# POLITECNICO DI TORINO

**Master's Degree in Computer Engineering**

**Automation and Intelligent Cyber-Physical Systems**


**Master's Degree Thesis**

# Design, Development, and Validation of a Wearable Haptic Thimble for Remote Palpation.


**Supervisors**

**Prof. Alessandro RIZZO**

**Prof. Domenico PRATTICHIZZO**

**Co-Supervisors**

**Dr. Eng. Nicole D'AURIZIO**

**Dr. Eng. Tommaso LISINI BALDI**

**Candidate**

**Michele POMPILIO**


**April 2022**

*To all my family,*
*my "shoulders of giants".*

# Abstract

During the last years, the need for telemedicine to face the challenge of health-care delivery in critical situations, as for example outbreaks of pandemic diseases, has acquired increasing importance. Despite that, there is no established technology to allow remote palpation in unstructured environments. Only few frameworks are available in literature, and the latters allow to perform skin indentation by sensing the contact force at a single contact point. In this thesis, a novel haptic device has been designed and tested by extending a former framework developed by the Siena Robotics and Systems Laboratory (SirsLab, UniSi). In the just mentioned framework a patient wears on his own finger a sensing device, in order to perform the palpation task himself, while at the same time a doctor wears a feedback device, able to replicate the pressure exerted by the patient on his own skin. The new developed device consists of a silicone haptic thimble holding four small form-factor force sensors, in this way the force exerted by the patient during the palpation task distributes on four different contact points represented by the sensing parts of the sensors. Thanks to this, the doctor is able to better understand how the patient's finger pressure distributes on the touched skin. Since the framework is based on a doctor that guides the patient through vocal instructions, a new video-call communication system has been developed by using multi-thread programming and TCP/IP sockets, able to manage both audio-video information and tactile information related to the telepalpation system.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The thesis aims at extending a previously developed framework for remote palpation in the telemedicine field. Briefly, the framework is based on two actors that interact with each other: a patient which is asked to indent (touch) their own skin with a finger while wearing a haptic thimble embedding one force sensor, and a doctor which guides the patient and wears a haptic device in charge of receiving and replicating on his own finger the force sensed by the patient. However, the haptic thimble used in this framework represents a limit to the ultimate goal, mainly because it has been designed to embed one single force sensor (in order to keep the thimble's overall dimension small). More in detail, a force sensor usually is made-up from a case holding the elettro-mechanic part and a sensing part that come out from the case structure (Fig. 1.1). Since the patient exerts a force on the sensor by using his own finger, a fraction of the force exerted is unloaded on the load cell structure because of the finger softness. So, in order to optimize the force sensing, an external rigid material has been inserted between the patient finger and the haptic thimble.

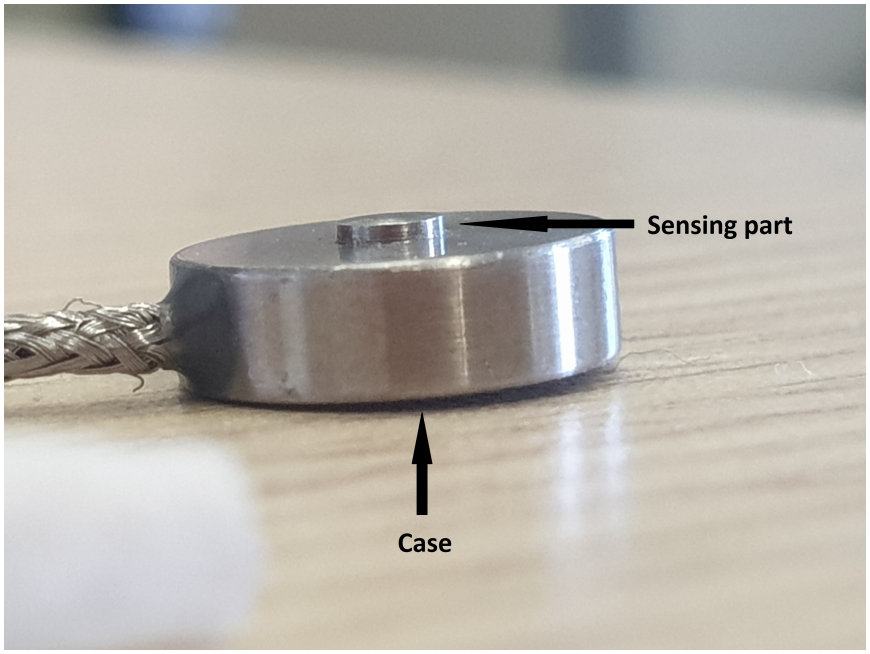



**Figure 1.1:** The old load cell used.

To overcome this limitation, a new approach has been studied: by choosing a new model (Fig. 1.2(a)) of force sensor (of suitable dimension), a new haptic thimble has been designed and realized, this time embedding not one but four force sensors. The four force sensors have been placed by ensuring that their sensing parts are the closest possible to each others, in this way the exerted force is applied only on the sensing parts, avoiding that a fraction of it is unloaded on the sensors' structure. Nevertheless, despite their positioning, a little space was still present between the sensing parts, this because of the sensors shape. Because of that, any soft enough material (like a finger) could have get in touch with the sensors structure, by still causing a force unloading (albeit to a lesser extent with respect to the old thimble). To optimize further the force sensing, also in this case a rigid material has been used (Fig. 1.2(b)), but this time the latter has been included inside the thimble structure and is no more an external body.

**(a)** **(b)**

**Figure 1.2:** (a) Comparison between the old and the new sensor. (b) Configuration of the four sensors to avoid force unloading. By using this configuration a rigid plane placed on the sensing part is perfectly parallel with respect to the sensors structure.

Furthermore, designing a haptic thimble embedding four sensors instead of one only brings additional advantages: since the contact force exerted by the patient's finger is now applied on four contact points, it is possible to understand how the exerted force distributes on the pressed area, allowing the doctor to perceive not only a one-dimensional (1D) force feedback but a more realistic three-dimensional (3D) force feedback (by using a suitable force rendering device). Since the framework was realized for a home-based telemedicine context, it has been considered also the realization of a video-call application to be used by the patient and the doctor. In particular, the realized application allows the two sides to see and speak to each other, like any other video-call application, but also to share the haptic information provided by the palpation framework.

In what follows, it is introduced the thesis structure and the chapters content, more in details:

- State-Of-The-Art (Ch.2): A brief overview of the studies and results achieved in the telemedicine field, with a focus on the current research state about the remote palpation. An overview about the framework developed by the SirsLab (UniSi) is also provided.
- Haptic Thimble (Ch.3): A detailed report about the realization and the achieved performances of the novel haptic thimble is provided, starting from the test of the new sensors up to the final calibration of the realized device.
- Video-call application (Ch.4): An overview about the video-call application developed, by focusing on how the different data are exchanged between the two sides. The performances achieved, in terms of latency introduced by the application during the data transmission, are reported and commented.
- Conclusion (Ch.5): A resume about the achieved results and possible future improvements/extensions.

# Chapter 2

# State-of-the-art

Telehealth is the remote provision of health care by means of a variety of telecommunication tools, including telephones, smartphones, and mobile wireless devices, with or without a video connection [1]. Despite in the last years the need for telemedicine has become of significant importance, especially during the pandemic, currently there are no established technologies and methods able to perform remote medical checks as, for example, the remote palpation. Palpation represents a limit to teleconsultations [2], mainly because of the haptic feedback present in the teleoperation system that introduces an intrinsic trade-off between stability and transparency [3]. Current systems show a remarkable performance gap between direct manipulation and operating the same task by means of a telerobot, where the times needed to the completion of the task are usually two orders of magnitude greater [3]. Furthermore, the hands-on examination has not only a practical value for the assessment of the organs and tissue conditions, but also for its affective properties [4]: in a qualitative investigation on doctors' perception conduct by Cocksedge *et al.* [5], procedural touch, *i.e.* the physical contact that occurs while a palpation task is performed, was valued as appropriate and also therapeutic, as if it was considered to be a reassuring practical process.

During the last years, several frameworks have been developed to allow operators to sense palpation, but mainly in virtual-reality settings for training purpose.

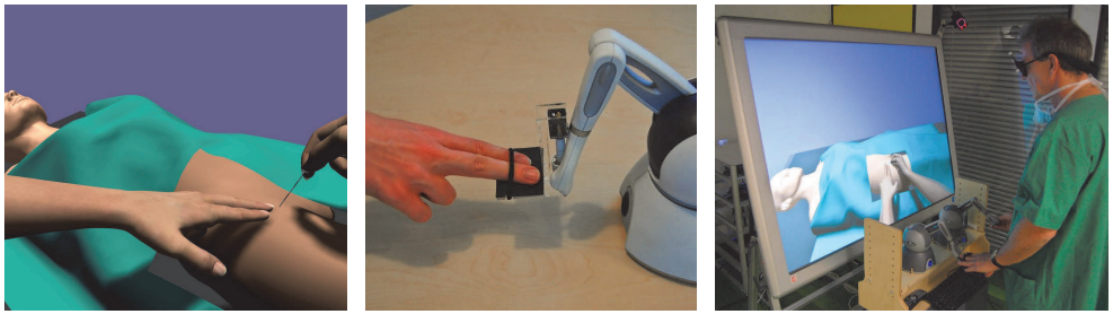
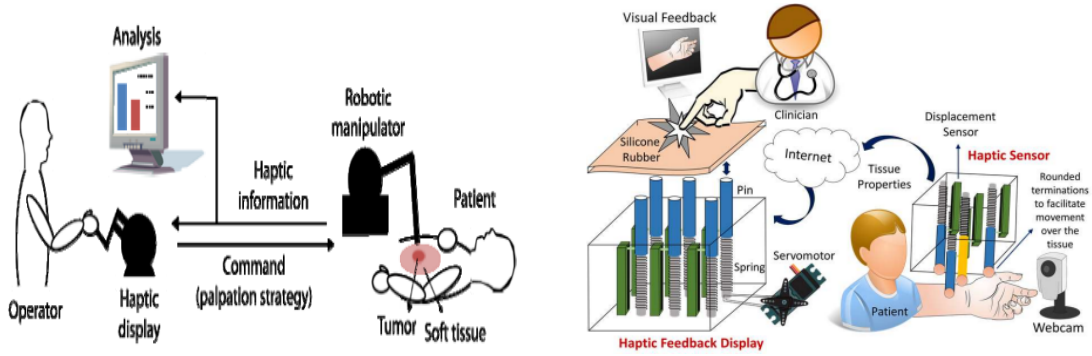


**Figure 2.1:** An example of haptic palpation framework for medical simulation in virtual environments.
Reproduced from [6] Copyright ©2012, IEE.

In Fig. 2.1 we can see a typical framework for haptic palpation in virtual environments: a close-up of palpation interaction on a virtual patient (left), a lightweight palpation pad as a hardware modification for a haptic device (middle), and a medical training simulator prototype used by a medical expert (right). Thanks to this system, it is possible to achieve several benefits among which fully controllable environments, unlimited repetitions, and automated assessments. On the contrary, developing technologies for remote palpation in real settings poses numerous additional challenges concerning the technology on the patient side, such as ensuring real-time interaction, trustworthy measurements, and devices usability. Due to these factors, only a few systems for remote palpation in real settings have been documented in literature. Furthermore, none of these systems are easily applicable to the home environments as they often require robotic manipulators on the teleoperated side (Fig. 2.2(a)). An example of these systems is the one developed by Kim *et al.* [7], where a user with a haptic device guides a robotic manipulator equipped with a force sensor to perform palpation tasks, and can detect the location of inclusions inside a soft medium (silicone-molded tissue phantom).

The measured haptic information is then transferred to the user through a haptic device. However, such a system is studied and developed to work in structured environments like hospitals, so it is not suitable for a home-based telemedicine approach, where patient and doctor are placed in unstructured environments (for example, the patients should install an entire robotic manipulator in their house in order to allow the doctor to perform the remote palpation task).

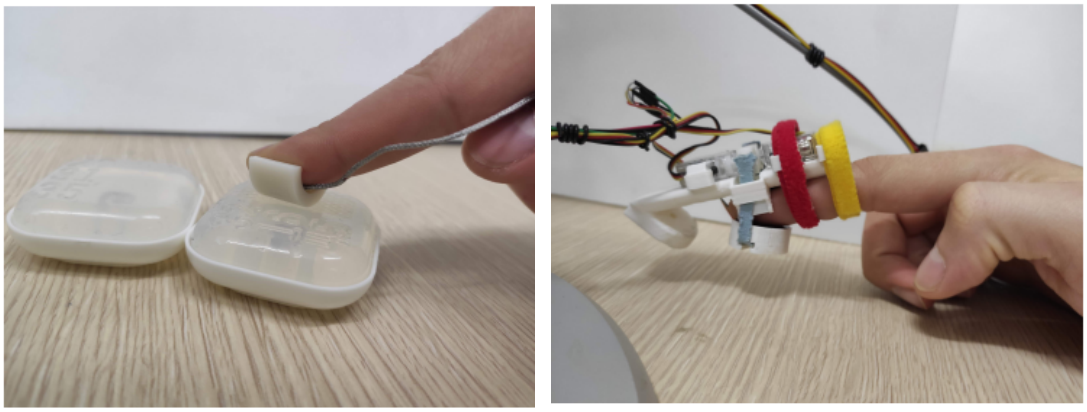



(a) (b)

**Figure 2.2:** (a) Conceptual diagram of haptic palpation system with a robotic manipulator on the teleoperated side. (b) Diagram of a haptic palpation system more in line with a home-based telemedicine approach.
(a) reproduced from [7] Copyright ©2009, IEE.
(b) reproduced from [8].

A system more in line with a home-based telemedicine approach is the device proposed by Hernandez-Ossa *et al.* [8] (Fig. 2.2(b)). The teletaction system presented incorporates a haptic sensor that records soft tissue properties according to mechanical imaging strategies, and then sends this information via internet to a distant central hub health facility, where the data are conveyed to the healthcare professional as tactile and kinesthetic feedback, through a haptic feedback display. However, in their work a system prototype design only has been presented, no physical realization has been provided or developed. Usually a typical framework that allows remote palpation in an unstructured environment exploits an uniaxial

thimble, worn by a patient, and a device able to repeat a one-dimensional (1D) force feedback, worn by the doctor. By exploiting this framework, it is possible to asses whether an objective measure of the indention dynamics is more effective than a subjective perception of it for performing nodules identification and stiffness discrimination in remote home-based palpation scenario.

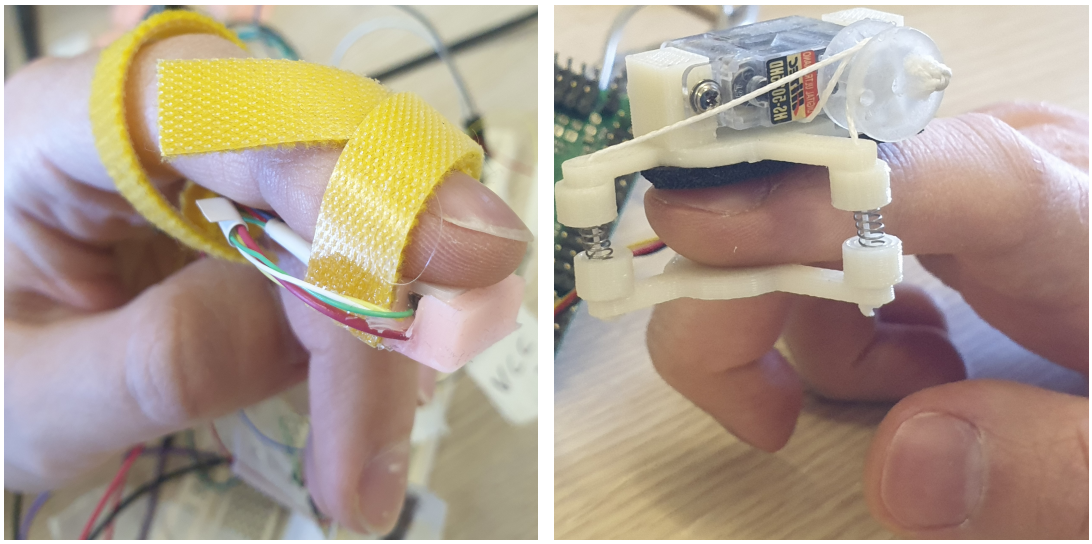


**(a)** **(b)**

**Figure 2.3:** System for remote palpation from [9]

On the base of the just mentioned framework D'Aurizio *et al.* [9] developed one of their systems, where the patient wears a simple cutaneous device housing a load cell (LLB 13 fsh02941, Futek, US) to sense the contact force during the self-examination (Fig. 2.3(a)) while the doctor wears a cutaneous device to perceive the palpation force (Fig. 2.3(b)). In particular, the cutaneous device applies a normal force on the fingertip by means of a wide fabric belt, whose tension is controlled through two servomotors and two pulleys. The consultation is led by the doctor by mean of an audio-video communication between the two sides, thus the doctor can always evaluate how the palpation is performed and the surface deformation under the indentation. On the basis of the latter framework, this thesis aims to design and test a new cutaneous device for the patient side. In particular, the new device holds four load cells of suitable dimension in order to achieve better palpation

performance by sensing the contact force applied on a larger skin contact area (Fig. 2.4(a)). In order to test if the new developed device is able to sense and transmit correctly the contact force, a haptic feedback device developed by L. Baldi *et al.* [10] has been used. The 1-DoF device is composed of two platforms: one placed on the nail side of the finger and the other in contact with the finger pad. Three cables and three springs connect the two parts, while one small servomotor controls the length of the cables. The idea is to move the platform toward or away from the finger pad, to display a force at the user's fingertip (Fig. 2.4(b)).



**(a)** **(b)**

**Figure 2.4:** (a) Thimble developed in this thesis. (b) Force feedback device from [10].

# Chapter 3

# Haptic Thimble

## 3.1 Force sensors used

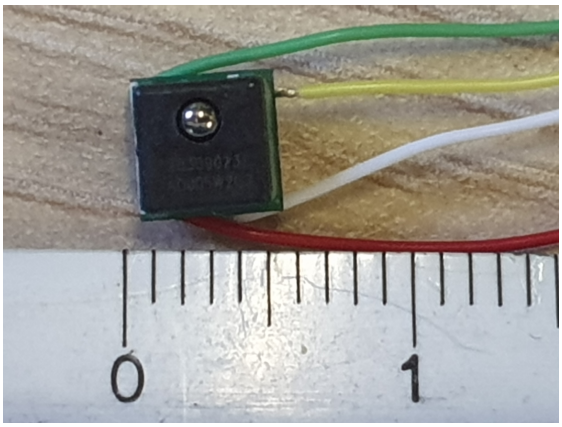



**Figure 3.1:** FMAMSDXX005WC2C3 force sensor.

One of the main goals was to increase the contact surface with the patient skin during the palpation task. In order to do this, it has been thought to increase to four the number of force sensors holded by the haptic thimble. In this way it is possible to identify a contact plane during the palpation task instead of only a contact point. To keep the haptic thimble of a suitable size to be worn on a human

finger the force sensors exploited should be small enough to be embedded in it, and have a good measurement precision in the force range typically used during a medical palpation task (usually the normal forces exploited during a palpation task have a mean magnitude of 3.2 N with a standard deviation of 0.16, Konstantinova *et al.* [11]). By keeping this in mind, the sensors chosen are the MicroForce FMA Series FMAMSDXX005WC2C3 by Honeywell [12], which are piezoresistive based force sensors offering a ratiometric digital output for reading force over the specified full scale force span (in our case 5 N). Thanks to their small form factor of 5 mm x 5 mm, it has been possible to design a haptic thimble embedding four sensors, able to measure the contact force distribution on four different points. The measurement returned by each sensor is a digital 14 bit output, which has to be converted in the corresponding force value by using the transfer function given by the manufacturer (Fig. 3.2)

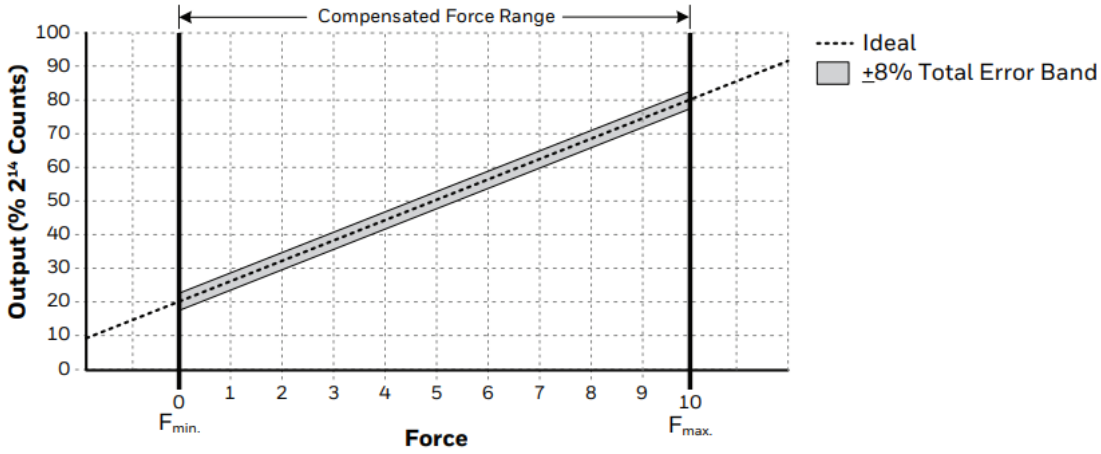



**Figure 3.2:** Force sensors' generic transfer function given by the manufacturer. Reproduced from [12] Copyright ©2020, Honeywell International Inc.

Where the sensor output is given by:

$$Output(\%\text{ of } 2^{14}\text{ counts}) = \frac{60\%}{Force_{range}} \times Force_{applied} + 20\%$$

Therefore, the force measured by the sensors is given by:

$$Force = \frac{Out - Out_{max}}{Out_{max} - Out_{min}} \times F_{rated}$$

where $Out$ is the digital force reading [counts], $Out_{max}$ is the digital output at maximum force (80% of $2^{14}$), $Out_{min}$ is the digital output at minimum force (20% of $2^{14}$) and $F_{rated}$ is the maximum value of the force range [N]. The measurements are transferred by means of the $I^2C$ protocol on an $I^2C$ bus, which provides good support for communication between different ICs across short circuit-board distances. To read the data, a Teensy 3.2 microcontroller (A.3) has been used. Since each sensor connected to the bus is software addressable by the same not modifiable unique address, a multiplexer (A.2) has been used between the microcontroller and the sensors (Fig. 3.3-3.4). The same kind of force sensors were available with a SPI digital output, which could have been more suitable for our project w.r.t. the $I^2C$ version, since it is a faster protocol appropriate for any application where data transfer speed is essential. Anyway, the SPI protocol is a four wires protocol (MOSI, MISO, SCL and SS, while $I^2C$ is a two wires protocol, SCL and SDA, where SCL is used for clock and SDA is used for data), which makes the corresponding sensors too much cumbersome since the cables size is not negligible w.r.t. the overall thimble desired dimension. Furthermore, by considering the max speed rate used for the data transfer in our project (200 Hz), the $I^2C$ protocol is more than suitable for our purpose. The main disadvantage of this configuration is the need for a multiplexer, which increases the overall system dimension. Anyway, our goal was design a haptic thimble of suitable size for a human finger, so we can say that we "moved" away the encumbrance from the finger to another place (for example the multiplexer could be moved on a PCB placed on the user's wrist).

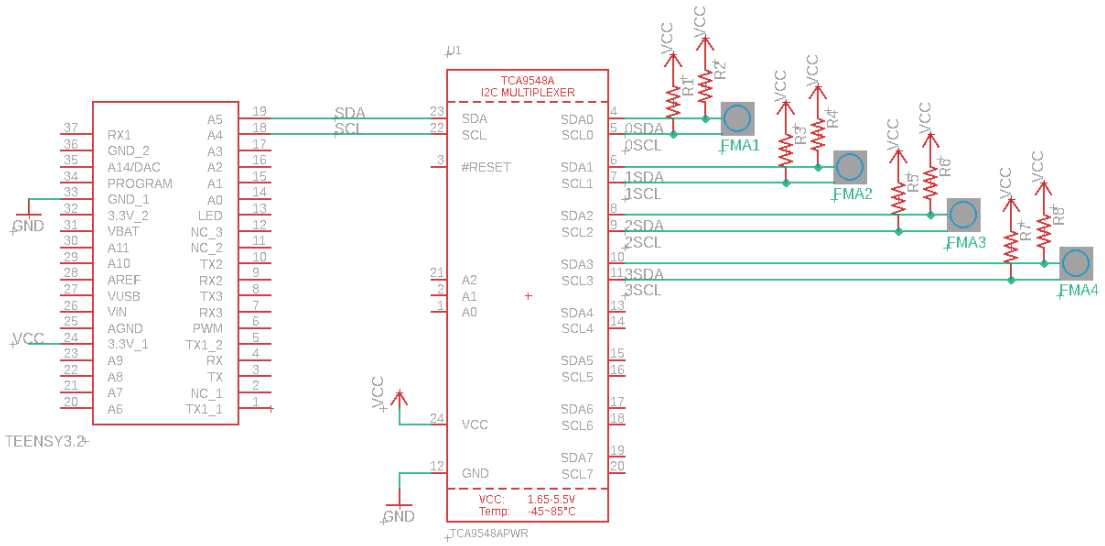



**Figure 3.3:** Thimble's principal components schematics

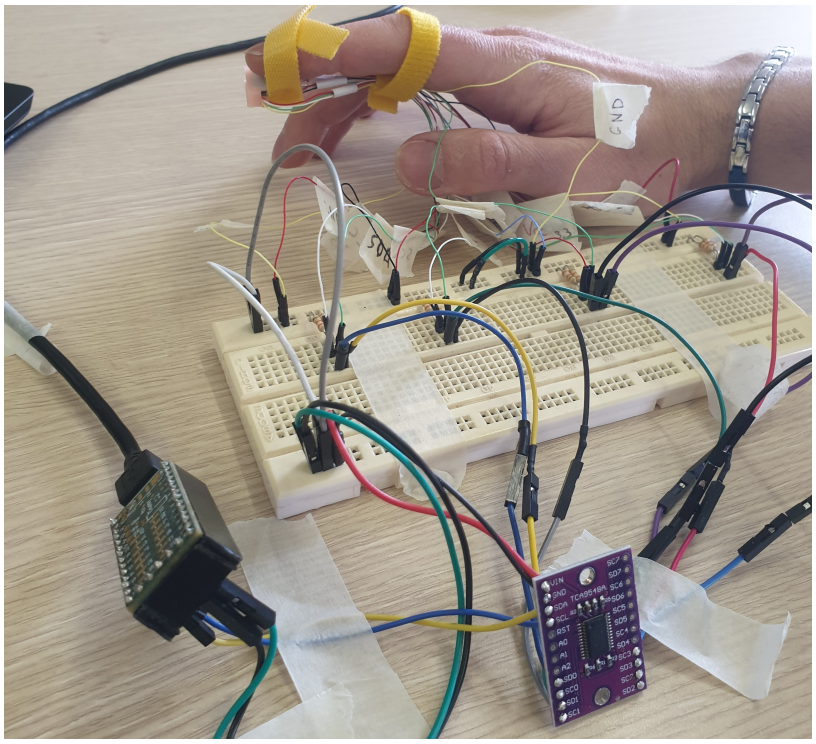


**Figure 3.4:** System's prototype physical realization.

### 3.1.1 Test and validation

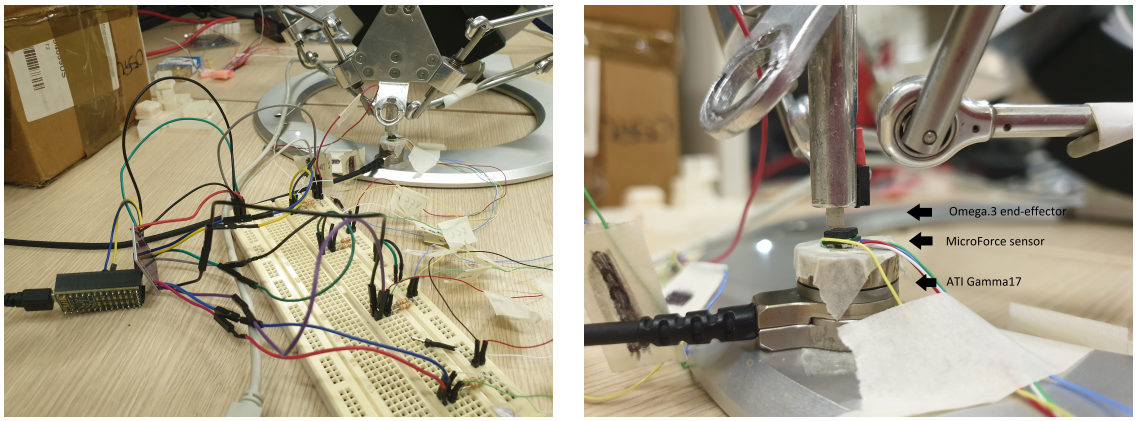



**Figure 3.5:** Test protocol.

As first step towards the development of a reliable haptic thimble for contact force measurement, single sensor performance was evaluated to establish a baseline for the thimble measurements quality. A high precision ATI Nano17 Force/Torque sensor (ATI Industrial Automation, USA) (see A.4) was used as ground truth reference to determine the accuracy of the MicroForce sensor. Thus, the two sensors were placed one on top of the other, with the $z$-axis of the ATI sensor coinciding with the sensing axis of the MicroForce sensor. The testing protocol provided for the exertion of a vertical force on the sensing sphere of the MicroForce sensor by using an Omega.3 haptic interface (Force Dimension, CH, A.1), hence the two sensors were securely hung to the Omega.3 base (Fig. 3.5). Data have been collected through ROS (B.1) and then analyzed by using MATLAB, according to the methods used by Cerveri *et al.* [13] to test the performances of a similar sensor. Different force signals have been exerted on the sensors (by writing an ad hoc code in C programming language by using Visual Studio 2017) to control the Omega.3's end-effector, in particular by moving the latter along the $z$-axis in order to obtain a force signal in the desired range, from 0 up to 5 N. Data have been collected with

a sampling rate of 200 Hz, since for higher frequencies ROS could not handle the rate of the transmitted data and that resulted in latency and errors during the acquisition. Three different force signals have been used:

- Sinusoidal: to simulate a slow progressive force loading/unloading, in order to verify that the sensors behave according to the transfer function given by the manufacturer.
- Stairs-step: to simulate ten progressive press and release maneuvers in order to verify the measurements accuracy given by the sensors.
- Step: to test the sensors' dynamic response.

In what follows the results obtained by the different acquisitions are shown.

### Sinusoidal input

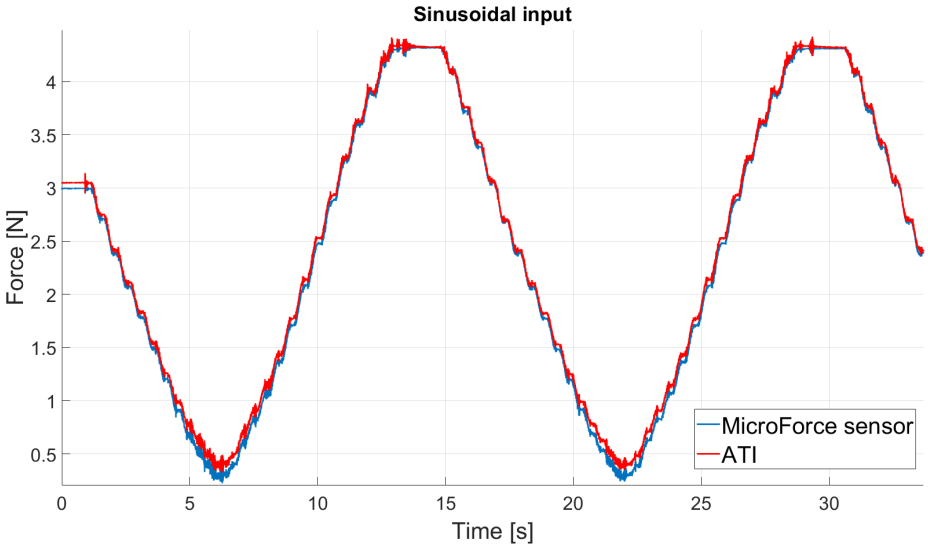



**Figure 3.6:** Sinusoidal input response.

In order to test the calibration quality, a sinusoidal input has been used, described by:

$$u(t) = F_0 + F_0 \sin(\frac{2\pi t}{T})$$

with $F_0$ equal to 2.5 N (to exploit the entire sensor force range) and T equal to 15 s. To obtain such response, the Omega.3's end-effector exerted a force step (starting from $F_0$) each 2 ms, described by

$$f(kt_0) = F_0 \sin(\frac{kx\pi}{180^\circ}) \qquad\qquad k = 0,1,2,...,N$$

where $t_0$ is the time step and $x$ is a parameter to control the force step-size equal to 5. The digital counts obtained by the MicroForce sensor have been compared with the corresponding force values measured by the ATI sensor; the obtained distribution is shown together with the sensor's transfer function in Fig. 3.7.

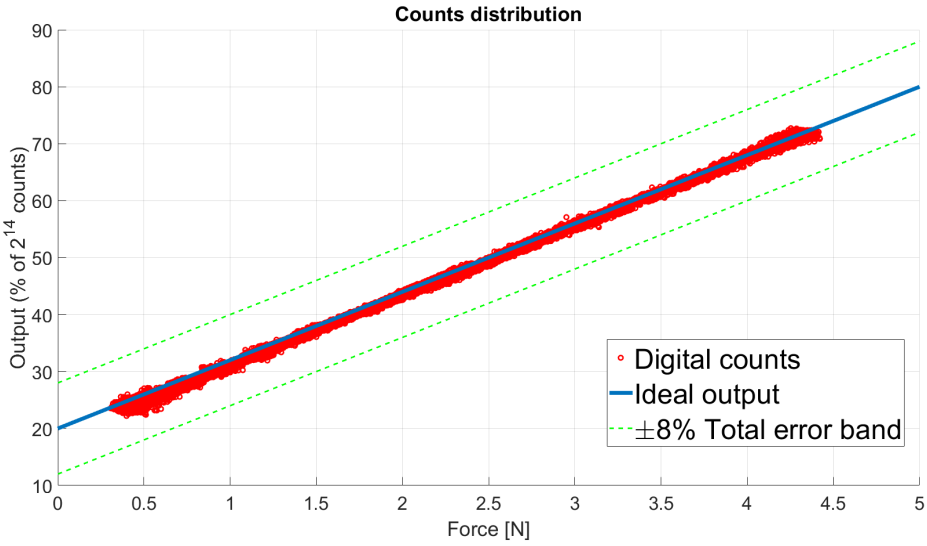



**Figure 3.7:** Scatter plot of the measured counts from the MicroForce sensor in relation with the measured force from ATI sensor. The Microforce sensor behave according to the transfer function shown in Fig. 3.2 (in this plot represented by the blue line).

It can be stated that the MicroForce sensor behaves according to the transfer function given by the manufacturer, so no calibration is needed.

### Stairs-step input

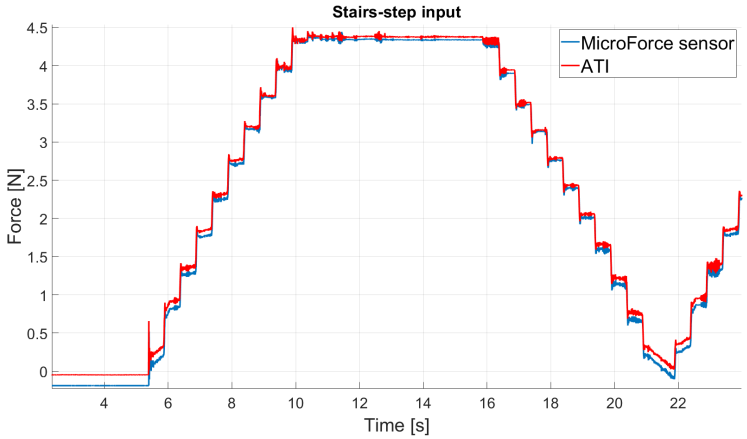



**Figure 3.8:** Stairs-step input response.

A progressive load from 0 to 4.5 N followed by a decrease of the force from 4.5 to 0 N in a time period of about 15 s has been exerted on the sensors, by using force steps of about 0.5 N progressively every half second. Eleven acquisitions were performed. It can be noticed how there is some difference between the output values obtained for the same input in both loading and unloading conditions (Fig. 3.9).

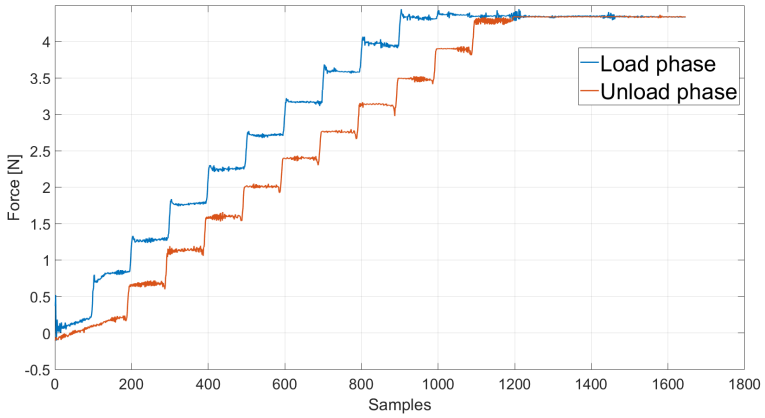



**Figure 3.9:** Difference between the output values obtained for the same input force in both loading and unloading conditions for one acquisition.

This effect is caused by the gravity force acting on the Omega.3 end-effector that should be compensated. It is possible to identify this error as hysteresis error, defined as the ratio between the average of the differences between the output values obtained for the same input in both loading and unloading condition and the full scale output (FSO) of the sensor ($E_H = \sum \frac{\Delta H}{FSO} \cdot 100$). For eleven acquisition the mean hysteresis error was about 2.5%, anyway this error is due only to the set-up of the test protocol, so it is irrelevant in order to analyze the sensors performances. The MicroForce sensors performed extremely well with mean difference between the two measurements (MAE) of 0.046 N and a difference standard deviation of 0.060 N and a root mean squared error (RMSE) of 0.073 N. The relation between the two sets was pretty linear with a Pearson correlation coefficient of about 0.99 (Fig. 3.10). The reproducibility coefficient (RPC) was 0.12 N and the coefficient of variation (CV) was 2.1%. The two curves provided no statistical difference with a $p - value$ near 0. Furthermore, the two distributions were statistically equivalent (two one-sided t-test) with 95 percent CI (Confidence Interval) within the equivalence interval $-0.01$:0.01 N.

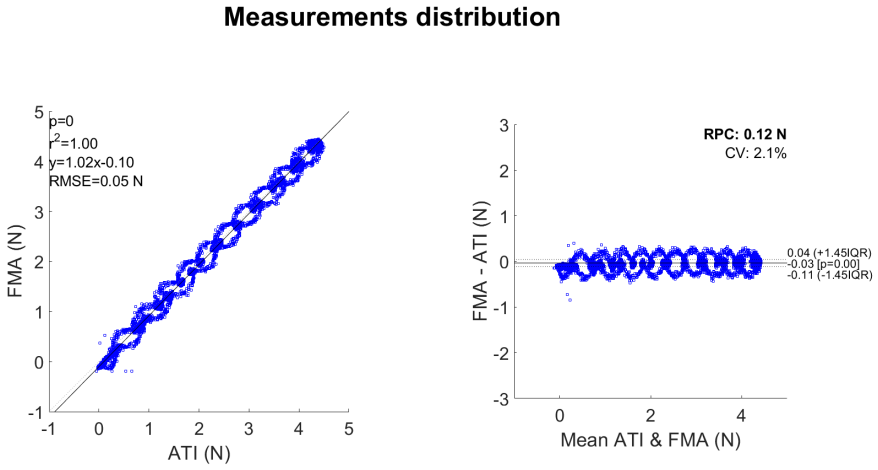



**Figure 3.10:** Distributions of the measured force for MicroForce (FMA) and ATI sensors (left panel). Bland-Altman plot (right panel).

### Step response

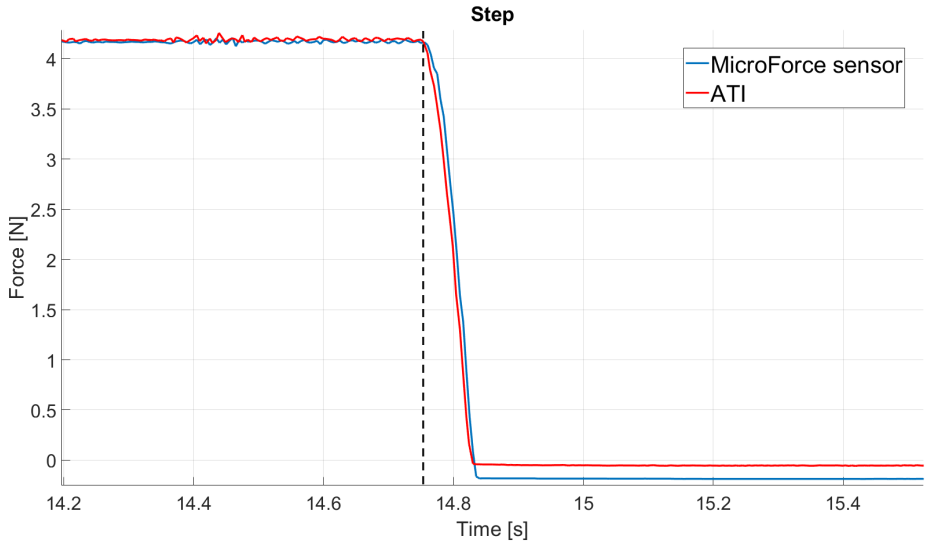
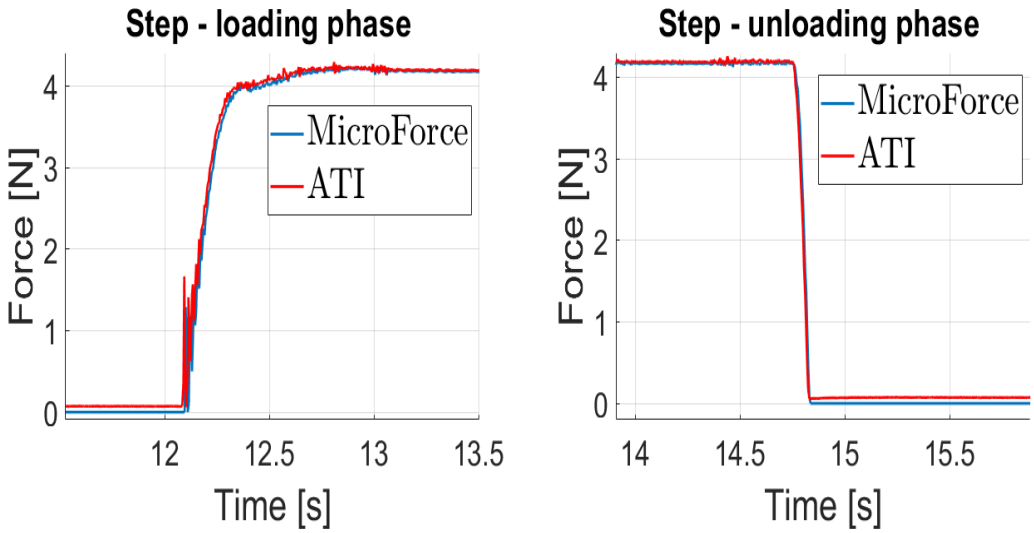



**Figure 3.11:** Step response obtained by using a preload of about 4 N. Vertical dashed lines indicate the moment when the sensors were unloaded.

In order to test the dynamic response of the sensors, a load-step response test has been designed, in which each sensor was preloaded and then suddenly unloaded. In this way, it has been possible to obtain a step signal with a closer behaviour to the one of an ideal step wave: during the loading phase, the step rise time actually depends by the time it took to the Omega.3 to reach the desired force, so the the step signal is not an ideal step wave. During the unloading phase, it is possible instead to raise up as fast as possible the end-effector from the sensors, thus obtaining a step signal with a behaviour closer to the ideal one (Fig. 3.12).




(a) (b)

**Figure 3.12:** (a) Step obtained during the loading phase. (b) Step obtained during the unloading phase.

To obtain such response, the Omega.3's end-effector moved in each 3 s time step to reach the force value described by

$$\begin{cases} F(0) = F_0 + \delta_f \\ F(kt_0) = (-1)^k \delta_f + F((k-1)t_0) \quad k = 1,2,3,...,N \end{cases}$$

where $F_0$ is the initial force exerted by the end-effector on the sensors (in our case equal to 0 N), $\delta_f$ is the force step equal to 4 N and $t_0$ is the duration of the load-unload condition equal to 3 s. To have a step response as accurate as possible during the transient state of the step, the end-effector moved with a velocity of 5 m/s during the unload phase. Sixteen repetitions of data acquisition were performed starting from a preload of about 4 N, results are shown in Tab. 3.1. Experiments showed that, on the average, the measured delay (fall time) for the MicroForce sensor was of about 106 ms and only 9 ms slower w.r.t. the ATI. Since the step's trailing edge represents the fastest dynamics for our application, it can

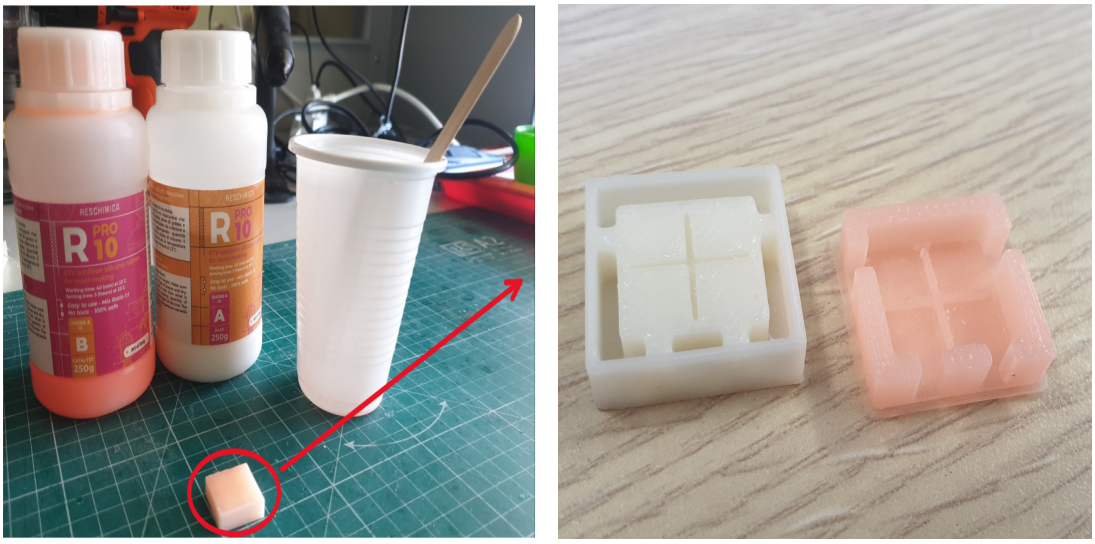
be stated that a sampling rate of 200 Hz (sampling time 5 ms) is suitable for our purpose.

| **Results** | | | | |
|---|---|---|---|---|
| No. | Start time $[s]$ | Fall time $[s]$ | | **Difference** [s] |
| | | ATI | MicroForce sensor | |
| 1 | 14.730 | 0.100 | 0.105 | **0.005** |
| 2 | 20.725 | 0.105 | 0.110 | **0.005** |
| 3 | 26.730 | 0.105 | 0.110 | **0.005** |
| 4 | 32.735 | 0.100 | 0.105 | **0.005** |
| 5 | 38.725 | 0.115 | 0.120 | **0.005** |
| 6 | 44.740 | 0.095 | 0.105 | **0.010** |
| 7 | 50.755 | 0.080 | 0.09 | **0.010** |
| 8 | 56.755 | 0.085 | 0.095 | **0.010** |
| 9 | 62.740 | 0.100 | 0.110 | **0.010** |
| 10 | 68.760 | 0.085 | 0.095 | **0.010** |
| 11 | 74.745 | 0.100 | 0.110 | **0.010** |
| 12 | 80.745 | 0.100 | 0.115 | **0.015** |
| 13 | 86.750 | 0.100 | 0.110 | **0.010** |
| 14 | 92.765 | 0.090 | 0.100 | **0.010** |
| 15 | 98.750 | 0.105 | 0.115 | **0.010** |
| 16 | 104.765 | 0.095 | 0.105 | **0.010** |
| Mean | | 0.098 | 0.106 | **0.009** |
| SD | | 0.009 | 0.008 | **0.003** |

**Table 3.1:** Results from the steps response

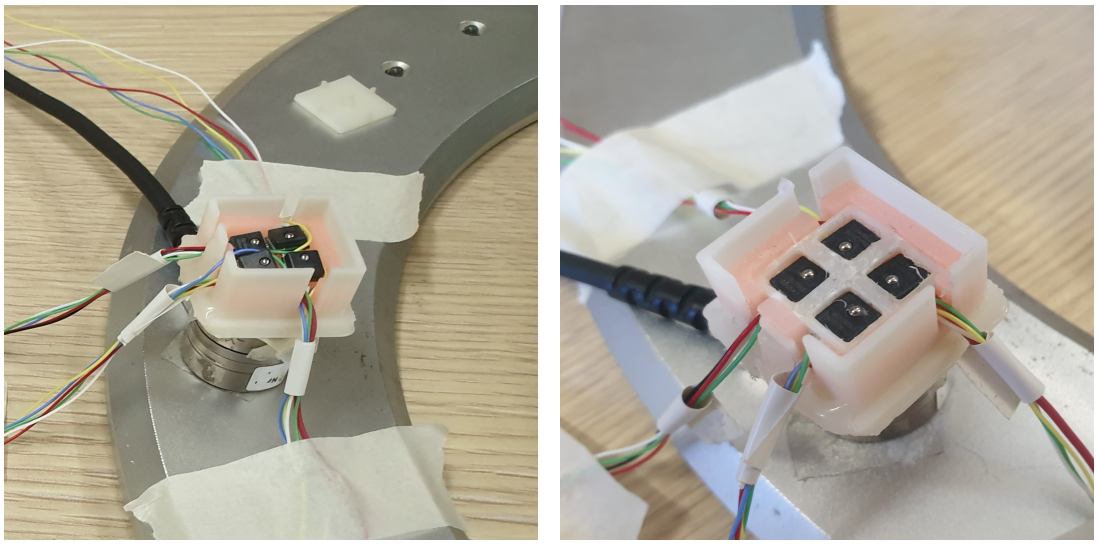
## 3.2 Haptic thimble realization

To allow the patient to perform the palpation task, a haptic thimble has been designed and then realized. The molds and all the ABS parts used during the realization process have been designed in Fusion360 (B.3) and then printed by using a 3D printer. The device consists of a square silicone case, with an overall dimension of 16.2 x 16.2 x 4.5 mm, with a hole of 12.6 x 12.6 x 4 mm placed in the center of it to hold the four sensors (Fig. 3.13(b)). Thanks to this design, when the sensors are placed inside the silicon case there is a thin 0.5 mm silicon layer between them and what is touched by using the haptic thimble. The material used is the *R-PRO-10* silicone (Reschimica s.r.l., IT) with shore hardness of A10. The *R-PRO-10* silicone is a bi-component addition rubber (base + platinum catalyst) of orange color that thanks to its low viscosity and high elasticity is versatile and easy to use. It comes with two parts: the base (A) and the catalyst (B). The optimal mixing ratio (weight or volume) to achieve a smooth and well cured elastomer is A/B = 1, the two parts have to be mixed for at least one minute in a clean and dry bowl (Fig. 3.13(a)). After the mixing, the liquid silicone has been poured into the ABS mold and left to cure for 2-3 hours (Fig. 3.13(b)). A challenge in the realization process has been to fix the four sensors in a suitable position inside the silicone case (Fig. 3.14(a)). To do that, an additive layer of liquid silicone has been poured inside the case, then a simple ABS grid has been used to hold the four sensors in the desired position (Fig. 3.14(b)). After the new silicone layer dried, the four sensors kept the correct position themselves and the ABS grid was removed. In order to fill the empty space left from the ABS grid, some more liquid silicone has been used to fill the empty region by using a syringe. Thanks to the syringe it has been possible to fill precisely the empty space left, without spill the liquid silicone on the sensors contact element.



(a) (b)

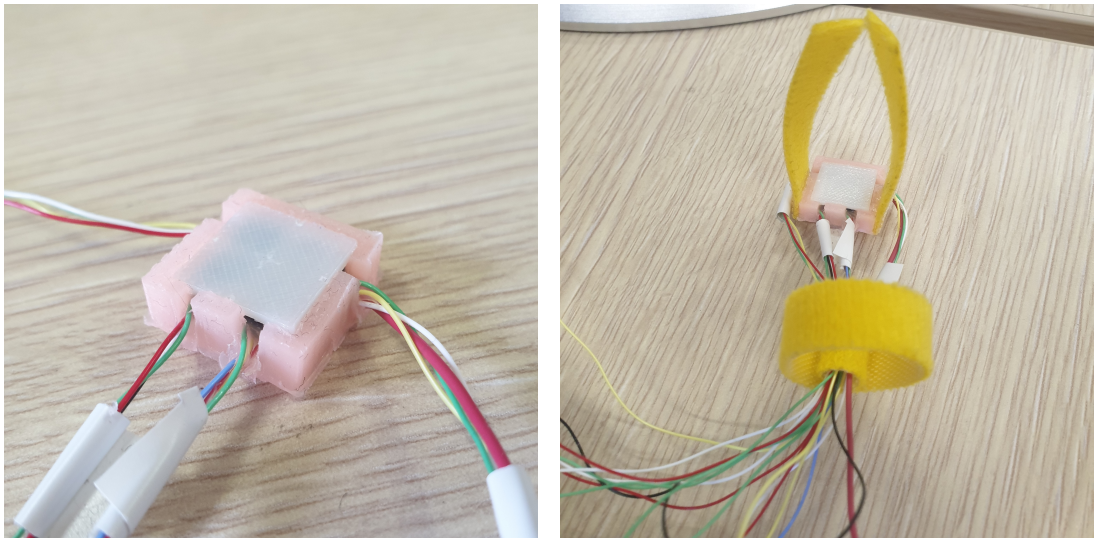
**Figure 3.13:** (a) Silicone used poured in the ABS mold. (b) Result after the drying time.



(a) (b)

**Figure 3.14:** Sensors placing before (a) and after (b) the the use of an ABS grid to keep them in place correctly.

Then, a rigid ABS layer of 12.6 x 12.6 x 1 mm has been used to close the silicon case and allow a correct distribution of the pressure exerted on the sensors by the user (Fig. 3.15(a)). After the new liquid silicone dried, to make the thimble wearable, some velcro stripes have been attached on it by using some glue 3.15(b)). The finished device, worn by a user, is shown in Fig. 3.16.



**(a)** **(b)**

**Figure 3.15:** (a) ABS layer closing the silicone case. (b) Velcro stripes to make the thimble wearable.

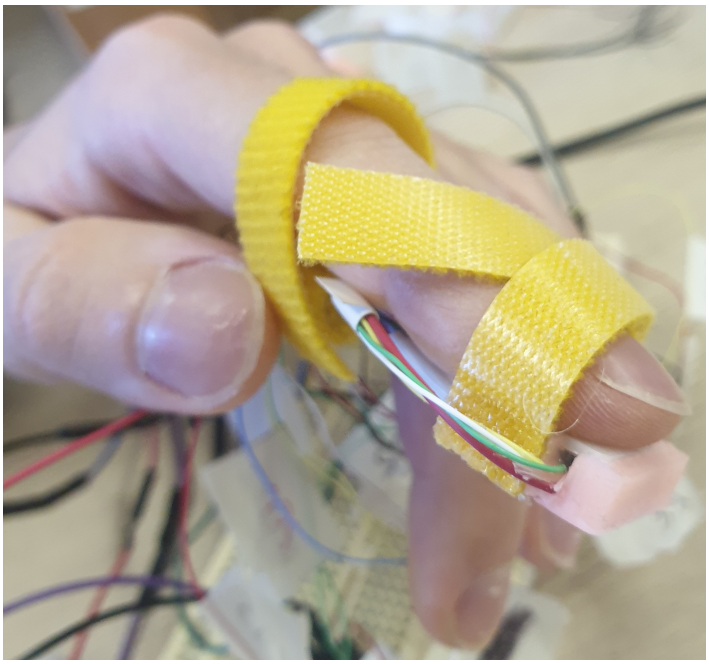


**Figure 3.16:** Thimble worn by a user.

## 3.3 Haptic thimble calibration

This haptic device is capable of measuring the distribution of the contact force among four different points represented by the contact elements of the sensors. At the end, we expect that the measurement of the contact force exerted on the haptic thimble is given by:

$$F_c = \sum_{i=1}^{4} F_i$$

where $F_c$ is the contact force exerted by the patient's finger and $F_i$ is the force measured by the $i$-th sensor. Even though the latter would be the ideal behaviour of the haptic thimble, in the real case the measurements obtained are affected by errors due the D.I.Y. realization of the device.

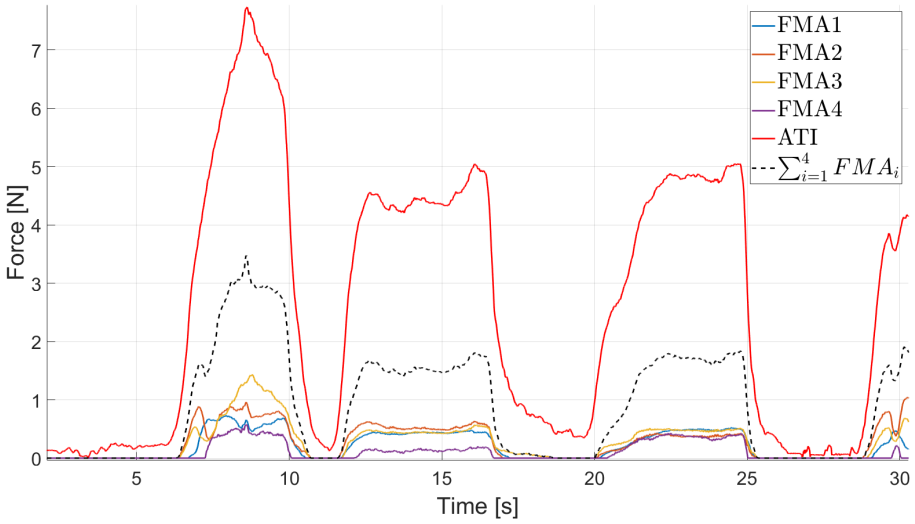



**Figure 3.17:** Comparison between the contact force (exerted by the user's index finger) measured by the haptic thimble and the ATI.

To evaluate the measurements quality of the haptic thimble the latter has been placed on the ATI transducer, and then an input force has been applied by means of the index finger, as the end user would do while wearing it (Fig. 3.17). It can be noticed that the forces measured by the haptic thimble are much smaller than those measured by the ATI, this because by realizing the device with a D.I.Y approach, the position of each sensor inside the thimble differs a little from the one expected. This caused the presence of some unloading force on the sensors structure that introduced some measurements errors. In any case, it is possible to improve the quality of measurements by calibrating the thimble. To calibrate the thimble, the same protocol performed in 3.1, to test a single sensor, has been adopted. The thimble has been placed on the top of the ATI transducer and then a stair-step input has been applied by using the Omega.3 (Fig. 3.18): in particular, a progressive load from 0 to 5 N followed by a decrease of the force from 5 to 0 N in a time period of about 15 s has been exerted on the thimble, by using force steps of about 0.5 N.

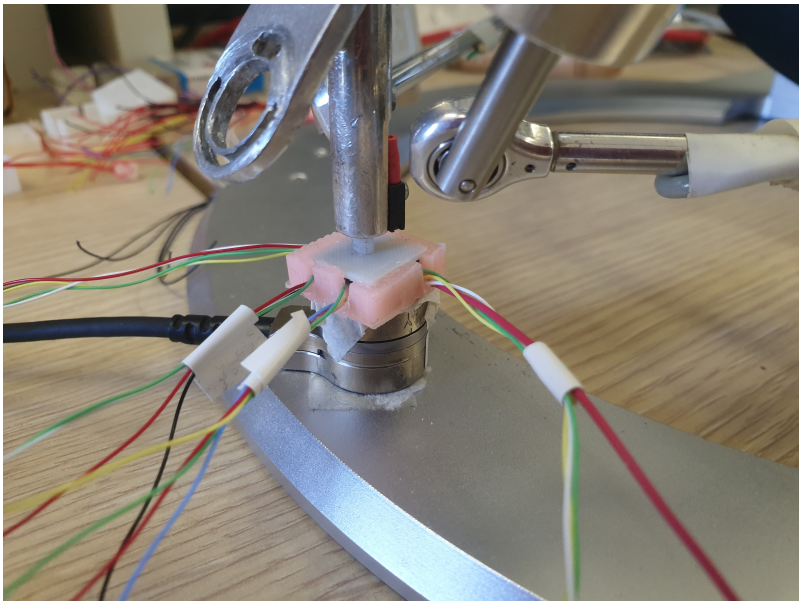


**Figure 3.18:** Omega.3 used to apply some input force on both the thimble and the ATI.

In this way it has been possible to collect the data to be used during the calibration procedure, by performing twenty-seven repetitions of data acquisition (Fig. 3.19).

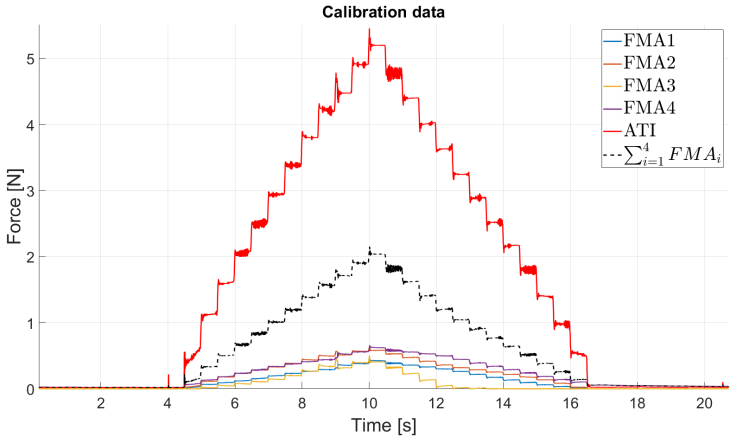



**Figure 3.19:** One of the 27 data acquisition performed to do the calibration.

A classical way to calibrate the thimble is to relate the measurements given by the thimble with the ones given by some ground-truth (in our case the ATI), and then create a new linear transfer function that characterizes the response of the thimble (Fig. 3.20).

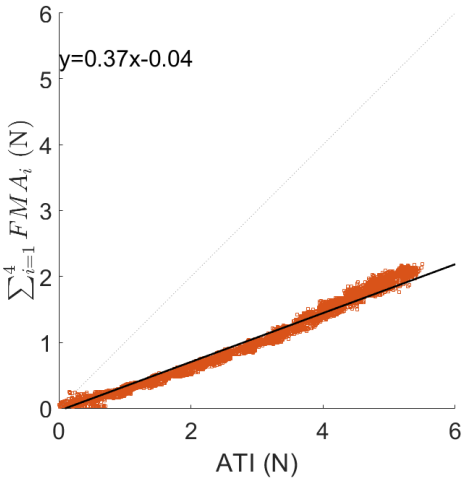



**Figure 3.20:** Relationship between the thimble and the ATI measurements.

As result, we can say that the force measured from the haptic thimble is given by:

$$F_c = \frac{1}{\beta}(\sum_{i=1}^{4} FMA_i + \alpha)$$

where $F_c$ is the contact force measured by the thimble, $FMA_i$ is the force measured by the i-th sensor and $\alpha, \beta$ are the calibration parameters equal respectively to 0.04 and 0.37. However, by using this approach, an important limit is introduced in our application: one of our goals is to use the measurements given by the four sensors to model how the contact force distributes among the four contact points represented from their sensing contact element, in this way it would be possible to understand how the patient's finger pressure distributes on the touched skin surface. By calibrating the thimble with this approach, we improve the overall contact force measurement accuracy, but the accuracy of the force measurements of each individual sensor is not improved.

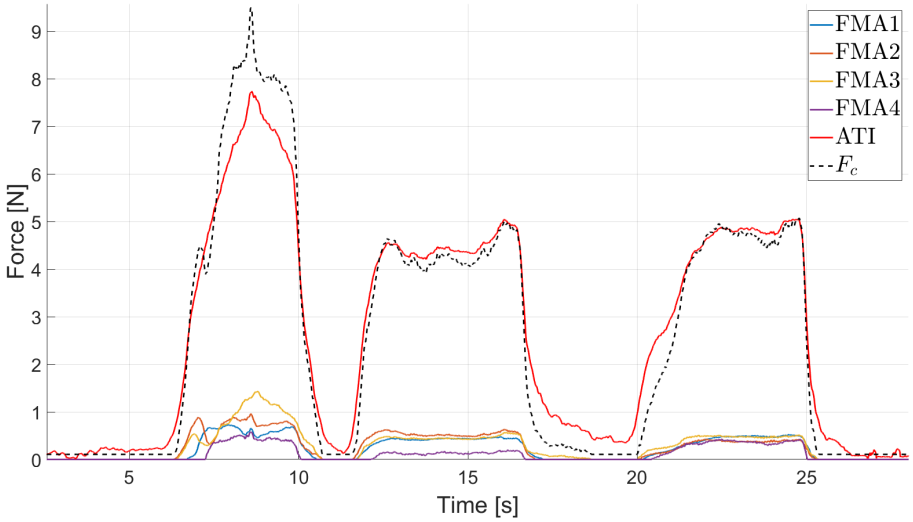



**Figure 3.21:** Same comparison as in Fig. 3.17 after calibration.

By applying this new transfer function to the same force signal shown in Fig. 3.17, the overall contact force measured by the thimble is more accurate (Fig. 3.21) with a RMSE of 0.71 N (before the calibration procedure the RMSE was 2.1 N). However the quality of measurements of each sensor taken individually remained unchanged. To avoid this limit, we need a calibration procedure able to give as result, a correction coefficient for each sensor in order to improve the quality measurement of the latter:

$$F_c = \sum_{i=1}^{4} x_i FMA_i$$

To do that, we can model a calibration procedure based on the estimation of four coefficients able to minimize the SSE (Sum of Squared Errors) between the measurements given by the thimble and a ground-truth (the ATI):

$$\begin{aligned} \min_{x} \quad & \|y - Ax\|_2^2 \\ \text{s.t.} \quad & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

where $y \in R^{N,1}$ is the vector containing the ground-truth measurements, $A \in R^{N,4}$ is the matrix containing the measurements of each sensor and $x \in R^{4,1}$ is the coefficients vector to be estimated ($N$ is the number of measurements). By leaving as it is this minimization problem, the solutions space is however too large and full of wrong solutions from a physics point of view; for example one possible solution could be a vector $x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T$ with a $\|x\|_0$ ($l_0$-*norm*, that is the number of nonzero entries of $x$) not equal to four, that is to say, a result that points to minimize the SSE by emphasizing the measurements given by only some of the four sensors embedded in the thimble. To limit our solutions space we have to introduce some constraints inside our minimization problem. By considering that a rigid layer is placed between the sensor and the patient finger, we know that if we exert a contact force exactly on the centre of the layer, the latter will equally distribute the force on all the four sensors (Fig. 3.22).

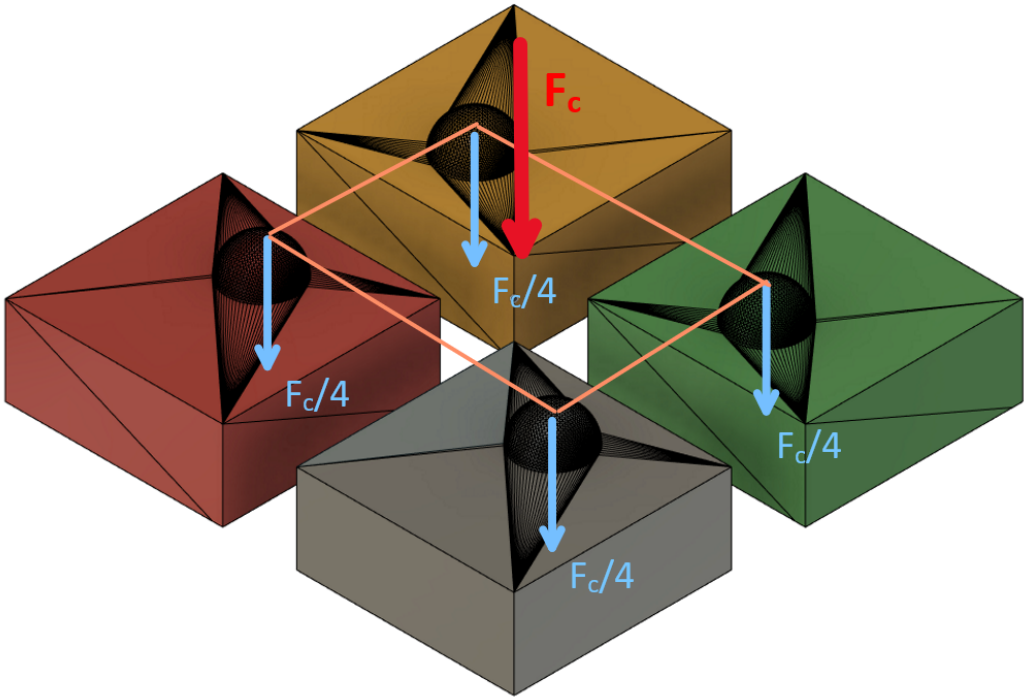



**Figure 3.22:** Simple scheme of how a rigid layer would distribute ideally the contact force exerted at the center of it on the four sensors.

As consequence, if during the data acquisition for the calibration, we exert the input force exactly at the centre of the thimble's rigid layer, we can affirm that in each time instant it must be true the equation

$$a_{ij}x_j - y_i/4 = 0 \qquad \forall j = 1,2,3,4 \qquad \forall i = 1,\ldots,N$$

where $a_{ij}$ is the i-th measurement of the j-th sensor, $x_j$ is the calibration coefficient of the j-th sensor and $y_i$ is the corresponding measurement of the ATI. However, these equality constraints are too strict and based on the idea that the realization quality of the thimble is perfect (which is not true since we want to calibrate it to eliminate the inaccuracies due the manufacturing defects). In order to relax the constraints, we can turn them from equality to inequality by introducing a

relaxation parameter:

$$|a_{ij}x_j - y_i/4| \leq \epsilon \qquad \forall j = 1,2,3,4 \qquad \forall i = 1,\ldots,N$$

where $\epsilon$ is the relaxation parameter to be tuned. With this in mind, we can write the final form of our minimization problem:

$$\begin{aligned} \min_x \quad & \|y - Ax\|_2^2 \\ \text{s.t.} \quad & |a_{ij}x_j - y_i/4| \leq \epsilon \qquad \forall j = 1,2,3,4 \\ & \qquad\qquad\qquad\qquad\quad \forall i = 1,\ldots,N \end{aligned}$$

It can be noticed that this problem it is nothing but a constrained linear least-squares problem, easily solved by using the *lsqlin(...)* MATLAB command (B.2). To use this MATLAB function, the constraints have to be specified in a matrix form, so that we have a constrained least-square problem written in the standard form:

$$\begin{aligned} \min_x \quad & \|y - Ax\|_2^2 \\ \text{s.t.} \quad & Cx \leq b \end{aligned}$$

In our case the $C$ matrix and the $b$ vector are

$$C = \begin{bmatrix} \text{diag}(A^{(i)}) \\ \vdots \\ \text{diag}(A^{(N)}) \\ \text{diag}(-A^{(i)}) \\ \vdots \\ \text{diag}(-A^{(N)}) \end{bmatrix}^{2\times4\times N,4} \qquad b = \begin{bmatrix} \mathbf{1}\cdot(y^{(i)}/4 + \epsilon) \\ \vdots \\ \mathbf{1}\cdot(y^{(N)}/4 + \epsilon) \\ \mathbf{1}\cdot(y^{(i)}/4 - \epsilon) \\ \vdots \\ \mathbf{1}\cdot(y^{(N)}/4 - \epsilon) \end{bmatrix}^{2\times4\times N,1}$$

where $A^{(i)}$ is the i-th row of the measurement matrix A, $y^{(i)}$ is the i-th ATI measurement and $\mathbf{1} \in R^{4,1}$ is an all-ones vector. By fixing the $\epsilon$ to 0.88 and by

setting the "*interior point*" algorithm for the optimizer the result is

$$x = \begin{bmatrix} 1.455 \\ 3.407 \\ 2.103 \\ 3.275 \end{bmatrix}$$

By using this new calibration procedure to correct the sensors measurements of the same force signal shown in Fig. 3.17, we obtain the result shown in Fig. 3.23.

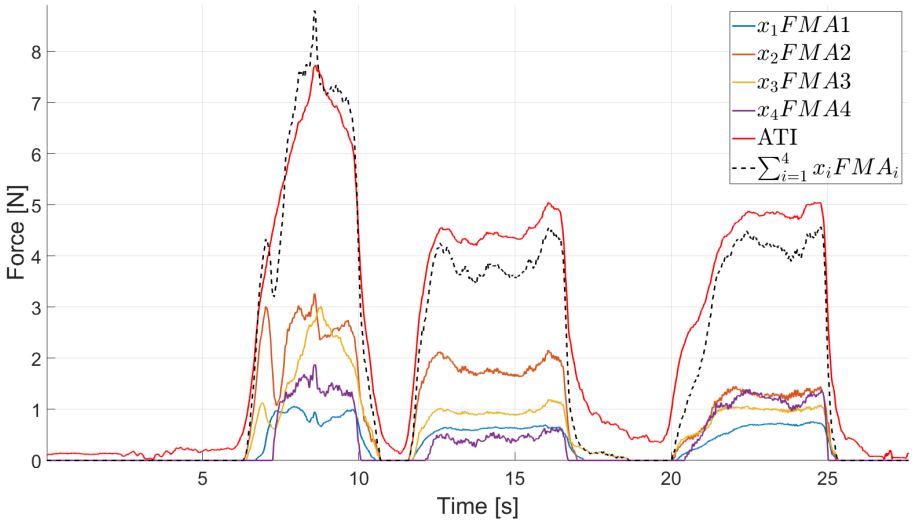



**Figure 3.23:** Same comparison as in Fig. 3.17 after the new calibration procedure.

It can be observed that with this new calibration procedure not only the overall measurement of the contact force has been improved, with a RMSE of 0.74 N (before the calibration procedure the RMSE was 2.1 N), but also the measurement given by each sensor taken individually.

### 3.3.1 Calibration quality analysis

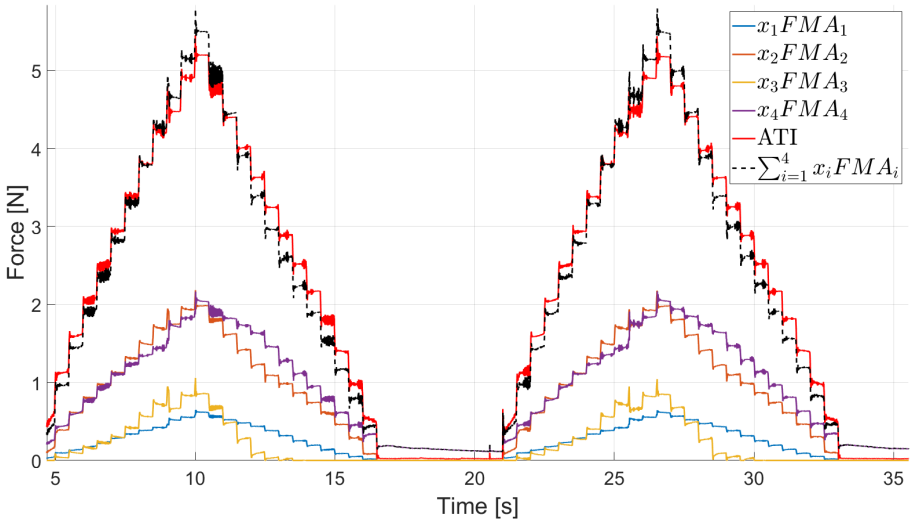



**Figure 3.24:** Stairs-step input force after applying the found calibration coefficients.

Since the data to perform the calibration have been collected by exploiting the same input force signal used in section 3.1.1 (Stairs-step input), it is possible to do a quick verification of the calibration quality in a similar way. All the 27 acquisitions collected during the calibration procedure have been used to perform the analysis. The haptic thimble performed well with a root mean square error (RMSE) of 0.18 N, a mean difference (MAE) of 0.16 N and a difference standard deviation of 0.086 N w.r.t. the ATI measurements. The relation between the two sets was pretty linear with a Pearson correlation coefficient (squared) of about 0.99 (Fig. 3.25). The reproducibility coefficient (RPC) was 0.36 N and the coefficient of variation (CV) was 8.8%. The two curves provided no statistical difference with a $p - value$ near 0. Furthermore, the two distributions were statistically equivalent (two one-sided t-test) with 95 percent CI (Confidence Interval) within the equivalence interval $-0.0116 : 0.0116$ N.

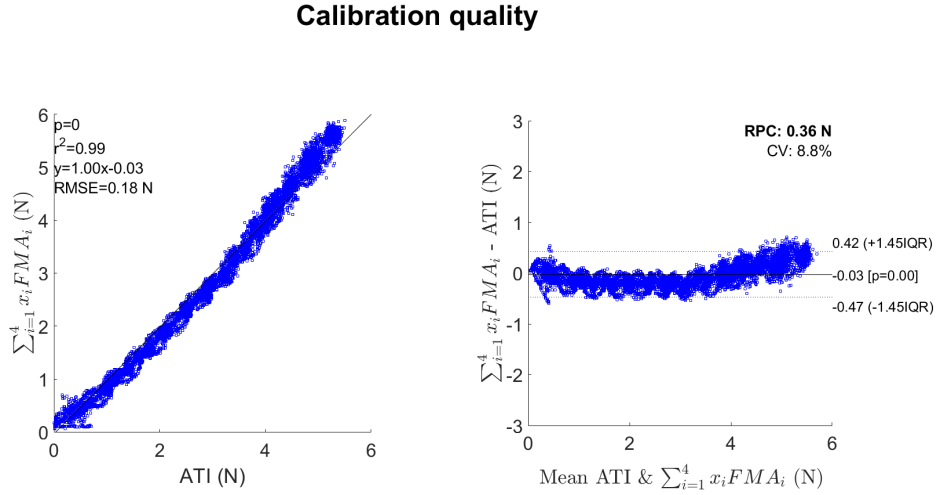



**Figure 3.25:** Distributions of the measured force for the thimble after the calibration and the ATI. Bland-Altman plot (right panel).

In order to understand of much the calibration improved the thimble performances, the same analysis was made without applying the calibration coefficients to the thimble measurements, then the two results were compared (Tab. 3.2).

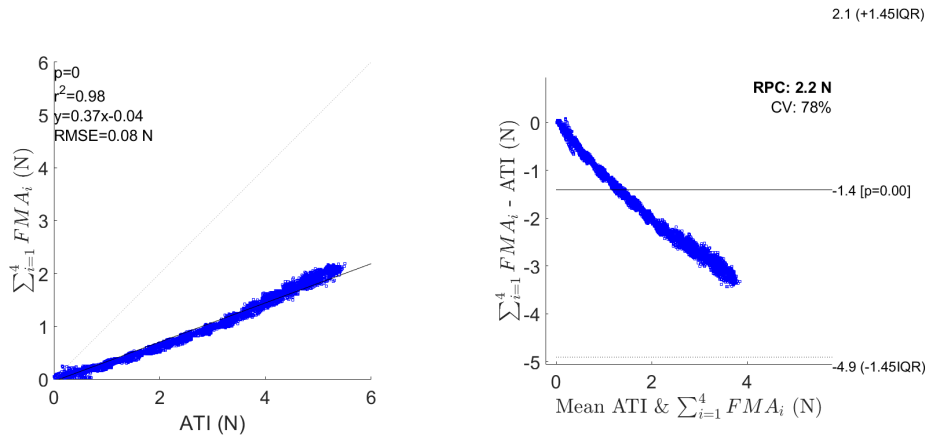



**Figure 3.26:** Distributions of the measured force for the thimble before the calibration and the ATI. Bland-Altman plot (right panel).

| Thimble performances | | |
|---|---|---|
| | without cal. coeff. | with cal. coeff. |
| Max Abs. Err. | 3.43 N | **0.71 N** |
| RMSE | 1.75 N | **0.18 N** |
| MAE | 1.37 N | **0.16 N** |
| Diff. SD | 1.1 N | **0.086 N** |
| RPC | 2.2 N | **0.36 N** |
| CV | 78% | **8.8 %** |
| Pearson corr. coeff. | 0.98 | **0.99** |

**Table 3.2:** Comparison between performances before and after the calibration procedure.

# Chapter 4

# Video-Call application

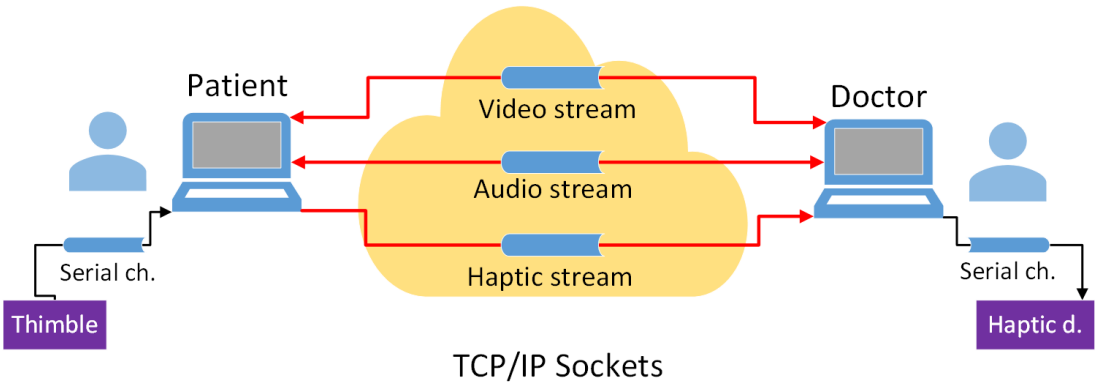



**Figure 4.1:** Video-Call application schematic.

Since the framework for the remote palpation reckon on a communication between a patient and a doctor, it was thought to develop a simple video-call application that allows to exchange not only video and audio data, but also the haptic information sent by the thimble (on the patient side) and received by a haptic feedback device (at the doctor side) (Fig. 4.1). The application was developed by using Python programming language, exploiting TCP/IP sockets and the multithreading programming paradigm, and it is based on two *.py* files (*Patient.py* and *Doctor.py*)

to be ran on the patient/doctor' s computers (more details about the application structure and the code development can be found in the appendix B.4). TCP/IP sockets have been choosed over the UDP sockets (more suitable to develop a data streaming application, since the UDP protocol is much faster than the TCP) in order to avoid the manual packets sequencing during the transmission and keep the application developing simple. For each data stream there is a server that sends the data and a client on the other side that receives it; each server and client run on an independent thread (Fig. 4.2).

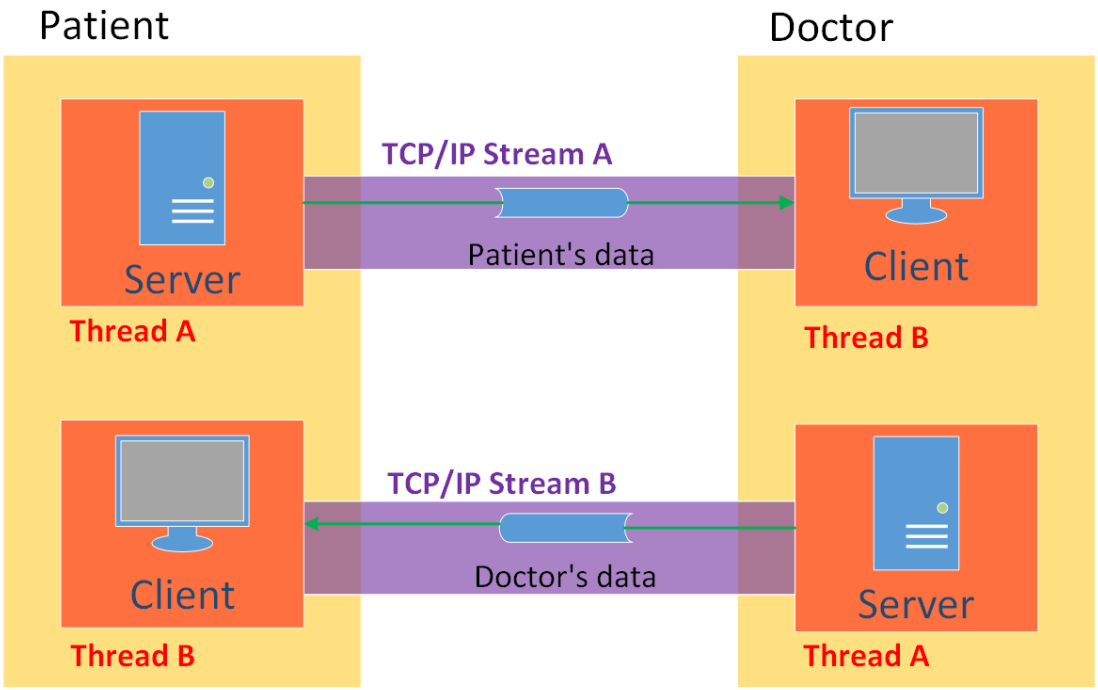



**Figure 4.2:** Data exchange between patient and doctor schematic.

For example, to send the video data from the patient to the doctor, a server is run on an independent thread, waiting for a connection from the client running on the doctor side (even the latter ran by an independent thread). When a connection between the two sides is established, the server will start to read the video frames

from the patient webcam (the video stream has been managed by using the OpenCV python library), and after having converted them in a *JPEG* coding, in order to reduce the data size, it will send them to the client. The client will receive the *JPEG* coding from the server, and after converting it to its original format, it will show the corresponding image on the monitor. The same communication mechanism has been implemented for the video stream from the doctor to the patient side and in an analogous way for any other data exchange between the two sides (audio and haptic data). By using this data exchanging mechanism is it possible for the patient and doctor to send and receive data simultaneously, without waiting each other response, and this reduces the latency during the data transmission. A little difference is present in the communication mechanism of the haptic data, since the latter have to be red from the thimble via serial protocol at the patient side, to be sent by their assigned TCP/IP stream to the doctor, and then to be written on a serial port in order to be red from the doctor's haptic feedback device (Fig. 4.3).

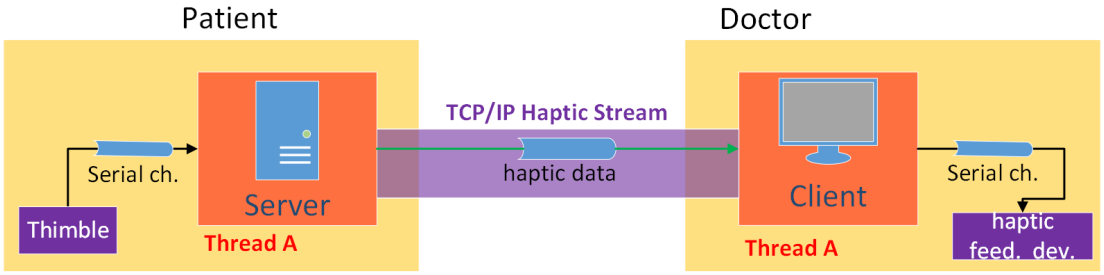



**Figure 4.3:** Haptic data exchange mechanism schematic

The used 1-DoF device is composed of two platforms: one placed on the nail side of the finger and the other in contact with the finger pad. Three cables and three springs connect the two parts, while one small servomotor controls the length of the cables, in this way the displacement between the two platform allows us to display a force at the user's fingertip. The device is wired to an Arduino *nano*,

which is in charge to read from the serial port an angle $\alpha$ in order to control the device's servomotor (Fig. 4.4).

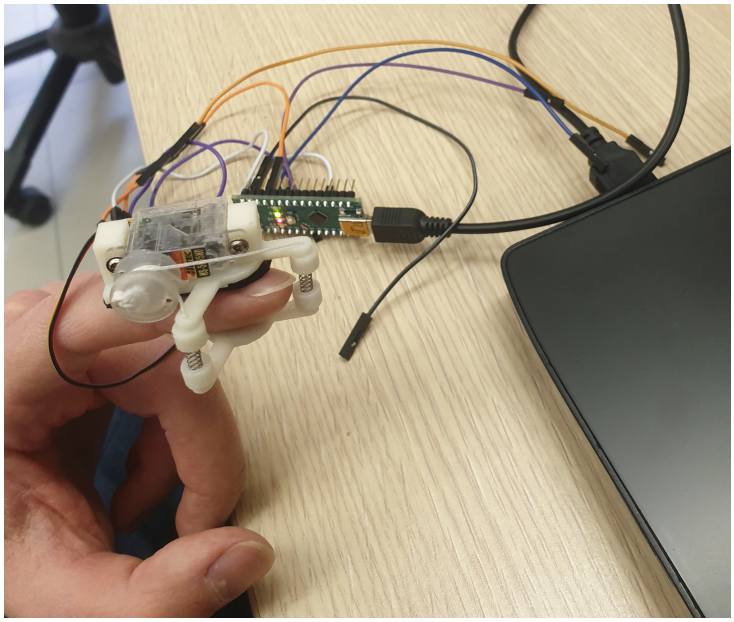


**Figure 4.4:** Haptic feedback device wired to an Arduino.

The relation between the platform displacement given by the servomotor's angle and the desired force $F_p$ to be applied to finger pad by the mobile platform (which is in our case the contact force measured by the haptic thimble) is given by:

$$\begin{cases} \Delta p = \Delta \alpha r \\ F_p = K_f^{-1} \Delta p \end{cases}$$

where $r = 5.5mm$ is the servomotor's pulley radius, the stiffness coefficient $K_f = 0.5N/mm$ model the the isotropic elastic behavior of the finger pad, and $\Delta p$ is the displacement of the platform with respect to its initial position, *i.e.*, when the platform is in contact with the finger pad without producing any skin deformation [10]. To have a measure of the latency introduced by the application during the communication, the latter has been ran on two PC belonging to the same LAN,

able to communicate each other by means of a wireless access point (scheme in Fig. 4.5).

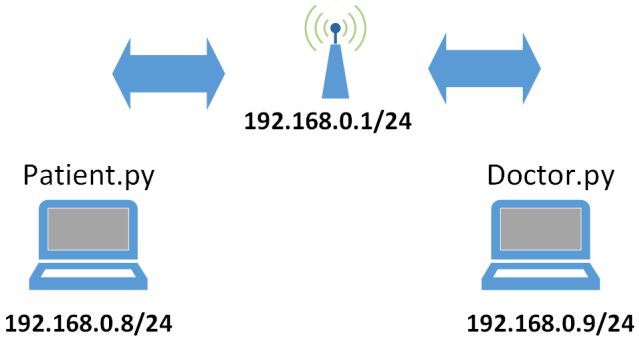



**Figure 4.5:** Network configuration used during tests.

The data have been labelled with a timestamp before being sent to their corresponding stream, and then, after being received on the other side, the transmission delay has been computed. Once computed, the delay has been registered on log files in order to do an evaluation of the latency introduced by the application through MATLAB (Fig. 4.6).


**Figure 4.6:** Latency introduced by the application during the audio-video data transmission.

The main results of the latency tests have been collected in Table 4.1.

| Latency tests results | | |
|---|---|---|
| Data type | Latency [s] | |
| | Mean | SD |
| Video | 0.61 | 0.02 |
| Audio | 0.79 | 0.01 |
| Haptic | 0.58 | 0.02 |

**Table 4.1:** Latency introduced by the application for the different data type.

It can be noticed how the latency introduced by the application during the tests remains below one second for all the different type of exchanged data, thus ensuring a good user experience during the audio-video communication. Since every different type of data have been exchanged through an independent TCP/IP stream, the latency introduced by the application is different for every data type. In other words, the different exchanged data are not perfectly synchronized during the communication, however since the difference between the various latencies is quite small, the audio-video communication experience is good enough (this statement is also confirmed by some volunteers who have be asked to try the application).

# Chapter 5

# Conclusions

This work of thesis aims at extending a previously developed framework for remote palpation in the telemedicine field. A novel haptic thimble has been designed and then validated. With respect to the sensing devices already documented in literature, the main difference lies in the use of four small form-factor force sensors, that allow a thimble overall size suitable to be worn on a human finger. In this way it has been possible to overcame the main limitation given by a similar device embedding only one force sensor: the contact force measurement inaccuracy due to the finger softness, which causes a force unloading on the sensor structure. By positioning the four force sensors so that their contact sensing elements are as close as possible to each other, it has been possible to extend the finger area that exterts the contact force and thus to reduce the measurement inaccuracy. The use of four sensors brought additional advantages: since the contact force exerted by the patient's finger is applied on four contact points, it is possible to understand how the exerted force distributes on the pressed area, allowing the doctor to perceive not only a one-dimensional force feedback but a more realistic three-dimensional force feedback (by using a suitable force rendering device). To develop a reliable haptic thimble for contact force measurement, single sensor performance was evaluated

to establish a baseline for the thimble measurement quality, then a proper case have been realized by using liquid silicone and some ad hoc ABS molds, printed by using a 3D printer. A rigid material layer has been placed on the sensors, in order to avoid the remaining force unloading phenomena. Since the device has been realized by using a D.I.Y. approach, some inaccuracies in the contact force measurements occurred. To overcame this problem two calibration procedures have been introduced and analyzed to evaluate which one is more suitable for our application. Since the previously developed framework reckon on a communication between a patient and a doctor, it was developed a simple video-call application that allows to exchange not only video and audio data, but also the haptic information sent by the thimble (on the patient side) and received by a haptic feedback device (at the doctor side). To evaluate the video-call application performances, the latter have been ran on two PC belonging to the same LAN, then the latency introduced during the communication has been measured.

The developed prototype system shows great potential and can be enhanced in future work. In this thesis the focus was the realization of the haptic thimble and the video-call software developement, but further investigation can be conducted to evaluate how the contact force measured by the thimble distributes on the patient skin surface, thanks to the four sensors embedded in it. For example, the four force measurements could be used to model a level curve force, in order to better understand how the patient finger pressure distributes on the touched skin (Fig. 5.1(a)). The measurements of the contact force distribution allow us to better understand how the patient's fingertip is oriented while the latter is touching his own skin (Fig. 5.1(b)). By exploiting this new feature it is possible to interface the haptic thimble with a 3-Dof force feedback device. A suitable device could be for example the one presented in [14] by Chinello *et al.*, which is composed of a static upper body and a mobile end-effector: the upper body is located on

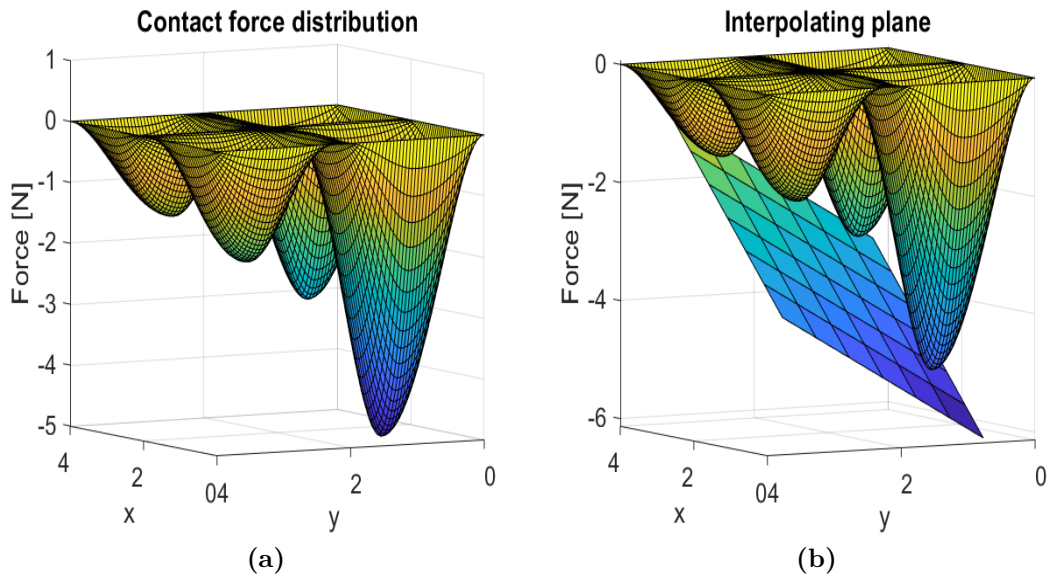



**Figure 5.1:** (a) Contact force distribution given by the four sensors. (b) A plane interpolating the four force curves, in order to better understand the patient's fingertip orientation while the latter is performing the indentation task.

the nail side of the finger, supporting three small servo motors, and the mobile end-effector is in contact with the finger pulp. The two parts are connected by three articulated legs, actuated by the motors. The end-effector can move toward the user's fingertip and rotate it to simulate contacts with arbitrarily-oriented surfaces (Fig. 5.2). A possible study could be conducted to find a relation between the force measurements given by the four haptic thimble's force sensors and the orientation angles of the haptic feedback device end-effector, in order to display an oriented force at the doctor's fingertip.

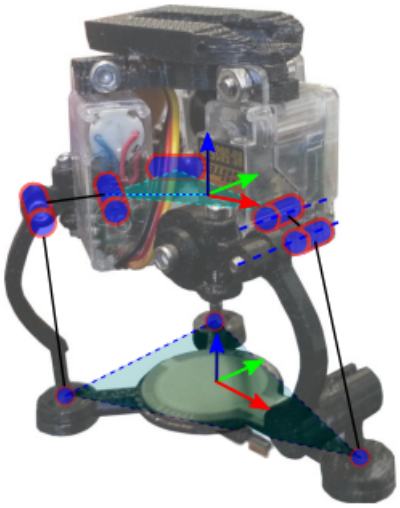


**Figure 5.2:** 3RRS wearable fingertip device.
Reproduced from [14].

# Appendix A

# Hardware description

## A.1 Omega.3

**Figure A.1:** `https://www.forcedimension.com/products`

Haptic devices which transmit digital information to the user through the sense of touch. This tactile form of interaction greatly enhances the ease to use and interact with complex 2D and 3D applications as this haptic device is capable of rendering with great accuracy the curvature, stiffness and texture properties of all forms of material. The omega.3 relies on a unique kinematic design that has been optimized for high-end force feedback. Its high mechanical stiffness, combined with

its embedded fast USB 2.0 controller, enables the rendering of crisp contact forces.

| | | |
|---|---|---|
| **omega.3** | | |
| **workspace** | translation | ∅ 160 x 110 mm |
| **forces** | translation | 12.0 N |
| **resolution** | translation | ‹ 0.01 mm |
| **stiffness** | closed-loop | 14.5 N/mm |
| **electronics** | | |
| **interface** | standard | USB 2.0 |
| | refresh rate | up to 4 KHz |
| **power** | universal | 110V - 240V |
| **software** | | |
| **platforms** | Microsoft | Windows |
| | Linux | all distributions |
| | Apple | macOS |
| | Blackberry | QNX |
| | WindRiver | VxWorks |
| **software** | haptic SDK | |
| | robotic SDK | |
| **features** | | |
| **structure** | delta-based parallel kinematics | |
| | active gravity compensation | |
| **calibration** | automatic | |
| | driftless | |
| **user input** | 1 programmable button | |
| **safety** | velocity monitoring | |
| | electromagnetic damping | |
| **option** | right- or left-handed | |

**Table A.1:** Omega.3 specs

Thanks to the Omega.3 it has been possible to test and validate the sensors and the haptic thimble by using it as a force signal generator. To do that, the Omega.3 end-effector has been programmed to move along its Z-axis in order to exert the desired force on the sensors. The code to move the end-effector has been written in C by using VisualStudio 2017 and the Omega.3 API.

## A.2 TCA9548A

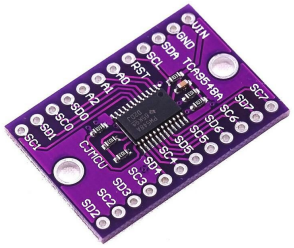



**Figure A.2:** TCA9548A multiplexer

The TCA9548A device has eight bidirectional translating switches that can be controlled through the I2C bus. The SCL/SDA upstream pair fans out to eight downstream pairs, or channels. Any individual SCn/SDn channel or combination of channels can be selected, determined by the contents of the programmable control register. These downstream channels can be used to resolve I2C slave address conflicts since the sensors used have all the same, not modifiable, physical address. Here below the Arduino function written to select the output channel is shown:

```
//Arduino function to select the output channel by writing the
    multiplexer control register
...
void tca_select(uint8_t i) {
  if (i > 7) return;
  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  int err = Wire.endTransmission();
  if(err!=0)
    Serial.println("MP err");
}
```

## A.3 Teensy 3.2

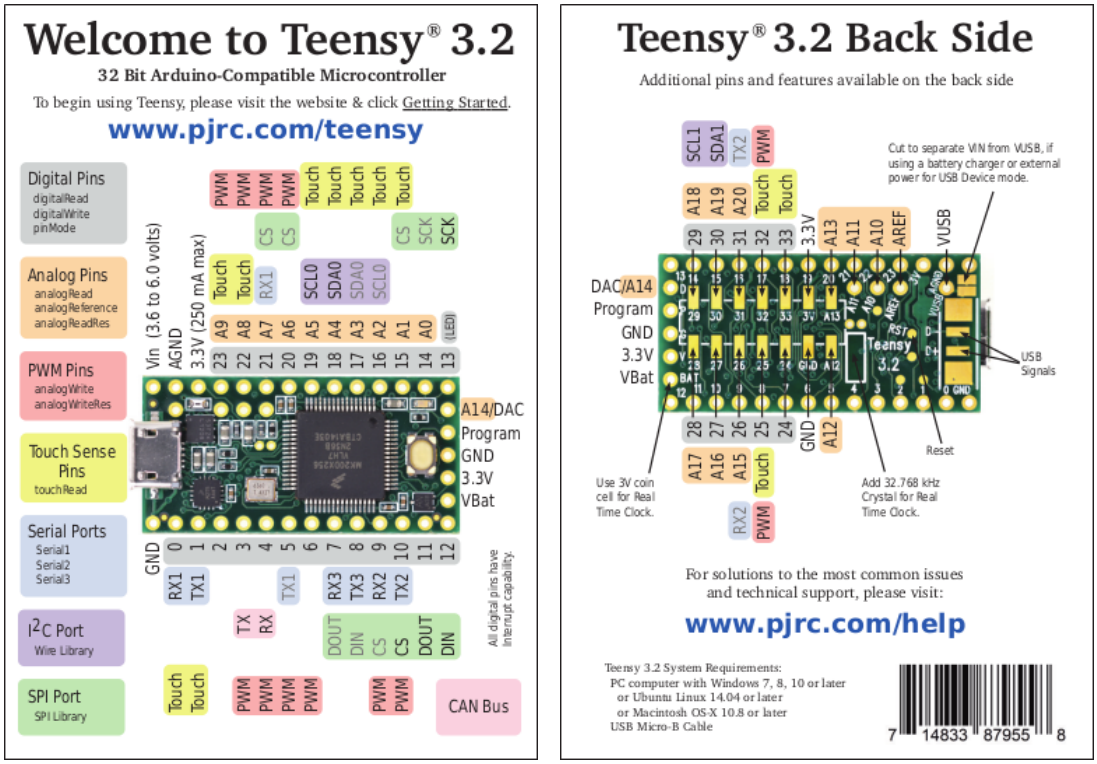


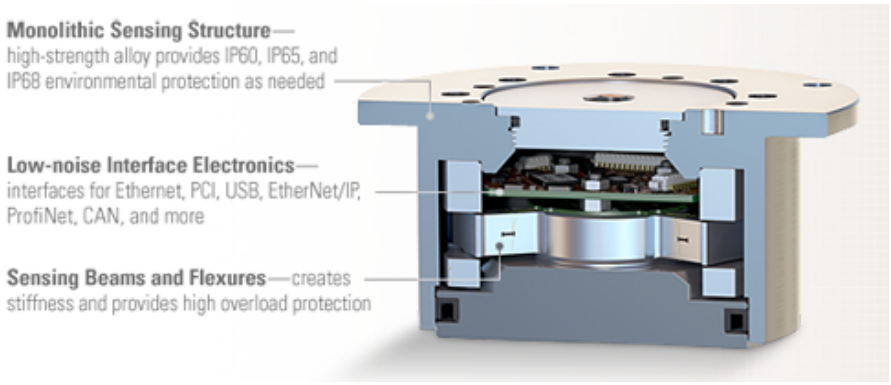

**Figure A.3:** `https://www.pjrc.com/store/teensy32.html#tech`

The Teensy is a breadboard-friendly development board with loads of features in a, well, teensy package. Each Teensy 3.2 comes pre-flashed with a bootloader so you can program it using the on-board USB connection so no external programmer needed. It is possible program for the Teensy in any editor using C or by installing the Teensyduino add-on for the Arduino IDE and write Arduino sketches. Two Arduino sketches have been written, one to make the teensy a ROS node able to publish the measured data on a proper topic and then collect them (used during the MicroForce sensors test and validation phase (3.1)), and one to write the measured data on a serial port in order to be able to read them in a more generic way. Here below the main parts of the written sketches have been reported:

```
//Part of the Arduino sketch to read data from sensors and publish
    them on a ROS topic
#include <i2c_t3.h>
#include <ros.h>
#include <fma_msgs/FMAdata.h>
ros::NodeHandle  nh;
fma_msgs::FMAdata data;
ros::Publisher pub_data("fma_data", &data);
uint16_t cnt[4] = {0};
...
void send_data(void){
  for(int k=0; k<4; k++){
    uint8_t a[2] = {0};
    tca_select(k);
    Wire.requestFrom(FMADDR, 2, I2C_STOP);
    Wire.read(a ,2);

    a[0] &= 0x3F;
    cnt[k] = (a[0] << 8) | a[1];
  }

  data.cnt1 = cnt[0];
  data.cnt2 = cnt[1];
  data.cnt3 = cnt[2];
  data.cnt4 = cnt[3];
  data.timeStamp = millis();

  pub_data.publish(&data);
  nh.spinOnce();
}
...
```

```
1  //Part of the Arduino sketch to read data from sensors and send them
       by serial protocol
2  #include <i2c_t3.h>
3
4  uint16_t cnt[4] = {0};
5  ...
6  void send_data(void){
7    for(int k=0; k<4; k++){
8      uint8_t a[2] = {0};
9      tca_select(k);
10     Wire.requestFrom(FMADDR, 2, I2C_STOP);
11     Wire.read(a ,2);
12
13     a[0] &= 0x3F;
14     cnt[k] = (a[0] << 8) | a[1];
15
16     Serial.print(cnt[k],DEC);
17     Serial.print(" ");
18   }
19
20   Serial.print(millis());
21   Serial.println("");
22 }
23 ...
```

## A.4 ATI NET F/T



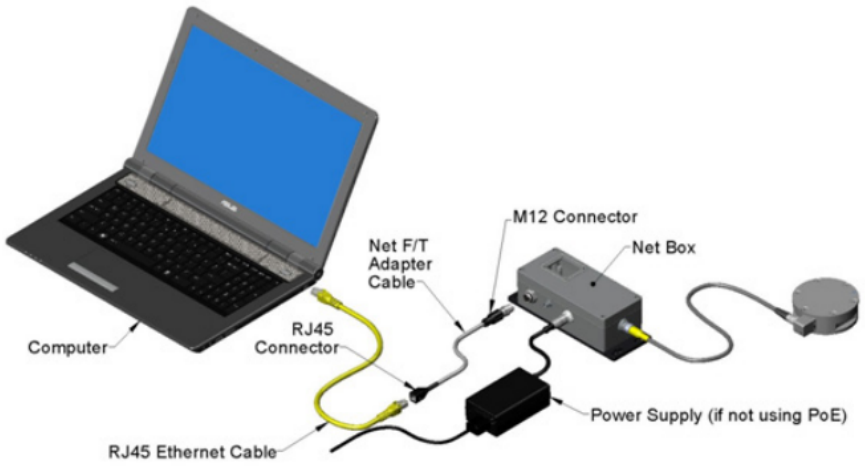




**Figure A.4:** `https://www.ati-ia.com/products/ft/ft_productDesc.aspx`

The ATI Multi-Axis Force/Torque Sensor system measures all six components of force and torque. It consists of a transducer, interface electronics and cabling. The compact and rugged monolithic transducer uses silicon strain gages to sense forces. The transducer's silicon strain gages provide high noise immunity and allow high overload protection, which is standard on all models.

| Table 5.9—Nano17 IP65/IP68 Physical Properties | | |
|---|---|---|
| **Single-Axis Overload** | **(US) Standard Units** | **(SI) Metric Units** |
| Fxy | ±56 lbf | ±250 N |
| Fz | ±110 lbf | ±480 N |
| Txy | ±14 inf-lb | ±1.6 Nm |
| Tz | ±16 inf-lb | ±1.8 Nm |
| **Stiffness (Calculated)** | | |
| X-axis & Y-axis forces (Kx, Ky) | $4.7 \times 10^4$ lb/in | $8.2 \times 10^6$ N/m |
| Z-axis force (Kz) | $6.5 \times 10^4$ lb/in | $1.1 \times 10^7$ N/m |
| X-axis & Y-axis torque (Ktx, Kty) | $2.1 \times 10^3$ lbf-in/rad | $2.4 \times 10^2$ Nm/rad |
| Z-axis torque (Ktz) | $3.4 \times 10^3$ lbf-in/rad | $3.8 \times 10^2$ Nm/rad |
| **Resonant Frequency** | | |
| Fx, Fy, Tz | 2200 Hz | 2200 Hz |
| Fz, Tx, Ty | 2200 Hz | 2200 Hz |
| **Physical Specifications** | | |
| Weight[1] | 0.09 lb | 0.0408 kg |
| Diameter[1] | 0.79 in | 20.1 mm |
| Height[1] | 0.873 in | 22.2 mm |
| Note:<br>1. Specifications include standard interface plates. | | |

**Table A.2:** Trandsucer specs

# Appendix B

# Software description

## B.1 ROS

ROS [15] is an open-source, meta-operating system for robots. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server. The employed ROS distribution for this thesis is ROS Noetic Ninjemys, released on May 23rd, 2020. It was decided to use ROS because of the need of collect the measured forces given by the ATI sensor and the MicroForce sensors in a centralized way. To exploit this solution have been used two ROS packages, *netft_utils* [16] and *rosserial_arduino* [17], so as to make both the ATI and the Teensy a ROS node. In this way the ATI could publish its own data on a custom topic called *netft_data* by using a standard *WrenchStamped*

message type. Differently, the Teensy published its data on a different topic called *fma_data* by using a custom message type called *FMAdata* structured as follow:

```
#FMAdata.msg

uint64 timeStamp     #reading time
uint32 cnt1          #value from sensor 1
uint32 cnt2          #value from sensor 2
uint32 cnt3          #value from sensor 3
uint32 cnt4          #value from sensor 4
```

To start the ROS core and the nodes in charge of publish the measured data a launch file has been written:

```
#start_sim.launch
<launch>

    <node pkg="rosserial_python" type="serial_node.py" name="
    teensy_bridge" respawn="false" args = "/dev/ttyACM0"/>

    <node pkg="netft_utils" type="netft_node" name="netft" respawn="
    true" args = "--address 192.168.200.11 --rate 200"/>

</launch>
```

The published data have been collected and saved in a *.bag* file through the ROS command:

```
rosbag record -o test /fma_data /netft_data
```

## B.2 Matlab

MATLAB (Matrix Laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages. Thanks to Matlab it has been possible to analyze and compare both the force measurements of the FMA MicroForce sensors and ATI sensor. To read the data collected in a *.bag* file, the ROS toolbox has been used:

```
bagSelect = rosbag(file_path);
%select 'fma_data' topic
fma_data = select(bagSelect,'Topic','/fma_data');
%select 'netft_data' topic
netft_data = select(bagSelect,'Topic','/netft_data');
%save topics' messages in some structs
msgStructs_fma = readMessages(fma_data,'DataFormat','struct');
msgStructs_netft = readMessages(netft_data,'DataFormat','struct');
%save the forces measured by the ATI in a vector
netft_z = cellfun(@(m) -double(m.Wrench.Force.Z),msgStructs_netft);
netft_y = cellfun(@(m) -double(m.Wrench.Force.Y),msgStructs_netft);
netft_x = cellfun(@(m) -double(m.Wrench.Force.X),msgStructs_netft);
force_netft = vecnorm([netft_x,netft_y,netft_z],2,2);
%save the forces measured by the 4 sensors in a matrix
cnt_fma = zeros(fma_data.NumMessages,4);
time_fma = cellfun(@(m) uint64(m.TimeStamp),msgStructs_fma);
time_fma = double(time_fma - time_fma(1))./10^3;
cnt_fma(:,1) = cellfun(@(m) double(m.Cnt1),msgStructs_fma);
cnt_fma(:,2) = cellfun(@(m) double(m.Cnt2),msgStructs_fma);
cnt_fma(:,3) = cellfun(@(m) double(m.Cnt3),msgStructs_fma);
cnt_fma(:,4) = cellfun(@(m) double(m.Cnt4),msgStructs_fma);
```

```
% MATLAB code to solve the optimization problem shown in the
    calibration paragraph

% A - sensors measurements matrix
% y - ATI measurements vector
eps=0.88; %calibration parameter

N = length(A(:,1)); %number of measurements

C = zeros(2*4*N,4);
b = zeros(2*4*N,1);

for i=0:N-1
    C(i*4+1:(i*4)+4,:) = diag(A(i+1,:));
    b(i*4+1:(i*4)+4) = ones(4,1)*y(i+1)/4 + ones(4,1).*eps;
end
k=1;
for i=N:2*N-1
    C(i*4+1:(i*4)+4,:) = diag(-A(k,:));
    b(i*4+1:(i*4)+4) =  ones(4,1).*eps - ones(4,1)*y(k)/4;
    k=k+1;
end

options = optimoptions('lsqlin','Algorithm','interior-point');
x = lsqlin(A,y,C,b,[],[],-Inf,Inf,[],options)
```

## B.3 Fusion 360

Fusion 360 is a cloud-based 3D modeling, CAD, CAM, and PCB software platform for product design and manufacturing. The latter was used to design, and then print by using a 3D printer, the ABS molds used during the haptic thimble realization process.

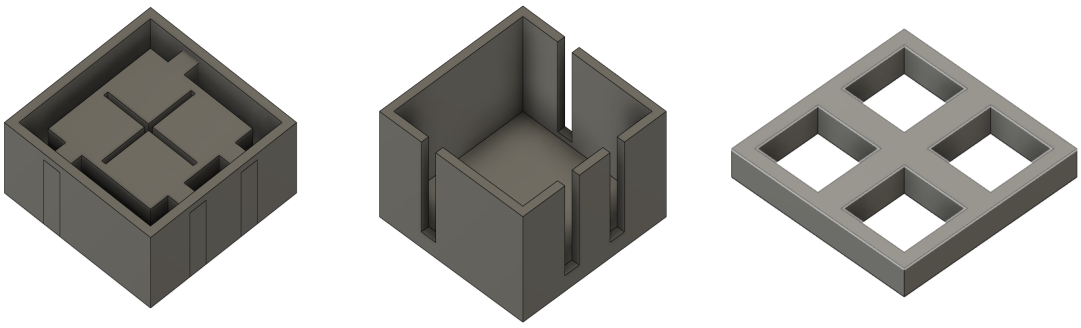


**Figure B.1:** Molds design in Fusion 360.

## B.4 Video-Call software

The Video-Call application described in Chapter 4 has been developed by using the Python version 3.9. As IDE, *PyCharm* version 2021.3 has been used. In order acquire the users' video, audio and haptic data, some external Python libraries have been exploited:

- OpenCV v4.5.3 [18]: to capture the video stream from the users' webcam and convert it in a JPEG code in order to be sent through a socket.
- pyaudio v0.2.11 [19] to capture the audio stream from the users' microphone and reproduce the incoming audio through their speakers.
- pyserial v3.4 [20] to read the haptic data through a serial port on the patient side and write them in the same way at the doctor side.

The application is based on four *.py* files:

- *consultation.py*, which contains all the classes and methods to manage the vdeo-call and the haptic data TCP/IP stream.
- *hapticData.py*, which contains all the classes and methods to:
  - read the haptic data sent by the thimble on a serial port when the application is running on the patient side.
  - write the haptic data, received from the patient, on the serial port where is connected a haptic feedback device when the application is running on the doctor side.
- *Patient.py*, which contains the *main* function to start the application on the patient side.
- *Doctor.py*, which contains the *main* function to start the application on the doctor side.

The communication between the patient and the doctor is based on TCP/IP sockets, where a server retrieves and sends the data, while a client, connected to the server, receives and shows them. To make the data communication asynchronous, a client and a server, running on independent threads, have been set-up for every data type. In what follow a brief explanation of the main application's functionality will be given, by showing the main code chunks. A server waits for a new connection from a client, after that it will start to send the avaiable data through the socket:

```
#...
#bind socket
self.ServerVideoSocket.bind((self.hostIP, port))
print("Server video: Socket Bind Successfully")
#wait for a connection
self.ServerVideoSocket.listen(5)
print("Server video LISTENING AT:", (self.hostIP, port))
try:
    while not end:
        client_socket, addr = self.ServerVideoSocket.accept()
        print('Server video GOT CONNECTION FROM:', addr)
        if client_socket:
            while true:
                #... get the data ...
                data = (code, datetime.now(timezone.utc))
                #serialize the data and create a message with their size
                a = pickle.dumps(data)
                message = struct.pack("Q", len(a)) + a
                #send data through the socket
                client_socket.sendall(message)
#...
```

On the other side, the client that sent the request for the new connection will start to receive the data:

```
#...
print("Client Video: Socket Accepted")

data = b""
payload_size = struct.calcsize("Q")
#...
try:
    while not end:
        #receive the data size from the server
        while len(data) < payload_size:
            packet = self.ClientVideoSocket.recv(2160)
            if not packet: break
            data += packet
        packed_msg_size = data[:payload_size]
        data = data[payload_size:]
        msg_size = struct.unpack("Q", packed_msg_size)[0]
        #receive the data payload
        while len(data) < msg_size:
            data += self.ClientVideoSocket.recv(2160)
        frame_data = data[:msg_size]
        data = data[msg_size:]
        #deserialize the data in their original form
        data, timeStamp = pickle.loads(frame_data)

        #... manage the received data ...

#...
```

Now it will be shown how the different type data have been acquired before being sent through their socket and, after being received on the other side, how they have been managed.

**Video data - Client**

```
#... connect to the server and start to receive the other user's frames
frame_data = data[:msg_size]
data = data[msg_size:]
code, timeStamp = pickle.loads(frame_data)
#save the frame in a shared variable in order to be shown in a window
self.videoLock.acquire()
self.Pframe = cv2.imdecode(code, cv2.IMREAD_UNCHANGED)
self.h, self.w = self.Pframe.shape[:2]
self.videoLock.release()
#...
```

### Video data - Server

```
#open the video stream
vid = cv2.VideoCapture(0)

#... wait for a connection ...

if client_socket:
    start = time.time()
    while vid.isOpened():
        #get a frame from the camera
        img, frame = vid.read()

        #send only 30 FSP
        if time.time() - start > 1./FPS:
            start = time.time()

            #flip the frame
            frame = cv2.flip(frame, 1)

            #get the JPEG encode
            ret, code = cv2.imencode('.JPEG', frame,
            ↪  [int(cv2.IMWRITE_JPEG_QUALITY), qual])
            data = (code, datetime.now(timezone.utc))

            #... send the JPEG encode through the socket ...

            #show the other user's image in a window, with the just
            ↪  read frame in the upper right corner
            if self.Pframe is not None:
```

```
27                  cv2.rectangle(frame, (0, 0), (frame.shape[1],
                    ↪  frame.shape[0]), (0, 255, 0), 3)
28                  frame = cv2.resize(frame, (int(0.25 * self.w), int(0.25
                    ↪  * self.h)), cv2.INTER_AREA)
29                  self.videoLock.acquire()
30                  self.Pframe[5:5 + frame.shape[0], int(0.75 *
                    ↪  self.w):int(0.75 * self.w) + frame.shape[1],
31                  :] = frame
32                  cv2.imshow("RECEIVING VIDEO", self.Pframe)
33                  self.videoLock.release()
34  #...
```

### Audio data - Server

```
1  def server_audio(self, port=5000, chunk=6024, FORMAT=pyaudio.paInt16,
   ↪  CHANNELS=1, RATE=44100):
2      #open the audio stream
3      p = pyaudio.PyAudio()
4      stream = p.open(format=FORMAT, channels=CHANNELS, rate=RATE,
       ↪  input=True, frames_per_buffer=chunk)
5      #... wait for a connection ...
6          if client_socket:
7              while True:
8                  #read a number of audio samples equal to 'chuncks'
9                  audio = stream.read(chunk)
10                 if audio:
11                     data = (audio, datetime.now(timezone.utc))
12                     #... send the data through the socket ...
13 #...
```

### Audio data - Client

```
def client_audio(self, destPort, chunk=1024, FORMAT=pyaudio.paInt16,
    CHANNELS=1, RATE=44100):
    #open the audio stream
    p = pyaudio.PyAudio()
    stream = p.open(format=FORMAT, channels=CHANNELS, rate=RATE,
        output=True, frames_per_buffer=chunk)
    #... connect to the server and start to receive the other user's
        audio ...
        audio_data = data[:msg_size]
        data = data[msg_size:]
        audio, timeStamp = pickle.loads(audio_data)
        stream.write(audio)
#...
```

### Haptic data

In order to read/write the haptic data on/from the users' serial ports, the class *HapticSerial* has been created:

```
class HapticSerial:

    def __init__(self, ttyStr, baud):
        try:
            self.usb = serial.Serial(ttyStr, baud)
        except:
            self.usb=None
        self.port = ttyStr
        self.baud = baud

```

```
11      def readData(self):
12          try:
13              if self.usb is None:
14                  self.usb = serial.Serial(self.port, self.baud)
15              data = str(self.usb.readline())
16              data = data[2:-5]
17              data = data.split(' ')
18              return [int(i) for i in data]
19          except:
20              if self.usb is not None:
21                  self.close()
22                  self.usb = None
23              return None
24
25      def writeData(self, data):
26          try:
27              if self.usb is None:
28                  self.usb = serial.Serial(self.port, self.baud)
29              self.usb.write(data)
30          except:
31              if self.usb is not None:
32                  self.close()
33                  self.usb = None
34              print('Serial write error')
35  #...
```

This class will be used from the server and the client in charge of the haptic data management:

- Server

```
    #...wait for a connection ...
    if client_socket:
        buff = []
        while True:
            meas = None
            if len(buff) <= BUFF_SIZE:
                meas = self.thimble.readData()
                if meas:
                    if len(meas) == 5:
                        buff.append(meas)
            else:
                data = (buff, datetime.now(timezone.utc))
                #...send the haptic data through the socket...
                buff = []
    #...
```

- Client

```
    #... connect to the server and start to receive the hapt. data
    ↪  ...
    haptic_data = data[:msg_size]
    data = data[msg_size:]
    reading, timeStamp = pickle.loads(haptic_data)
    for k in reading:
        self.thimble.write(k)
```

# Bibliography


[1] E. Ray Dorsey and Eric J. Topol. «State of Telehealth». In: *New England Journal of Medicine* 375.2 (2016), pp. 154–161. URL: https://doi.org/10.1056/NEJMra1601705 (cit. on p. 5).

[2] S Joseph Sirintrapun and Ana Maria Lopez. «Telemedicine in cancer care». In: *American Society of Clinical Oncology Educational Book* 38 (2018), pp. 540–545 (cit. on p. 5).

[3] Dale A Lawrence. «Stability and transparency in bilateral teleoperation». In: *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*. IEEE. 1992, pp. 2649–2655 (cit. on p. 5).

[4] Deborah Lupton and Sarah Maslen. «Telemedicine and the senses: a review». In: *Sociology of Health & Illness* 39.8 (2017), pp. 1557–1571. DOI: https://doi.org/10.1111/1467-9566.12617. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-9566.12617 (cit. on p. 5).

[5] Simon Cocksedge, Bethan George, Sophie Renwick, and Carolyn A Chew-Graham. «Touch in primary care consultations: qualitative investigation of doctors' and patients' perceptions». In: *British Journal of General Practice* 63.609 (2013), e283–e290. ISSN: 0960-1643. DOI: 10.3399/bjgp13X665251. URL: https://bjgp.org/content/63/609/e283 (cit. on p. 5).

[6] Sebastian Ullrich and Torsten Kuhlen. «Haptic palpation for medical simulation in virtual environments». In: *IEEE Transactions on Visualization and Computer graphics* 18.4 (2012), pp. 617–625 (cit. on p. 6).

[7] Jungsik Kim, Bummo Ahn, Yeongjin Kim, and Jung Kim. «Inclusion detection with haptic-palpation system for medical telediagnosis». In: *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2009, pp. 4595–4598 (cit. on pp. 6, 7).

[8] Kevin Hernandez-Ossa, Arnaldo Leal Junior, Anselmo Frizera, Teodiano Bastos-Filho, Kim Adams, and Martin Ferguson-Pell. «Haptic Feedback for Remote Clinical Palpation Examination». In: (2019) (cit. on p. 7).

[9] Nicole D'Aurizio, Tommaso Lisini Baldi, Leonardo Franco, and Domenico Prattichizzo. *Remote Palpation for Home-Based Telemedicine: a Comparison of frameworks*. Not published yet (cit. on p. 8).

[10] Tommaso Lisini Baldi, Stefano Scheggi, Leonardo Meli, Mostafa Mohammadi, and Domenico Prattichizzo. «GESTO: A glove for enhanced sensing and touching based on inertial and magnetic sensors for hand tracking and cutaneous feedback». In: *IEEE Transactions on Human-Machine Systems* 47.6 (2017), pp. 1066–1076 (cit. on pp. 9, 40).

[11] Jelizaveta Konstantinova, Giuseppe Cotugno, Prokar Dasgupta, Kaspar Althoefer, and Thrishantha Nanayakkara. «Palpation force modulation strategies to identify hard regions in soft tissue organs». In: *PloS one* 12.2 (2017), e0171706 (cit. on p. 11).

[12] *MicroForce FMA Series*. `https://sps.honeywell.com/us/en/products/sensing-and-iot/sensors/force-sensors/microforce-fma-series` (cit. on p. 11).

[13] Pietro Cerveri, Mauro Quinzi, Dario Bovio, and Carlo Albino Frigo. «A novel wearable apparatus to measure fingertip forces in manipulation tasks based on MEMS barometric sensors». In: *IEEE transactions on haptics* 10.3 (2016), pp. 317–324 (cit. on p. 14).

[14] Francesco Chinello, Claudio Pacchierotti, Monica Malvezzi, and Domenico Prattichizzo. «A three revolute-revolute-spherical wearable fingertip cutaneous device for stiffness rendering». In: *IEEE Transactions on Haptics* 11.1 (2017), pp. 39–50 (cit. on pp. 44, 46).

[15] *Robotic Operating System.* `https://www.ros.org/` (cit. on p. 55).

[16] *netft_utils, C++ class and ROS node for ATI force/torque sensors.* `https://github.com/UTNuclearRoboticsPublic/netft_utils` (cit. on p. 55).

[17] *rosserial_arduino, ROS support for Arduino compatible boards.* `https://github.com/ros-drivers/rosserial` (cit. on p. 55).

[18] *OpenCV, Open Source Computer Vision library.* `https://opencv.org/about/` (cit. on p. 60).

[19] *PyAudio, Python Bindings for PortAudio.* `http://people.csail.mit.edu/hubert/pyaudio/` (cit. on p. 60).

[20] *pyserial, Python Serial Port Extension.* `https://github.com/pyserial/pyserial` (cit. on p. 60).