The design of an intuitive interface for the GyBAR

Simplifying the controls of a balance assistance device for optimal adoption by physical therapists

by

Daniela Estrada

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday December 9, 2022 at 02:00 PM

Student number:	4373804	
Project duration:	May 23, 2022 – December 9, 2022	
Thesis committee:	Prof. Dr-Ing H. Vallery,	TU Delft, supervisor
	MSc. B. Sterke,	TU Delft, Erasmus University, daily supervisor
	Dr. K. Poggensee,	TU Delft, Erasmus Univerity, daily supervisor

Acknowledgements

This report shows my final work as a master student at the TU Delft, which also marks the end of an era for me. I know it sounds cliché, but that is what this text is for right? I have been studying in Delft for about 8 years, following my own path to the finish line. After deciding Mechanical Engineering was not my passion I decided to switch to Industrial Design Engineering to finally end up at 3ME again. It may have cost me a couple of study years, but I don't regret any of my choices. I have really enjoyed combining both, using the design methods I have learned in my bachelor and applying them to topics in Biomedical Engineering.

This thesis has showed me how I can use my background as an advantage. I have had moments of doubt, thinking I missed background knowledge because I did not obtain my bachelor's degree in Mechanical Engineering. I would like to thank Bram Sterke, my supervisor, for helping me to trust my abilities and to guide me to the right direction. I would like to thank Katie Poggensee, my other daily supervisor, for always being open to my questions and giving me the tools to continue my work. I would like to thank my supervisor Heike Vallery for asking me the right questions and giving me useful feedback. Additionally, I would like to thank everyone in the lab for always making me feel welcome and letting me use your personal desks as a flex workspace. I really enjoyed getting to know you and spending the last year with you.

I have been very lucky to be surrounded with people who have been very supportive. Thank you Lianne (moemie), for the phone calls that helped me straighten my thoughts and giving me your insights.

Gracias Papá, por creer en mí y siempre tratarme como tu princesa.

Sebastián and Adrián, my big brothers, thank you for always setting the example and giving your support.

Lastly, Koen, thank you for dealing with me being a little grumpy (sometimes) and making our new apartment feel like home while I was busy studying.

Daniela Estrada
Delft, December 2022

Summary

According to the WHO, falls are responsible for over 38 million disability-adjusted life years lost each year, globally. Additionally, an estimated 684.000 individuals die of falling each year, making it the second leading cause of unintentional death. One of the most effective physical activity interventions to reduce the risk of falling, is targeted exercise that safely challenges balance. One tool, that is currently being developed, which can aid physical therapists with these interventions is the GyBAR. A wearable device that uses gyroscopes to apply moments on the patient. This can be used to either provide balance assistance or challenge balance by applying perturbations. The goal of this research was to develop an interface for the GyBAR. The interface should contribute to the acceptance of the GyBAR. This can be achieved by an excellent perceived ease of use, which combined with perceived usefulness are indicators for the Technology Acceptance Model (TAM). The interface inspired by a Voodoo-doll was developed by going through the following design process steps; the creation of a list of requirements; a brainstorm to generate ideas; the development of three concepts; and the selection and development of one concept into a prototype. The Voodoo Doll controls the GyBAR by a handheld model of the patient, which can be manipulated and translates the movements of the model to the actual patient. User tests were performed to validate the design. With a SUS-score of 82.81 (SD=7.48), which is within the 90-95 percentile, it can be concluded that the interface has an excellent perceived ease of use. Further development is encouraged are several recommendations for points of improvement are given.

Contents

1 Introduction

According to the WHO, falls are responsible for over 38 million disability-adjusted life years lost each year, globally. Additionally, an estimated 684.000 individuals die of falling each year, making it the second leading cause of unintentional death [6]. One of the most effective physical activity interventions to reduce the risk of falling, is targeted exercise that safely challenges balance [22].

Research shows an increased risk of falls among persons with physiological impairments associated with neurological conditions, compared to healthy age matched individuals [24, 2]. An impaired balance control has a negative influence on the reactive balance of the individual which can lead to a fall. The occurrence of a fall due to failing to recover after a postural perturbation is one of the most common causes of injuries [6]. On top of that, an increased risk of falling comes with fear, which leads to less confidence [22]. This can result in self-imposed activity restrictions, affecting ones independence.

Research shows that reactive balance can be improved with Perturbation-Based Balance Training (PBT). This is a balance training intervention that shows improvement in the control of balance recovery reactions, which reduces the risk of falling [14, 16, 13]. The working principle of PBT is based on training the exact mechanism that is needed to recover from a perturbation in daily life. Instead of training voluntary movements, an individual is exposed to repeated perturbations that evoke rapid balance reactions, allowing them to improve control of their reactive balance. Perturbations can be applied by using a treadmill that changes speed or direction [23], by a therapist who exerts manual pushes on the body [21], or by a robotic device that applies mechanical perturbations on the body [15].

One innovative tool that can aid therapists with PBT, the GyBAR [12], is under development. The GyBAR is a wearable device that uses a gyroscope to apply moment forces on the patient. Additionally to PBT, it can provide balance assistance (BA), countering loss of balance. This gives the patient more time to recover when losing their balance. In practice this can be used for training different aspects of rehabilitation which usually are not possible because the patient is focused on their balance.

The GyBAR has been shown to improve balance [12] and can be used to replace overhead support devices. It has the advantage that it is not restricted to a treadmill and can provide overground balance support without obstructing the hands and legs. The GyBAR, just like other innovative tools that are being developed, shows promise to improve patient care. However, often new technologies are not easily adopted in practice [10].

New technologies often have complicated looking user interfaces, which lead to unclear functionalities. This finally results in slow or non-optimal adoption of new technologies within the intended use scenario.

The Technology Acceptance Model (TAM) shows that adoption is dependent on the perceived usefulness and perceived ease of use [9]. Users need to see the value of using the new technology instead of their current way of working. The new technology should provide the user equal or better results and should be seen as easy to use.

In case of the GyBAR, the usefulness is dependent on the intended applications, which are based on the functions of the device. The perceived ease of use can be improved with an interface that is accessible, easy to learn, intuitive and should invite physical therapists to use the product optimally. The goal of this project is to design an interface that controls the GyBAR when used for physical therapy purposes, specifically for gait rehabilitation consisting of BA and PBT. Currently a Python Graphical User Interface (GUI) is used to control the GyBAR, which is not easy to understand for non-engineers and is assumed to have a low perceived ease of use. For rehabilitation tools used during therapy, the physical therapist is the one who controls the device, making them the user of the device in this research, not the patient.

This report describes the process leading to the prototype of an interface for the GyBAR and the validation of this prototype. This report consists of two parts: (I) The Interface Design and (II) The Interface Validation. This is followed by a discussion of the results and limitations, resulting in a conclusion and recommendations for further research.

Part I

Interface Design

2 Requirements

In order to determine the requirements for the interface the user, the environment and the intended application were analyzed.

2.1. Users

The users are physical therapists that treat people with balance problems, caused by a variety of pathologies. During a conversation with two therapists working at a progressive rehabilitation centre in the Netherlands, several needs and wishes came up;

- Patient profiles, to plan training sessions and track progress.
- A quick setup, to allow for the required limited time in between patients.
- An interface close to the device, to always be present for use.
- An automatically charged interface, to always be ready for use.
- A training environment that mimics reality as closely as possible, to prepare patients optimally.

Furthermore, some general characteristics of the physical therapists are assumed (shown in figure 2.1. Many exercises involve therapists physically supporting, guiding or challenging their patients during therapeutic sessions. Therefore, it is assumed that the therapists have a hands on approach. Additionally, they have an education in physical therapy and are assumed to not have a background in engineering, therefore it can not be assumed that new technologies will be understood automatically.

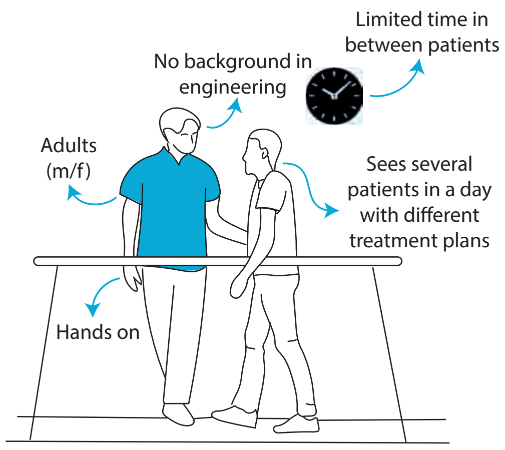

Limited time in between patients
No background in engineering
Adults (m/f)
Sees several patients in a day with different treatment plans
Hands on

Figure 2.1: General characteristics of the user: a physical therapist.

2.2. Intended Application

The interface will be used to control the GyBAR, which is used for BA and PBT. The BA is a continuous mode, where loss of balance is detected and countered with a force in the opposite direction. For PBT two modes are available. During automatic (continuous) mode, a moment is applied in the direction the loss of balance occurs. The manual (discrete) mode allows the physical therapist to manually apply the perturbation during the session.

In order to get an idea of how the GyBAR will be used in practice, a use scenario is shown in figure 2.2. The interface is shown as a black box, since this design had yet to be determined. The therapy session consists of four phases where the interface will be used:

1. Before the session, when uploading the patient profile and training program.
2. At the start of the session, the mode will be set to BA, automatic PBT or manual PBT.
3. During the session, changing the magnitude of the moments, changing modes, pausing the session, and applying perturbations in manual PBT mode.
4. After the session, analyzing the session and saving progress.

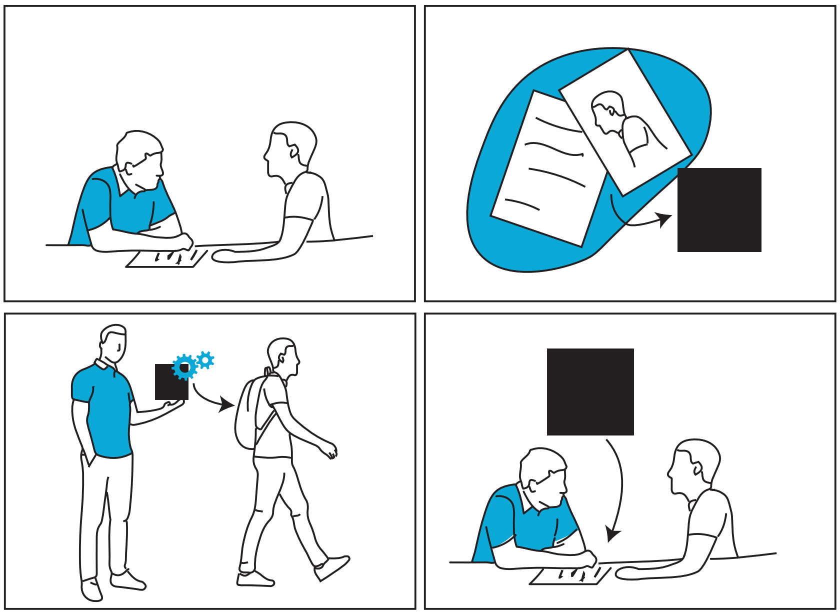


Figure 2.2: The user scenario where the interface (depicted as a black box) is used for uploading the patient profile and training program, to make adjustments during session and to look in to the data after the training session.

The actions that are done during the session, should be able to be done from a distance of the patient, without them being aware of the adjustments made.

2.3. Environment

Training takes place in the exercise room in a rehabilitation centre (shown in figure 2.3, the hallways and occasionally outside. Since the GyBAR will be a mobile device it could be used in all different settings. In the exercise room, multiple physical therapists and patients are present. Therefore, the interface should not interfere with others present. As mentioned before, one of the preferences of the physical therapists is to mimic reality as closely as possible. Since this device allows mobile use, in the future it might also be used in other environments, like public places or at home.

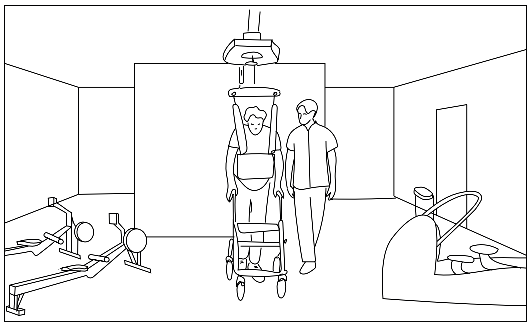


Figure 2.3: An exercise room in a rehabilitation centre, containing different devices used for physical therapy.

2.4. Requirements

Based on previous findings a list of requirements is drafted, categorized according to Roozenburg and Eekels [19].

Performance

The interface must:

- provide training sessions with balance assistance.
- provide training sessions with automatic perturbations.
- provide training sessions with manual perturbations.
- show predetermined training sessions.
- allow for making adjustments during a training session.
- store patient information.
- allow the therapist to have an insight on previous training sessions per patient.
- be able to control the device from a distance.
- must be intuitive to use

Maintenance

- Small maintenance of the interface must be able to be performed by in-house technicians.

Aesthetic, Appearance and Finish

- All functionalities/applications of the device must be clear at first look.

Ergonomics

- The interface must be used by adult males and females, without the need for different sizes or configurations.

Storage

- The interface must be stored close to the device.
- The interface must be charged automatically during storage.

Safety

- A stop button must be integrated in the interface, with minimal delay.

Installation and Initiation of Use

- The interface must take less or equal time to start as the gyroscope.

3 Concepts

3.1. Idea generation

The interface can be divided in different components: display, storage, data storage, controller and power supply. The characteristics of these components can be determined based on the requirements that are stated in the previous chapter. The display, the storage, the data storage and the power supply can be constructed using prefabricated components. Since these components are being produced at high quality and a high range of specifications, it is not necessary to redesign them. The controller component of the interface allows for the most freedom of design, therefore three concepts were developed based around an idea for this component.

First a brainstorm session with a product designer took place. During this session no limitations were set and anything was possible. This resulted in an array of ideas on how the GyBAR can be controlled (shown in Appendix A). These ideas were clustered, which led to the following categories:

- Remote controller
- Remote controller with motion control
- Motion controller
- Voice controller/Mind controller
- Controller on GyBAR or display

Since one of the requirements was that the interface must be able to be used from a distance, the controls on the GyBAR or display are rejected. Furthermore, mind control is too complex and voice control does not follow the requirement that the patient must be unaware of adjustments the therapist applies during the session. A concept was created for each of the remaining categories: remote controller, remote controller with motion control and motion controller.

3.2. Concept ideas

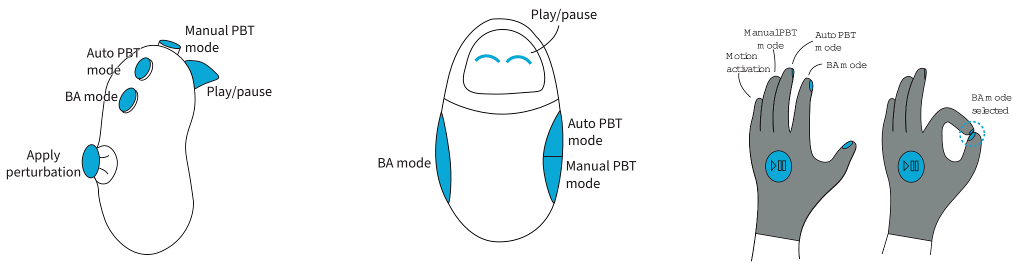
The three concepts (shown in figure 3.1 were designed to meet all the performance requirements described in chapter 2.

3.2.1. Remote controller: The Ergonomic Remote

The Ergonomic Remote (shown in figure 3.1a) controls the GyBAR from a distance with the use of buttons. The design is inspired by game-controllers, made to fit in the palm of your hand naturally, with all the buttons within reach. The three modes can be selected using the round buttons on the front of the remote. The modes can be started or paused by pressing the button on the back of the remote. The manual perturbations can be applied with the joystick, in the right direction.

3.2.2. Remote controller with motion control: The Voodoo Doll

The Voodoo Doll (shown in figure 3.1b) controls the GyBAR by a handheld model of the patient, which can be manipulated and translates the



(a) The Ergonomic Remote, an ergonomic design with buttons in reach during grip. **(b)** The Voodoo Doll, manipulate the GyBAR by controlling a handheld model. **(c)** The Motion Glove, use hand gestures to control the GyBAR while being hands-free.

Figure 3.1: The three developed concepts.

movements of the model to the actual patient. The different modes can be selected using buttons, with the same concept as the Ergonomic Remote. The perturbations can be applied by activating the motion detection and moving the handheld model.

3.2.3. Motion controller: The Motion Glove

The Motion Glove (shown in figure 3.1c) is a wearable that uses hand gestures to control the GyBAR. Modes can be selected and intensity can be adjusted through hand gestures. For example when thumb and index finger are touching the BA mode is selected. The modes can be started and paused by pressing a button. A perturbation can be applied by activating motion detection by pressing a button and waving in the desired direction.

3.3. Concept Selection

A Harris profile [3] was used to select the final concept. A list of selection criteria based on the requirements, described in chapter 2, was used. Criteria that were assumed to have the most influence on perceived ease of use were listed and ranked from most to least important:

1. The interface should be able to be used intuitively.
2. The interface should appear to be simple to use.
3. The interface should be easy to set up.
4. The functions of the interface must be visible.
5. There must be a quick learning curve to use the interface.
6. The interface should be easy to physically.
7. The interface should be lightweight.
8. The interface should be universally applicable.

Each concept was graded per criteria. The Harris profile is a visual method, criteria are graded by filling in squares with values -2 to +2. It is important to use one color for the squares, the concept where the visual “leans” most to the right meets the criteria the best.

The result of the Harris profile (figure 3.2) shows that the Voodoo doll concept is the most promising and is therefore selected.

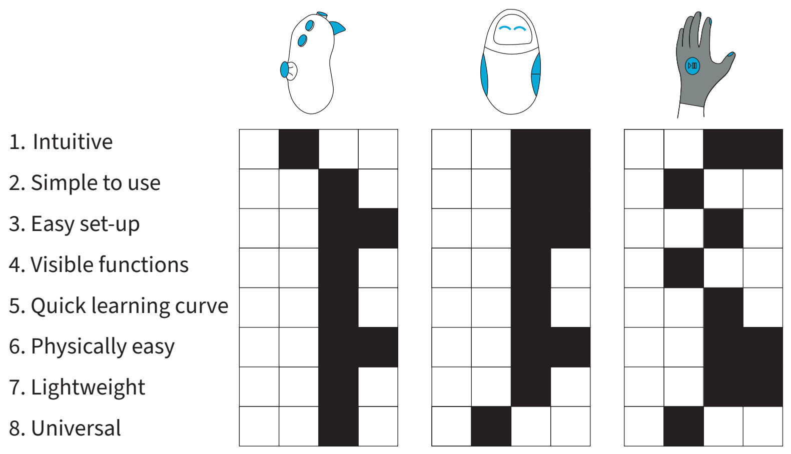


Figure 3.2: The Harris profiles of the three concepts, from left to right: the Ergonomic Remote, the Voodoo Doll and the Motion Glove

Part II

Interface Validation

4 Prototype

A prototype of the Voodoo Doll was made to be used for user tests, to validate the perceived ease of use of the interface.

4.1. Prototype wishes

For the experiment a working prototype is required, to test perceived ease of use. In order to create an intuitive design, reaction time and amount of errors made should be recorded. This will give insights in whether the design is clear and if changes should be made. The motion detection that is used for manual PBT should only be activated when the user wants to apply a perturbation. In order to get an insight in how the user will orient the interface when applying a perturbation, orientation data should be recorded. For the experiment, the prototype will not control the GyBAR, instead it is connected to a computer. Any actions that are performed will control a Python script, which shows a simple looking simulation of the device on a computer screen. The Python script is also used to track all interaction of the participants with the interface.

4.2. Physical Design

The physical design of the Voodoo Doll is built consists of a Wiimote [1], an Arduino Nano 33 IoT [17] and 3D printed components. A 3D-printed cover is used to change the appearance of the Wiimote. This is important since the Wiimote is a familiar device and the user should not associate the interface with the original application of the Wiimote. The redundant buttons are covered up and the icons on the buttons that will be used are hidden. An abstract face is visualized on the cover to portray the patient that will wear the GyBAR. This could be perceived as an indicator of how the interface is oriented relative to the patient. The three buttons aligned horizontally in the middle of the interface are used to select the different modes. The play/pause button is used to start and pause the modes. The two buttons on the bottom of the interface are used to increase and decrease the magnitude of the moments. The button on the back of the interface is used to activate the motion detection used for applying a perturbation. The 3-axis accelerometer sensor of the LSM6DS3 module on the Arduino Nano 33 IoT is used to determine the relative orientation of the interface. The Wiimote and Arduino are connected to eachother using the plug of the Wii Nunchuck (shown in figure 4.1). Note that this is only a physical connection, there are no signals being sent between the Wiimote and the Arduino. The Wiimote is connected to the computer containing the Python script through a bluetooth connection and the Arduino Nano with a USB cable.

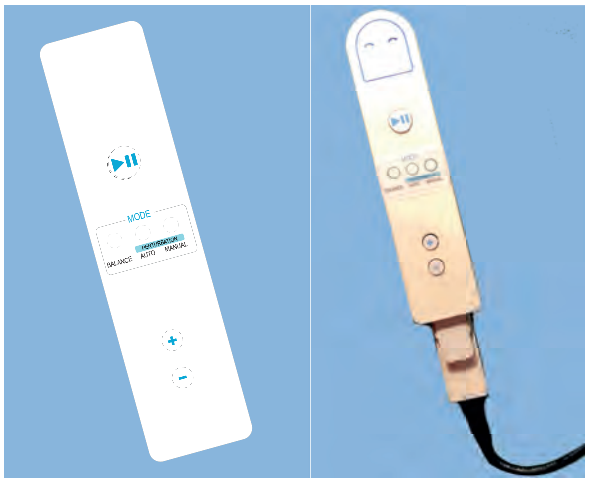


Figure 4.1: Final design of the prototype of the Voodoo Doll.

4.3. Code

A Python script (shown in Appendix B) was written to give visual feedback to the tasks performed. A full screen with bright text shows the action that is performed by pressing buttons. The accelerometer data is processed with an Arduino code (shown in appendix C). The motion detection is visualized with a colored 3D cube which moves in the same directions as the accelerometer (shown in figure 4.2). Additionally, all actions are printed in the terminal, with a timestamp, which will not be visible for the participant.

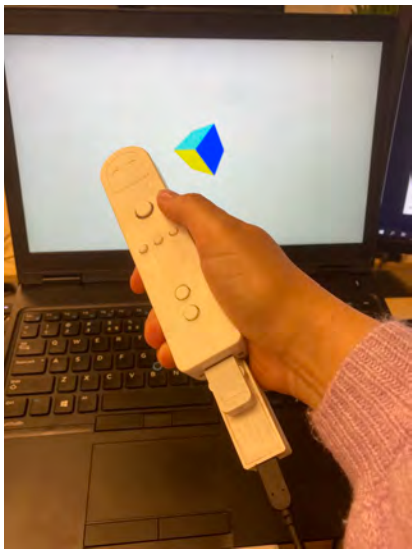


Figure 4.2: The orientation of the Voodoo Doll is visualized with a colored cube.

5 Experiment

5.1. Method

In order to test the perceived ease of use, an experiment was performed where the participants were asked to perform certain tasks with the interface. Different measures were used to get insights on the interaction between the user and the interface.

5.1.1. Participants

Young adults (n=8, all female) with limited technological knowledge and familiar with the intended use scenarios were selected to join the experiment. Students and PhD students from a large academic hospital working on rehabilitation studies were recruited via the professional network of the supervisors of this master thesis. Since their working area includes rehabilitation, it was assumed that they had sufficient knowledge on the intended use scenarios.

5.1.2. Materials

The experimental setup consisted of the interface and a laptop with a running Python script, that could be controlled using the interface. The System Usability Scale (SUS) [9] questionnaire was used to assess the perceived ease of use of the interface. It consists of ten five-point items with alternating positive and negative tone, shown in Appendix D.

5.1.3. Procedure

First an application for ethical approval was submitted. Once this application was approved, contact with the participants was made through the professional network of the supervisors of this research. The research was conducted in one day at a large academic hospital, the workplace of the participants. The whole experiment lasted for about 20 minutes per participant. The participants were given a presentation about the GyBAR, by one of the responsible researchers, three days before the experiment was conducted. During this presentation, they were not yet given specific information on the interface.
First the participant was asked to fill in the consent form, which contains a short introduction on the experiment (shown in Appendix E). Thereafter more information on the procedure was given. The intended use scenarios of the GyBAR in combination with the interface were explained.
During the explanation, the interface was already handed to the participant, for them to explore it. Thereafter, a short explanation about the different functions of the interface was given. Before starting with the tasks, the participant was asked to put the interface down. They were asked about their own view on new technologies and their ease of use. This was partly done to distract the participant and prevent them to work from their short term memory.

After five minutes of distracting the participant, they are asked to pick up the interface again and start performing the different tasks (shown in Appendix F).

All actions on the interface showed the participant what they were doing on a full screen window on the laptop. Additionally their actions were recorded with a timestamp. The orientation of the device was recorded from the moment the participant first pressed the motion detection button, whilst being in active automatic PBT mode. When the task was given, a key on the keyboard was pressed at the start and end of the instruction. This was also recorded with a timestamp. The recorded data with timestamps were not visible for the participants.

After all tasks were completed the participant was asked to fill in the SUS-questionnaire (found in Appendix D) with their immediate response, rather than thinking about it for a longer period. The participant was asked to score ten statements on the scale of zero (completely disagree) to five (completely agree).

5.1.4. Analysis

As described before, three types of data were extracted during the experiment: the SUS-score, the reaction time/completion time and the orientation of the device. All data was prepared in Microsoft

Excel and analysed using Matlab (scripts can be found in Appendix G, H and I).

The outcome of the SUS-questionnaire was evaluated. The given scores from zero to five were converted in the range of zero to four according to Brooke's scoring [4].
The scoring differed for the odd- and even-numbered questions. The odd-numbered questions ($s(n_{odd})$) are scored by subtracting the given score by c_{odd} (equals one), shown in equation 5.1. The even-numbered questions ($s(n_{even})$) were scored by subtracting the given score from c_{even} (equals 5), shown in equation 5.2. The sum of the scores of the odd-numbered questions (S_{odd}) and the even-numbered questions (S_{even}) is multiplied by α (equals 2.5), shown in equation 5.3.

$$S_{\text{odd}} = s(n_{\text{odd}}) - c_{odd} \tag{5.1}$$

$$S_{even} = c_{even} - s(n_{even}) \tag{5.2}$$

$$\text{SUS-score} = (S_{odd} + S_{even}) \cdot \alpha \tag{5.3}$$

This resulted in the SUS-score in the range of 0-100. Only the final score is relevant, since the score of all questions combined make up the SUS-score.
The mean SUS-score is compared to the "average" SUS-score. This average equals a SUS-score of 68, "the average", any score above 68 can be considered above average and scores lower than 68, below average [20]. According to the Shapiro-Wilk test (p=0.000823) the SUS scores follow a normal distribution. Therefore a T-test was performed to determine the difference between the mean SUS score and the average SUS score. Additionally, the SUS score was interpreted using the Sauro-Lewis curved grading scale [11].

The reaction time and the completion time were both analyzed. The reaction time was the time it took the participant to press the first (correct) button after the task was given. This was computed by subtracting the time when the first button was pressed, with the start time of the question. The completion time is the time it took to complete the task, which could consist of multiple actions. The completion time of the task was computed by subtracting the time when the task was completed with the start time of the question. The mean of the reaction time per task was compared to research on choice reaction time.

The X and Y-rotation was recorded in degrees against time. Sensor noise was filtered out using a 4th order Butterworth filter with a cutoff frequency of 4 Hz. The orientation data of each participant was visualised in two graphs, X rotation against time and Y rotation against time. This graph was analysed to see if there exists a pattern between the participants.

5.2. Results

During the experiment, three types of data were extracted: the SUS-score, the reaction time and the orientation of the interface. In this section the main results are explained.

The SUS scores are shown in figure 5.1. The scores ranged from 72.5 to 92.5. The mean SUS-score is 82.81 (SD=7.49), which differs significantly from the average SUS-score of 68 (p=0.000823).

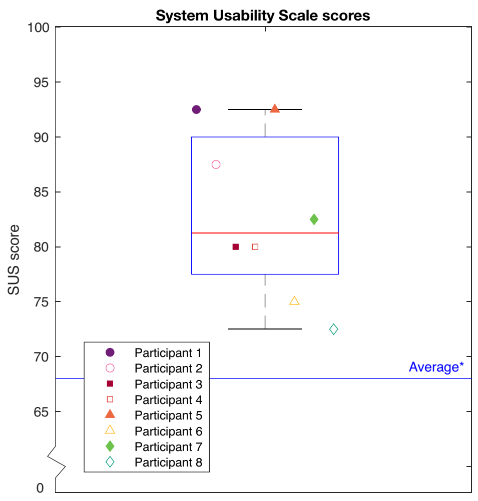


Figure 5.1: The SUS-scores given by the participants and the mean (82.81 ±7.49). *Average SUS-score which is seen as the threshold for an adequate perceived ease of use [20].

The reaction time was determined by subtracting the start time of when the question was asked from the time of the first action of the sequence that completed that task. As shown in figure 5.2 and table 5.1, the reaction time and completion time differ per task. All tasks, but the final task, have a reaction time in the range of the compared reaction times.

There were almost no errors made. The only errors were made by one participant, who pressed the wrong button 9 times for the final task. There is a limited number of participant and data per participant, zero errors for seven participants and one participant only mader errors for on of the tasks. It was therefore decided that the outlier was not to be excluded.

In figure 5.3 the filtered orientation data is visualized with X and Y rotation against time.

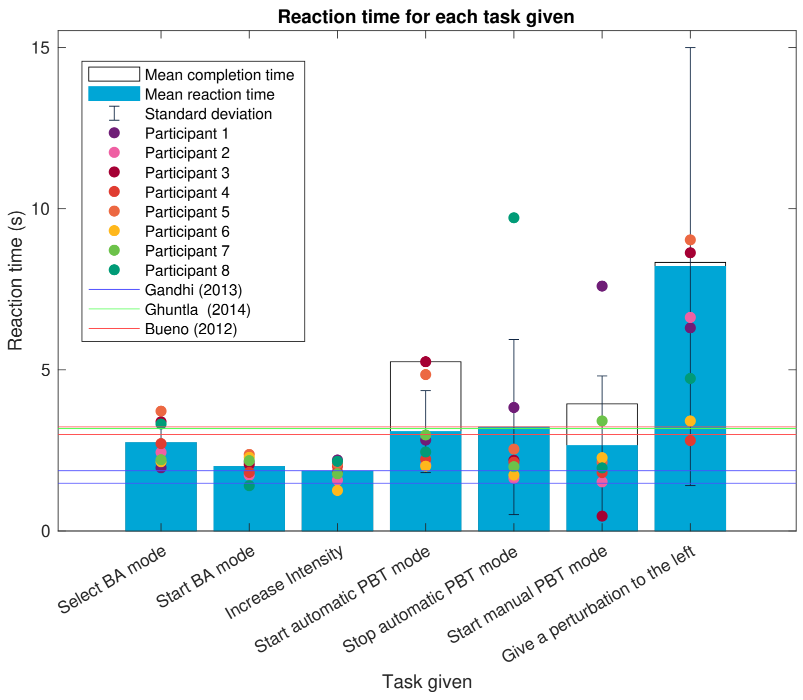


Figure 5.2: The reaction time and completion time for each task given per participant, mean and standard deviation. Compared to reaction time of related research.

Table 5.1: Reaction time, Completion time and number of errors made for each task performed.

Task	Reaction time (s)	Completion time (s)	Error
Select BA mode	2.74(SD=0.69)	2.74 (SD=0.69)	0 (SD=0)
Start BA mode	2.00 (SD=0.33)	2.00 (SD=0.33)	0 (SD=0)
Increase Intensity	1.86 (SD=0.32)	1.86 (SD=0.32)	0 (SD=0)
Start Automatic PBT mode	3.08 (SD=1.27)	5.25 (SD=1.54)	0 (SD=0)
Stop Automatic PBT mode	3.22 (SD=2.71)	3.22 (SD=2.71)	0 (SD=0)
Start Manual PBT mode	2.65 (SD=2.16)	3.94 (SD=2.16)	0 (SD=0)
Apply a Perturbation to the left	8.21 (SD=6.79)	8.34 (SD=7.14)	1.13 (SD=3.18)

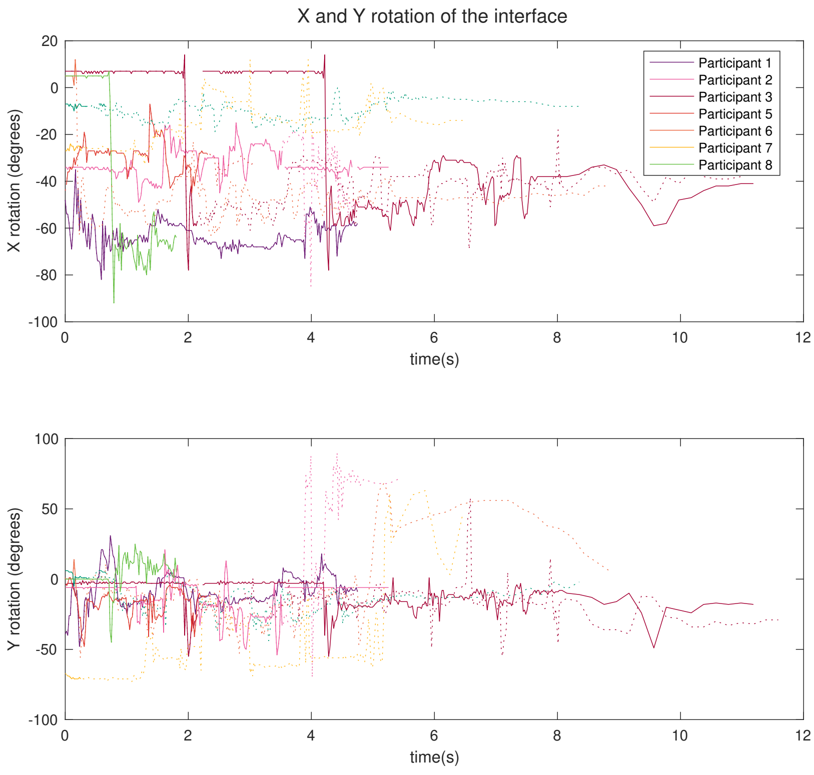


Figure 5.3: The orientation visualised as the X and Y rotation for all the participants. The dotted lines show the orientation from the moment the participant first pressed the motion detection button. The solid line shows when the participant held the button down.

6 Discussion & Conclusion

6.1. Discussion

The interface inspired by a Voodoo Doll was developed by going through the following design process steps; the creation of a list of requirements: a brainstorm to generate ideas; the development of three concepts; and the selection and development of one concept into a prototype. The Voodoo Doll controls the GyBAR with a handheld model of the patient, which can be manipulated and translates the movements of the model to the actual patient. User tests were performed to validate the design.
The perceived ease of use of the interface was determined by the SUS-score. The obtained score was 82.81 (SD = 7.48), which is in the 90-95 percentile range according to the Sauro-Lewis curved grading scale [11]. This can be compared to other devices used in rehabilitation. Research has shown that the Lokomat ® (Hocoma AG, Volketswil, Switzerland) has a SUS-score of 54.4 (SD = 13.95) [18]. Based on this it can be concluded that the design of the Voodoo doll has an excellent perceived ease of use.

The mean reaction times of all, except the final task, were in the same range as the reaction time of similar research. The lines on the graph (shown in fig. 5.2) present the range of reaction times found in literature [7, 5, 8]. It is indicated that performing the tasks with the Voodoo-doll was similar in intuition. The reaction time of the final task ('Give a perturbation to the left') was longer than for the other tasks. The cause for this may be the wording of the task giving, which might be harder to process for the participant. Another cause could be the fact that the button for activating the motion detection is located on the back of the Voodoo-doll. Visibility might had an influence on the reaction time of the participant. Additionally, the errors were only made during the final tasks. Both reaction time and errors made suggest that applying the perturbation with the Voodoo-doll is not intuitive.

There does not appear to be a pattern between the orientation of the different participants. This suggests that the hand movements for certain directions are subjective.

6.1.1. Limitations

The set up of the experiment was a promising starting point for first prototype testing of the new interface design. However, there were some limitations. The design of the prototype required a cable to connect the IMU to the laptop with the Python script. Limiting the range of motion while holding the interface and the possibility to use it from a distance.

The intended user for the interface of the GyBAR are physical therapists. Within this research it was not possible to recruit physical therapists, due to budget. Therefore, it was decided to recruit participants that did have knowledge on the intended use applications but are not working as physical therapists. This could influence the results since they are not used to treating patients daily. Although familiar with the theory behind the intended application, lack of experience results in lower intuition in regards to applying the treatment.

Since the Voodoo-doll was not connected to the GyBAR, virtual feedback was given. This gave a limited version of the real life feedback. The motion detection should only be activated when the respective button is pressed, in order to avoid unintentional movements being applied to the patient. The motion detection simulation did not stop working when the button was released, due to technical difficulties with the Python script. Therefore, it was unclear for the participants that the button should be continuously pressed.

The reaction time was measured to determine whether the interface was intuitive to use. It was measured by taking the start time from when a task was given and subtract it from the timestamp when the participant performed the right action. The researcher pressed the key when they started speaking, which gave a timestamp to the start time. However, this is prone to human error, therefore it should be taken in account that the reaction time data has a low accuracy. No research has been found where the reaction time of similar devices was recorded. Instead the reaction time was compared to results of simple choice reaction time tests, where a stimulus was used for a specific button and no consecutive choices were required.

6.1.2. Recommendations

The Voodoo Doll shows potential for increasing the perceived ease of use of the GyBAR. It is therefore recommended to continue development of the interface. The prototype can be improved by doing research with the Voodoo doll connected to the actual GyBAR. This will give the user more realistic feedback. It is important to implement the complete TAM, where perceived ease of use and perceived usefulness are indicators for the acceptance of new technology. The physical design can be improved by integrating the IMU inside the casing of the Voodoo doll and making it wireless.

Another improvement of the Voodoo-doll is to calibrate the orientation for each user. The results of this research indicate that each participant has their own perception of ”left”, therefore a personal approach is advised. This can be achieved by letting the participant perform movements in all directions and adjusting the system to the participant.

Further research should be done on the configuration of the buttons on the Voodoo-doll. One of the main objectives could be whether the visibility of the buttons would have an influence on the reaction time.

Within this research only the rehabilitation session was analysed, by focusing on the controller. The interface also consists of a display, which is used for setting up a training program and analysing the sessions. It is advised to expand the interface by looking at all components described in chapter 3, according to the formulated requirements.

For a more realistic user test, the experiments should be conducted with actual physical therapists as participants. They should be familiar with the procedures used during the different use scenarios.

6.2. Conclusion

The goal of this research was to develop an interface for the GyBAR, with a high perceived ease of use to stimulate physical therapists to use the device optimally. It can be concluded that the Voodoo-doll is a promising prototype to serve as an interface for the GyBAR. The perceived ease of use is excellent. Furthermore, this research has highlighted several aspects in which further research can be conducted to further develop and optimize the Voodoo-doll .

Bibliography

[1] *Accessoires | Wii | Nintendo | Wii | Nintendo*. URL: https://www.nintendo.nl/Wii/Accessoires/Accessoires-Wii-Nintendo-626430.html.

[2] Frances A. Batchelor et al. “Falls after Stroke”. In: *International Journal of Stroke* 7.6 (2012), pp. 482–490. DOI: 10.1111/j.1747-4949.2012.00796.x.

[3] Annemiek van Boeijen, Jaap Daalhuizen, and Jelle Zijlstra. *Brainstorming & Brainwriting*. 2nd ed. BIS, 2017, p. 168.

[4] John Brooke. “SUS: A quick and dirty usability scale”. In: *Usability Eval. Ind.* 189 (Nov. 1995).

[5] Viviane Freire Bueno and Luiz Eduardo Ribeiro do Valle. “Effects of visual and auditory stimuli in a choice reaction time task.” In: *Psychology & Neuroscience* 5 (2 2013), p. 199. ISSN: 1983-3288. DOI: 10.3922/J.PSNS.2012.2.10. URL: /fulltext/2012-35338-010.html.

[6] *Falls*. URL: https://www.who.int/news-room/fact-sheets/detail/falls. (accessed: 16.01.2022).

[7] Pritesh Hariprasad Gandhi et al. “A Comparative Study of Simple Auditory Reaction Time in Blind (Congenitally) and Sighted Subjects”. In: *Indian Journal of Psychological Medicine* 35 (2014). DOI: 10.4103/0253-7176.119486. URL: www.ijpm.info.

[8] Tejas P. Ghuntla et al. “Effect of number of stimuli on auditory reaction time in healthy subjects of Bhavnagar region”. In: *Indian Journal of Otology* 19 (4 Oct. 2013), p. 179. ISSN: 0971-7749. DOI: 10.4103/0971-7749.124510. URL: https://www.indianjotol.org/article.asp?issn=0971-7749;year=2013;volume=19;issue=4;spage=179;epage=181;aulast=Ghuntla%20https://www.indianjotol.org/article.asp?issn=0971-7749;year=2013;volume=19;issue=4;spage=179;epage=181;aulast=Ghuntla;type=0.

[9] Paul j. Hu et al. “Examining the Technology Acceptance Model Using Physician Acceptance of Telemedicine Technology”. In: *Journal of Management Information Systems* 16 (2 1999), pp. 91–112. URL: https://sci-hub.se/10.1080/07421222.1999.11518247.

[10] Odai Y. Khasawneh. “Technophobia: Examining its hidden factors and defining it”. In: *Technology in Society* 54 (Aug. 2018), pp. 93–100. ISSN: 0160-791X. DOI: 10.1016/J.TECHSOC.2018.03.008.

[11] Urška Lah, James R Lewis, and Boštjan Šumak. “Perceived Usability and the Modified Technology Acceptance Model”. In: *International Journal of Human-Computer Interaction* 36 (13 2020), pp. 1216–1230. DOI: 10.1080/10447318.2020.1727262. URL: https://www.tandfonline.com/action/journalInformation?journalCode=hihc20.

[12] D. Lemus et al. “Controller Synthesis and Clinical Exploration of Wearable Gyroscopic Actuators to Support Human Balance”. In: *Scientific Reports* 10 (2020), p. 10412. DOI: 10.1038/s41598-020-66760-w.

[13] Forough Madehkhaksar et al. “The effects of unexpected mechanical perturbations during treadmill walking on spatiotemporal gait parameters, and the dynamic stability measures by which to quantify postural response”. In: *PLOS ONE* 13 (4 Apr. 2018), e0195902. ISSN: 1932-6203. DOI: 10.1371/JOURNAL.PONE.0195902.

[14] Avril Mansfield et al. “Does Perturbation-Based Balance Training Prevent Falls? Systematic Review and Meta-Analysis of Preliminary Randomized Controlled Trials”. In: *Physical Therapy* 95 (5 May 2015), pp. 700–709. ISSN: 0031-9023. DOI: 10.2522/PTJ.20140090.

[15] D Martelli, J Kang, and S K Agrawal. “A Perturbation-based Gait Training with Multidirectional Waist-Pulls Generalizes to Split-Belt Treadmill Slips”. In: (2018). DOI: 10.1109/BIOROB.2018.8487618.

[16] Christopher McCrum et al. “A systematic review of gait perturbation paradigms for improving reactive stepping responses and falls risk among healthy older adults”. In: *European Review of Aging and Physical Activity* 14 (1 Mar. 2017), pp. 1–11. ISSN: 18137253. DOI: `10.1186/S11556-017-0173-7/TABLES/3`.

[17] *Nano 33 IoT | Arduino Documentation | Arduino Documentation*. URL: `https://docs.arduino.cc/hardware/nano-33-iot`.

[18] Nawel Ouendi et al. “The rehabilitation robot: factors influencing its use, advantages and limitations in clinical rehabilitation”. In: *Disability and Rehabilitation: Assistive Technology* (2022). DOI: `10.1080/17483107.2022.2107095`.

[19] N.F.M. Roozenburg and J. Eekels. *Productontwerpen, structuur en methoden*. 2nd ed. Lemma, 1998. ISBN: 9789051897067.

[20] Jeff Sauro. “SUStisfied? Little-Known System Usability Scale Facts User Experience Magazine”. In: *User Experience Magazine* 10 (3 Aug. 2011). URL: `https://uxpamagazine.org/sustified/`.

[21] Nicola Smania et al. “Effect of balance training on postural instability in patients with idiopathic Parkinson’s disease”. In: *Neurorehabilitation and neural repair* 24 (9 Nov. 2010), pp. 826–834. ISSN: 1552-6844. DOI: `10.1177/1545968310376057`. URL: `https://pubmed-ncbi-nlm-nih-gov.tudelft.idm.oclc.org/21045119/`.

[22] *Step safely: strategies for preventing and managing falls across the life-course*. World Health Organization, 2021.

[23] Bhatt Tanvi, Yang Feng, and Pai Yi-Chung. “Learning to resist gait-slip falls: long-term retention in community-dwelling older adults”. In: *Archives of physical medicine and rehabilitation* 93 (4 Apr. 2012), pp. 557–564. ISSN: 1532-821X. DOI: `10.1016/J.APMR.2011.10.027`. URL: `https://pubmed-ncbi-nlm-nih-gov.tudelft.idm.oclc.org/22341989/`.

[24] David J. Thurman, Judy A. Stevens, and Jaya K. Rao. “Practice Parameter: Assessing patients in a neurology practice for risk of falls (an evidence-based review)”. In: *Neurology* 70.6 (2008), pp. 473–479. ISSN: 0028-3878. DOI: `10.1212/01.wnl.0000299085.18976.20`. eprint: `https://n.neurology.org/content/70/6/473.full.pdf`. URL: `https://n.neurology.org/content/70/6/473`.

Part III

Appendices

A
Brainstorm

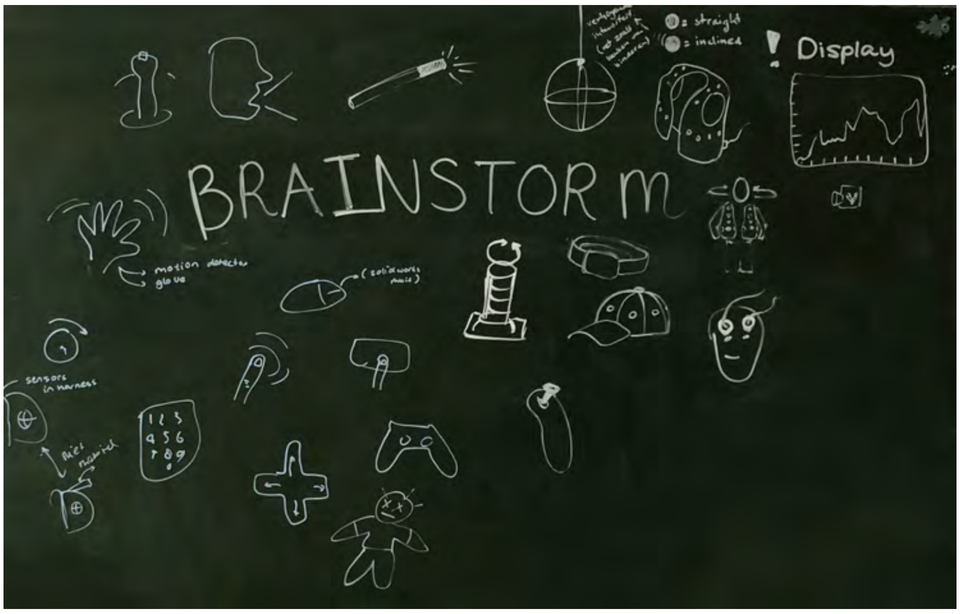


Figure A.1: Brainstorm ideas.

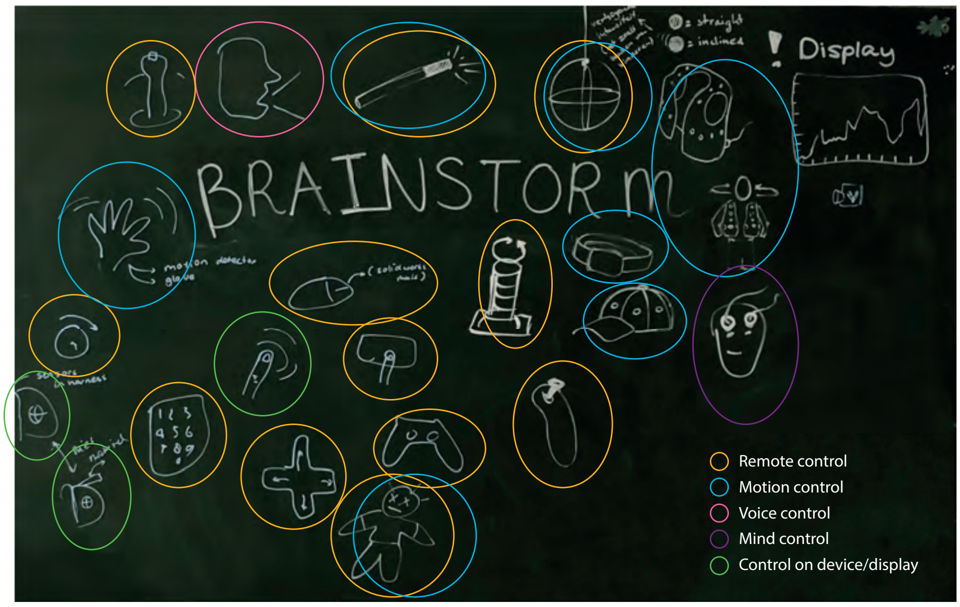


Figure A.2: Brainstorm ideas clustered into five categories.

B
Python code

```python
import pygame
import sys
import datetime
from IMU import Simulation  #import program that reads accelerometer data

pygame.init() #initialize pygame

# Define colors and display settings
WHITE = (255, 255, 255)
BLUE = (0, 0, 255)
GREEN = (0, 201, 87)

font = pygame.font.Font(pygame.font.get_default_font(), 36)                    #define font displayed on screen
screen = pygame.display.set_mode((0,0),pygame.FULLSCREEN)                      #set up of full screen display
homescreen = font.render('GyBAR Interface: select mode', True,BLUE)            #Homescreen text when system is started
screen.fill(WHITE)                                                              #Homescreen background
screen.blit(homescreen, homescreen.get_rect(center=screen.get_rect().center)) #Center text on screen
pygame.display.update()                                                         #update screen to display homescreen

#Connect Wii Remote using the joystick module
try:
    j=pygame.joystick.Joystick(0)
except pygame.error:
    print("Joystick not connected.")

j.init() #initialize joystick module

#Define text displayed on screen per mode
b0=font.render('intensity increased',True,BLUE)
b1=font.render('intensity decreased',True,BLUE)
b2=font.render('play/pause',True,BLUE)
b3=font.render('Motion detection activated',True,BLUE)
b4=font.render('Manual perturbation mode selected',True,BLUE)
```

```python
b5=font.render('Balance assistance mode selected',True,BLUE)
b6=font.render('Automatic perturbation mode selected',True,BLUE)
b7=font.render('Balance assistance mode activated',True,WHITE)
b8=font.render('Automatic perturbation mode activated',True,WHITE)
b9=font.render('Manual perturbation mode activated',True,WHITE)

#Initial modes
mode=0  #initial mode wii controller
q=0     #Initial mode question (no question asked)

#Add timestamp to all printed actions
old_f = sys.stdout
class F:
    def write(self, x):
        old_f.write(x.replace("\n", " [%s]\n" % str(datetime.datetime.now())))
sys.stdout = F()

# Generate a pygame.JOYBUTTONDOWN event when a button is pressed and pygame.JOYBUTTONUP event when released
try:
    while True:
        events = pygame.event.get()
        for event in events:
            if event.type == pygame.JOYBUTTONDOWN and j.get_button(0): #Shows intensity increased on display when Wii Remote "1"-button is pressed and prints this in terminal with timestamp.
                screen.fill(WHITE)
                screen.blit(b0, b0.get_rect(center = screen.get_rect().center))
                mode=0
                pygame.display.update()
                print('Intensity increased pressed')
            elif event.type == pygame.KEYDOWN and q == 0:               #Prints "start question" with timestamp in terminal when a key is pressed, when q=0 which indicates that the question had not been started before.
                print("Start question")
                q = 1
            elif event.type == pygame.KEYDOWN and q == 1:             #Prints "end question" with timestamp in terminal when a key is pressed, when q=1 which indicates that the question had been started before.
                print("End question")
                q = 0
            elif event.type == pygame.JOYBUTTONDOWN and j.get_button(1):  #Shows intensity decreased on display when Wii Remote "2"-button is pressed and prints this in terminal with timestamp.
                screen.fill(WHITE)
                screen.blit(b1, b1.get_rect(center = screen.get_rect().center))
                pygame.display.update()
                mode=0
                print('Intensity decreased pressed')
            elif event.type == pygame.JOYBUTTONDOWN and j.get_button(3) and mode == 33: #When "B"-button is pressed while manual perturbation mode is activatedc, the cube simulation that shows the orientation from the IMU script is activated.
                print('Motion detection pressed')
                Simulation().run()
                mode=333
            elif event.type == pygame.JOYBUTTONDOWN and j.get_button(4): #Shows manual perturbation mode selected on display when "home"-button is pressed and prints in terminal with timestamp
                screen.fill(WHITE)
                screen.blit(b4, b4.get_rect(center = screen.get_rect().center))
                pygame.display.update()
                mode=3
                print('Manual perturbation mode pressed')
            elif event.type == pygame.JOYBUTTONDOWN and j.get_button(5):  #Shows balance assistance mode selected on display when "-"-button is pressed and prints in terminal with timestamp
                screen.fill(WHITE)
                screen.blit(b5, b5.get_rect(center = screen.get_rect().center))
                pygame.display.update()
                mode = 1
```

```python
                print('Balance assistance mode pressed')
            elif event.type == pygame.JOYBUTTONDOWN and j.get_button(6):
                screen.fill(WHITE)
                screen.blit(b6, b6.get_rect(center = screen.get_rect().center)) #Shows automatic perturbation mode selected on display when "+"-button is pressed and prints in terminal with timestamp
                pygame.display.update()
                mode = 2
                print('Automatic perturbation mode pressed')
            elif event.type == pygame.JOYBUTTONDOWN and j.get_button(2) and mode == 1: #Shows green screen with balance assistance mode started when "play"-button is pressed, while in balance assistance mode
                screen.fill(GREEN)
                screen.blit(b7, b7.get_rect(center = screen.get_rect().center))
                pygame.display.update()
                mode=11
                print('Play/pause pressed')
            elif event.type == pygame.JOYBUTTONDOWN and j.get_button(2) and mode == 11:  #Goes back to balance assistance mode selcted display while  "play"-button is pressed, while in activated balance assistance mode
                screen.fill(WHITE)
                screen.blit(b5, b5.get_rect(center=screen.get_rect().center))
                pygame.display.update()
                mode = 1
                print('Play/pause pressed')
            elif event.type == pygame.JOYBUTTONDOWN and j.get_button(2) and mode==2:  #Shows green screen with automatic perturbation mode started when "play"-button is pressed, while in automatic perturbation mode
                screen.fill(GREEN)
                screen.blit(b8, b8.get_rect(center = screen.get_rect().center))
                pygame.display.update()
                mode=22
                print('Play/pause pressed')
            elif event.type == pygame.JOYBUTTONDOWN and j.get_button(2) and mode==22: #Goes back to automatic
```

```
perturbation mode selcted display while  "play"-button is pressed, while in activated automatic perturbation mode
                screen.fill(WHITE)
                screen.blit(b6, b6.get_rect(center = screen.get_rect().center))
                pygame.display.update()
                mode=2
                print('Play/pause pressed')
            elif event.type == pygame.JOYBUTTONDOWN and j.get_button(2) and mode == 3:  #Shows green screen with manual perturbation mode started when "play"-button is pressed, while in manual perturbation mode
                screen.fill(GREEN)
                screen.blit(b9, b9.get_rect(center=screen.get_rect().center))
                pygame.display.update()
                mode = 33
                print('Play/pause pressed')
            elif event.type == pygame.JOYBUTTONDOWN and j.get_button(2) and mode == 33:  #Goes back to manual perturbation mode selcted display while  "play"-button is pressed, while in activated manual perturbation mode
                screen.fill(WHITE)
                screen.blit(b4, b4.get_rect(center=screen.get_rect().center))
                pygame.display.update()
                mode = 3
                print('Play/pause pressed')






            else:
                pass

except KeyboardInterrupt: #Quit system when key is pressed
    print("EXITING NOW")
    j.quit()

```

```
import math
import sys
import pygame
import serial
from operator import itemgetter

def find(s, ch):
    return [i for i, char in enumerate(s) if char == ch]


#connect accelerometer through the used USB port
ser = serial.Serial('COM4', 9600, stopbits=1, bytesize=8)


class Point3D:
    def __init__(self, x=0, y=0, z=0):
        self.x, self.y, self.z = float(x), float(y), float(z)

    def rotateX(self, angle):
        """rotates the point around the x-axis by the given angle in degrees."""
        rad = angle * math.pi / 180
        cosa = math.cos(rad)
        sina = math.sin(rad)
        y = self.y * cosa - self.z * sina
        z = self.y * sina + self.z * cosa
        return Point3D(self.x, y, z)

    def rotateY(self, angle):
        """rotates the point around the y-axis by the given angle in degrees."""
        rad = angle * math.pi / 180
        cosa = math.cos(rad)
        sina = math.sin(rad)
        z = self.z * cosa - self.x * sina
        x = self.z * sina + self.x * cosa
        return Point3D(x, self.y, z)

    def rotateZ(self, angle):
        """rotates the point around the z-axis by the given angle in degrees."""
        rad = angle * math.pi / 180
        cosa = math.cos(rad)
```

```python
        sina = math.sin(rad)
        x = self.x * cosa - self.y * sina
        y = self.x * sina + self.y * cosa
        return Point3D(x, y, self.z)

    def project(self, win_width, win_height, fov, viewer_distance):
        """Transforms this 3d point to 2d using a perspective projection"""
        factor = fov / (viewer_distance + self.z)
        x = self.x * factor + win_width / 2
        y = -self.y * factor + win_height / 2
        return Point3D(x, y, 1)


class Simulation: #Set up simulation
    def __init__(self):
        pygame.init() #intialize pygame

        self.screen = pygame.display.set_mode((0, 0), pygame.FULLSCREEN) #set up display
        pygame.display.set_caption("3D Wireframe Cube Simulation")
        self.clock = pygame.time.Clock()

        self.vertices = [
            Point3D(-1, 1, -1),
            Point3D(1, 1, -1),
            Point3D(1, -1, -1),
            Point3D(-1, -1, -1),
            Point3D(-1, 1, 1),
            Point3D(1, 1, 1),
            Point3D(1, -1, 1),
            Point3D(-1, -1, 1)
        ]

        # define the  vertices that compose each of the 6 faces, these numbers are indices to the vertices list
        # defined above
        self.faces = [ (5, 4, 7, 6),(3, 2, 6, 7), (4, 0, 3, 7),(1, 5, 6, 2), (0, 4, 5, 1), (0, 1, 2, 3)]

        # define colors for each face
        self.color = [(255, 0, 255), (255, 0, 0), (0, 255
, 0), (0, 0, 255), (0, 255, 255), (255, 255, 0)]

        self.angleX, self.angleY, self.angleZ = 0, 0, 0

    def run(self):

        while (1):
            for event in pygame.event.get():
                if event.type == pygame.QUIT: #quit system when pygame is quit
                    sys.exit()
                elif event.type == pygame.KEYDOWN: #quit system when keyboard key is pressed while in simulation
                    sys.exit()
                elif event.type == pygame.JOYBUTTONUP:  # Print when "B"-button is released
                    print('Button released')
                elif event.type == pygame.JOYBUTTONDOWN: #Print when "B"-button is pressed
                    print('Motion button pressed')
            self.clock.tick(50)
            self.screen.fill((255, 255, 255))

            # will hold transformed vertices.
            t = []

            for v in self.vertices:
                # rotate the point around x-axis, then around y-axis, and finally around z axis
                r = v.rotateX(self.angleX).rotateY(self.angleY).rotateZ(self.angleZ)
                # transform the point from 3d to 2d
                p = r.project(self.screen.get_width(), self.screen.get_height(), 256, 4)
                # put the point in the list of transformed vertices
                t.append(p)

            avg_z = []
            i = 0

            for f in self.faces:
                z = (t[f[0]].z + t[f[1]].z + t[f[2]].z + t[f[3]].z) / 4.0
                avg_z.append([i, z])
                i = i + 1

            for tmp in sorted(avg_z, key=itemgetter(1), reverse=True):
                face_index = tmp[0]
                f = self.faces[face_index]
                pointlist = [(t[f[0]].x, t[f[0]].y), (t[f[1]].x, t[f[1]].y),
                             (t[f[1]].x, t[f[1]].y), (t[f[2]].x, t[f[2]].y),
                             (t[f[2]].x, t[f[2]].y), (t[f[3]].x, t[f[3]].y),
                             (t[f[3]].x, t[f[3]].y), (t[f[0]].x, t[f[0]].y)]
                pygame.draw.polygon(self.screen, self.color[face_index], pointlist)
            pygame.display.flip()

            # Read IMU data into code
            s = str(ser.readline())
            comma = find(s, ',')
            bpos = s.find('b')
            posx = s[bpos + 2:comma[0]]
            posy = s[comma[0] + 1:comma[1]]
            posz = s[comma[1] + 1:comma[2]]


            # Set angles from IMU data
            self.angleX = float(posx)
            self.angleY = float(posy)
            self.angleZ = float(posz)

            print(posx,posy)

            pygame.display.flip()



if __name__ == "__main__":
    Simulation().run()
    ser.close()
```

C
Arduino code

```
/*
  Arduino LSM6DS3 - Simple Accelerometer

  This script reads the acceleration values from the LSM6DS3
  sensor and continuously prints them to the Serial Monitor
  or Serial Plotter. The standard Arduino code for the LSMDS3 module is used as a starting
    point and built upon~\cite{arduinocode}

*/

#include <Arduino_LSM6DS3.h>
float x, y, z;
int degreesX = 0;
int degreesY = 0;


void setup() {
  Serial.begin(9600);
  while (!Serial);

  if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");

    while (1);
  }

  Serial.print("Accelerometer sample rate = ");
  Serial.print(IMU.accelerationSampleRate());
  Serial.println(" Hz");
}
void loop() {
  z = 10 * z;
  if (IMU.accelerationAvailable()) {
    IMU.readAcceleration(x, y, z);

  }

  x = 100 * x;
  degreesX = map(x, -100, 97, -90, 90);
  y = 100 * y;
  degreesY = map(y, -100, 97, -90, 90);

  Serial.print(degreesX);
  Serial.print(",");
  Serial.print(degreesY);
  Serial.print(",");
  Serial.print(z);
  Serial.println(",");
  delay(200);
}
```

Listing C.1: Blink.ino

D
SUS questionnaire given to the participants

Usability of the GyBAR with the new interface
The system usability scale (SUS) is used to determine your perspective on the usability of the interface for the GyBAR. Please fill in your immediate response to each item, rather than thinking about items for a long time.

	Strongly Disagree				Strongly Agree
	1	2	3	4	5
1. I think that I would like to use this system frequently.					
2. I found the system unnecessarily complex.					
3. I thought the system was easy to use.					
4. I think that I would need the support of a technical person to be able to use this system.					
5. I found the various functions in this system were well integrated.					
6. I thought there was too much inconsistency in this system.					
7. I would imagine that most people would learn to use this system very quickly.					
8. I found the system very cumbersome to use.					
9. I felt very confident using the system.					
10. I needed to learn a lot of things before I could get going with this system.					

Thank you for participating!

E
Consent form

You are being invited to participate in a research study titled "Creating an interface for a balance assistive device" This study is being done by Daniela Estrada, a Biomedical Engineering student from the TU Delft.

The purpose of this research study is to determine the perceived ease of use of the interface controlling a balance assistance device. It will take you approximately 30 minutes to complete.
We will ask you to perform certain tasks using the interface. Afterwards we will ask you to fill in a questionnaire on how you perceive the ease of use of the interface.
Your movements while interacting with the interface will be recorded through a program which tracks the buttons you press and the orientation of the device.

If you are in agreement, the data of your interaction with the interface and your answers on the questionnaire will be analysed. Conclusions will be drawn and described in Daniela Estrada's Master thesis. The thesis will be uploaded to the TU Delft repository and will be available to the public.
All data are anonymized, you will receive a randomized participant number that corresponds with your data. The researcher is not aware of which participant number is connected to which participant. If you wish to withdraw from the study, you may notify the researcher at any time before final publication. The data corresponding to the participant number, which you will provide, will then be deleted.

Your participation in this study is entirely voluntary and you can withdraw at any time. You are free to omit any questions.

Any comments can be directed to B. Sterke.

Thank you for your participation.
Daniela Estrada

PLEASE TICK THE APPROPRIATE BOXES	Yes	No
A: GENERAL AGREEMENT – RESEARCH GOALS, PARTICPANT TASKS AND VOLUNTARY PARTICIPATION		
1. I have read and understood the study information dated 30/08/2022, or it has been read to me. I have been able to ask questions about the study and my questions have been answered to my satisfaction.	☐	☐
2. I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.	☐	☐
3. I understand that taking part in the study involves: performing timed tasks with a new interface and filling out a questionnaire on perceived ease of use.	☐	☐
4. I understand that the study will end on approximately December 31st.	☐	☐
B: POTENTIAL RISKS OF PARTICIPATING (INCLUDING DATA PROTECTION)		
5. I understand that the data collected will correspond to a random participant number and cannot be traced back to me.	☐	☐
6. I understand that no personal information about me that can identify me, such as my name and contact information, will be collected.	☐	☐
C: RESEARCH PUBLICATION, DISSEMINATION AND APPLICATION		

PLEASE TICK THE APPROPRIATE BOXES	Yes	No
7. I understand that after the research study the de-identified information I provide will be used for a Master Thesis.	☐	☐
8. I agree that my responses, views or other input can be quoted anonymously in research output.	☐	☐
D: (LONGTERM) DATA STORAGE, ACCESS AND REUSE		
9. I give permission for the de-identified questionnaire answered that I provide to be archived in the TU Delft education repository so it can be used for future research and learning.	☐	☐
10. I understand that access to this repository is open to the public.	☐	☐

Signatures

_____________________________ _____________________________ _________

Name of participant [printed] Signature Date

I, as researcher, have accurately read out the information sheet to the potential participant and, to the best of my ability, ensured that the participant understands to what they are freely consenting.

Daniela Estrada

___________________________ ____________________ _________

Researcher name Signature Date

Study contact details for further information: *Daniela Estrada*

F Experiment Protocol

1. Participants are given the consent form and are asked to read and sign it if they agree with it.
2. Participant is given the interface while being explained what the intended applications and the functions of the interface are.
3. The interface is put down and the participant is asked about personal experiences with interfaces to distract them.
4. The researcher asks the participant to perform certain tasks. At the beginning and end of each of the following tasks the researcher presses a button on the keyboard.
    - Select balance assistance mode
    - Increase intensity
    - Start automatic perturbation mode
    - Stop automatic perturbation mode
    - Start manual perturbation mode
    - Apply a perturbation to the left
5. Participant fills in SUS questionnaire
6. Python data is put into excel file to prepare for processing.

G
Matlab Code SUS-score

```
clear all; clc; close all;

% import xls file
SUS = xlsread('SUS_scores.xlsx');

% Define variables
scores = SUS(:,2);  %SUS score

% Find mean and standard deviation
M = mean(scores);   % mean
S = std(scores);    % standard deviation

co1 = '#6F1D77';
co2 = '#EF60A3';
co3 = '#A50034';
co4 = '#E03C31';
co5 = '#EC6842';
co6 = '#FFB81C';
co7 = '#6CC24A';
co8 = '#009B77';

cotu = "#00A6D6";
cotu2 = '#0C2340';


% b = bar(M);
% b.FaceColor = cotu;
% b.EdgeColor = cotu; hold on
boxplot(scores); hold on
plot(1,scores(1,:),'o','MarkerEdgeColor',co1,'MarkerFaceColor',co1); hold on
plot(1,scores(2,:),'o','MarkerEdgeColor',co2,'MarkerFaceColor',co2); hold on
plot(1,scores(3,:),'o','MarkerEdgeColor',co3,'MarkerFaceColor',co3); hold on
plot(1,scores(4,:),'o','MarkerEdgeColor',co4,'MarkerFaceColor',co4); hold on
plot(1,scores(5,:),'o','MarkerEdgeColor',co5,'MarkerFaceColor',co5); hold on
plot(1,scores(6,:),'o','MarkerEdgeColor',co6,'MarkerFaceColor',co6); hold on
plot(1,scores(7,:),'o','MarkerEdgeColor',co7,'MarkerFaceColor',co7); hold on
plot(1,scores(8,:),'o','MarkerEdgeColor',co8,'MarkerFaceColor',co8); hold on
yline(68,'-b','Average*');
ylim([0 100])
ylabel('SUS score')
legend('Participant 1','Participant 2','Participant 3','Participant 5','Participant 6','
    Participant 7','Participant 8')
xticklabels({'Mean'})
title('System Usability Scale scores')
```

appendix/SUS_scores.m

H

Matlab Code Reaction time

```matlab
clc; clear all; close all;

% Import participant data
[numbers, TEXT1, everything] = xlsread('220890.xlsx','Python');
P1 = TEXT1;
[numbers, TEXT2, everything] = xlsread('170730.xlsx','Python');
P2 = TEXT2;
[numbers, TEXT3, everything] = xlsread('210730.xlsx','Python');
P3 = TEXT3;
[numbers, TEXT4, everything] = xlsread('120352.xlsx','Python');
P4 = TEXT4;
[numbers, TEXT5, everything] = xlsread('251060.xlsx','Python');
P5 = TEXT5;
[numbers, TEXT6, everything] = xlsread('260782.xlsx','Python');
P6 = TEXT6;
[numbers, TEXT7, everything] = xlsread('160496.xlsx','Python');
P7 = TEXT7;
[numbers, TEXT8, everything] = xlsread('170117.xlsx','Python');
P8 = TEXT8;

%% Find the time of the start of the questions and the completion of the assignments.Convert
    time to duration in seconds.

%Find start time of the questions
[row11,column]= find(ismember(P1,'Start question')); %Participant 1
SQ1 = P1(row11,2);
[row22,column]= find(ismember(P2,'Start question')); %Participant 2
SQ2 = P2(row22,2);
[row33,column]= find(ismember(P3,'Start question')); %Participant 3
SQ3 = P3(row33,2);
[row44,column]= find(ismember(P4,'Start question')); %Participant 4
SQ4 = P4(row44,2);
[row55,column]= find(ismember(P5,'Start question')); %Participant 5
SQ5 = P5(row55,2);
[row66,column]= find(ismember(P6,'Start question')); %Participant 6
SQ6 = P6(row66,2);
[row77,column]= find(ismember(P7,'Start question')); %Participant 7
SQ7 = P7(row77,2);
[row88,column]= find(ismember(P8,'Start question')); %Participant 8
SQ8 = P8(row88,2);

%Find first correct reaction (first right button of sequence to completion)
[row11,column]= find(ismember(P1,'first')); %Participant 1
F1 = P1(row11,2);
[row22,column]= find(ismember(P2,'first')); %Participant 2
F2 = P2(row22,2);
[row33,column]= find(ismember(P3,'first')); %Participant 3
F3 = P3(row33,2);
[row44,column]= find(ismember(P4,'first')); %Participant 4
F4 = P4(row44,2);
[row55,column]= find(ismember(P5,'first')); %Participant 5
F5 = P5(row55,2);
[row66,column]= find(ismember(P6,'first')); %Participant 6
F6 = P6(row66,2);
[row77,column]= find(ismember(P7,'first')); %Participant 7
F7 = P7(row77,2);
[row88,column]= find(ismember(P8,'first')); %Participant 8
```

```
F8 = P8(row88,2);


%Find time of completion of assignments
[row11,column]= find(ismember(P1,'complete')); %Participant 1
C1 = P1(row11,2);
[row22,column]= find(ismember(P2,'complete')); %Participant 2
C2 = P2(row22,2);
[row33,column]= find(ismember(P3,'complete')); %Participant 3
C3 = P3(row33,2);
[row44,column]= find(ismember(P4,'complete')); %Participant 4
C4 = P4(row44,2);
[row55,column]= find(ismember(P5,'complete')); %Participant 5
C5 = P5(row55,2);
[row66,column]= find(ismember(P6,'complete')); %Participant 6
C6 = P6(row66,2);
[row77,column]= find(ismember(P7,'complete')); %Participant 7
C7 = P7(row77,2);
[row88,column]= find(ismember(P8,'complete')); %Participant 8
C8 = P8(row88,2);

%Convert time into duration in seconds
F = 'hh:mm:ss.SSSSSS'; %time format

%participant 1
du1 = duration(SQ1,'InputFormat',F,'Format',F);
format longg
SQ1 = seconds(du1);
du11 = duration(C1,'InputFormat',F,'Format',F);
format longg
C1 = seconds(du11);
du111 = duration(F1,'InputFormat',F,'Format',F);
format longg
F1 = seconds(du111);

%participant 2
du2 = duration(SQ2,'InputFormat',F,'Format',F);
format longg
SQ2 = seconds(du2);
du22 = duration(C2,'InputFormat',F,'Format',F);
format longg
C2 = seconds(du22);
du222 = duration(F2,'InputFormat',F,'Format',F);
format longg
F2 = seconds(du222);

%participant 3
du3 = duration(SQ3,'InputFormat',F,'Format',F);
format longg
SQ3 = seconds(du3);
du33 = duration(C3,'InputFormat',F,'Format',F);
format longg
C3 = seconds(du33);
du333 = duration(F3,'InputFormat',F,'Format',F);
format longg
F3 = seconds(du333);

%participant 4
du4 = duration(SQ4,'InputFormat',F,'Format',F);
format longg
SQ4 = seconds(du4);
du44 = duration(C4,'InputFormat',F,'Format',F);
format longg
C4 = seconds(du44);
du444 = duration(F4,'InputFormat',F,'Format',F);
format longg
F4 = seconds(du444);

%participant 5
du5 = duration(SQ5,'InputFormat',F,'Format',F);
format longg
```

```
SQ5 = seconds(du5);
du55 = duration(C5,'InputFormat',F,'Format',F);
format longg
C5 = seconds(du55);
du555 = duration(F5,'InputFormat',F,'Format',F);
format longg
F5 = seconds(du555);

%participant 6
du6 = duration(SQ6,'InputFormat',F,'Format',F);
format longg
SQ6 = seconds(du6);
du66 = duration(C6,'InputFormat',F,'Format',F);
format longg
C6 = seconds(du66);
du666 = duration(F6,'InputFormat',F,'Format',F);
format longg
F6 = seconds(du666);

%participant 7
du7 = duration(SQ7,'InputFormat',F,'Format',F);
format longg
SQ7 = seconds(du7);
du77 = duration(C7,'InputFormat',F,'Format',F);
format longg
C7 = seconds(du77);
du777 = duration(F7,'InputFormat',F,'Format',F);
format longg
F7 = seconds(du777);

%participant 8
du8 = duration(SQ8,'InputFormat',F,'Format',F);
format longg
SQ8 = seconds(du8);
du88 = duration(C8,'InputFormat',F,'Format',F);
format longg
C8 = seconds(du88);
du888 = duration(F8,'InputFormat',F,'Format',F);
format longg
F8 = seconds(du888);


%% Find reaction time & completion time

%Find first reaction time (substract start question time (SQ) from reaction
%time (F)
RT1 = F1 - SQ1;
RT2 = F2 - SQ2;
RT3 = F3 - SQ3;
RT4 = F4 - SQ4;
RT5 = F5 - SQ5;
RT6 = F6 - SQ6;
RT7 = F7 - SQ7;
RT8 = F8 - SQ8;

%Find completion time (substract start question time (SQ) from completion
%time (C)

RT11 = C1 - SQ1;
RT22 = C2 - SQ2;
RT33 = C3 - SQ3;
RT44 = C4 - SQ4;
RT55 = C5 - SQ5;
RT66 = C6 - SQ6;
RT77 = C7 - SQ7;
RT88 = C8 - SQ8;

%% Find mean reaction time per question

RT = [RT1 RT2 RT3 RT4 RT5 RT6 RT7 RT8];
RT = RT.';
```

```
M = mean(RT);
S = std(RT);

CT = [RT11 RT22 RT33 RT44 RT55 RT66 RT77 RT88];
CT = CT.';
MC = mean(CT);
SC = std(CT);
%% Plot reaction time
%Colors for the plot
co1 = '#6F1D77';
co2 = '#EF60A3';
co3 = '#A50034';
co4 = '#E03C31';
co5 = '#EC6842';
co6 = '#FFB81C';
co7 = '#6CC24A';
co8 = '#009B77';

cotu = '#00A6D6';
cotu2 = '#0C2340';

x=1:1:7;


figure
b = bar(x,M);
b.FaceColor = cotu;
b.EdgeColor = cotu; hold on
errorbar(x,M,S,"LineStyle","none",'Color',cotu2);  hold on
plot(x,RT(1,:),'o','MarkerEdgeColor',co1,'MarkerFaceColor',co1);
plot(x,RT(2,:),'o','MarkerEdgeColor',co2,'MarkerFaceColor',co2);
plot(x,RT(3,:),'o','MarkerEdgeColor',co3,'MarkerFaceColor',co3);
plot(x,RT(4,:),'o','MarkerEdgeColor',co4,'MarkerFaceColor',co4);
plot(x,RT(5,:),'o','MarkerEdgeColor',co5,'MarkerFaceColor',co5);
plot(x,RT(6,:),'o','MarkerEdgeColor',co6,'MarkerFaceColor',co6);
plot(x,RT(7,:),'o','MarkerEdgeColor',co7,'MarkerFaceColor',co7);
plot(x,RT(8,:),'o','MarkerEdgeColor',co8,'MarkerFaceColor',co8);
legend('Mean reaction time','Standard deviation','Participant 1','Participant 2','Participant
     3','Participant 5','Participant 6','Participant 7','Participant 8')
title('Reaction time for each task given')
xlabel('Task given')
ylabel('Reaction time (s)')
xticklabels({'Select BA mode','Start BA mode','Increase Intensity','Start automatic PBT mode'
    ,'Stop automatic PBT mode','Start manual PBT mode','Give a perturbation to the left'})

figure
b2 = bar(x,M);
b2.FaceColor = cotu;
b2.EdgeColor = cotu; hold on
errorbar(x,MC,SC,"LineStyle","none",'Color',cotu2);  hold on
plot(x,CT(1,:),'o','MarkerEdgeColor',co1,'MarkerFaceColor',co1);
plot(x,CT(2,:),'o','MarkerEdgeColor',co2,'MarkerFaceColor',co2);
plot(x,CT(3,:),'o','MarkerEdgeColor',co3,'MarkerFaceColor',co3);
plot(x,CT(4,:),'o','MarkerEdgeColor',co4,'MarkerFaceColor',co4);
plot(x,CT(5,:),'o','MarkerEdgeColor',co5,'MarkerFaceColor',co5);
plot(x,CT(6,:),'o','MarkerEdgeColor',co6,'MarkerFaceColor',co6);
plot(x,CT(7,:),'o','MarkerEdgeColor',co7,'MarkerFaceColor',co7);
plot(x,CT(8,:),'o','MarkerEdgeColor',co8,'MarkerFaceColor',co8);
legend('Mean reaction time','Standard deviation','Participant 1','Participant 2','Participant
     3','Participant 5','Participant 6','Participant 7','Participant 8')
title('Completion time for each task given')
xlabel('Task performed')
ylabel('Reaction time (s)')
xticklabels({'Balance assistance mode selected','Balance assistance mode started','Intensity
    increased','Automatic perturbation mode started','Automatic perturbation mode stopped','
    Manual perturbation mode started','Motion detection activated'})
```

appendix/Reaction_time.m

Matlab Code Orientation

```
clc; clear all; close all;

% Import participant orientation data
[xy1, timestamp1, everything] = xlsread('220890.xlsx','Orientation'); %Participant 1
[xy2, timestamp2, everything] = xlsread('170730.xlsx','Orientation'); %Participant 2
[xy3, timestamp3, everything] = xlsread('210730.xlsx','Orientation'); %Participant 3
[xy4, timestamp4, everything] = xlsread('120352.xlsx','Orientation'); %Participant 4
[xy5, timestamp5, everything] = xlsread('251060.xlsx','Orientation'); %Participant 5
[xy6, timestamp6, everything] = xlsread('260782.xlsx','Orientation'); %Participant 6
[xy7, timestamp7, everything] = xlsread('160496.xlsx','Orientation'); %Participant 7
[xy8, timestamp8, everything] = xlsread('170117.xlsx','Orientation'); %Participant 8

% Import participant performed actions data
[numbers, action1, everything] = xlsread('220890.xlsx','Python'); %Participant 1
[numbers, action2, everything] = xlsread('170730.xlsx','Python'); %Participant 2
[numbers, action3, everything] = xlsread('210730.xlsx','Python'); %Participant 3
[numbers, action4, everything] = xlsread('120352.xlsx','Python'); %Participant 4
[numbers, action5, everything] = xlsread('251060.xlsx','Python'); %Participant 5
[numbers, action6, everything] = xlsread('260782.xlsx','Python'); %Participant 6
[numbers, action7, everything] = xlsread('160496.xlsx','Python'); %Participant 7
[numbers, action8, everything] = xlsread('170117.xlsx','Python'); %Participant 8


%% Convert time to duration

F = 'hh:mm:ss.SSSSSS'; %time format

%participant 1
du1 = duration(timestamp1,'InputFormat',F,'Format',F);
format longg
t1 = seconds(du1);
L1 = ones(length(t1),1)*t1(1,1);                       %create vector with first timestamp,
    with the same length as the number of timestamps recorded
t1 = t1-L1;                                            %substract first timestamp from all
    timestamp, so the duration starts at 0

%participant 2
du2 = duration(timestamp2,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                           %Long fixed format
t2 = seconds(du2);                                     %Convert duration into seconds
L2 = ones(length(t2),1)*t2(1,1);                       %Create vector with first timestamp,
    with the same length as the number of timestamps recorded
t2 = t2-L2;                                            %Substract first timestamp from all
    timestamp, so the time starts at 0

%participant 3
du3 = duration(timestamp3,'InputFormat',F,'Format',F);
format longg
t3 = seconds(du3);
L3 = ones(length(t3),1)*t3(1,1); %create vector with first timestamp, with the same length as
     the number of timestamps recorded
t3 = t3-L3; %substract first timestamp from all timestamp, so the duration starts at 0

%participant 4
du4 = duration(timestamp4,'InputFormat',F,'Format',F);
format longg
t4 = seconds(du4);
```

```
L4 = ones(length(t4),1)*t4(1,1); %create vector with first timestamp, with the same length as the number of timestamps recorded
t4 = t4-L4; %substract first timestamp from all timestamp, so the duration starts at 0

%participant 5
du5 = duration(timestamp5,'InputFormat',F,'Format',F);
format longg
t5 = seconds(du5);
L5 = ones(length(t5),1)*t5(1,1); %create vector with first timestamp, with the same length as the number of timestamps recorded
t5 = t5-L5; %substract first timestamp from all timestamp, so the duration starts at 0

%participant 6
du6 = duration(timestamp6,'InputFormat',F,'Format',F);
format longg
t6 = seconds(du6);
L6 = ones(length(t6),1)*t6(1,1); %create vector with first timestamp, with the same length as the number of timestamps recorded
t6 = t6-L6; %substract first timestamp from all timestamp, so the duration starts at 0

%participant 7
du7 = duration(timestamp7,'InputFormat',F,'Format',F);
format longg
t7 = seconds(du7);
L7 = ones(length(t7),1)*t7(1,1); %create vector with first timestamp, with the same length as the number of timestamps recorded
t7 = t7-L7; %substract first timestamp from all timestamp, so the duration starts at 0

%participant 8
du8 = duration(timestamp8,'InputFormat',F,'Format',F);
format longg
t8 = seconds(du8);
L8 = ones(length(t8),1)*t8(1,1); %create vector with first timestamp, with the same length as the number of timestamps recorded
t8 = t8-L8; %substract first timestamp from all timestamp, so the duration starts at 0

%% Find timesstamps of when motion detection button was pressed and released
%Find timestamp when participant pressed the motion detection button and
%concvert in to seconds
[row1,column]= find(ismember(action1,'Motion detection pressed')); %Participant 1
PR1 = action1(row1,2);
dup1 = duration(PR1,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                     %Long fixed format
tp1 = seconds(dup1);                             %Convert duration into seconds
Lp1 = ones(length(tp1),1)*tp1(1,1);               %Create vector with first timestamp, with the same length as the number of timestamps recorded
tpr1 = tp1-Lp1;                                   %Substract first timestamp from all timestamp, so the time starts at 0

[row2,column]= find(ismember(action2,'Motion detection pressed')); %Participant 2
PR2 = action2(row2,2);
dup2 = duration(PR2,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                     %Long fixed format
tp2 = seconds(dup2);                             %Convert duration into seconds
Lp2 = ones(length(tp2),1)*tp2(1,1);              %Create vector with first timestamp, with the same length as the number of timestamps recorded
tpr2 = tp2-Lp2;                                   %Substract first timestamp from all timestamp, so the time starts at 0

[row3,column]= find(ismember(action3,'Motion detection pressed')); %Participant 3
PR3 = action3(row3,2);
dup3 = duration(PR3,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                     %Long fixed format
tp3 = seconds(dup3);                             %Convert duration into seconds
Lp3 = ones(length(tp3),1)*tp3(1,1);              %Create vector with first timestamp, with the same length as the number of timestamps recorded
tpr3 = tp3-Lp3;                                   %Substract first timestamp from all timestamp, so the time starts at 0

[row4,column]= find(ismember(action4,'Motion detection pressed')); %Participant 4
PR4 = action4(row4,2);
```

```
dup4 = duration(PR4,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                     %Long fixed format
tp4 = seconds(dup4);                             %Convert duration into seconds
Lp4 = ones(length(tp4),1)*tp4(1,1);              %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
tpr4 = tp4-Lp4;                                   %Substract first timestamp from all
    timestamp, so the time starts at 0

[row5,column]= find(ismember(action5,'Motion detection pressed')); %Participant 5
PR5 = action5(row5,2);
dup5 = duration(PR5,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                     %Long fixed format
tp5 = seconds(dup5);                             %Convert duration into seconds
Lp5 = ones(length(tp5),1)*tp5(1,1);              %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
tpr5 = tp5-Lp5;                                   %Substract first timestamp from all
    timestamp, so the time starts at 0

[row6,column]= find(ismember(action6,'Motion detection pressed')); %Participant 6
PR6 = action6(row6,2);
dup6 = duration(PR6,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                     %Long fixed format
tp6 = seconds(dup6);                             %Convert duration into seconds
Lp6 = ones(length(tp6),1)*tp6(1,1);              %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
tpr6 = tp6-Lp6;                                  %Substract first timestamp from all
    timestamp, so the time starts at 0

[row7,column]= find(ismember(action7,'Motion detection pressed')); %Participant 7
PR7 = action7(row7,2);
dup7 = duration(PR7,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                     %Long fixed format
tp7 = seconds(dup7);                             %Convert duration into seconds
Lp7 = ones(length(tp7),1)*tp7(1,1);              %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
tpr7 = tp7-Lp7;                                  %Substract first timestamp from all
    timestamp, so the time starts at 0

[row8,column]= find(ismember(action8,'Motion detection pressed')); %Participant 8
PR8 = action8(row8,2);
dup8 = duration(PR8,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                     %Long fixed format
tp8 = seconds(dup8);                             %Convert duration into seconds
Lp8 = ones(length(tp8),1)*tp8(1,1);              %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
tpr8 = tp8-Lp8;                                  %Substract first timestamp from all
    timestamp, so the time starts at 0



%Find timestamp when participant released the motion detection button
%Participant 1 and 7 did not release the button during the session,
%therefore this part can be ignored for those participants.

% [row11,column]= find(ismember(action1,'Motion detection released')); %Participant 1
% RE1 = action1(row11,2);
% dur1 = duration(RE1,'InputFormat',F,'Format',F); %Read excel data with the right format
% format longg                                     %Long fixed format
% tr1 = seconds(dur1);                             %Convert duration into seconds
% Lr1 = ones(length(tr1),1)*tp1(1,1);              %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
% tr1 = tr1-Lr1;                                   %Substract first timestamp from all
    timestamp, so the time starts at 0

[row22,column]= find(ismember(action2,'Motion detection released')); %Participant 2
RE2 = action2(row22,2);
dur2 = duration(RE2,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                     %Long fixed format
tr2 = seconds(dur2);                             %Convert duration into seconds
Lr2 = ones(length(tr2),1)*tp2(1,1);              %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
```

```
tr2 = tr2-Lr2;                                  %Substract first timestamp from all
    timestamp, so the time starts at 0

[row33,column]= find(ismember(action3,'Motion detection released')); %Participant 3
RE3 = action3(row33,2);
dur3 = duration(RE3,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                    %Long fixed format
tr3 = seconds(dur3);                            %Convert duration into seconds
Lr3 = ones(length(tr3),1)*tp3(1,1);             %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
tr3 = tr3-Lr3;                                  %Substract first timestamp from all
    timestamp, so the time starts at 0

[row44,column]= find(ismember(action4,'Motion detection released')); %Participant 4
RE4 = action4(row44,2);
dur4 = duration(RE4,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                    %Long fixed format
tr4 = seconds(dur4);                            %Convert duration into seconds
Lr4 = ones(length(tr4),1)*tp4(1,1);             %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
tr4 = tr4-Lr4;                                  %Substract first timestamp from all
    timestamp, so the time starts at 0


[row55,column]= find(ismember(action5,'Motion detection released')); %Participant 5
RE5 = action5(row55,2);
dur5 = duration(RE5,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                    %Long fixed format
tr5 = seconds(dur5);                            %Convert duration into seconds
Lr5 = ones(length(tr5),1)*tp5(1,1);             %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
tr5 = tr5-Lr5;                                  %Substract first timestamp from all
    timestamp, so the time starts at 0

[row66,column]= find(ismember(action6,'Motion detection released')); %Participant 6
RE6 = action6(row66,2);
dur6 = duration(RE6,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                    %Long fixed format
tr6 = seconds(dur6);                            %Convert duration into seconds
Lr6 = ones(length(tr6),1)*tp6(1,1);             %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
tr6 = tr6-Lr6;                                  %Substract first timestamp from all
    timestamp, so the time starts at 0

% [row77,column]= find(ismember(action7,'Motion detection released')); %Participant 7
% RE7 = action7(row77,2);
% dur7 = duration(RE7,'InputFormat',F,'Format',F); %Read excel data with the right format
% format longg                                    %Long fixed format
% tr7 = seconds(dur7);                            %Convert duration into seconds
% Lr7 = ones(length(tr7),1)*tp7(1,1);             %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
% tr7 = tr7-Lr7;                                  %Substract first timestamp from all
    timestamp, so the time starts at 0

[row88,column]= find(ismember(action8,'Motion detection released')); %Participant 8
RE8 = action8(row88,2);
dur8 = duration(RE8,'InputFormat',F,'Format',F); %Read excel data with the right format
format longg                                    %Long fixed format
tr8 = seconds(dur8);                            %Convert duration into seconds
Lr8 = ones(length(tr8),1)*tp8(1,1);             %Create vector with first timestamp, with
    the same length as the number of timestamps recorded
tr8 = tr8-Lr8;                                  %Substract first timestamp from all
    timestamp, so the time starts at 0

%% Divide the X Y rotation data into pressed and released button

%Participant 2 pressed and released 3 times
MD21 = t2(t2(:,1) > tpr2(1,1) & t2(:,1) < tr2(1,1));
MD22 = t2(t2(:,1) > tpr2(2,1) & t2(:,1) < tr2(2,1));
MD23 = t2(t2(:,1) > tpr2(3,1) & t2(:,1) < tr2(3,1));
[rowMD21,column]= find(t2(t2(:,1) >= tpr2(1,1) & t2(:,1) < tr2(1,1)));
```

```
[rowMD22,column]= find(t2(t2(:,1) >= tpr2(2,1) & t2(:,1) < tr2(2,1)));
[rowMD23,column]= find(t2(t2(:,1) >= tpr2(3,1) & t2(:,1) < tr2(3,1)));
xy21 = xy2(rowMD21,:); %X and Y rotation between the first press and release
xy22 = xy2(rowMD22,:); %X and Y rotation between the second press and release
xy23 = xy2(rowMD23,:); %X and Y rotation between the third press and release

%Participant 3 pressed and released 2 times
MD31 = t3(t3(:,1) > tpr3(1,1) & t3(:,1) < tr3(1,1));
MD32 = t3(t3(:,1) > tpr3(2,1) & t3(:,1) < tr3(2,1));
[rowMD31,column]= find(t3(t3(:,1) >= tpr3(1,1) & t3(:,1) < tr3(1,1)));
[rowMD32,column]= find(t3(t3(:,1) >= tpr3(2,1) & t3(:,1) < tr3(2,1)));
xy31 = xy3(rowMD31,:); %X and Y rotation between the first press and release
xy32 = xy3(rowMD32,:); %X and Y rotation between the second press and release


%Participant 4 pressed and released 1 time
MD41 = t4(t4(:,1) > tpr4(1,1) & t4(:,1) < tr4(1,1));
[rowMD41,column]= find(t4(t4(:,1) >= tpr4(1,1) & t4(:,1) < tr4(1,1)));
xy41 = xy4(rowMD41,:); %X and Y rotation between the first press and release

%Participant 5 pressed and released 1 time
MD51 = t5(t5(:,1) > tpr5(1,1) & t5(:,1) < tr5(1,1));
[rowMD51,column]= find(t5(t5(:,1) >= tpr5(1,1) & t5(:,1) < tr5(1,1)));
xy51 = xy5(rowMD51,:); %X and Y rotation between the first press and release

%Participant 6 pressed and released 1 time
MD61 = t6(t6(:,1) > tpr6(1,1) & t6(:,1) < tr6(1,1));
[rowMD61,column]= find(t6(t6(:,1) >= tpr6(1,1) & t6(:,1) < tr6(1,1)));
xy61 = xy6(rowMD61,:); %X and Y rotation between the first press and release

%Participant 3 pressed and released 2 times
MD81 = t8(t8(:,1) > tpr8(1,1) & t8(:,1) < tr8(1,1));
MD82 = t8(t8(:,1) > tpr8(2,1) & t8(:,1) < tr8(2,1));
[rowMD81,column]= find(t8(t8(:,1) >= tpr8(1,1) & t8(:,1) < tr8(1,1)));
[rowMD82,column]= find(t8(t8(:,1) >= tpr8(2,1) & t8(:,1) < tr8(2,1)));
xy81 = xy8(rowMD81,:); %X and Y rotation between the first press and release
xy82 = xy8(rowMD82,:); %X and Y rotation between the second press and release

%% Filter data using a first order butterworth filter and threshold value

% fc = 10;     % cutoff frequency [Hz]
% fs = 104;    % sampling frequency [Hz]
% n_order = 1; % filter order
%
% [b,a] = butter(n_order,fc/(fs/2)); %filter coefficients
% xy1 = filtfilt(b,a,xy1); %filter the data




%% Plot XY against time
% Plot colors
co1 = '#6F1D77';
co2 = '#EF60A3';
co3 = '#A50034';
co4 = '#E03C31';
co5 = '#EC6842';
co6 = '#FFB81C';
co7 = '#6CC24A';
co8 = '#009B77';

% Plot figure with X and Y rotation against time
figure
t = tiledlayout(2,1);
nexttile


%plot solid line off orientation data while motion detection button is
%pressed
%First Press and Release
```

```
plot(t1,xy1(:,1),'Color',co1); hold on    %Plot X rotation (degrees) against time (seconds)
    of participant 1
plot(MD21,xy21(:,1),'Color',co2); hold on %Plot X rotation (degrees) against time (seconds)
    of participant 2 during first press and release
plot(MD31,xy31(:,1),'Color',co3); hold on %Plot X rotation (degrees) against time (seconds)
    of participant 3 during first press and release
plot(MD41,xy41(:,1),'Color',co4); hold on %Plot X rotation (degrees) against time (seconds)
    of participant 4 during first press and release
plot(MD51,xy51(:,1),'Color',co5); hold on %Plot X rotation (degrees) against time (seconds)
    of participant 5 during first press and release
plot(MD61,xy61(:,1),'Color',co6); hold on %Plot X rotation (degrees) against time (seconds)
    of participant 5 during first press and release
plot(t7,xy7(:,1),'Color',co7); hold on    %Plot X rotation (degrees) against time (seconds)
    of participant 7
plot(MD81,xy81(:,1),'Color',co8); hold on %Plot X rotation (degrees) against time (seconds)
    of participant 8 during first press and release

%Second Press and Release
plot(MD22,xy22(:,1),'Color',co2); hold on %Plot X rotation (degrees) against time (seconds)
    of participant 2 during second press and release
plot(MD32,xy32(:,1),'Color',co3); hold on %Plot X rotation (degrees) against time (seconds)
    of participant 3 during second press and release
plot(MD82,xy82(:,1),'Color',co8);         %Plot X rotation (degrees) against time (seconds)
    of participant 8 during second press and release

%Third Press and Release
plot(MD23,xy23(:,1),'Color',co2); hold on %Plot X rotation (degrees) against time (seconds)
    of participant 2 during third press and release

%plot dotted line of all orientation data
plot(t1,xy1(:,1),':','MarkerSize',2,'Color',co1); hold on %Plot X rotation (degrees) against
    time (seconds) of participant 1
plot(t2,xy2(:,1),':','MarkerSize',2,'Color',co2); hold on %Plot X rotation (degrees) against
    time (seconds) of participant 2
plot(t3,xy3(:,1),':','MarkerSize',2,'Color',co3); hold on %Plot X rotation (degrees) against
    time (seconds) of participant 3
plot(t4,xy4(:,1),':','MarkerSize',2,'Color',co4); hold on %Plot X rotation (degrees) against
    time (seconds) of participant 4
plot(t5,xy5(:,1),':','MarkerSize',2,'Color',co5); hold on %Plot X rotation (degrees) against
    time (seconds) of participant 5
plot(t6,xy6(:,1),':','MarkerSize',2,'Color',co6); hold on %Plot X rotation (degrees) against
    time (seconds) of participant 6
plot(t7,xy7(:,1),':','MarkerSize',2,'Color',co7); hold on %Plot X rotation (degrees) against
    time (seconds) of participant 7
plot(t8,xy8(:,1),':','MarkerSize',2,'Color',co8); hold on %Plot X rotation (degrees) against
    time (seconds) of participant 8

ylabel('X rotation (degrees)');
xlabel('time(s)');
legend('Participant 1','Participant 2','Participant 3','Participant 5','Participant 6','
    Participant 7','Participant 8')
nexttile
%plot dotted line of all orientation data
plot(t1,xy1(:,2),':','MarkerSize',2,'Color',co1); hold on %Plot Y rotation (degrees) against
    time (seconds) of participant 1
plot(t2,xy2(:,2),':','MarkerSize',2,'Color',co2); hold on %Plot Y rotation (degrees) against
    time (seconds) of participant 2
plot(t3,xy3(:,2),':','MarkerSize',2,'Color',co3); hold on %Plot Y rotation (degrees) against
    time (seconds) of participant 3
plot(t4,xy4(:,2),':','MarkerSize',2,'Color',co4); hold on %Plot Y rotation (degrees) against
    time (seconds) of participant 4
plot(t5,xy5(:,2),':','MarkerSize',2,'Color',co5); hold on %Plot Y rotation (degrees) against
    time (seconds) of participant 5
plot(t6,xy6(:,2),':','MarkerSize',2,'Color',co6); hold on %Plot Y rotation (degrees) against
    time (seconds) of participant 6
plot(t7,xy7(:,2),':','MarkerSize',2,'Color',co7); hold on %Plot Y rotation (degrees) against
    time (seconds) of participant 7
plot(t8,xy8(:,2),':','MarkerSize',2,'Color',co8);         %Plot Y rotation (degrees) against
    time (seconds) of participant 8

%plot solid line off orientation data while motion detection button is
```

```
%pressed
plot(t1,xy1(:,2),'Color',co1); hold on    %Plot Y rotation (degrees) against time (seconds)
     of participant 1
plot(MD21,xy21(:,2),'Color',co2); hold on %Plot Y rotation (degrees) against time (seconds)
     of participant 2 during first press and release
plot(MD22,xy22(:,2),'Color',co2); hold on %Plot Y rotation (degrees) against time (seconds)
     of participant 2 during second press and release
plot(MD23,xy23(:,2),'Color',co2); hold on %Plot Y rotation (degrees) against time (seconds)
     of participant 2 during third press and release
plot(MD31,xy31(:,2),'Color',co3); hold on %Plot Y rotation (degrees) against time (seconds)
     of participant 3 during first press and release
plot(MD32,xy32(:,2),'Color',co3); hold on %Plot Y rotation (degrees) against time (seconds)
     of participant 3 during second press and release
plot(MD41,xy41(:,2),'Color',co4); hold on %Plot Y rotation (degrees) against time (seconds)
     of participant 4 during first press and release
plot(MD51,xy51(:,2),'Color',co5); hold on %Plot Y rotation (degrees) against time (seconds)
     of participant 5 during first press and release
plot(MD61,xy61(:,2),'Color',co6); hold on %Plot Y rotation (degrees) against time (seconds)
     of participant 5 during first press and release
plot(t7,xy7(:,2),'Color',co7); hold on    %Plot Y rotation (degrees) against time (seconds)
     of participant 7
plot(MD81,xy81(:,2),'Color',co8); hold on %Plot Y rotation (degrees) against time (seconds)
     of participant 8 during first press and release
plot(MD82,xy82(:,2),'Color',co8);         %Plot Y rotation (degrees) against time (seconds)
     of participant 8 during second press and release
ylabel('Y rotation (degrees)');
xlabel('time(s)');

% nexttile([1 2])
% plot3(xy1(:,1),xy1(:,2),t1);
% xlabel('x(degrees)');
% ylabel('y(degrees)');
% zlabel('time(s)');

title(t,'X and Y rotation of the interface')

% figure
% plot3(xy2(:,1),xy2(:,2),t2);
% xlabel('x(degrees)');
% ylabel('y(degrees)');
% zlabel('time(s)');
%
% figure
% plot3(xy3(:,1),xy3(:,2),t3);
% xlabel('x(degrees)');
% ylabel('y(degrees)');
% zlabel('time(s)');
%
```

appendix/Orientation.m