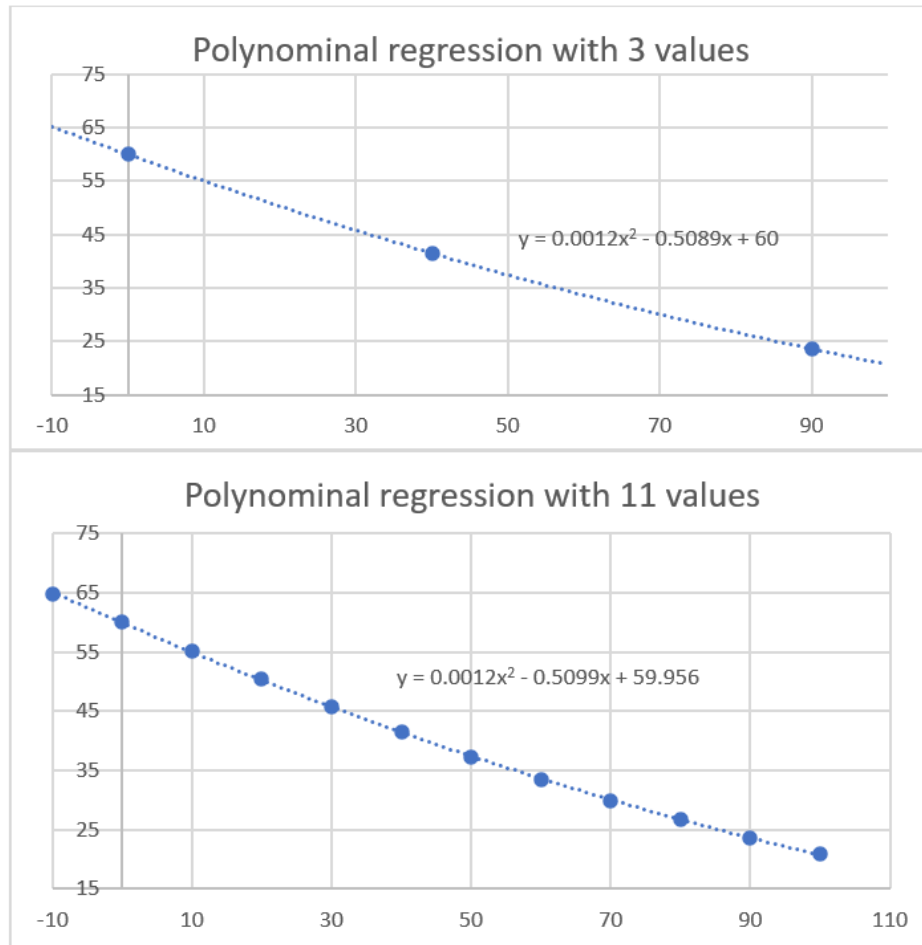


Figure 34: Comparison between polynomial regression with 11 and 3 values

Therefore, to calibrate the MCP_x, PIP and the wrist, only 3 measurements will be taken at the angles:

- 0, 45 and 90 for MCP_x and the PIP joints since they can reach those angles
- -45, 0 and 45 for the wrist joint since it can only reach those angles.

However, the user does not know when it reaches those angles. Therefore, the interface is here to help. In the following Figures, the interface shows how the hand should be positioned to have the correct angles. Furthermore, there is a possibility to open the webcam and to show the video feedback with an overlay using OpenCV. The overlay displays three lines representing different parts: the hand and/or the phalanges or the arm. The user can superpose its hand on the lines to replicate the same joint angles. In Figure 35. On the left, the picture shows the position to replicate. On the right, there is the possibility to activate the webcam that shows the video with the overlay. Once the button pressed, the webcam feedback will show up with the overlay as shown in the following pictures.



Figure 35: First step of calibration with possibility of showing webcam

The following figures (Figure 37, Figure 36, Figure 38, Figure 39, Figure 40) are the step to calibrate, with the webcam feedback and the glove positioned behind the instructions. Once the hand is in correctly positioned, the button “Done!” can be pressed and the values are saved, and the next window will appear with new instructions.

1. Values measured for the calibration: Initial MCP_z , initial $CMC_{x,y}$, 0° MCP_x , 0° PIP, 0° wristx (Figure 36)

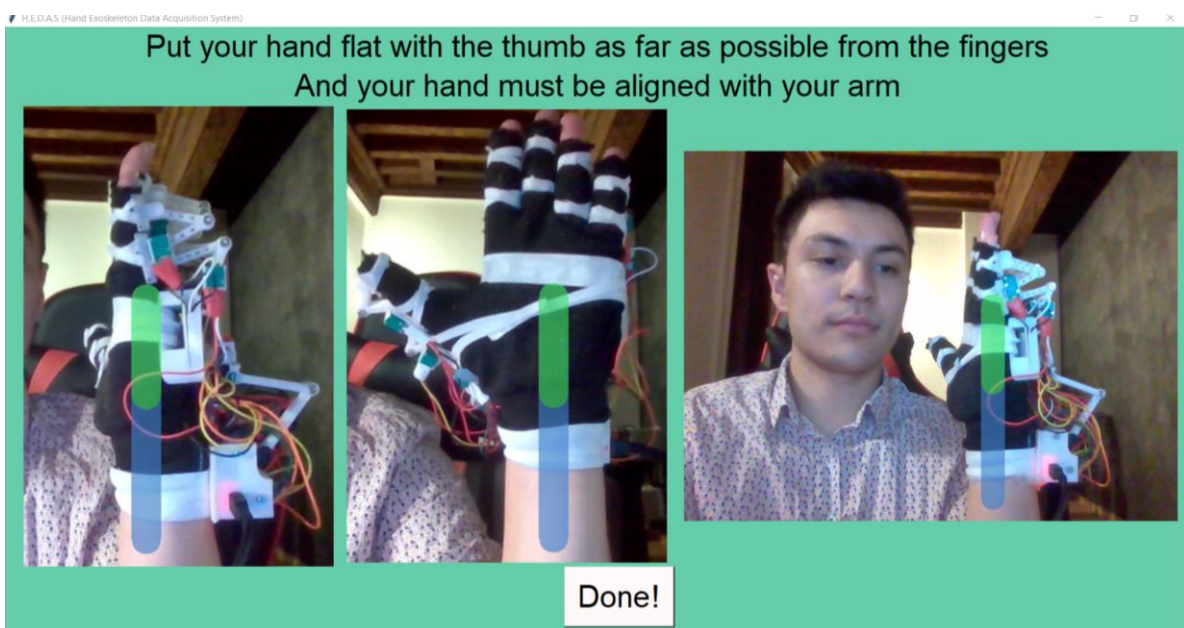


Figure 36: Tkinter window, for Hand flat, finger close and thumb far away.

2. Values measured for the calibration: 45° MCP_x, 45° PIP (Figure 37)

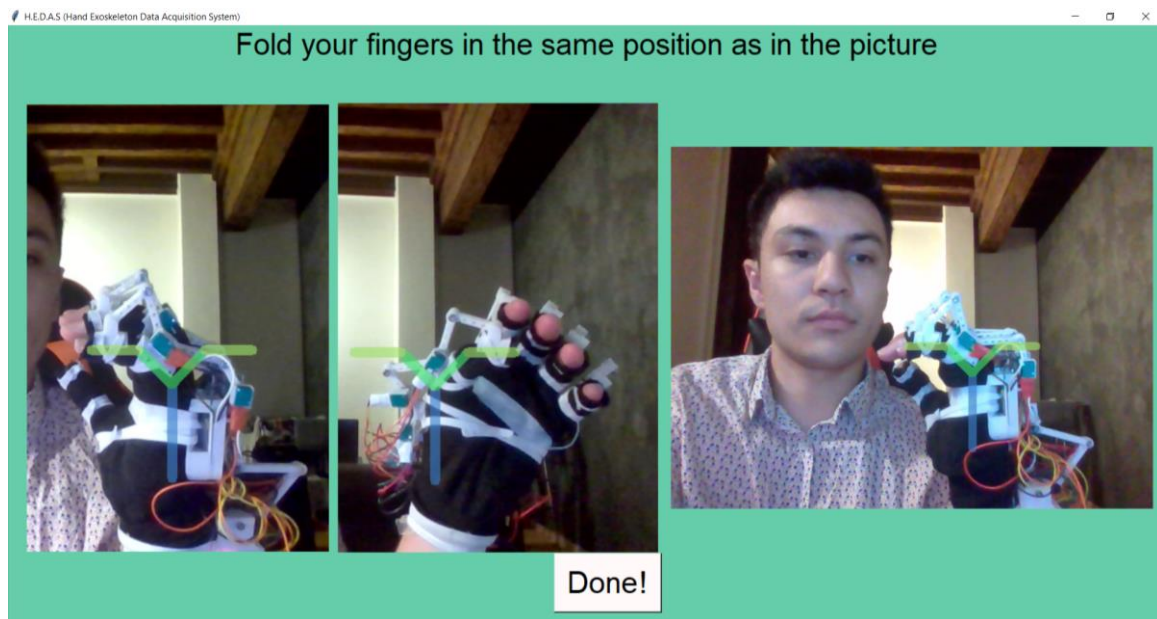


Figure 37: Tkinter window, for Hand halfway closed

3. Values measured for the calibration: 90° MCP_x, 90° PIP (Figure 38)

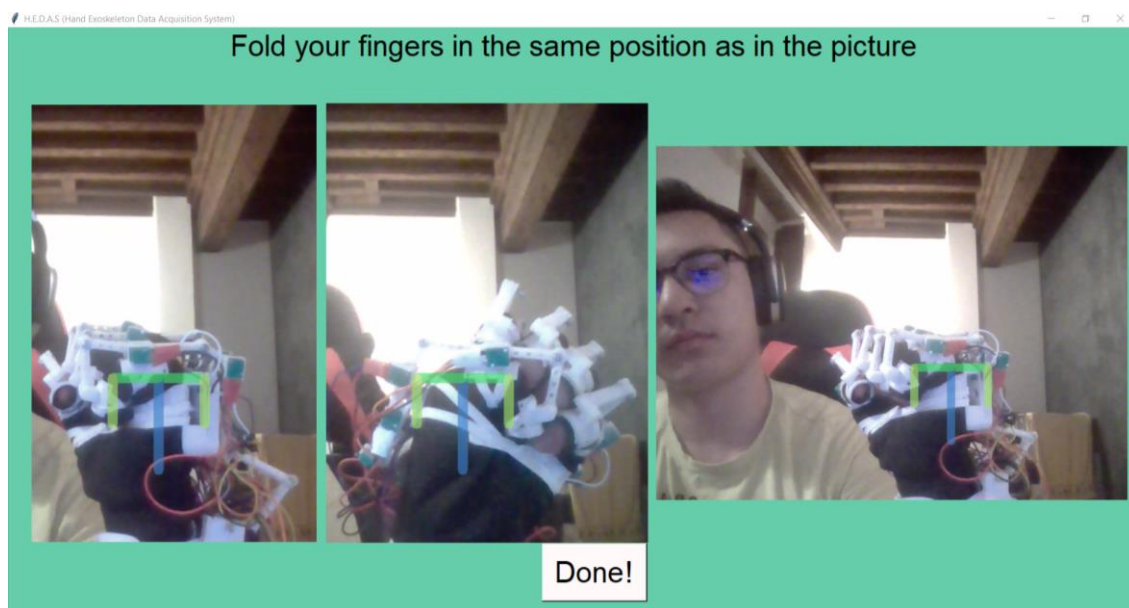


Figure 38: Tkinter window, for Hand closed

4. Value measured for the calibration: -45° wrist_x (Figure 39)

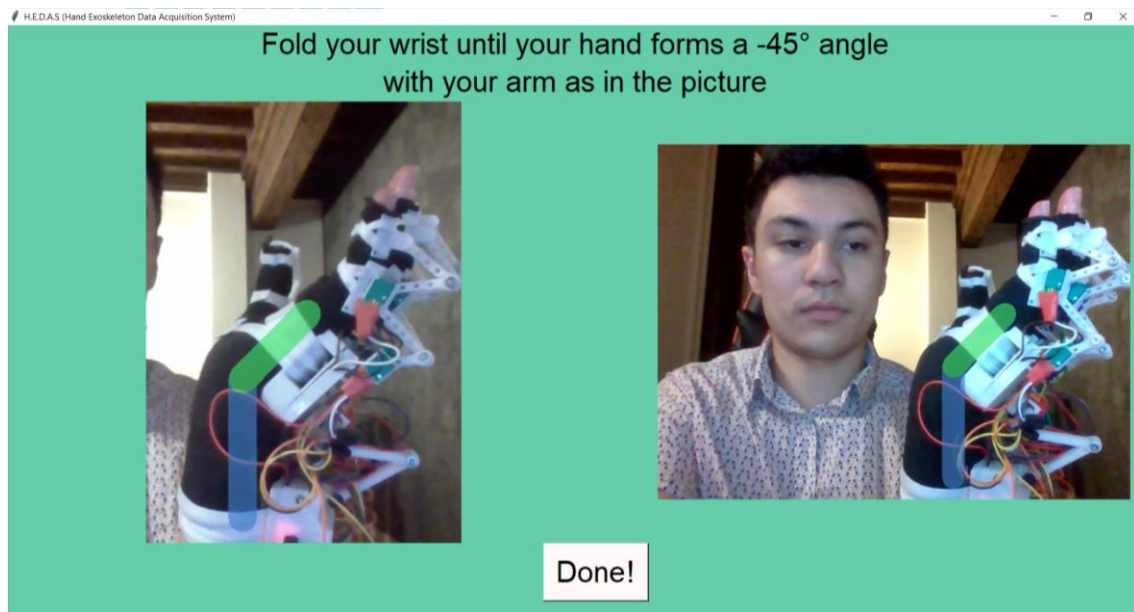


Figure 39: Tkinter window, for Wrist folds at -45°

5. Value measured for the calibration: 45° wrist_x (Figure 40)

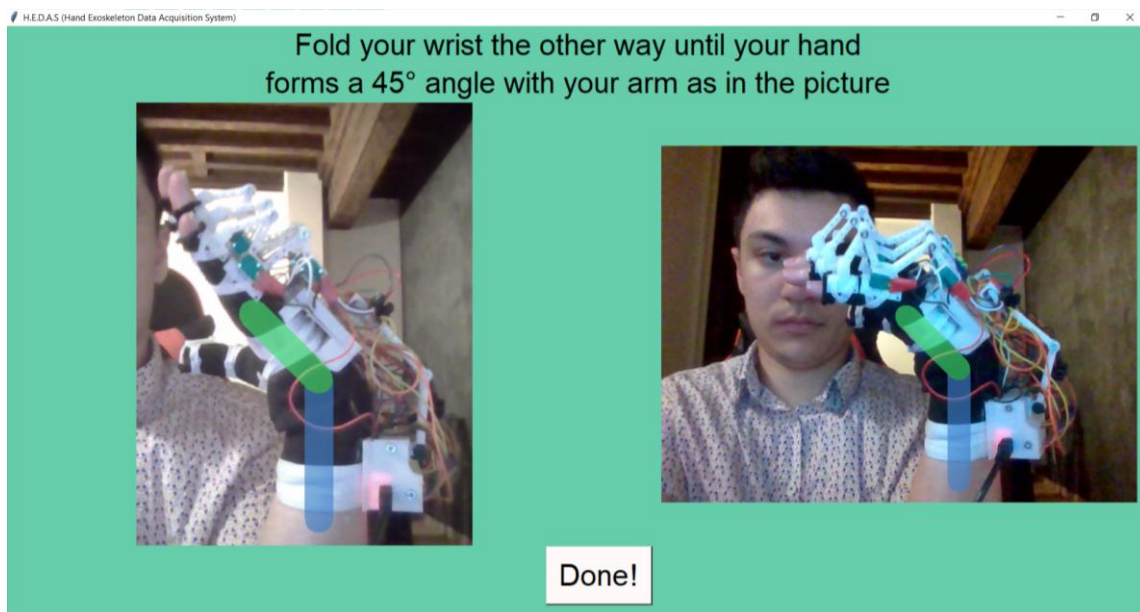


Figure 40: Tkinter window, for Wrist folds at 45°

Once the calibration is finished the values are entered in the polyfit function from the NumPy library. The function will return the three coefficients from the estimated quadratic equation in a form of an array. The following Figure shows the result of a polynomial regression with all the point and with a reduced data set (only 3 values).

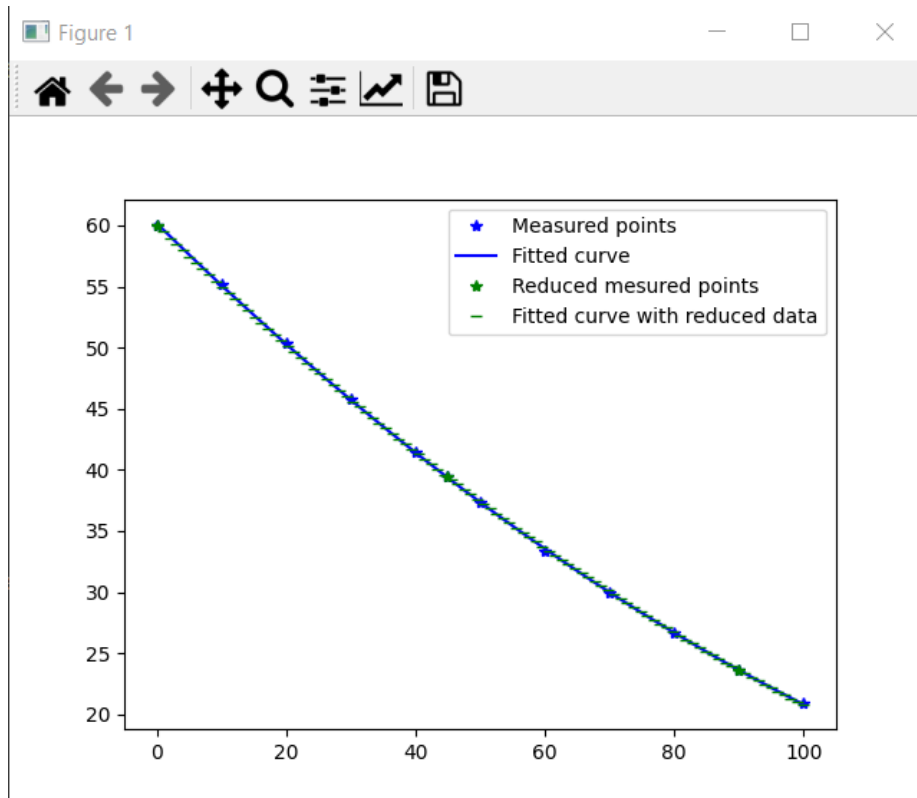


Figure 41: Plot of the regression using the Python polyfit function

The curves have identical shapes because the coefficient obtained are very similar:

$$\text{Coefficients} = [1.25324982e - 03, -5.19165129e - 01, 6.01762753e + 01]$$

Reduced set coefficients

$$= [1.18518519e - 03, -5.11111111e - 01, 6.00000000e + 01]$$

By averaging the difference between the two curves, the error of the approximation can be obtained. The result for this specific example is -0.0016 which is negligible. These result of course are theoretical, in practice, the user could perform an incorrect hand position with a bad angle. It will introduce a large error since the approximation is power 2.

4.3.3 Saving the calibration with JSON

Once the calibration is done, the name of the calibration is asked in Figure 42. The goal was to save the calibration to avoid calibrating each time the user wants to use the device.

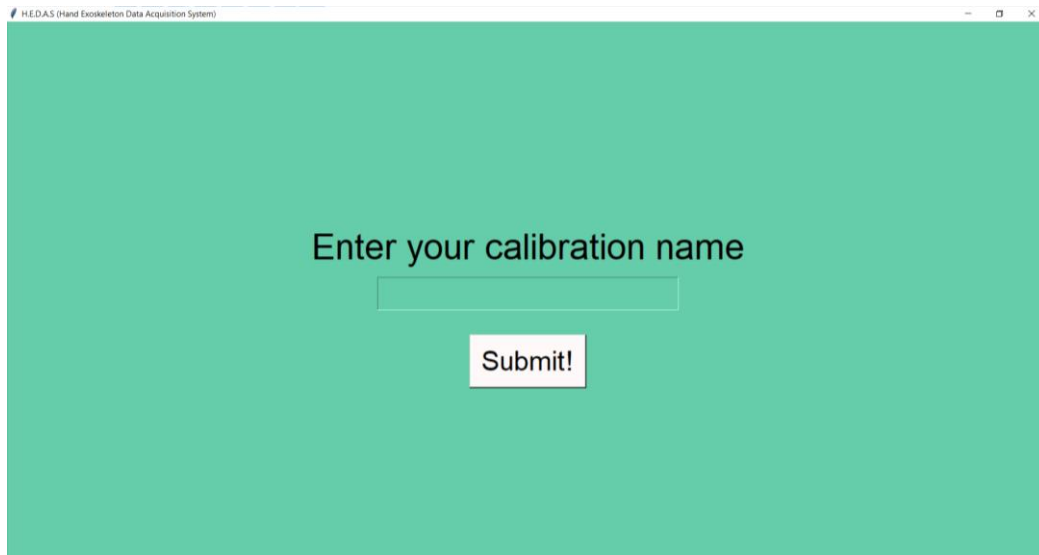


Figure 42: Tkinter window, asking for the name of the calibration

Now, to save the data from the calibration, there is multiple options such as a text file. However, more than storing, the data should be easily accessible with Python. Therefore, a JSON file is perfect for this purpose. A JSON file is a file consisting of a list of variables, each variable is assigned a value. The value can be of any type: string, int, float or even an array (1D or 2D). All the variables are separated by a comma, the name of the variables must be defined using double quotes. It is followed by two points and the values of the variables. The Figure 43 shows the JSON file created or open by the code. “Username” is the array containing the names with the corresponding coefficients.

```
calibration - Bloc-notes
Fichier Edition Format Affichage Aide
{"Usernames":
  [{
    "name": "Thomas",
    "PIP": [[0.0014059957241775649, -2.1293954816682343, 806.0699128881023],
            [0.0012639534378664796, -0.6465838509316767, 82.43543413108634],
            [0.0007719553174098624, -0.30428662246844035, 28.68240850059028],
            [0.001409363691688115, -0.7702495823628691, 105.2141715613684],
            [0.0014135036294585074, -2.373423367581308, 994.7163904436273]],
    "MPCx": [[-0.0001186338130093875, -0.23270022421790099, 362.20605506981394],
            [-0.0005138637733574473, 0.15935352622061755, 90.29546413502052],
            [-0.00016030508588977843, -0.07073672842631004, 125.12959400627733],
            [-0.0019449181912044847, 1.5282131774139032, -219.1569193071003],
            [-0.0037617351072550403, 3.0788949615740213, -544.5010488462708]],
    "MPCz": [542.0, 539.0, 485.0, 477.0, 474.0, 337.0, 910.0],
    "wrist": [-0.0015541964901034063, 2.2859050165381936, -764.8890724313663]
  ]
}
```

Figure 43: JSON file storing all the data of calibration

To store a new calibration, the following Python lines are used. First, the file must be opened, and it is load within a variable:

```
if os.path.isfile(os.path.join(path + os.sep, "calibration.json")):
    with open(os.path.join(path + os.sep, "sample.json")) as json_open:
        data = json.load(json_open)
```

data being the variable where the json is load. Then, the file is modified by adding a “Username” with all the information:

```
data['Usernames'].append({'name': name, 'PIP': coeff_PIP, 'MCPx': coeff_MCP, 'MPCz': z_MCP, 'wrist': coeff_wrist})
```

If there exists no file, after the calibration, a JSON file with all the data must be created:

```
data = {}
data['Usernames'] = []
data['Usernames'].append({'name': name, 'PIP': coeff_PIP, 'MCPx': coeff_MCP, 'MPCz': z_MCP, 'wrist': coeff_wrist})
```

And write in the a new file called “sample.json”:

```
with open(os.path.join(path + os.sep, "sample.json"), "w") as json_create:
    json.dump(data, json_create)
```

data is defined, then it is filled with data and finally, the JSON file is created, and the data is written inside it.

Now, to access the data in a JSON file, it must be open using its path then load the file. Then the data can be accessed using the name of the variables. For instance, to get the PIP coefficient, the line of code would be:

```
PIP_coeffs = data['Usernames'][0]['PIP']
```

with “0” being the first username.

Now, instead of calibrating, the user can use a calibration profile previously calibrated to his hand.

4.3.4 Processing the data

The coefficients of the polynomial regression and the initial angles for the z-axis joints obtained, the data coming from Arduino can be processed. Each type of joint has a different process:

- **PIP:** *poly_reg(coefficients PIP, new value of PIP)*
- **MCPx:** *poly_reg(coefficients MCPx, new value of MCPx)*

- **MCPz**: $\text{potToAngle}(\text{initial value of MCPz}) - \text{potToAngle}(\text{new value of MCPz})$
- **wirstz**: $180 - \text{potToAngle}(\text{new value of wristz})$
- **wirstx**: $\text{poly_reg}(\text{coeffs wristx}, \text{new value of wristz})$
- **DIP**: $\text{PIPangle} * 0.88$

with:

- **poly_reg** being a function that have for inputs the coefficient of the polynomial regression and the raw values coming from the Arduino. It returns the angle of the joints

```
def poly_reg(coeff,val):
    return coeff[0] * pow(val,2) + coeff[1]*val + coeff[2]
```

- **potToAngle** being a function that maps the values of the potentiometer to an actual angle. To get MCPz, CMCx, CMCy, and wristz angles, their initial values subtracted by the data received. The potentiometer can only rotate 330° and it returns a value between 0 and 1023, therefore, the function is the following:

```
def potToAngle(val):
    return val * 330 / (1023 + 15)
```

While being processed, the data is also reorganized in the **angles array** with a specific order:

Table 5: Organisation in the array of angles

PIP	MCPx	MCPz, CMCx, CMCy	wirstz	wirstx	DIP
0 → 4	5 → 9	10 → 16	17	18	19 → 23

4.4 Summary

The code is separated in multiple software and languages as every language has their advantages and disadvantages. The role of Arduino is simple, collecting the data, organising it and sending it. Python then receives the data to process it according to the calibration. The calibration is an objectively short process that requires the full attention of the user where accuracy is key to the good operation of the glove. The user has therefore the choice to save the calibration to avoid repeat the same thing every time. In the next part, the Unity part of the code will be covered since it concerns the testing part.

5 Testing the glove

5.1 Testing with Unity for a fast visual verification

Unity real engine is a free software to develop mostly 3D games. It is widely known and therefore, VR games can be easily created from scratch. Indeed, *Oculus*, a VR headset brand, offers hand models in in their package for free. The model contains two hands, with all the joints on the fingers with the complete degrees of freedom. The different phalanges can be folded by changing locally the values of the joints' angle. By posing the hand in various position, a quick visual verification can be performed to verify the good operation of the glove. The code is vastly explained, the code can be found in the Appendix B. Unity program.

5.1.1 Communicating with Python

Now, to communicate the values of the angles, the package “Socket” will be the bridge from Python to Unity. Unity's language is C#, but socket is one of library available. A socket is an object that allows the user to open a connection with a machine, locally or remotely, and exchange data with it. There is a Client-Server architecture, and there is usually one server and several clients. The server is a machine which processes the client's requests and possibly sends him a response.

There are therefore two types of application installed on the machines:

- The “server”: listens while waiting for connections from clients.
- “client”: connect to the server.

For the client to connect to the server and vice versa, both needs two pieces of information:

- The server's IP address, or its host name, which identifies a machine on the Internet or on a local network.
- Host names are used to represent IP addresses more clearly

Here, the Python code is the server waiting for a connection from the client, Unity. Once the request is sent from the client and the connection established, the two programs can start exchanging data. The data sent to Unity are angles of every joint of the hand.

5.1.2 Moving the virtual hand

Before moving the hand, the hand must be in a known position, a “zero position” that the user can reproduce easily. This position is the first performed during the calibration with the finger near each other, the hand flat and the thumb as far as possible (Figure 36). This particular

position is important since the “zero angles” of the MCPz and CMCy,x are determined at this moment.

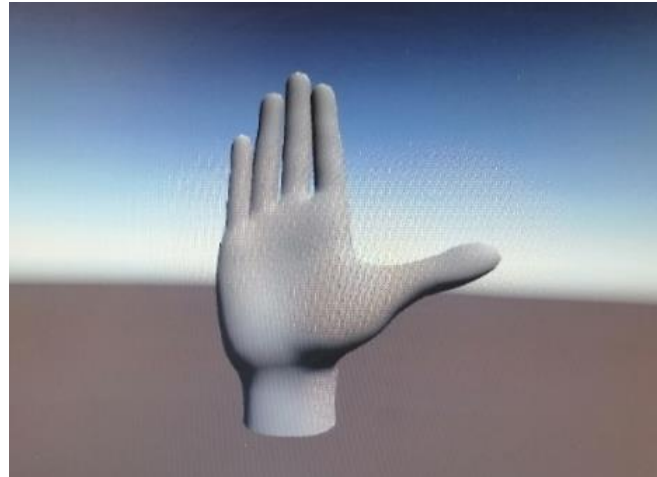
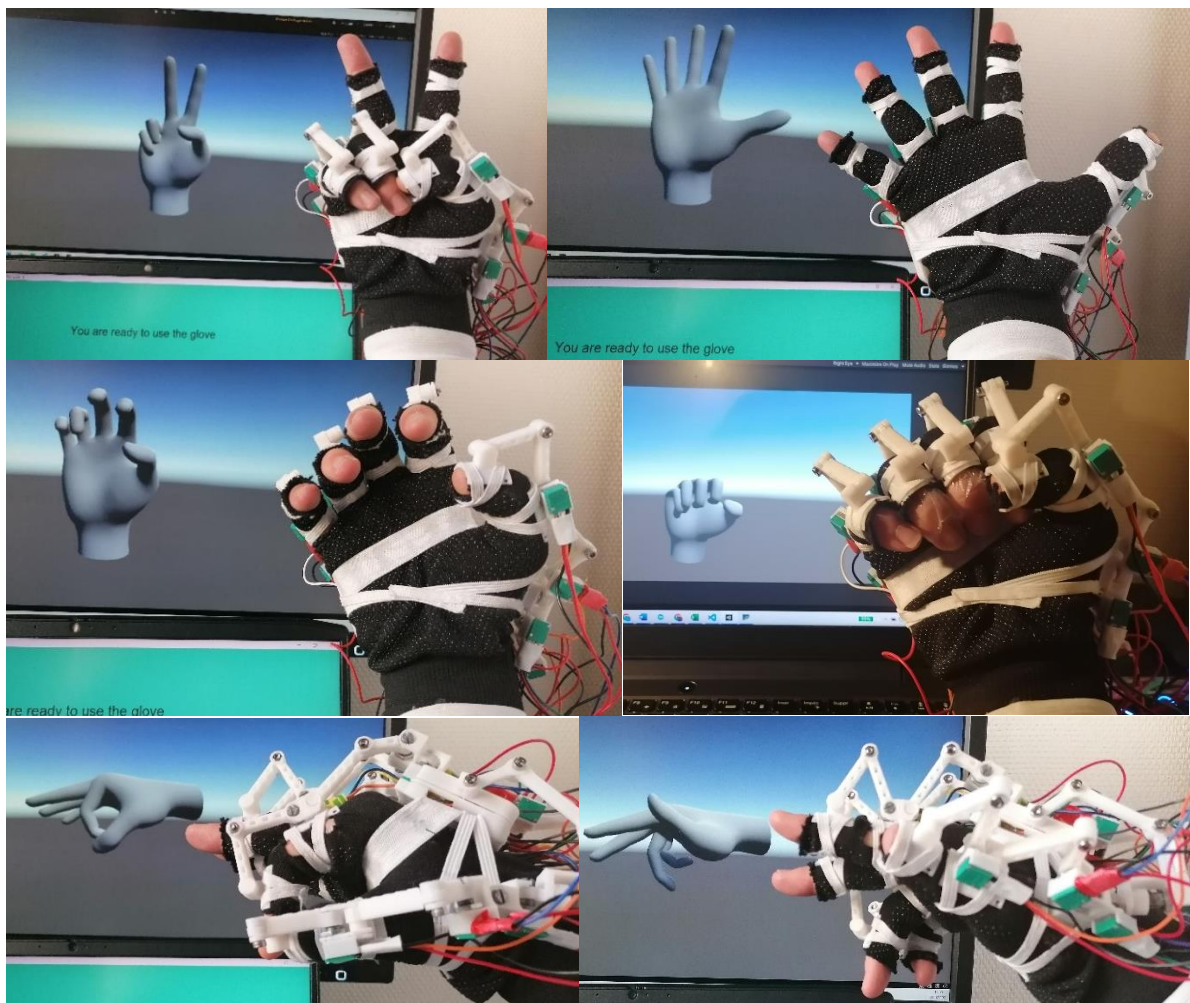


Figure 44: Initial position in simulation

Therefore, once the programmed launched, the angles between the simulation and the glove are going to match. The test consists on moving the hand in various poses to test the glove capability to reproduce the movements, but the also the reactivity of the glove.



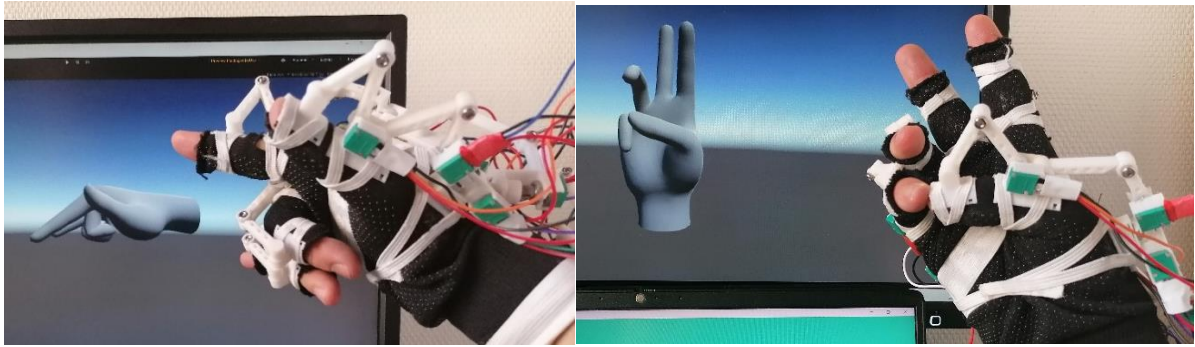


Figure 45: Testing different hand pose in Unity without using the wrist

5.1.3 Visual feedback

Overall, moving the hand while looking at the Unity simulation at the same time, the movements of the hands are synchronised and accurate. Every joint in the simulation moves according to your hand movements. Furthermore, the response time is immediate and there is no noticeable latency.

5.1.4 DIP joint issues

Despite a great approximation of the DIP joint, some inaccuracies can be noticed, especially in the three last pictures. Indeed, the DIP approximation does not work in certain condition, it is possible that the DIP joint cannot move while the PIP joint is moving. This behaviour can observe when one finger is folded without forcing to move the distal phalange (the tip of the finger). This can be also observed with two, three or all the fingers as soon as the distal phalange is not moving willingly. The negligence of the DIP was at first meant to save material, response time and to have a better comfort. Therefore, by observing the movement of the hand, some mechanical behaviours are noticeable. When the finger is forced to close the DIP, the finger next to it move as well, especially the PIP. Therefore, from this, determining if the DIP moves or not depending on the movement of the PIP joints of the finger nearby is possible. This is not an easy task, and the approximation of the DIP angle should be for further work.

Unfortunately, it will not solve the problem when the DIP is fold against its will (pushed against a table for example). Depending on the application, this solution can still be viable, VR games for instance. However, in motion capture, the interaction with object is inevitable, which makes the solution obsolete. The conclusion to this issue could be that the DIP should be also equipped with sensors.

Anyway, concluding on the accuracy of the device with a visual verification is not enough and further tests must be performed.

Figure 46: Test with the overlay at 30° and 60° for both joints

Although the results look very good, like Unity, this test is only a visual test. Indeed, it does not provide a numerical value giving the real accuracy. Therefore, another test must be performed to obtain the accuracy.

5.2.2 Comparison with an IMU

As seen in the literature review, multiple gloves use an Inertial Measurement Unit (IMU) to measure the angle of the joints. Since the IMU was not part of the project anymore, it will be used to measure the joint angles while wearing the glove. However, the I²C pins were taken by the potentiometers which means that another board will be used. The Raspberry Pi was the choice since it can be connected to the internet, Socket can be once more utilize to communicate data. The setup of the test is shown in Figure 47. The IMU is strapped to the phalange and the hand is parallel to the ground. Then, the finger is fold where the IMU is (the MCP_x, and the angle is measured by the accelerometer of the IMU while the Python code running on the computer is sending the data to the Raspberry using Socket with a similar approach as sending data to Unity.



Figure 47: Accuracy test with an IMU connected to a Raspberry Pi

The glove and IMU were running at their own frequency, so to synchronise the data obtained by the sensors, the data was collected when the Raspberry Pi received the glove information. Furthermore, since both devices have a high rate of data collection, to avoid having too much data to treat, only one value every 100 is kept. Once the experiment done, the data is saved in a text file, analysed, and visualised in Excel. The result is shown in Figure 48.

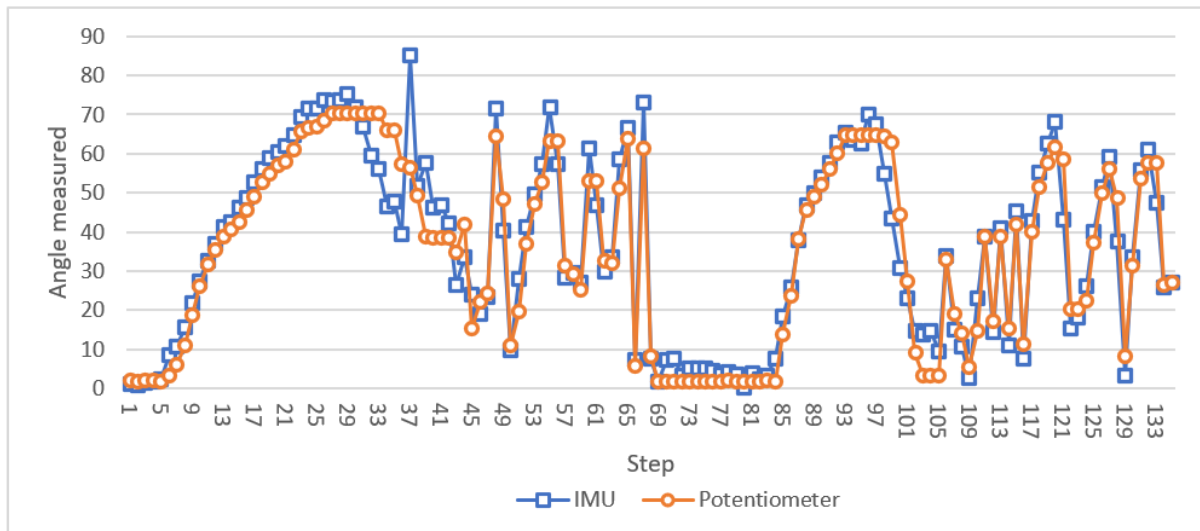


Figure 48: Comparison of measurements of the MPCx from the device and the IMU

The accuracy can be determined by averaging the difference between the values obtained by the IMU and the glove. The accuracy obtained was $\pm 5^\circ$, the difference is huge for this type of device. Of course, this result is obtained without considering the inaccuracy of the IMU. Indeed, the IMU values are very noisy despite the implementation of a median filter. The spikes and errors in the IMU sensor can also be caused by the hand movements during the measure. The hand moving is affecting the results of the IMU since it is sensitive to any movement whereas the potentiometers values are only sensitive to the finger movement. To perfect the experiment, the hand should not be moving, by being strapped to a table similarly to the test performed by Salchow-Hömmen *et al* [1]. However, when the angles are increasing or decreasing slowly, the curves merge and there is almost difference. Only the MCP_x is tested here, but since the solution is the same for the PIP and the wrist, the conclusion is the same.

5.2.3 MCP_z accuracy

To test the MCP_z joints, a different method will be used, similar to the first one using an overlay. Instead of using the overlay, a 3D part shaped with an angle of 20° has been printed to replace it. The part goes between two fingers as shown in Figure 49 on the left picture. The angle detected by the glove are printed as well as the two fingers' MCP_z value.

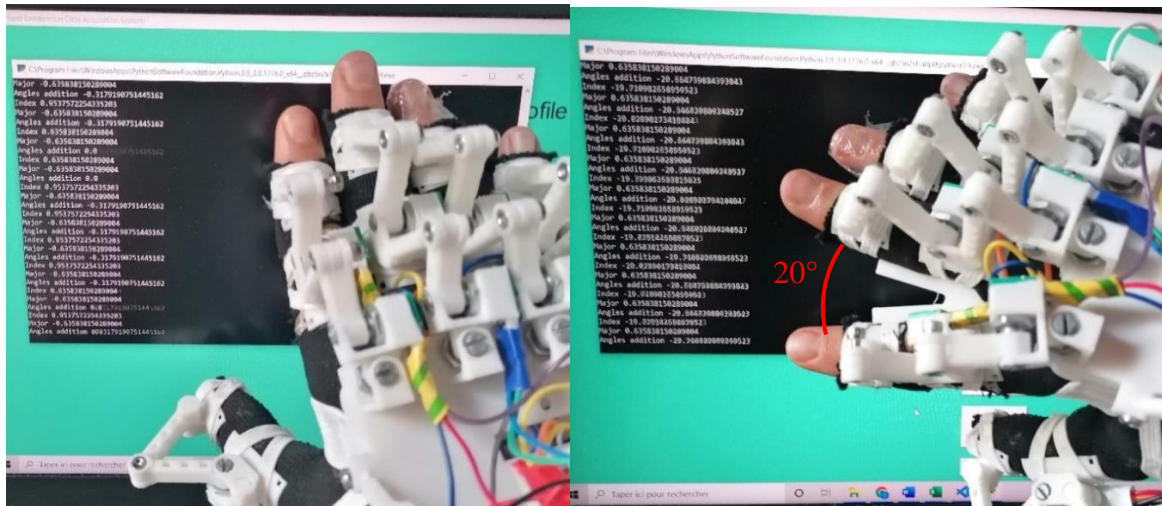


Figure 49: MPCz test with major and Index closed, then separated using a 20° 3D printed spacer

The result in both cases is very good. The repeatability is great as well because the body moving up and down does not affect those sensors.

5.3 Repeatability

To test the precision, a repeatability test should be performed. The repeatability test is testing the reliability of the device to successively obtain the same measure. From the accuracy test, the ability to obtain the same measure should have already proven to be acceptable. However, another test should be done to prove it.

The test consists in closing the hand five times, then to lay the hand flat on the table, and repeat this process multiple times. The goal is to see if the glove is capable to measure the same value in a similar position. The result of the test is in the following Figure 50, the test has been performed on one finger only.

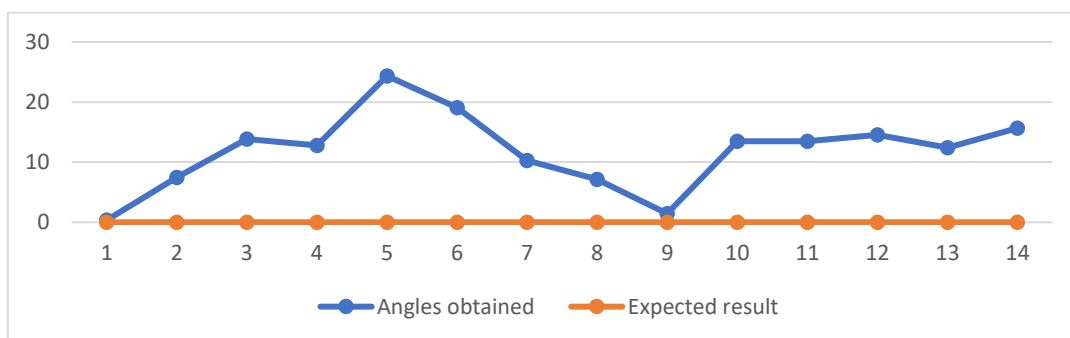


Figure 50: Repeatability test on MCPx

For a successful repeatability test, the curve should be a constant line with the value 0. In Figure 50 however, the curve is far from a straight line, the values are not equal. The repeatability of the MCP_x is very poor, at least, when the hand is flat. The sensor cannot get the same data after

folding the finger multiple times because the 3D parts are not really embedded to the glove. The parts attached to the phalanges with elastic bands are not moving. However, for the main body of the glove, the elastic band cannot hold it close to the hand and the body moves up and down. Therefore, when the hand is closed, the body is close to the hand, but when the hand is open and flat, the body is pushed and far away from the hand. Since the body is moving, the values obtained from the sensors are different after folding the hand. For the repeatability, this is not a good sign because the elasticity adds to the system some randomness, which explains the terrible result obtained here. Acceptable results are obtained in the accuracy test mainly because hand was in constant movement.

The repeatability of the PIP and the MCP_z or wrist_z are much better. The joint measured in the z axis have a linear relation and they are not affected by up and down movement of the body. The PIP, despite being measured in x, has shown a very good repeatability because the two parts fixed to the phalanges are attached tightly on the finger.

5.4 Testing the device on participant's hand

The glove is not meant to be used by only one person, which made the conception quite difficult. The device has been tested on various hand size, slim or large, short, or long fingers. The glove could adapt in every case with adults, but it does not fit a child's hand because the parts are too long for a small phalange.

5.5 Response time

The response time of this device can be measured by getting the sum of the time that one loop takes in Arduino and Python. The Arduino loop takes 11ms to collect all the data, reorganizing it and sending it to the Serial port, while a Python loop takes 0.5ms to receive the data, process it and sending it. In total, the glove takes **11.5ms** to send the data which is equivalent to a frequency of 88.6Hz, which is impossible to notice by the human eye.

5.6 Summary

The result was satisfying with a great experience whilst moving the hand using UNITY. The movement are synchronised and similar. However, from the simulation, the issue on the DIP not moving exactly as expected has been discovered. For VR games, this is already quite good as the user might not notice. Otherwise, the calibration could be extended to find more relations between the DIP and the PIP joint. This solution cannot work when the distal phalange is forced. Therefore, the best solution remains to have a sensor measuring the DIP as well.

Some accuracies and repeatability tests were performed. The result on the accuracy showed some good results, whereas the repeatability was poor. The solution proposed was to attach the 3D printed parts to the hand much tighter to not allow parasite movements.

6 Conclusion

6.1 Comparison to objectives

To conclude, all the objectives were completed. After searching in many articles, the potentiometer was promising because it is accurate by itself, it is unsensitive to noise, it is the most inexpensive sensor, and it is also the most accessible. Then, after designing and testing multiple design by printing different part and different solution, a functional design has been found. In the meantime, the interface to help the user to calibrate was developed. Accuracies tests and a repeatability test were performed and allowed to conclude on the performance of the glove. Finally, the aim of designing an open-source project the most inexpensive possible has been successful as the artifact costed less than £20. The program is available in the GitHub link in the appendices with the images for the calibration as well.

6.2 Critical appraisal

6.2.1 Performance

During the testing phases, the results in Unity seemed very promising. The hand and fingers movements were replicated with high accuracy, at least, visually. The real accuracy has been tested with multiple ways, using a webcam overlay, an IMU or 3D printed parts. The accuracy was satisfying in every test, but the repeatability is very low. Since the 3D parts are not completely fixed to the hand, some inaccuracies can be noticed. The way to attach the part to the body can greatly improve.

6.2.2 Calibration

Although, the calibration with the webcam is a good idea, this is not the most accurate way to have the perfect angles. Another method should be found to calibrate with the angle required by the program, for instance using 3D parts to shape your finger with the good angles (similarly to the test performed in part 5.2.3: *MCPz accuracy*). Most importantly, it would be even easier to get rid of the calibration by designing a device that have a linear relationship between the finger joints movement and the potentiometer movements.

6.2.3 Electrical components

Unfortunately, the wires are exposed, and they can be caught in something while moving. Therefore, there should be a case or a piece of fabric to hide them and to protect them.

The size of the potentiometers could have been smaller, the glove is not heavy, but the design was complicated because of the size of them.

6.3 Future work

The market research helped for this question since the questionnaire included a question about other features that the device could have.

6.3.1 Adding the second hand

The project was focused on designing the glove for only one hand. The next step should be to implement the second hand. A second Arduino with another Serial port could be a solution.

6.3.2 Adding a wireless communication

One of the most important features for this kind of device is the possibility to be wireless. Carrying a wire is not adding a lot of value for the device. Indeed, the Oculus Rift or a webcam with machine learning can measure joints angles with ease. Therefore, the advantage of wearing a glove is the possibility to move around without restriction. However, a wire adds a restriction. The device can still be used freely in space, but it all depends on the length of the wire.

The wireless communication can be performed easily with a Bluetooth module plugged to the Arduino. Moreover, in part 3.3.2, the power consumption has been computed and it has been proven to be very low. An external battery could be used in this case, and it could power the system for a long time depending on the capacity.

6.3.3 Adding a haptic feedback

A haptic feedback is a physical feedback felt by the user when a force is applied against the virtual model. For virtual reality, some gloves have the motors that drives the fingertips if a player touches an object, adding resistance to its movements to simulate the sensation of touch. Some projects use a vibrator on the fingertips to simulate the touch as well, but it is less immersive. Another option is using electromagnetic bars which stops the folding when supplied with electricity.

6.3.4 PCB (Printed circuit board)

A PCB is a support, making it possible to maintain and electrically connect a set of electronic components to one another, with the aim of producing a complex electronic circuit. PCBs are

designed by the customer, and it will be manufactured by a company. Therefore, the advantage of this is to choose the size, the components which can reduce the price and the waste. However, it usually is cheaper if multiple PCB are manufactured in one order.

The circuit for this glove should be composed of a microcontroller, a multiplexer of 16 inputs and a USB adapter for communication and powering.

References

- [1] Park Jaeyoung, Hwang Inchan and Lee Woochan. (2020). *Wearable Robotic Glove Design Using Surface-Mounted Actuators*. Frontiers in Bioengineering and Biotechnology.
- [2] S. Ghate, L. Yu, K. Du, C. T. Lim and J. C. Yeo. (2020). *Sensorized fabric glove as game controller for rehabilitation*. IEEE SENSORS. Rotterdam, Netherlands .
- [3] Y. Mori and M. Toyonaga. (2018). *Data-Glove for Japanese Sign Language Training System with Gyro-Sensor*. Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), Toyama, Japan.
- [4] sparkfun.com. Flex sensor SEN-08606, *Spectra Symbol*. [online] Available at: <https://www.sparkfun.com/products/8606>

- [5] aliexpress.com, eyewink 001. *Flex sensor SEN-08606, Spectra Symbol*. [online] Available at: https://fr.aliexpress.com/item/4001053241357.html?spm=a2g0o.productlist.0.0.5c193458sMzpef&algo_pvid=1f98a97a-4e2d-46db-a937474ab2faa327&algo_expid=1f98a97a-4e2d-46db-a937-474ab2faa3270&btsid=2100bdd716182437770148171ef85b&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_
- [6] aliexpress.com. *Slider potentiometer 10kOhm*. [online] Available at: <https://fr.aliexpress.com/item/32990553553.html?spm=a2g0s.9042311.0.0.41f06c374vhzgE>
- [7] aliexpress.com. *IMU 6-axis*. [online] Available at: https://fr.aliexpress.com/item/33009159455.html?spm=a2g0o.productlist.0.0.386745a23l78xx&algo_pvid=f037dd36-b9f5-40ae-815a1b38723fd24d&algo_expid=f037dd36-b9f5-40ae-815a-1b38723fd24d1&btsid=2100bdec16182427367641348ef252&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_
- [8] aliexpress.com. *Rotary potentiometer 10kOhm*. [online] Available at: <https://fr.aliexpress.com/item/33011428749.html?spm=a2g0s.9042311.0.0.41f06c374vhzgE>
- [9] NotionTheory, Medium.com.(2017) “BUILDING A HAPTIC FEEDBACK GLOVE FOR VIRTUAL REALITY “, [online] Available at: <https://medium.com/@notiontheory/building-a-haptic-feedback-glovefor-virtual-reality-77232999a2d7>.
- [10] M. Ariyanto, R. Ismail, A. Nurmianto, W. Caesarendra, Paryanto and J. Franke. (2016). *Development of a low cost anthropomorphic robotic hand driven by modified glove sensor and integrated with 3D animation*. IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES). Kuala Lumpur, Malaysia.

- [11] Ivan Petrov. (2016). *ARDUINO BLUETOOTH GLOVES PART 2 – PUTTING EVERYTHING TOGETHER*, deviceplus.com, [online] Available at: <<https://www.deviceplus.com/arduino/bluetooth-gloves-part-2/>>.
- [12] 5DT (2004). *Data glove Ultra*. [online] Available at: <<http://www.5dt.com/downloads/dataglove/ultra/5DT%20Data%20Glove%20Ultra%20Manual%20v1.3.pdf>>.
- [13] Y. Park, J. Lee and J. Bae. (2015). *Development of a Wearable Sensing Glove for Measuring the Motion of Fingers Using Linear Potentiometers and Flexible Wires*. vol. 11, no. 1, pp. 198-206, in IEEE Transactions on Industrial Informatics.
- [14] J. Connolly, J. Condell, B. O’Flynn, J. T. Sanchez and P. Gardiner. (2018). *IMU Sensor-Based Electronic Goniometric Glove for Clinical Finger Movement Analysis*. vol. 18, no. 3, pp. 1273-1281. in IEEE Sensors Journal.
- [15] Salchow-Hömmen, Christina; Callies, Leonie; Laidig, Daniel *et al.* (2019). *A Tangible Solution for Hand Motion Tracking in Clinical Applications*. Technische Universität Berlin.
- [16] Hrabia, Christopher-Eyk & Wolf, Katrin & Wilhelm, Mathias. (2013). *Whole Hand Modeling Using 8 Wearable Sensors: Biomechanics for Hand Pose Prediction*. International Conference Proceeding Series.
- [17] Mehlmanmedical.com. (2021). Wrist and hand – Abduction, adduction, opposition – MEHLMANMEDICAL. [online] Available at: <<https://mehlmanmedical.com/wrist-and-hand-abduction-adduction-opposition/>>.
- [18] Ti.com. (2021). [online] Available at: <https://www.ti.com/lit/ds/symlink/cd74hc4067.pdf?ts=1628201499826&ref_url=https%253A%252F%252Fwww.google.com%252F>.

Appendices

A. GitHub link

This link goes to a GitHub project where the Python and Arduino are:

<https://github.com/ThAUJAS/H.E.D.A.S-Exoskeleton-glove-sensor->

B. Unity program

Since the Unity program is not explained in the dissertation, the code is provided.

```
using System.Collections;
using System.Collections.Generic;
using System.Net;
using System.Net.Sockets;
using System.Text;
using UnityEngine;
using System.Threading;

public class CSharpForGIT : MonoBehaviour
{
    Thread mThread;
    public string connectionIP = "127.0.0.1";
    public int connectionPort = 25001;
    IPAddress localAdd;
    TcpListener listener;
    TcpClient client;
    float[] receivedData = new float[23];

    bool running;

    //updates the values of the joints
    private void Update()
    {
        GameObject thumb1 = GameObject.Find("b_r_thumb1");
        GameObject thumb2 = GameObject.Find("b_r_thumb2");
        GameObject thumb3 = GameObject.Find("b_r_thumb3");
        thumb3.transform.localRotation = Quaternion.Euler(new Vector3(0,0,-
receivedData[0]));
        thumb2.transform.localRotation = Quaternion.Euler(new Vector3(0,receiv
edData[10],-receivedData[5]+20));
        thumb1.transform.localRotation = Quaternion.Euler(new Vector3(received
Data[16]-7,receivedData[10]-10,receivedData[15]));
        GameObject index1 = GameObject.Find("b_r_index1");
        GameObject index2 = GameObject.Find("b_r_index2");
        GameObject index3 = GameObject.Find("b_r_index3");
        index3.transform.localRotation = Quaternion.Euler(new Vector3(0,0,-
receivedData[19]));
```



```

        index2.transform.localRotation = Quaternion.Euler(new Vector3(0,0,-
receivedData[1]));
        index1.transform.localRotation = Quaternion.Euler(new Vector3(0,receiv
edData[11]+5,-receivedData[6]));
        GameObject middle1 = GameObject.Find("b_r_middle1");
        GameObject middle2 = GameObject.Find("b_r_middle2");
        GameObject middle3 = GameObject.Find("b_r_middle3");
        middle3.transform.localRotation = Quaternion.Euler(new Vector3(0,0,-
receivedData[20]));
        middle2.transform.localRotation = Quaternion.Euler(new Vector3(0,0,-
receivedData[2]));
        middle1.transform.localRotation = Quaternion.Euler(new Vector3(0,recei
vedData[12],-receivedData[7]));
        GameObject ring1 = GameObject.Find("b_r_ring1");
        GameObject ring2 = GameObject.Find("b_r_ring2");
        GameObject ring3 = GameObject.Find("b_r_ring3");
        ring3.transform.localRotation = Quaternion.Euler(new Vector3(0,0,-
receivedData[21]));
        ring2.transform.localRotation = Quaternion.Euler(new Vector3(0,0,-
receivedData[3]));
        ring1.transform.localRotation = Quaternion.Euler(new Vector3(0,receiv
edData[13]-3,-receivedData[8]));
        GameObject pinky0 = GameObject.Find("b_r_pinky0");
        GameObject pinky1 = GameObject.Find("b_r_pinky1");
        GameObject pinky2 = GameObject.Find("b_r_pinky2");
        GameObject pinky3 = GameObject.Find("b_r_pinky3");
        pinky3.transform.localRotation = Quaternion.Euler(new Vector3(0,0,-
receivedData[22]));
        pinky2.transform.localRotation = Quaternion.Euler(new Vector3(0,0,-
receivedData[4]));
        pinky1.transform.localRotation = Quaternion.Euler(new Vector3(0,receiv
edData[14]-10,-receivedData[9]));
        pinky0.transform.localRotation = Quaternion.Euler(new Vector3(-
5,10,0));
        GameObject wrist = GameObject.Find("b_r_wrist");
        wrist.transform.localRotation = Quaternion.Euler(new Vector3(0,-
receivedData[17],-receivedData[18]));
    }

    private void Start()
    {
        ThreadStart ts = new ThreadStart(GetInfo);
        mThread = new Thread(ts);
        mThread.Start();
    }

    //connects to Python
    void GetInfo()
    {

```

```

        localAdd = IPAddress.Parse(connectionIP);
        listener = new TcpListener(IPAddress.Any, connectionPort);
        listener.Start();

        client = listener.AcceptTcpClient();

        while (true)
        {
            SendAndReceiveData();
        }
        listener.Stop();
    }

    //receiving the data and converting it
    void SendAndReceiveData()
    {
        NetworkStream nwStream = client.GetStream();
        byte[] buffer = new byte[client.ReceiveBufferSize];
        print("hello");

        //---receiving Data from the Host---
        //Getting data in Bytes from Python
        int bytesRead = nwStream.Read(buffer, 0, client.ReceiveBufferSize);
        //Converting byte data to string
        string dataReceived = Encoding.UTF8.GetString(buffer, 0, bytesRead);

        if (dataReceived != null)
        {
            //---Using received data---
            receivedData = StringToArray(dataReceived); //<-
- assigning receivedPos value from Python
        }
    }

    public static float[] StringToArray(string sVector)
    {
        // Remove the parentheses
        if (sVector.StartsWith("[") && sVector.EndsWith("]"))
        {
            sVector = sVector.Substring(1, sVector.Length - 2);
        }

        // split the items
        string[] sArray = sVector.Split(',');

        // store as a float[]
        float[] result = new float[23];

        for(int i=0; i<23; i++){

```

```
        result[i] = float.Parse(sArray[i])/1000;
    }
    return result;
}
}
```