Digital Interface for Upper Limb Amputees

António Figueira, *ist1102448,* João Freitas, *ist1103218,* and Guilherme Costa-Ferreira, *ist1103394*

I. Introduction

IN 2022, in the United States of America (USA) around 2 million Americans suffered from some form of limb loss, whether due to disease (diabetes mellitus, for example) or due to an accident or war. These figures are expected to rise to 3.6 million by 2050.

The intersection between assistive technology and improving digital accessibility for people with upper limb amputations has been a growing field of research. Although prostheses are often considered the most complete solution, they do not necessarily result in higher levels of satisfaction among individuals with amputated limbs. [7]. For this reason, assistive technologies continue to be developed.

In 2020, a study used a gyroscope and an electromyogram (EMG) to develop a computer mouse. The gyroscope, capable of measuring orientation and angular velocity on three axes, was used to measure the orientation of the hand and encode cursor control on the screen, while the EMG in the biceps and triceps muscles encoded the mouse's functionalities. In addition, the device's performance was improved with the use of *auto-thresholding* algorithms and [5] muscle contraction detection.

In 2022, an interface called EMKEY was designed to simulate computer mouse and keyboard functions through facial and voice recognition, replacing the use of hands. This interface is divided into two modules: one responsible for translating facial movements into cursor movement, and the second module dedicated to recognizing voice commands and transcribing them into text [6].

The assistive technologies mentioned are part of the field of the person-machine interface, which is becoming increasingly accessible to people with amputations [2].

However, it is important to note that these technologies are not yet advanced enough to be used reliably in everyday life. Therefore, the development of devices with more specialized functionalities is essential.

II. Initial Concept

Originally, we proposed designing a digital interface using an EMG sensor and an accelerometer, taking advantage of the movement of the residual limb in the different axes (and respective directions) that make it up and muscle contraction to translate into effective mouse control actions. Depending on the action, it would be possible to manipulate the different functionalities in the digital interface (movement, clicking and menu creation, for example). The user's choice of upper limb to operate the device would lead to the use of a sock that would incorporate the accelerometer (ensuring its stabilization in the distal area of the arm) and EMG sensors would be implanted along the residual limb.

III. Proof of Concept

Before proceeding to explain the approach taken, it should be noted that, over time, it was necessary to adapt the work plan to be followed, to make the project feasible depending on the resources available. In any case, all the changes are described in this report and continue to focus on the initial objective: restoring the *workflow* and importance in the workplace of upper limb amputees.

Thus, the final prototype shown in figure 1 was developed.

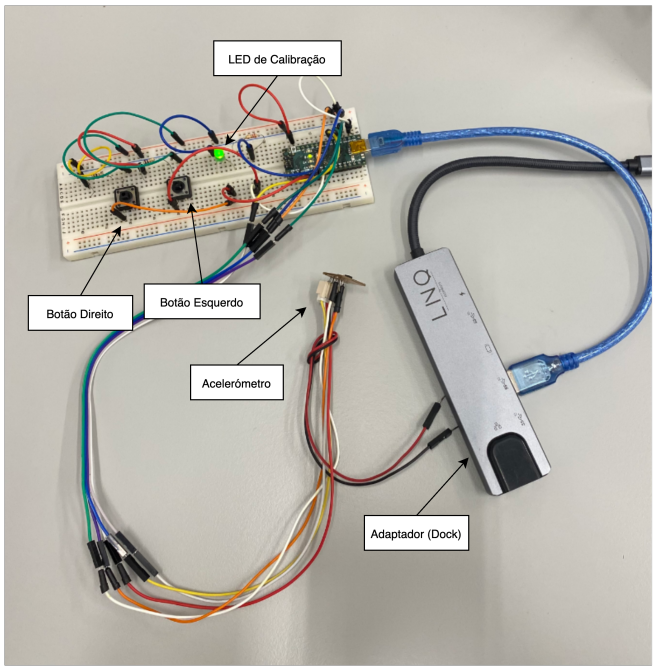

Fig. 1. Captioned image of the prototype

We felt that it wouldn't be necessary to use the EMG approach. The essential functions of an ordinary computer mouse are only provided by the use of the accelerometer (movements) and buttons (clicks). Another substantial change to the initial proposal is the use of the sock, which we didn't do because, contextualizing it in the working environment of the *breadboard* and the added components (wires and buttons, for example), wouldn't be practical or, in a way, feasible to integrate this entire device at the end of the residual limb. We therefore created a *proof of concept* for the solution presented.

To indicate to the user that the prototype is working, the green LED is on and, during calibration, switches to a flashing state.

To conclude this section, it is important to recognize the problem formulation of this specific case. By using an accelerometer, the data obtained from it - after converting the voltage to discrete levels (between 0 and 1023) in the Arduino Nano's Analog to Digital Converter (ADC) - can be expressed as a vector at each instant n:

$$x(n) = [x_0, x_1, x_2]^{\mathrm{T}} \tag{1}$$

where each vector's component corresponds to the accelerations in the 3 main axes (x, y and z), respectively.

IV. Circuit

To visualize the structure of the circuit, we present a ThinkerCAD image that simulates the physical version of the *breadboard* along with the various components. This representation is shown in figure 2.

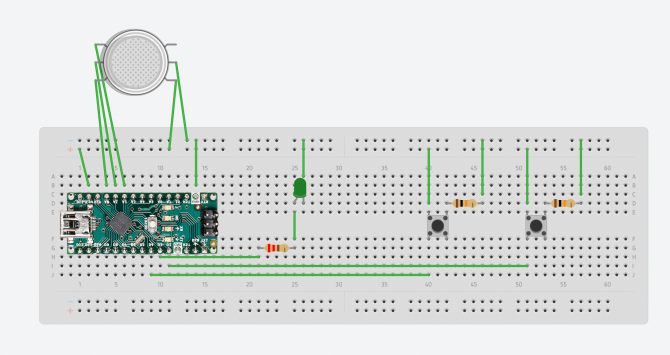

Fig. 2. Circuit diagram

Due to the limitations of ThinkerCAD, we used a smoke sensor as an analogue sensor (it has the same number of inputs/outputs).

V. Process

It's important to note that, to have a common thread to guide the group's thinking, we based our steps on the lectures provided by the teaching staff, both theoretical and laboratory, without forgetting the key methods for correctly acquiring, processing and analyzing information.

Indeed, when we investigated how the accelerometer works, we realized that the best way to use this device, taking into account the functionalities of the *mouse*, is to vary the accelerations associated with the movement in its different components, i.e. in the directions of the 3 main axes x, y and z - the other approach to using the accelerometer consists of acquiring variations relating exclusively to the z-axis (*default* approach). In this way, the mechanism described resembles that of a *joystick*, whereby tilting the accelerometer forwards or backwards (x-axis) is associated with an upward or downward movement on the digital interface, respectively. Similarly, tilts to the left or right (y-axis) correspond to the same movement on the screen in the same directions.

The next point to address is the design of the code associated with the Arduino IDE and Python.

Concerning the Arduino IDE code, there are several functions which, in a nutshell, are responsible for collecting: the values of the signal obtained by the accelerometer and its time stamp, but also information about the buttons and their activity. This data is sent to Python. In the *loop* there is also transmission of *data* in the reverse direction (Python - Arduino IDE) in which, if the accelerometer is in calibration, Arduino receives information to change the status of the LED on the *breadboard*.

The extraction of data from the accelerometer and the consequent processing and study of the signal were developed in Python, as was the interaction with the *mouse*. We can therefore divide this section into two groups: execution of the action and study of the signal - the latter allowing the former to be improved throughout the project using different parameters and methods.

A. Signal Processing

By studying the signal, we were able to adapt our approach to the problem. To do this, we grouped all its functions into a `loadData` function which, from the outset of its creation, allowed us to conclude our signal by drawing up a PSD (Power Spectral Density) graph, which represents the proportion of each frequency's contribution to the signal's total power. In this case study, the movements made are characterized by low frequencies (less than 1Hz), thus encouraging the use of a low-pass filter (with a cut-off frequency equal to a value between 1 and 5Hz). As can be seen in the following figure, filtering helps to smooth out the signal and obtain a result free of noise and undesirable artefacts (figure 3).

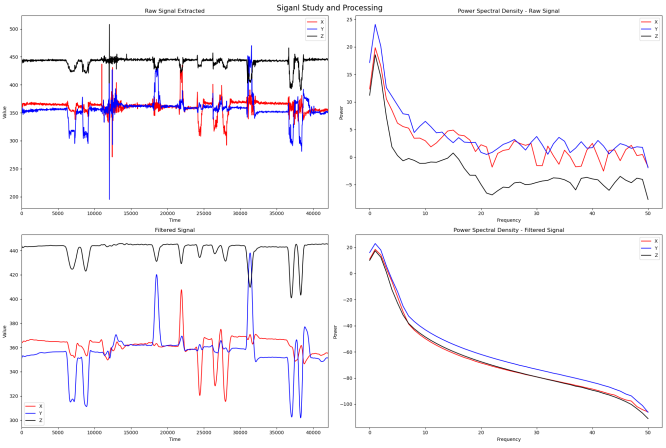

Fig. 3. Low-pass results with a cut-off frequency of 1Hz

As the signal is obtained by acceleration variations, the units commonly used are m/s^2 or g's. As the signal sent by the Arduino is ADC, by performing a calibration exercise (slow rotation on the z-axis), we can convert the units.

$$\mathrm{ACC(g)} = \frac{ADC - C_{min}}{C_{max} - C_{min}} \times 2 - 1 \tag{2}$$

Using the equation 2 and substituting ADC for the signal values, C_{max} and C_{min} for the extreme values, we obtained the results in the figure 4.

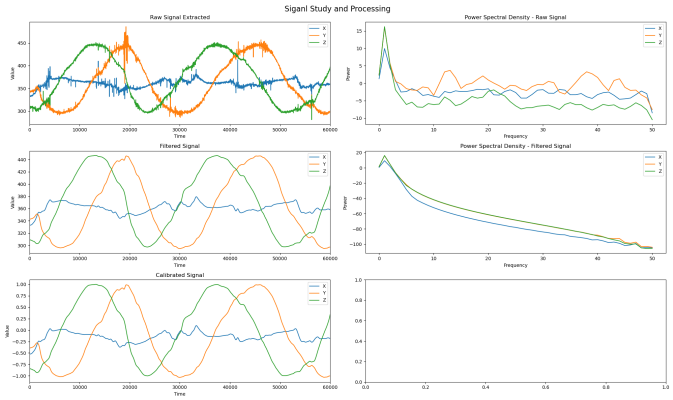

Fig. 4. Calibration exercise

Through this approach, we realized that the best way to discretize the signal would be through a movement *joystick* where each action would have a clear and distinct reading. This was defined after analyzing the data in the figure 5. Thus, it would be necessary to add *thresholds* and, consequently, *baselines* (obtained through a calibration period) to our system.

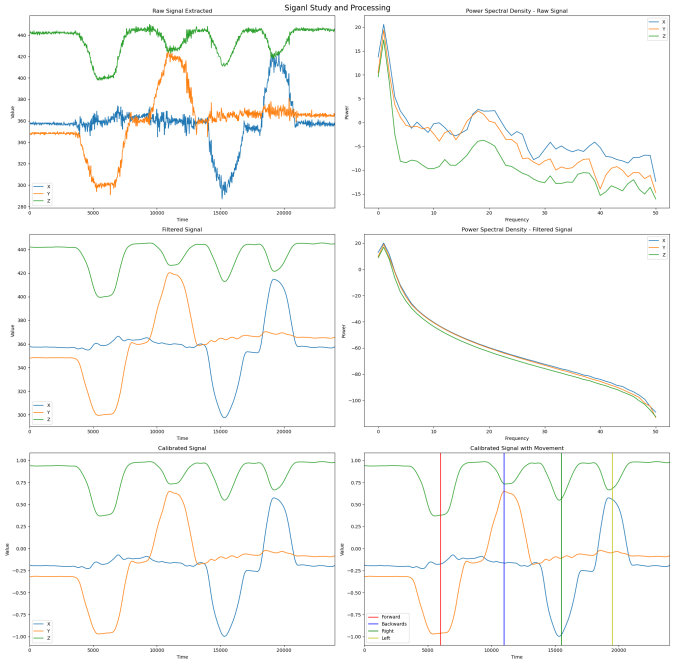

Fig. 5. Movement test analysis

In the first iterations of signal acquisition, there was an accumulative *delay* in signal propagation, visible in the prolonged movement of the *mouse*. This phenomenon is due to the time that the operating systems need to receive and read the information that is communicated between them, which, even though it is very short, accumulates over the many iterations of this process [9]. To solve this problem holistically, the sampling frequency was lowered (from 100Hz to 10Hz) and, consequently, the amount of information reaching the computer in a given time interval was reduced, reducing the magnitude of this problem. This change did not compromise our signal, respecting Nyquist's law - the sampling frequency must be at least twice the value of the maximum frequency of the analogue signal (as mentioned above, less than 1Hz) [8]. This way, we only get results where the graph is less "dense". As shown in figure 6.

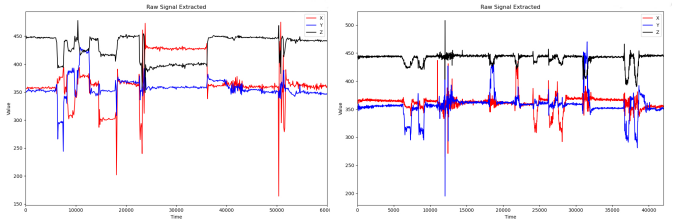

Fig. 6. Comparison between a 10Hz *test run* (left) and a 100Hz *test run* (right)

B. Signal Execution

The execution of the action is based exclusively on the `extractData` function. As its name suggests, it is responsible for obtaining data from the Arduino and translating it into movement and/or clicking the *mouse*.

As concluded in the previous phase, we need to use *thresholds* (to ensure that movement only occurs when desired) and, as such, we need a *baseline* to define them. Thus, during this translation process, there is an initial calibration stage which, during the first 5 seconds of the prototype's action, calculates the respective average values (in other words, the *baselines*) for each axis (x, y and z).

It should be noted that the direction of the movements can be inferred from the oscillations of the x and y values and that, regardless of the direction, a decrease in the z values occurs.

The *baselines* will allow the calculation of the *thresholds* that will be essential in controlling the digital movement according to the physical movement. A *baseline* will result in two *thresholds* which, when the signal is recorded, will prompt the action of the *mouse* if these thresholds are crossed in two of the axes (one of the directional axes and z). This ensures that accidental movements such as repositioning the arm to a more suitable/comfortable position for the user (a situation in which only one of the axes registers a change in value) are considered by the system to be redundant and not translatable into a digital effect.

As new calibrations may be required, the upper *threshold* of z (hitherto unused) is used as input for a new calibration to take place.

As mentioned above, in developing our project, we took into account the approaches presented in the Fundamentals of Bioinstrumentation classes. When defining our filtering with a low-pass filter, we had to implement it to work in real-time. It was therefore adapted to a filter called *Moving Average*. It is characterized by being a temporal low-pass filter [8] and, as its name suggests, it averages a certain set of values while the system is receiving the signal, so it is a non-causal filter - it only depends on the present and past inputs. This explanation can be translated into the expression 3 [1]:

$$y_i[n] = \frac{1}{N}\sum_{i=0}^{N-1}(x_i[n-i]) \tag{3}$$

Where N is the size of the sample window used. Note that the weight associated (1/N) with each input is the same for all the elements in a given sample. In addition, it is noticeable that the smaller the window, the smoother the final result becomes, as the large variations in the original signal are smoothed out. On the other hand, the larger the window, the trendier the signal becomes, as more points are incorporated into each average made [1]. This equation will be applied to the successive vectors described in the equation 1.

To convert the sample size into a cut-off frequency value, we use the equation 4 [3], which, using the code in [3], allowed us to conclude that a size of 3 approximates the *Moving Average* of a signal filtered with a 1Hz low-pass.

$$\sin^2\left(\frac{f_c N}{2}\right) - \frac{N^2}{2}\sin^2\left(\frac{f_c}{2}\right) = 0 \tag{4}$$

Thus, our signal is ready to use, typically having a *test run* as represented in the graphs in 7.

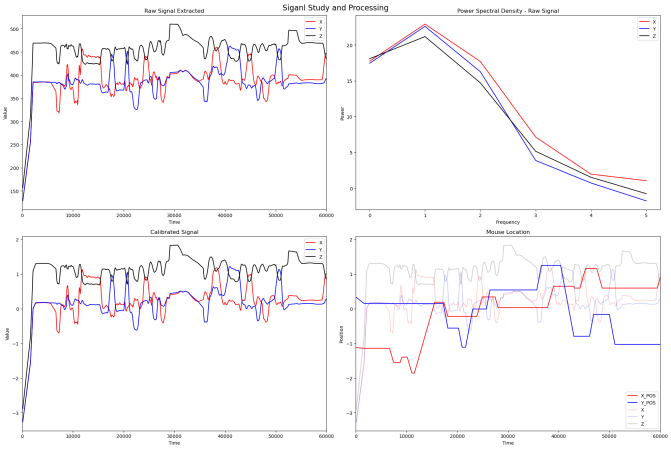

Fig. 7. *Test Run* of 1 minute with the new parameters

VI. Improvements and Future Challenges

Following on from the adaptation mentioned at the beginning of this report, a future improvement would be to incorporate the prototype into the glove which, by attaching to the end of the residual limb, would allow the *mouse* to be handled reliably to the user's movement. In terms of software, although it is not essential to achieving the main objective, code could be implemented to allow the *mouse* to be moved and clicked simultaneously, unlocking the drag function. Another feature that could be introduced is the "*scroll*" action that can be performed on a standard computer mouse. Finally, we describe what would be a challenge related to the acceleration principle itself. If the prototype were tested in motion (on a train, for example), accelerations external to the use of the accelerometer would influence the values recorded.

VII. Conclusion

For people without disabilities, technology makes things easier. For people with disabilities, technology makes things possible [4].

It was with this quote that we began this journey and which also helps us to complete a prototype that ensures the proposed objective: to restore the *workflow* of amputees with part of their upper limbs, making it possible to move the mouse and click on a digital interface.

References

[1] Yan Chen, Dan Li, Yanhai Li, Xiaoyuan Ma, and Jianming Wei. Use moving average filter to reduce noises in wearable ppg during continuous monitoring. In *Use Moving Average Filter to Reduce Noises in Wearable PPG During Continuous Monitoring*, volume 181, pages 193–203, 06 2016.

[2] Daniele Esposito, Jessica Centracchio, Emilio Andreozzi, Gaetano D. Gargiulo, Ganesh R. Naik, and Paolo Bifulco. Biosignal-based human–machine interfaces for assistance and rehabilitation: A survey. *Sensors*, 21(20), 2021.

[3] Geoffrey Hunter.

[4] Xueliang Huo. Introduction and preliminary evaluation of the tongue drive system: Wireless tongue-operated assistive technology for people with little or no upper-limb function. *The Journal of Rehabilitation Research and Development*, 45(6):921–930, Dec 2008.

[5] Md. Rokib Raihan, Abdullah Bin Shams, and Mohiuddin Ahmad. Wearable multifunctional computer mouse based on emg and gyro for amputees. In *2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT)*, pages 129–134, 2020.

[6] Pablo Ramos, Mireya Zapata, Kevin Valencia, Vanessa Vargas, and Carlos Ramos-Galarza. Low-cost human–machine interface for computer control with facial landmark detection and voice commands. *Sensors*, 22(23), 2022.

[7] Julie Rekant, Lee E. Fisher, Michael L. Boninger, Robert A. Gaunt, and Jennifer L. Collinger. Amputee, clinician, and regulator perspectives on current and prospective upper extremity prosthetic technologies. *Assistive Technology*, page 1–13, Feb 2022.

[8] Andrew G Webb and Cambridge University Press. *Principles of Biomedical Instrumentation*. Cambridge University Press, Cambridge, Uk ; New York ; Melbourne ; Delhi ; Singapore, 2019.

[9] Jinlong Xing, Gongliu Yang, and Tijing Cai. Modeling and calibration for dithering of mdrlg and time-delay of accelerometer in sins. *Sensors*, 22(1), 2022.