university of groningen / faculty of science and engineering / biomedical engineering

Development of an intention detection system for the EduExo Pro exoskeleton

Juan Fernández-Martos Fernández

S4686004

Engineering and Technology Institute Groningen, Discrete Technology and Production Automation

15/04/2024 - 15/07/2024

Bachelor’s project

1st Examiner: Dr. Elisabeth Wilhelm, Assistant Professor Engineering and Technology Institute Groningen (ENTEG)

2nd Examiner: Dr. Ir. Charissa Roossien, Bio-inspired MEMS and Biomedical Devices

Abstract

Exoskeletons are tools used for movement amplification or medical rehabilitation. However, these exoskeletons try to understand what the user wants to do through an intent detection system. Most intention detection systems rely on residual muscle function, making them difficult to use for certain subgroups of patients who lack these abilities. The development of systems that do not rely on residual function would be beneficial for the users. For this purpose, an intention detection system was developed for a one-degree-of-freedom upper limb exoskeleton focused on users with muscle weakness. Two MPU6050 sensors were used to determine the position of the head relative to the body. Two Grove-EMG sensors were used to perform vertical and horizontal electrooculography to detect the movement of the user's eyes. The sensors were connected to two Arduino Uno R3 boards and combined via Python. The developed algorithm was tested on a user without vision problems and in a static position. Obtaining an average accuracy of 56% in the detection of eye movement directions. However, being unable to compute the head and shoulder movement in the yaw axis, a limitation that compromises an essential function of the system. Further research focused on improving yaw axis motion measurement, the blinking detection capacity, and the detection of more complex eye movement patterns, might enhance the user experience.

Table of contents

1. Introduction

Exoskeletons are crucial tools in both personal use and medical rehabilitation. They play a key role in restoring lost skills due to age or disabilities. Exoskeletons are used in a wide range of fields, such as neuromuscular impairment compensation (Carmeli et al., 2010), post-stroke rehabilitation (Kim & Deshpande, 2017), or force amplification in industrial environments (Nussbaum et al., 2018). These devices interact with the human body by amplifying force, assisting or replacing motor function (Gull et al., 2020). They generate a considerable amount of force to support, assist, or stop the patient's movements (Jarrassé et al., 2014). Most notably, these devices are designed to comprehend human movement intentions and execute them accordingly (Wang et al., 2023).

Exoskeletons are classified according to their use: those designed for movement amplification and those dedicated to medical rehabilitation. Exoskeletons intended for medical rehabilitation are generally mounted on a mobile or stationary platform that cannot be operated by the user alone or can only be used in controlled environments. They help patients suffering from amyotrophic lateral sclerosis or post-stroke paralysis. Exoskeletons dedicated to assisting movement help to reduce the load or help patients with muscle weakness who have difficulty with daily activities (Gull et al., 2020). An example of an exoskeleton designed to assist movement is the EksoVest exoskeleton. This device reduces the load carried by the wearer and the spinal load by approximately 10%, particularly during overhead tasks such as drilling (Nussbaum et al., 2018).

A crucial point when using an exoskeleton is how the user can initiate the desired movement. Intent can be obtained by assessing the interactions between the human and the exoskeleton. Human-robot interaction (HRI), which can be cognitive or physical, is how the exoskeleton interacts with the user and vice versa. Cognitive human-robot interaction-based systems (cHRI) often monitor the electrical signals the central nervous system uses to control the musculoskeletal system, using them as inputs to control the exoskeleton. The intention is identified before the movement occurs. The speed and/or force needed to assist the motion the user intends to carry out can be predicted by the algorithm. The motors of the exoskeleton can then provide the necessary torque to support the motion. On the other hand, physically human-robot interaction-based systems (pHRI) measure the force or change in position produced by the user's movement. The measured change in position or interaction forces is used as an input to the exoskeleton (Gull et al., 2020).

This study aims to develop an intention detection system for the EduExo (Auxivo AG, n.d.) exoskeleton that does not rely on residual muscle function of the arm. Implementation of the developed algorithm in the device's control structure is beyond this report's scope. For this, two Arduino One R3 boards will be connected with two MPU6050 and two Grove-EMG sensors to determine the position of the shoulders and head and to perform vertical and horizontal electrooculography (EOG), respectively. The information obtained by the sensors is sent to the computer by the boards through the USB-A ports. Once received, the data is processed with Python.

2. Problem analysis

2.1 Market research

Different sensors and systems are combined to detect the user's intent. Each exoskeleton uses its own combination. Some systems include tongue movement (Kim et al., 2013), speech (Simpson & Levine, 2002), or manual triggers (Gantenbein et al., 2022). The most common ways to detect intentions include the following strategies:

Electromyography (EMG) detects signals generated during muscle activation and is used in more than 40% of exoskeletons. It is simple to use and obtain and can be applied in various ways, including sleeves and bands. However, it is not particularly resistant to changes in electrode position, sweating, or muscle fatigue (Gantenbein et al., 2022). sEMG signals are stochastic signals affected by various noise sources. Including noise generated from the movement of the electrodes and their cables, the crosstalk signal generated by interference from other muscles, and electromagnetic interference from power lines and radio transmission. EMG signals are user-dependent and change with time (Hameed et al., 2019). If the patient has some kind of neurological disorder, there may be alterations in motor neuron properties, resulting in muscle spasticity, weakness, and contracture. Paretic muscles can produce involuntary motor activity, such as spurious background spikes, making reading these signals and implementing any algorithm much more complicated (Hameed et al., 2019).

Other systems include measuring mechanical muscle contraction by measuring changes in muscle stiffness patterns (FMG) or low-frequency vibrations of the muscle fibers (MMG). The operation and uses of force myography (FMG) are very similar to EMG. Both are simple to integrate but depend on the user's muscle health, muscle fatigue, and sensor placement. On the other hand, they are robust to moisture and resistant to electromagnetic noise (Gantenbein et al., 2022).

It is also possible to detect the intention through the rotation of the joints. However, this system requires sufficient residual muscle function to work. Sensors such as inertial measurement units (IMUs) are used to detect residual limb movement. Similarly, the intention can be detected by measuring the isometric forces generated by the user on the exoskeleton because the desired motion is "constrained" by the exoskeleton structure (Gantenbein et al., 2022).

Another intention detection technique that has advanced rapidly in recent years is Brain-computer interfaces (BCIs). This technique allows patients with zero muscle control, such as paralysis, to communicate and interact with the environment. However, 20% of the users are unable to use them because they cannot modulate their brain response correctly. BCIs require a very complex setup and extensive knowledge of the device. They also generate a large amount of fatigue since they need a lot of concentration, focus, and awareness to be able to respond to environmental stimuli (Mridha et al., 2021). Electroencephalography (EEG) is the least invasive version, using electrodes placed on the scalp to measure the electrical activity of the cerebral cortex. These systems have a limitation: only some things that are thought of are performed. This can create uncomfortable or dangerous situations for the user; a normally harmless thought, due to its non-execution, can be carried out by the device (Gantenbein et al., 2022). For example, during a fierce argument, the exoskeleton user has the idea of punching the other person. Usually, this thought is not expected to turn into action, but BCIs cannot distinguish between this thought and the decision to act, so they throw the punch directly (Rainey et al., 2020). When a vague intention activates a device, it may indicate that, from a neuroanatomical perspective, two signals were sufficiently similar to be interpreted as commands (Rainey et al., 2020). This makes it complex to apply in an exoskeleton as it must be able to differentiate between thoughts and actions. BCIs are usually used when there is no other option because the device can work as long as the brain is functioning. They allow the user to do simple tasks like typing or moving the computer cursor.

Other intent detection strategies use eye movements. Eye movements play a fundamental role in motion planning, obtaining information about the object and its surroundings. Most neurological deficits affecting the upper limbs do not affect the eyes. Making this a viable method for a large population. The difficulty of this approach is knowing how to differentiate between when the user is simply looking around and when they are looking to initiate movement. For this reason, eye movement systems are often used in combination with other intention detection systems (Gantenbein et al., 2022). There are different methods available for tracking the eye. Video-oculography measures the position of the eye through corneal reflection (Gantenbein et al., 2022). Among the problems with this method are the high cost of the devices, low temporal resolution, and possible reduction of the range of motion of the head (Bamiou & Luxon, 2005). Electrooculography (EOG) measures the potential difference between the cornea and the retina with electrodes placed around the eye (Gantenbein et al., 2022). In other fields, electrooculography is sometimes used as a cheaper alternative with higher time resolution (Barea et al., 2002).

2.2 Problem definition

The EduExo Pro (Auxivo AG, n.d.) exoskeleton by AUXIVO covers the shoulder and elbow with 2 degrees of freedom at the shoulder, allowing internal/external rotation and shoulder flexion/extension but blocking shoulder abduction/adduction and 1 degree of freedom in the elbow actively supporting forearm extension/flexion. However, supination/pronation is also possible (Auxvivo AG, 2022). The EduExo Pro exoskeleton features a force sensor in the lower arm, an angle sensor in the elbow joint, and an EMG sensor to measure muscle activity.

Image 1. EduExo Pro exoskeleton (Auxivo AG, n.d.)

Due to time constraints, the scope of this research is limited. Therefore, it has been decided to focus the research on the intention detection system and design a system that could be combined with the existing exoskeleton in the future. The design must transmit the commands corresponding to the detected intent but not execute the movements. The system is intended to be used in a static or nearly static position without sudden movement.

The objective is to find the combination of sensors that can be added to the existing exoskeleton to detect the user's intentions. For the purpose of this study, it is assumed that the user has limited upper arm and forearm muscle control, thus ruling out the use of the EMG provided. The use of force sensors is also ruled out for similar reasons. Suppose the user tries to pick up a heavy object. In that case, the sensor will detect the force generated by this weight, which the user's arm cannot compensate for due to muscle weakness, and move to accompany this force, reacting in the opposite direction to that required to lift the weight.

3. Design

3.1 Inertial measurement units (IMUs)

Normally, we make movements with the upper limbs where the head is not involved, movements where haptic perception is essential. These movements require great precision to modify the position millimetrically. The user does not have normal use of the arm and the

exoskeleton does not have the level of precision required when combined with the user's arm, so certain functions are limited, and different alternatives need to be explored to alleviate these problems.
For this purpose, it was decided that all movements require the head. The head must face the side where the exoskeleton is located to be able to use it, partly substituting the lack of touch-based precision with sight. This works as a safety measure to prevent unintentional use of the device when performing other actions, such as moving the opposite arm.
For this purpose, two IMUs (MPU6050) were placed on the back and head, as shown in Figure 2b, to know the head's relative position and where the user is looking. The IMU placed on the back will also be considered to determine if the user wants to start the movement. In case of a forward movement with the shoulder, the arm will be moved forward. The sensors were connected to an Arduino one R3 board following the schematic in Figure 2a. For the sensor on the back (Sensor 1), the 3.3V port is connected to the AD0 pin to change the write address of the sensor from 0x68 to 0x69, allowing the use of two sensors at the same time. Then, both sensors sensitivity were set to ±251 º/s and ±2g for the gyroscope and accelerometer, respectively.

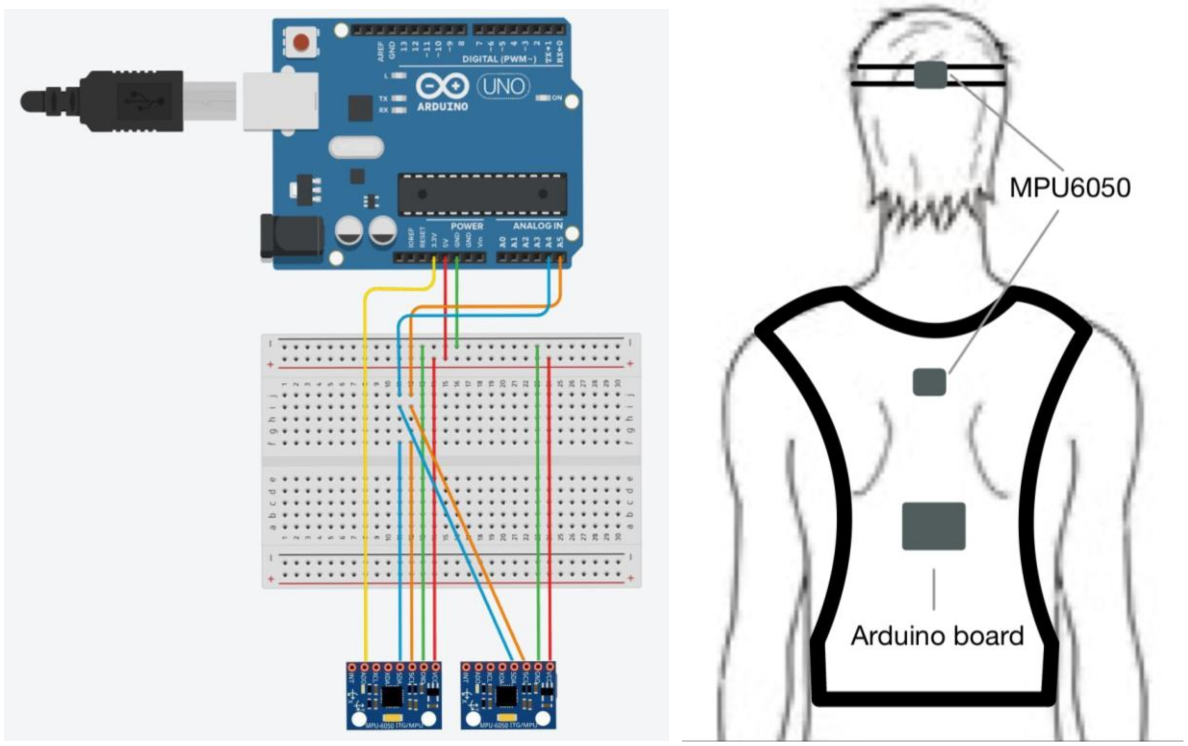

Figure 2: a) Arduino circuit adapted from (Autodesk, 2011) b) sensor placement.

The accelerometer uses the inertia of a mass, which accelerates slower than the surroundings. When the sensor is moved, this causes a small delay in the displacement of the bigger mass compared to the smaller mass. Most often, a capacitive sensor is used to detect this small displacement. Due to gravity, we constantly see 1 g in the axis that points towards the center of the earth. Through the measurements, it is possible to know the rotation in the axes X (roll) and Y (pitch), but not being able to know the rotation in the Z axis (yaw), because gravity is so large that small accelerations are lost in a quantization error. A representation of the angles in the human body is shown in Figure 3.

To complement the accelerometer, it is possible to use the gyroscope, which measures the angular acceleration on each axis. The angle can be obtained by integrating the angular velocity obtained from the gyroscope as a function of time. In order to set the initial values of the angle, the sensors must be calibrated before starting. The problem with the gyroscope is that if a small error increases the measurement constantly, the error will affect the angle measurement more and more.

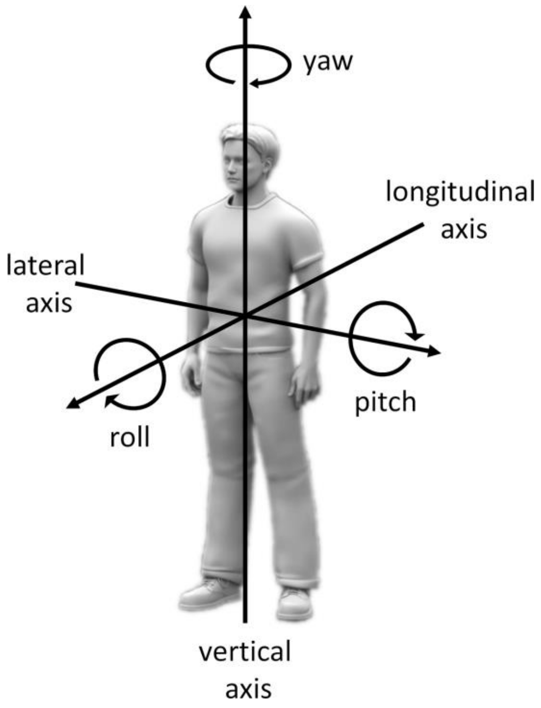

Figure 3. Roll pitch and yaw equivalents in the human body (Arnold et al., 2019).

Therefore, the most accurate way to know the angle through the MPU6050 is to combine the information from the gyroscope and the accelerometer. To calibrate the sensors, they were left still in the same position, and the values were measured 200 times in succession. The average of these values was then used as the offset error in each of the axes. The Arduino code can be found in Appendix 1.

Formulas 1 and 2 were used to calculate the angle with the accelerometer. The orientation of the device was calculated through a sequence of rotations.

$$(1) \quad ARoll = atan\frac{AccPitch}{\sqrt{AccRoll^2 + AccYaw^2}} * 180/\pi$$

$$(2)\quad APitch = atan\frac{-AccRoll}{\sqrt{AccPitch^2 + AccYaw^2}} * 180/\pi$$

To calculate the angle with the gyroscope, the value obtained is integrated over the time between measurements. This is added to the previous values, obtaining the current value (equations 3, 4, and 5).

$$(3)\, GRoll = GRoll + GyroRoll * ElapsedTime$$

$$(4)\, GPitch = GPitch + GyroPitch * ElapsedTime$$

$$(5)\, Yaw = Yaw + GyroYaw * ElapsedTime$$

Roll and pitch combine 96% weighting to the gyroscope and 4% weighting to the accelerometer with respect to the angle, as shown in equations 6 and 7.

$$(6)\, Roll = 0.96 * GRoll + 0.04 * ARoll$$

$$(7)\, Pitch = 0.96 * GPitch + 0.04 * APitch$$

A Kalman filter was used to solve the drift problem. This uses the accelerometer measurements, which are very noisy but not affected by drift, to predict the real gyroscope input (Franklin, 2020). This filter gives very accurate measurements but can only be applied on the axis where the accelerometer provides usable data.

As an alternative method, it was attempted to use the Digital Motion Processor (DMP) of the MPU6050 to obtain the characteristic quaternions of each angle. The DMP reduces the load that would usually be placed on the microprocessor by combining information from the gyroscope and accelerometer to obtain a quaternion that represents the orientation of the sensor (White, 2019). Quaternions are a more efficient way of expressing the rotation of a body than Euler angles because they do not have the gimbal lock problem, which can lead to a singularity that causes the robot to crash. Quaternions express any rotation or sequence of rigid body rotations around a fixed point as a single rotation of an angle θ around a particular axis (Nelli, 2020). The quaternion is then converted to yaw pitch roll angles. Once the data is converted, the results obtained are sent via serial communication with the computer. For all this, the MPU6050_6Axis_MotionApps12 library was used. The Arduino code can be found in Appendix 2.

In order to know the rotation of the head with respect to the body, the average of the last 10 measurements of the yaw angle of the body is subtracted from the average of the head angle. This way, the impact of sensor errors is reduced. If a value differs greatly from the previous one, it will not have a big effect. Only when the angle is maintained, the difference is seen. If the difference in angle is between 10 and 90 degrees, in the robotic arm quadrant, it is possible to start the movement.

While the head is in the correct quadrant, it is calculated if the body's position has changed rapidly. For this, the average of the last five values and the previous five values is calculated, and then the difference between them is calculated. If the difference is greater than 10 degrees, it is considered that the forward motion is fast enough to initiate the movement. The arm would then move forward.

3.2 Electrooculography (EOG)

Once the movement has been initiated, it is necessary to be able to direct the arm to the proper position. For this, the measurements collected by an EOG were used, with Skintact® Silver–Silver Chloride electrodes positioned as shown in Figure 4a. For this purpose, a similar setup to the one proposed by (Barea et al., 2002) to manage wheelchairs was used. This system allows the user to control the direction of the arm by looking at the different directions, using a double blink to stop the arm's movement. In addition, this system allows the user to choose the depth to which the arm reaches. Once the movement has started and the arm has been thrown forward, the double blink determines when it stops. If the target has not been reached, the user can repeat the initial trigger with the shoulder, and the arm will continue moving forward. This method creates a two-dimensional control with the eyes and a depth control with the shoulders.

Two Grove-EMG Detectors (Huang, 2023) were used to carry out the EOG. They were connected to an Arduino Uno R3 board following the schematics shown in Figure 4b. EMG, EKG, and EOG all measure biopotentials, the only differences being the strength and frequency of the signal. These biopotentials can be measured with the same sensor as long as the device is capable of measuring the amplitudes and frequencies characteristic of each signal (Thakor, 2014). The theoretical amplitude of the EOG ranges from 0.01 to 0.1 mV, although the measurements obtained are closer to a max of 0.35 mV. The maximum voltage of the Grove EMG sensor is 3.3V. So, it is suitable for an EOG.

Different studies point out that information is no longer relevant above 10 Hz (Thakor, 2014) (Banerjee et al., 2014). In order to improve the received signal, low-pass filters were used at 8.7 Hz (R = 470 Ω, C = 39 μF) and 10,3 Hz (R = 470 Ω, C = 33 μF) for the horizontal and vertical channels, respectively. The cutoff frequency was calculated with formula 8. A 10 Hz filter in both channels was not possible due to a lack of materials in the laboratory, so a solution as close as possible was sought. Although the filter of the vertical channel is below 10Hz, no problems have been encountered when measuring the signal.

$$(8)\ f_c = \frac{1}{2\pi RC}$$

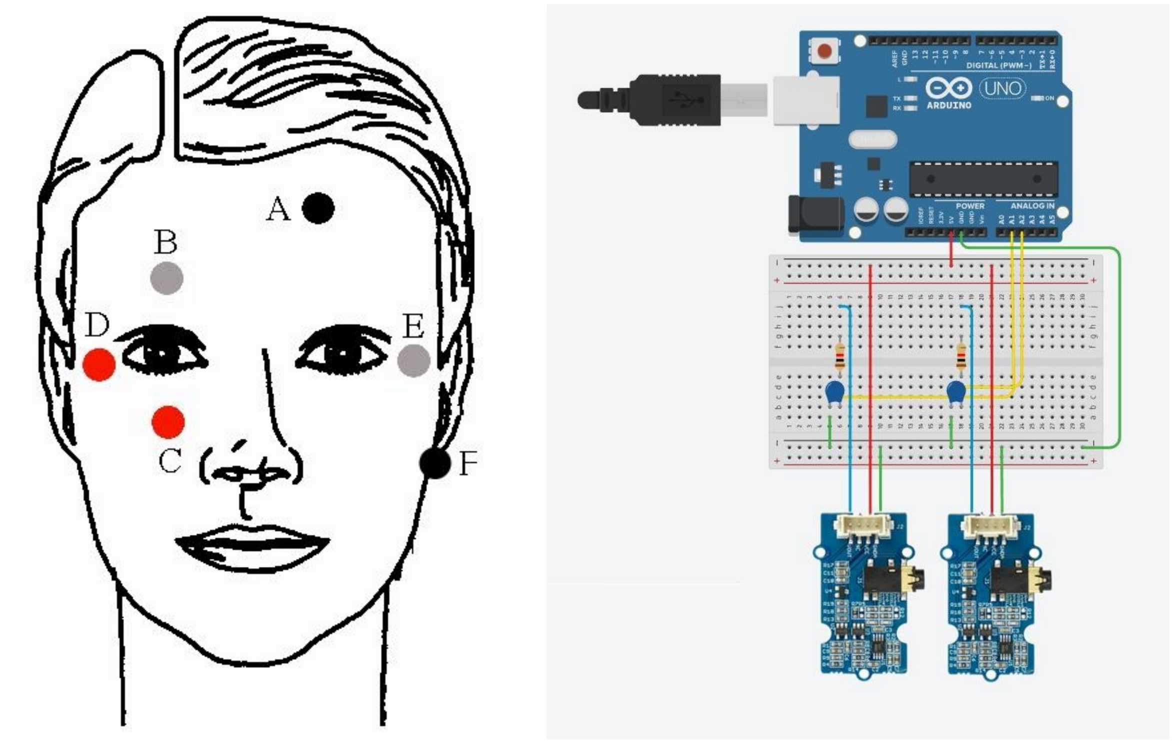

Figure 4. a) Electrode positioning (adapted from (Barea et al., 2002)) b) Arduino circuit (adapted from (Autodesk, 2011))

The PyFirmata library was used to read the information from the Grove sensors. It allows Python to control the Arduino board. Python version 3.9 was used because the PyFirmata library is incompatible with the latest version (3.12). In Python, the information provided by the sensors on the analog ports is read and stored by channels (Horizontal and Vertical). Then, the 150 values following the first 50, which are discarded because the sensors take time to start and return null values, are taken at a standstill position, looking at the center. These are then averaged and subtracted from all other values. In this way, the drift that the devices can accumulate each time the measurement is started is eliminated.

The objective is to be able to detect five different movements: up, down, right, left, and blinking. Thresholds are then used to differentiate the movements (Lv et al., 2009). Several tests were carried out to determine these thresholds. In these tests, it was found that looking right-left or up-down are opposite movements. Looking to the right or up causes first a negative and then a positive polarization. Looking to the left or down causes the opposite effect. Blinking has a similar effect as looking up but faster and more negative. To know if one of these events is occurring, a value 100 values prior to the last value taken is chosen (n), and it is checked if any values in the range of the following m meet the criteria of the initial peak for each movement, Table 1 and 2 show the ranges and thresholds. Then, it is checked if any of the following q values coincide with the next peak. In case one of the events is detected, the search for further events is stopped, and the command for the triggered event is sent. It is then that the search for the double-blink starts. The same procedure is followed, with the difference that finding two events in a row is necessary. When this is detected, the signal is sent, and the event search is reactivated. The Python code can be found in Appendix 3.

Each time the measurement is started, it is necessary to slightly modify the thresholds because the slightest differences in the electrode positions can change the measurement.

		Range of comprobation		Range of comprobation
Threshold order	Negative peak		Positive peak	
Right	n > - 0.020	[n: n + 75]	n > 0.03	[n + 25: n + 100]
Threshold order	Positive peak		Negative peak	
Left	0.025 > n > 0.010	[n: n + 25]	n < - 0.026	[n + 25: n + 50]

Table 1. Horizontal channel thresholds and range

		Range of comprobation		Range of comprobation
Threshold order	Negative peak		Positive peak	
Up	- 0.025 > n > - 0.04	[n: n + 75]	n > 0.025	[n + 25: n + 100]
Threshold order	Positive peak		Negative peak	
Down	0.05 > n > 0.01	[n: n + 25]	-0.04 < n < -0.016	[n + 25: n + 50]
Threshold order	Negative peak		Positive peak	
Blink	n < -0.030	[n: n + 25]	0.025 > n > 0.01	[n + 25: n + 75]

Table 2. Vertical channel thresholds and range

4. Results

4.1 Inertial Measurement Units (IMUs)

After calibrating the sensors, the following errors were obtained, which are subtracted from each of the values obtained. These values are calculated each time the measurement is started. The values in Table 3 show the errors calculated during one of the measurements.

Sensor	Accelerometer Error X	Accelerometer Error Y	Gyroscope Error X	Gyroscope Error Y	Gyroscope Error Z
1	0.56	-3.62	0.09	-5.37	1.05
2	0.40	-2.62	-1.17	1.44	0.96

Table 3. Error calculation

The calculated angles increased non-stop as if the sensor was spinning. When the Kalman filter was applied, the roll and pitch angles remained stable, but the yaw angle had the same problem, as it is not calculable through this filter.

First, the two sensors were placed in parallel, one in front of the other, so that the angle between them was 0º, as shown in Figure 5a. The program was started, and the difference between the two yaw angles was recorded. Then, the process was repeated with one sensor at 90 degrees to the other, as shown in Figure 5b. In both cases, the results obtained were totally random. The obtained difference kept jumping between -55 and 35 when the angle was 0º and between -45 and 25 when the angle was 90º. There was never a moment when the values remained stable. During the measurements, the sensors were as still as possible, but they were carried out on a table without any vibration isolation. Additionally, one of the sensors was rotated 360º to detect when the angle was between 10º and 90º. At no time did the sensors give values close to these angles.

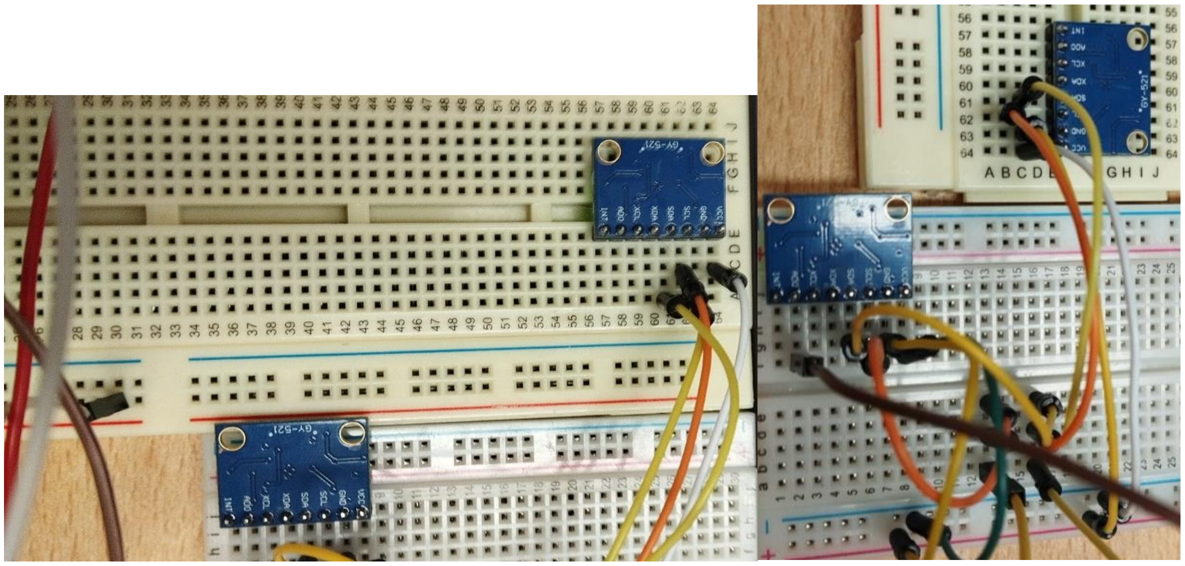

Figure 5. a) Sensor placement at 0º and b) 90º.

4.2 Electrooculography (EOG)

To set the thresholds, patterns were recorded on both channels. First, in the horizontal channel, the pattern right, right, left, left, and in the vertical channel, up, up, down, down, blink, blink, were repeated twice each. Then, the patterns were combined by looking right, left, up, down, blink. Each of the channels was analyzed, and the last patterns were represented, as shown in Figures 6 and 7.

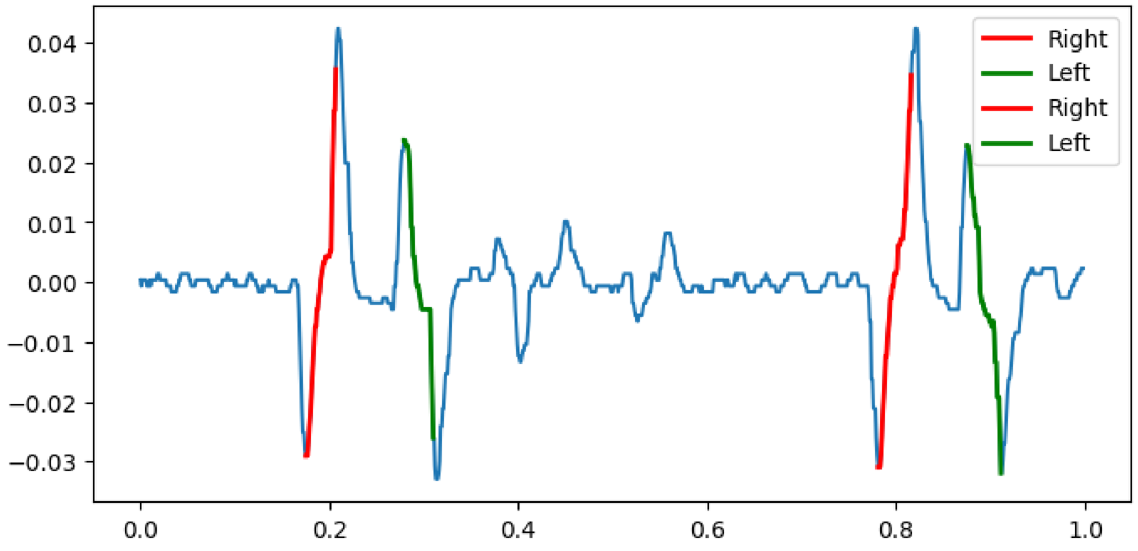

Figure 6. Horizontal channel readings + movement detection

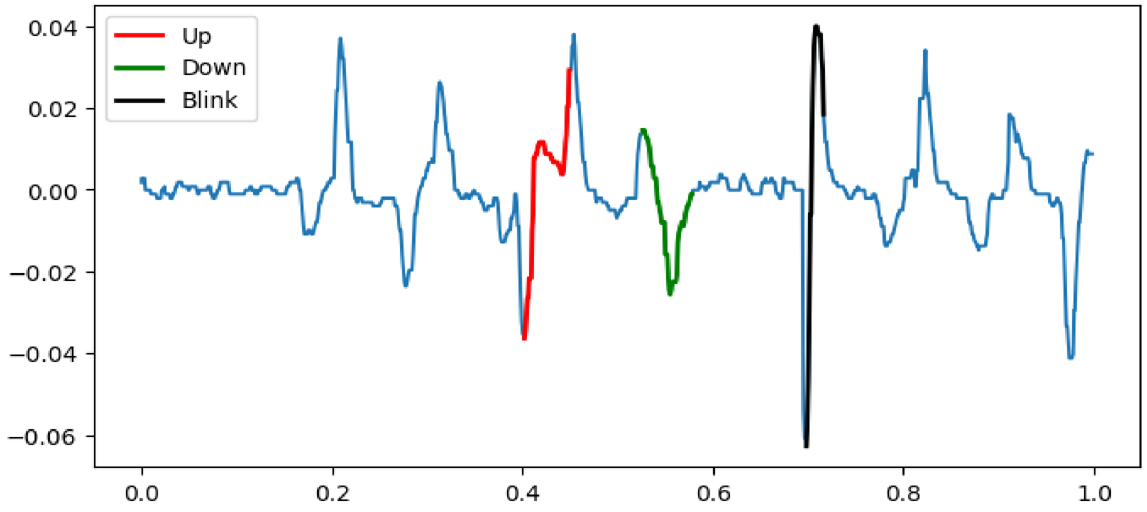

Figure 7. Vertical channel readings + movement detection

Once it was verified that it was able to detect the movements, each movement was repeated ten times and recorded whether the result was obtained on the first or second attempt to make the move. The result was recorded in Table 4. To be considered as movement, the user needs to look in the intended direction and immediately look back to the center.

Movement	First try	Second try	No response
Right	9	0	1
Left	7	2	1
Up	9	1	0
Down	2	5	3
Blink	2	0	8

Table 4. Movement detection results

5. Discussion

The MPU6050 sensors are not accurate at measuring the yaw angle. The results are neither accurate nor precise, which compromises this design's fundamental function. To solve this problem, the MPUs could be combined with a magnetometer or a sensor that includes it, obtaining 9 DOF and more accurate and precise long-term readings of all the angles. Other methods include using only the gyroscope for the yaw angle calculation, subtracting the average drift, assuming that the drift remains constant (Borowska-Terka & Strumiłło, 2023).

The EOG gives good results for right, left, and up movements. 90% of the movements to the right were correctly recorded, while only 70% of the movements to the left were recorded. This may be due to the fact that, as shown in Figure 7, these movements are weaker. This may be due to the right eye being dominant, meaning that the right eye is used more, has better vision, or fixes the view better. However, it is not able to detect the down and blink movements, as it only detects them 20% of the time. The objective was to use the blinking as a stopping mechanism. This low detection capacity makes its combination with the exoskeleton dangerous because if the movement is not stopped, it can harm the user.

On the other hand, all upward movements were detected, unlike the downward movements, which were hardly registered. This may be because upward movements are easier to make. After all the tests, eye pain was reported, especially when looking down. However, the studies suggest that there is a greater range of downward movement compared to upward movement, approximately 20º. Which could explain results contrary to those obtained (Lee et al., 2019). The more movement, the greater the potential difference. One possible explanation for these measurements is the positioning of the ground electrode, which, although standard, is placed high in the middle of the forehead. It may be that these differences are influenced by the proximity of the electrode to the vertical axis of the eyes.

The blink recognition failure may be due to the fact that not all blinks are the same. It is very complicated to always blink the same way. A blink usually lasts between 0.1 and 0.4 seconds (Chudler, n.d.). The force with which the eyes are closed also modifies the measurement. When you try to blink on purpose, it is more complicated to control the force because it is not automatic. It may be possible to regulate the response depending on the force. However, further research is needed to learn more about this relationship.

The system proposed by (Wu et al., 2013) has a 100% accuracy in 10 different movements, including up-down, right-left, and combinations of both channels, plus clockwise and counterclockwise eye movements. This system uses the slope of the maximum and minimum voltage peaks to differentiate the movements. They show that the movements are sufficiently different from each other to be able to always identify them. However, for its use, the user needs to stand 1.05 meters away from the wall and follow the target paths, so it still needs to be refined to be used in more specific contexts, such as virtual keyboards or the control of a wheelchair.

Two different approaches are proposed to improve the EOG. Improving the received signal or improving the signal processing. To improve the received signals, the first thing that can be done is to incorporate at least an amplifier between each sensor and the Arduino board. It is also possible to include more filters to improve the signal quality. Currently, only a low-pass filter is used. Among the options could be to increase the order of this filter to make it more accurate. Incorporating a high-pass filter near 0.15 Hz to decrease the signal noise could also be an option.

From the signal processing point of view, using variable thresholds to convert the signal to high or low can help simplify the detection of movements. Right now, the system can only detect quick movements to the sides. More complex mechanisms to know the angle at which the user is looking can allow more complex algorithms that depend not only on the direction but also on the angle. Knowing where the subject is looking permits faster and more precise control. For example, creating a gradient of commands that allows speed control. With this improved system, even patterns or combinations of the vertical and horizontal channels could also be introduced, and the corners of the visual field could be used as extra commands.

In terms of design, more comfortable alternatives should be considered. Right now, the electrode cables fall on the user's face and are not excessively comfortable. Among the options proposed is a system similar to the one proposed by (Ryu et al., 2019) that makes it easier to wear with glasses, in case the user needs them, or similar to the EOG forehead proposed by (Heo et al., 2017), which locates the system on a band of cloth placed on the forehead.

It is also important to keep in mind that the measured potential changes with the intensity of the surrounding light and the dilation of the pupil. The more light, the higher the potential (Denney & Denney, 1984). Furthermore, the system has only been tested on one person, so the thresholds may not work for other subjects. This is why the system should be tested on a larger number of users and even on users with eye problems. Also, the range of motion is affected by age. The most affected movement is looking upward, being approximately 33º at ten years of age and 16º at 90 years of age. Looking down, left and right vary little with age. Right and left are similarly reduced by 5° throughout life. The depression angle slightly increases with age (Lee et al., 2019). This may make the thresholds ineffective in people of different ages, thus requiring further research on the effect of age on the measured potential. It is also important to note that this research has been carried out during the spring. Pollen allergy causes inflammation in the eyes that can affect the eyes and the measurements. Also, the electrodes are intended to be used in the next seven days after the bag is opened. When the research started, the bag was already open.

At the end of the research, it was realized that the Pyfirmata library was unnecessary. Initially, it was intended to be used to help control the MPUs simultaneously with the EOG. In the end, the MPUs were used independently, so there was no need for the Pyfirmata library. By the time this conclusion was reached, the research was well underway, and it was decided that there was not enough time to start again. This may have affected the frequency at which EOG is measured. Although an attempt was made to use a frequency of 156 Hz, during the measurements, it was found that the ADC failed to sample that fast, as it was approximately five times lower. Nevertheless, the measurements are still analyzable and allow conclusions to be drawn.

6. Conclusion

The suggested system can detect the user's intention with an overall accuracy of 58% for the EOG, but it fails to know the relationship between the head's and body's position. Although that does not satisfy the needs of Exoskeleton users, it could be promising if the problems with the MPUs were fixed and the blinking detection capacity was improved. Future research should test whether the results can be replicated with other users and connect this algorithm to the low-level control that steers the Exoskeleton motors.

7. References

Arnold, G., Sarlegna, F., Fernandez, L., & Auvray, M. (2019, March 11). Somatosensory loss influences the adoption of Self-Centered versus decentered perspectives. *Frontiers in psychology, 10*, 11. Retrieved April 30, 2024 from https://doi.org/10.3389/fpsyg.2019.00419

Autodesk. (2011). Tinkercad - From mind to design in minutes. *2024*. Retrieved June 17, 2024 from AUTODESK Tinkercad: https://www.tinkercad.com/

Auxivo AG. (n.d.). *Exoskeleton | EduExo Pro.* Retrieved April 16, 2024 from Auxivo: https://www.auxivo.com/eduexo-pro

Auxvivo AG. (2022). *EduExo Pro The Advanced Robotic Exoskeleton Kit Handbook and Tutorial* (1 ed.).

Bamiou, D.-E., & Luxon, L. (2005). Diseases of the eighth cranial nerve. In *Peripheral Neuropathy* (4 ed., Vol. 2, pp. 1253-1272).

Banerjee, A., Pal, M., Datta, S., Tibarewala, D., & Konar, A. (2014, November 1). Eye movement sequence analysis using electrooculogram to assist autistic children. *Biomedical signal processing and control, 14*, 134-140. Retrieved April 17, 2024 from https://doi.org/10.1016/j.bspc.2014.07.010

Barea, R., Boquete, L., Mazo, M., & Lopez, E. (2002, December 1). System for assisted mobility using eye movements based on electrooculography. *IEEE transactions on neural systems and rehabilitation engineering, 10*(4), 209-218. Retrieved April 16, 2024 from https://doi.org/10.1109/tnsre.2002.806829

Carmeli, E., Peleg, S., Bartur, G., Elbo, E., & Vatine, J.-J. (2010, August 25). HandTutorTM enhanced hand rehabilitation after stroke — a pilot study. *Physiotherapy research international, 16*(4), 191-200. Retrieved June 20, 2024 from https://doi.org/10.1002/pri.485

Chudler, E. (n.d.). *Brain Facts and Figures.* Retrieved July 10, 2024 from faculty.washington: https://faculty.washington.edu/chudler/facts.html

Denney, D., & Denney, C. (1984, April 1). The eye blink electro-oculogram. *British journal of ophthalmology, 68*(4), 225-228. Retrieved June 20, 2024 from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1040301/

Franklin, W. (2020, December 31). *Kalman Filter Explained Simply.* Retrieved June 10, 2024 from The Kalman Filter: https://thekalmanfilter.com/kalman-filter-explained-simply/

Gantenbein, J., Dittli, J., Meyer, J., Gassert, R., & Lambercy, O. (2022, February 21). Intention Detection Strategies for Robotic Upper-Limb Orthoses: A scoping review considering usability, daily life application, and user evaluation. *Frontiers in neurorobotics, 16*, 21. Retrieved April 15, 2024 from https://doi.org/10.3389/fnbot.2022.815693

Gull, M., Bai, S., & Bak, T. (2020, March 17). A review on design of upper limb exoskeletons. *Robotics, 9*(1), 16. Retrieved May 15, 2024 from https://doi.org/10.3390/robotics9010016

Hameed, H., Hassan, W., Shafie, S., Ahmad, S., & Jaafar, H. (2019, December 16). A review on Surface Electromyography-Controlled Hand Robotic Devices used for rehabilitation and assistance in activities of daily living. *Journal of prosthetics and orthotics, 32*(1), 3-13. Retrieved May 10, 2024 from https://doi.org/10.1097/jpo.0000000000000277

Heo, J., Yoon, H., & Park, K. (2017, June 23). A novel wearable forehead EOG measurement system for human computer interfaces. *Sensors, 17*(7), 14. Retrieved April 16, 2024 from https://doi.org/10.3390/s17071485

Huang, J. (2023, January 6). *Grove - EMG Detector.* Retrieved April 16, 2024 from Seeed Studio Wiki: https://wiki.seeedstudio.com/Grove-EMG_Detector/

Jarrassé, N., Proietti, T., Crocher, V., Robertson, J., Sahbani, A., Morel, G., & Roby-Brami, A. (2014, December 1). Robotic Exoskeletons: A perspective for the rehabilitation of arm coordination in stroke patients. *Frontiers in human neuroscience, 8*, 13. Retrieved May 20, 2024 from https://doi.org/10.3389/fnhum.2014.00947

Kim, B., & Deshpande, A. (2017, April 1). An upper-body rehabilitation exoskeleton Harmony with an anatomical shoulder mechanism: Design, modeling, control, and performance evaluation. *The international journal of robotics research, 36*(4), 414-435. Retrieved June 10, 2024 from https://doi.org/10.1177/0278364917706743

Kim, J., Bulach, C., Richards, K., Wu, D., Butler, A., & Ghovanloo, M. (2013, November 1). An apparatus for improving upper limb function by engaging synchronous tongue motion. *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, 1574-1577. From https://doi.org/10.1109/ner.2013.6696248

Lee, W., Kim, J., Shin, Y., Hwang, S., & Lim, H. (2019, March 5). Differences in eye movement range based on age and gaze direction. *Eye, 33*(7), 1145-1151. Retrieved June 20, 2024 from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6707237/#:~:text=The%20mean%20values%20for%20angles,%C2%B1%208.0%CB%9A%20in%20depression.

Lv, Z., Wu, X.-P., Li, M., & Zhang, D.-X. (2009, January 1). Development of a human computer Interface system using EOG. *Health, 1*(1), 39-46. Retrieved June 15, 2024 from https://doi.org/10.4236/health.2009.11008

Nelli, F. (2020, September 25). *Hamilton's quaternions and 3D rotation with Python.* Retrieved June 1, 2024 from Meccanismo Complesso: https://www.meccanismocomplesso.org/en/hamiltons-quaternions-and-3d-rotation-with-python/

Nussbaum, M., Esfahani, M., Alemi, M., Jia, B., & Rashedi, E. (2018, July 1). Assessing the influence of a passive, upper extremity exoskeletal vest for tasks requiring arm elevation: Part II – "Unexpected" effects on shoulder motion, balance, and spine loading. *Applied Ergonomics/Applied ergonomics, 70*, 323-330. Retrieved July 1, 2024 from https://doi.org/10.1016/j.apergo.2018.02.024

Rainey, S., Maslen, H., & Savulescu, J. (2020, January 2). When Thinking is Doing: Responsibility for BCI-Mediated Action. *AJOB neuroscience, 11*(1), 46-58. From https://doi.org/10.1080/21507740.2019.1704918

Ryu, J., Lee, M., & Kim, D.-H. (2019, October 1). EOG-based eye tracking protocol using baseline drift removal algorithm for long-term eye movement detection. *Expert systems with applications, 131*, 275-287. Retrieved May 10, 2024 from https://doi.org/10.1016/j.eswa.2019.04.039

Simpson, R. C., & Levine, S. P. (2002, June 1). Voice control of a powered wheelchair. *IEEE Transactions on Neural Systems and Rehabilitation Engineering, 10*(2), 122-125. Retrieved May 10, 2024 from https://ieeexplore.ieee.org/document/1031981

Thakor, N. (2014). Biopotentials and Electrophysiology Measurement. In *Measurement, Instrumentation, and Sensors Handbook: Electromagnetic, Optical, Radiation, Chemical, and Biomedical Measurement* (2 ed., p. 1921). Taylor & Francis Group.

Wang, D., Gu, X., & Yu, H. (2023, March 1). Sensors and algorithms for locomotion intention detection of lower limb exoskeletons. *Medical engineering & physics, 113*, 8. Retrieved May 17, 2024 from https://doi.org/10.1016/j.medengphy.2023.103960

White, M. (2019, July 26). *The MPU6050 Explained.* Retrieved June 1, 2024 from Programming Robots: https://mjwhite8119.github.io/Robots/mpu6050

Wu, S.-L., Liao, L.-D., Lu, S.-W., Jiang, W.-L., Chen, S.-A., & Lin, C.-T. (2013, August 1). Controlling a Human–Computer Interface System With a Novel Classification Method that Uses Electrooculography Signals. *IEEE Journals & Magazine | IEEE Xplore, 60*(8), 2133-2141. Retrieved June 30, 2024 from https://ieeexplore.ieee.org/document/6468076

8. Appendix

8.1 MPU6050 Arduino Gyroscope + Accelerometer

```
#include <Wire.h>
const int MPU_addrs[] = { 0x68, 0x69 };

float AcX[2], AcY[2], AcZ[2], Tmp[2], GX[2], GY[2], GZ[2]; // definition of variables
float roll[2], pitch[2], yaw[2];
float AccErrorX[2], AccErrorY[2], GyroErrorX[2], GyroErrorY[2], GyroErrorZ[2];
float elapsedTime[2], currentTime[2], previousTime[2];
float accAngleX[2],  accAngleY[2], accAngleZ[2], gyroAngleX[2], gyroAngleY[2],
gyroAngleZ[2];

void setup(){

  Serial.begin(19200);
  for(byte b=0; b<2; b++)
  {
    Wire.begin();
    Wire.beginTransmission(MPU_addrs[b]);
    Wire.write(0x1A);  //Switch on low pass filter (10Hz)
    Wire.write(0x05);
    Wire.endTransmission(true);

    Wire.beginTransmission(MPU_addrs[b]);
    Wire.write(0x6B);  // PWR_MGMT_1 register
    Wire.write(0);     // set to zero (wakes up the MPU-6050)
    Wire.endTransmission(true);
    Wire.beginTransmission(MPU_addrs[b]);
    Wire.write(0x1B);                    //Accessing the register 1B - Gyroscope Configuration
(Sec. 4.4)
    Wire.write(0x00000000);
    Wire.endTransmission(true);
    Wire.beginTransmission(MPU_addrs[b]);
    Wire.write(0x1C);                    //Accessing the register 1C - Acccelerometer
Configuration (Sec. 4.5)
    Wire.write(0b00000000);            //Setting the accel to +/- 2g
    Wire.endTransmission(true);

  }
  calculate_IMU_error();
}

void loop()
{
  for(byte b=0; b<2; b++)
  {
    Wire.beginTransmission(MPU_addrs[b]);
    Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
    Wire.endTransmission(true);
    Wire.requestFrom(MPU_addrs[b],8);
    Wire.requestFrom(MPU_addrs[b], 8, true); // request a total of 8 registers
    AcX[b] = Wire.read() << 8 | Wire.read()/ 16384; // 0x3B (ACCEL_XOUT_H) & 0x3C
(ACCEL_XOUT_L)
    AcY[b] = Wire.read() << 8 | Wire.read()/ 16384; // 0x3D (ACCEL_YOUT_H) & 0x3E
(ACCEL_YOUT_L)
    AcZ[b] = Wire.read() << 8 | Wire.read()/ 16384; // 0x3F (ACCEL_ZOUT_H) & 0x40
(ACCEL_ZOUT_L)
    Tmp[b] = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42
(TEMP_OUT_L)

    Wire.beginTransmission(MPU_addrs[b]);
    Wire.write(0x43);
    Wire.endTransmission(true);
    Wire.requestFrom(MPU_addrs[b],6);
    GX[b] = Wire.read() << 8 | Wire.read()/ 131;
    GY[b] = Wire.read() << 8 | Wire.read()/ 131;
    GZ[b] = Wire.read() << 8 | Wire.read()/ 131;

// Correct the outputs with the calculated error values
    accAngleX[b] = (atan(AcY[b] / sqrt(pow(AcX[b], 2) + pow(AcZ[b], 2))) * 180 / PI) ;
    accAngleY[b] = (atan(-1 * AcX[b] / sqrt(pow(AcY[b], 2) + pow(AcZ[b], 2))) * 180 / PI);

    previousTime[b] = currentTime[b];        // Previous time is stored before the actual time
read
    currentTime[b] = millis();            // Current time actual time read
    elapsedTime[b] = (currentTime[b] - previousTime[b]) / 1000; // Divide by 1000 to get
seconds

    GX[b] = GX[b] - GyroErrorX[b]; // GyroErrorX
    GY[b] = GY[b] - GyroErrorY[b]; // GyroErrorY
    GZ[b] = GZ[b] - GyroErrorZ[b]; // GyroErrorZ
    // Currently the raw values are in degrees per seconds, deg/s, so we need to multiply by
seconds (s) to get the angle in degrees
    gyroAngleX[b] = gyroAngleX[b] + GX[b] * elapsedTime[b]; // deg/s * s = deg
    gyroAngleY[b] = gyroAngleY[b] + GY[b] * elapsedTime[b];
    yaw[b] =  yaw[b] + GZ[b] * elapsedTime[b];

    // Complementary filter - combine acceleromter and gyro angle values
    roll[b] = 0.96 * gyroAngleX[b] + 0.04 * accAngleX[b];
    pitch[b] = 0.96 * gyroAngleY[b] + 0.04 * accAngleY[b];

    Serial.print(b+1);
    Serial.print(" ");
    Serial.print(roll[b]);
    Serial.print(" ");
    Serial.print(pitch[b]);
    Serial.print(" ");
    Serial.print(yaw[b]);
    Serial.println(" ");


  }
  //delay(100);
}

void calculate_IMU_error() {
  // Calculate error - IMU flat
  // Read accelerometer values 200 times
  for(byte b=0; b<2; b++)
  {
    //while (c < 200) { //No me gustan los while
    for (int i = 0; i < 200; i++){

      Wire.beginTransmission(MPU_addrs[b]);
      Wire.write(0x3B);
      Wire.endTransmission(false);
      Wire.requestFrom(MPU_addrs[b], 6, true);
      AcX[b] = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
      AcY[b] = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
      AcZ[b] = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
      Wire.beginTransmission(MPU_addrs[b]);
      Wire.write(0x43);
      Wire.endTransmission(false);
      Wire.requestFrom(MPU_addrs[b], 6, true);
      GX[b] = Wire.read() << 8 | Wire.read();
      GY[b] = Wire.read() << 8 | Wire.read();
      GZ[b] = Wire.read() << 8 | Wire.read();

      // Sum all readings
      AccErrorX[b] = AccErrorX[b] + ((atan((AcY[b]) / sqrt(pow((AcX[b]), 2) + pow((AcZ[b]),
2))) * 180 / PI));
      AccErrorY[b] = AccErrorY[b] + ((atan(-1 * (AcX[b]) / sqrt(pow((AcY[b]), 2) +
pow((AcZ[b]), 2))) * 180 / PI));
      // Sum all readings
      GyroErrorX[b] = GyroErrorX[b] + (GX[b] / 131.0);
      GyroErrorY[b] = GyroErrorY[b] + (GY[b] / 131.0);
      GyroErrorZ[b] = GyroErrorZ[b] + (GZ[b] / 131.0);

    }
    //Divide the sum by 200 to get the error value
    AccErrorX[b] = AccErrorX[b] / 200;
    AccErrorY[b] = AccErrorY[b] / 200;
    //Divide the sum by 200 to get the error value
    GyroErrorX[b] = GyroErrorX[b] / 200;
    GyroErrorY[b] = GyroErrorY[b] / 200;
    GyroErrorZ[b] = GyroErrorZ[b] / 200;

  }
}
```

8.2 MPU6050 Arduino DMP Quaternions

```
#include <Wire.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps612.h"

MPU6050 mpu1(0x68);  // MPU device 1 (AD0 low)
MPU6050 mpu2(0x69);  // MPU device 2 (AD0 high)

bool dmpReady1 = false, dmpReady2 = false;
uint8_t mpuIntStatus1, mpuIntStatus2;
uint16_t packetSize1, packetSize2;
uint16_t fifoCount1, fifoCount2;
uint8_t fifoBuffer1[64], fifoBuffer2[64];
```

```
float yr, pr, rr, y, p, r;

Quaternion q1, q2;
VectorFloat gravity1, gravity2;
float ypr1[3], ypr2[3];

void setup() {
   Wire.begin();
   Serial.begin(38400);

   mpu1.initialize();
   mpu2.initialize();

   Serial.println(mpu1.testConnection() ? "MPU1 connection successful" : "MPU1
connection failed");
   Serial.println(mpu2.testConnection() ? "MPU2 connection successful" : "MPU2
connection failed");

   uint8_t devStatus1 = mpu1.dmpInitialize();
   uint8_t devStatus2 = mpu2.dmpInitialize();

   if (devStatus1 == 0 && devStatus2 == 0) {
      mpu1.setDMPEnabled(true);
      mpu2.setDMPEnabled(true);
      packetSize1 = mpu1.dmpGetFIFOPacketSize();
      packetSize2 = mpu2.dmpGetFIFOPacketSize();
      dmpReady1 = true;
      dmpReady2 = true;
   } else {
      Serial.println("DMP Initialization failed.");
   }
}
void loop() {
   if (!dmpReady1 || !dmpReady2) return;

   manageFIFO(mpu1, fifoCount1, packetSize1, fifoBuffer1, q1, gravity1, ypr1, "MPU1");
   manageFIFO(mpu2, fifoCount2, packetSize2, fifoBuffer2, q2, gravity2, ypr2, "MPU2");

   delay(100);  // Adjust delay based on your needs
}

void manageFIFO(MPU6050 &mpu, uint16_t &fifoCount, const uint16_t packetSize,
uint8_t *fifoBuffer, Quaternion &q, VectorFloat &gravity, float *ypr, String label) {
   fifoCount = mpu.getFIFOCount();

   if (fifoCount < packetSize) return;  // Ensure we have enough data for a full packet

   while (fifoCount >= packetSize) {
      mpu.getFIFOBytes(fifoBuffer, packetSize);
      fifoCount -= packetSize;
   }

   mpu.dmpGetQuaternion(&q, fifoBuffer);
   mpu.dmpGetGravity(&gravity, &q);
   mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

   Serial.print(label + ":");

   Serial.print(ypr[0] * 180/M_PI);
   Serial.print(",");
   Serial.print(ypr[1] * 180/M_PI);
   Serial.print(",");
   Serial.print(ypr[2] * 180/M_PI);
   Serial.println(";");
}
```

8.3 EOG Python code

```
import threading
import pyfirmata
import time
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation


#setup initial values
n = 100  # seconds of experiment
freq = fs = 256  # frequency (timepoints per sec)
aveV = 0
aveH = 0
#arduino setup
port = 'COM3'
board = pyfirmata.Arduino(port)  # Initialize the communication
with the Arduino card
```

```
a1 = board.get_pin('a:1:o')
a2 = board.get_pin('a:2:o')

it = pyfirmata.util.Iterator(board)
it.start()

# Left right + Up down checking activated
ON = True
global detect
detect = True
dataV = []
dataH = []

global blink
blink = False

def EOG(): #EOG get data
   print("starting")
   global dataV
   global dataH

   for i in range(200):
       valueH = a1.read()
       valueV = a2.read()
       dataH.append(valueH)
       dataV.append(valueV)
       time.sleep(5 / freq)
   aveV = np.average(dataV[50:200:])
   aveH = np.average(dataH[50:200:])
   dataH = list(dataH[50::] - aveH)
   dataV = list(dataV[50::] - aveV)

   for i in range(n*freq):
       valueH = a1.read() - aveH
       valueV = a2.read() - aveV
       dataH.append(valueH)
       dataV.append(valueV)
       time.sleep(5 / freq)

   return dataV, dataH
i = 1

def detection():
   global detect
   print("LRUD started")
   while ON == True:
       if detect == True:
           i = len(dataH) - 201
           if (i > 0):
               print("start")
       while ((detect == True) and (i > 0)):
           # print(len(dataH)-i)
           print("")
           if (((min(dataH[i + 25:i + 50]) < -0.026)) and (0.025 >
max(dataH[i:i + 25]) > 0.010)):
               detect = False
               print("LEFT")

           elif (((-0.020 > min(dataH[i:i + 75]))) and
(max(dataH[i + 25:i + 100]) > 0.03)):
               detect = False
               print("RIGHT")

           elif (((-0.025 > min(dataV[i:i + 75]) >-0.04)) and
(max(dataV[i + 25:i + 100]) > 0.025)):
               detect = False
               print("UP")

           elif (((-0.04 < min(dataV[i + 25:i + 50]) < -0.016))
and (0.05 > max(dataV[i:i + 25]) > 0.01)):
               detect = False
               print("DOWN")


           elif (((min(dataV[i:i + 25]) < -0.030))and (0.025
>max(dataV[i+25:i + 75]) > 0.01)):
               detect = False
               print("Blink")

           if i < (len(dataH) - 100):
               i += 50
```