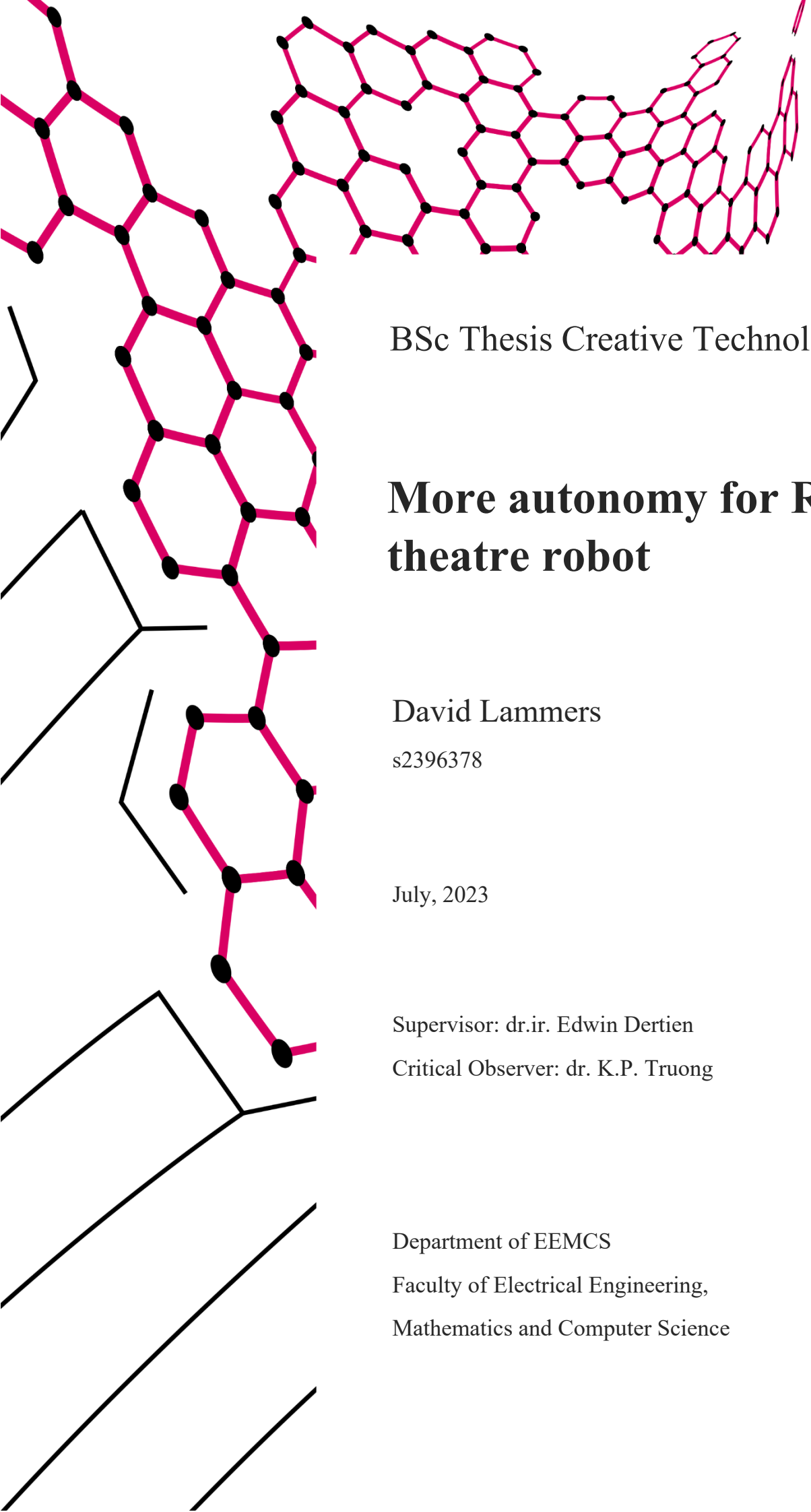


BSc Thesis Creative Technology

More autonomy for Ravi the theatre robot

David Lammers

s2396378

July, 2023

Supervisor: dr.ir. Edwin Dertien

Critical Observer: dr. K.P. Truong

Department of EEMCS

Faculty of Electrical Engineering,

Mathematics and Computer Science

Abstract

In recent years the field of robotics has made a significant impact on various industries. One of those industries is entertainment & theatre. The theatre group Sonnevanck did a theatre performance including a robot named Ravi. Currently the robot has no autonomous behaviour, the aim of this graduation project is to explore the use of autonomous behaviour on the robot Ravi, highlighting benefits, limitations and also the possibilities that it will create for the future of performing art on stage. This is achieved through designing and testing several prototypes while meeting the stakeholder requirements. With the utilization of IMU sensor data we were able to capture head movement and angular changes, translating it into real-time rotation of a robot head. Overall, the graduation work presented in the project showcases the potential of implementing partial autonomous behavior on theatrical robots. The research contributes to the advancement of robotics technology in theatrical environments and is a solid base for future research regarding mimicking body movements for robotics.

Acknowledgement

I would like to express my gratitude towards the following individuals for their contributions and support throughout my thesis project:

First and foremost, I would like to thank my supervisor Dr.ir. Edwin Dertien for his guidance, expertise and continuous support throughout the entire project. His insights and knowledge were of great importance for the outcome of this work.

I would also like to thank Dr. K.P. Khiet Truong for playing the role as a critical observer and taking the time to critically review my documentation, providing valuable insights and feedback on the development of this project.

Special thanks to Daniel van Klaveren and Michiel Bijmans from Theatre Sonnevanck. Their input and requirements were of great importance for the way this project has developed.

I would like to acknowledge Sjoerd de Jong for providing me with access to the GroveBot robotic head, which significantly influenced the prototyping process.

Each of these individuals had a significant role in the successful completion of this project.

Table of contents

Contents

List of Figures

Chapter 1: introduction

In recent years the field of robotics has made a significant impact on various industries. One of those industries is entertainment & theatre [1]. Robots overall, but specifically autonomous robots have become increasingly popular in theatre as they offer innovative and unique ways of performing, as well as engaging with the audience. The use of autonomous robots can offer a wide range of interactions, performances and complex tasks to create revolutionary performances on stage. However, most robots used in theatre are currently puppeteered or pre-programmed, limiting the movement and autonomy. The core of this paper is to explore the process of creating autonomy for a robot that can perform on stage.

The thesis project challenge is based on an already existing, currently puppeteered robot called ‘Ravi’. The robot Ravi has been used previously by the theatre group ‘Theater Sonnevanck’. Currently the robot has no autonomous behaviour so the goal is to explore the use of autonomous behaviour on the robot Ravi, highlighting benefits, limitations and also the possibilities that it will create for the future of performing art on stage.

Throughout the process of creating partial autonomy for Ravi, this paper aims to contribute to the current, ongoing process of the use of robots in theatre and shows it’s potential. By incorporating previous researches and theatre plays, as well as documenting our own experiences and tests, we aim to answer the main research question “How can (partially) autonomous behaviour be used on a robot for theatrical use.”

'Based on a short literature survey of existing use of (partially) autonomous robots on stage, interesting tools and methods will be selected to explore during the study. Therefore, the aim of the background research is to get an inside of the already existing autonomy on and off stage, as well as looking into the technological part of how autonomous robots operate. Providing unbiased information about researches and on/off stage performances. Creating autonomy for a puppeteered robot is a complex process that requires a multidisciplinary approach. Equipping the robot with sensors and algorithms, as well as synchronizing this with the performance’s timing is crucial to make it perform on stage (partially) autonomous.

Chapter 2: Background research

2.1 State of the Art

To enhance our understanding of the progress made in the development of autonomous behaviour in theatrical robots, we engage in a literature research. The first topic to be explored will focus on the types of robots that are currently being used on stage, or which have performed in the past. The second topic will dive more into what kind of autonomy is actually being used on stage, and what kind of autonomy the robots possess. To get a better understanding for our design process of how autonomy for robots works, not specifically focussing on theatre, an overview of the primarily utilized sensors is given as the third topic of the State of the Art research.

2.1.1 What kind robots are already performing on stage?

Previous performances have shown the many possibilities that robots can bring to the world of performing, such as a mesmerizing experience of the performance “Pendulum Choir” [2], where 9 performers where strapped on to 2 hydraulic jacks each. This allowed the singers to be tilted and twisted in all directions up to a 45º angle. The machine operated in real time on pre-programmed sequences controlled from the control booth.

Figure 1 Pendulum Choir [2]

“My Square Lady” [3] is a collaboration between the Gob Squad and, in 2015, one of the Germany’s leading Opera Houses. An autonomous robot with the name Myon, specialized in autonomous learning, is replacing Eliza Doolittle in the classic opera performance “My Fair Lady”. The humanoid robot is not remotely controlled, nor has any predetermined algorithms to do certain tasks. All Myon knows is a result of self-teaching, which causes the actions of Myon to be “Spontaneous [4]. The aim of the play was to introduce and teach Myon about the world of Opera. Professor Manfred Hild mentioned that the focus of Myon is primarily based on detected emotions,

“The task of the opera staff is thus to make their engagement with Myon emotionally capturing enough for the robot to focus its attention on them.” [4]. The implementation of an autonomous robot on stage challenged the performers to improvise on stage, since no the reaction of Myon was pre-programmed.

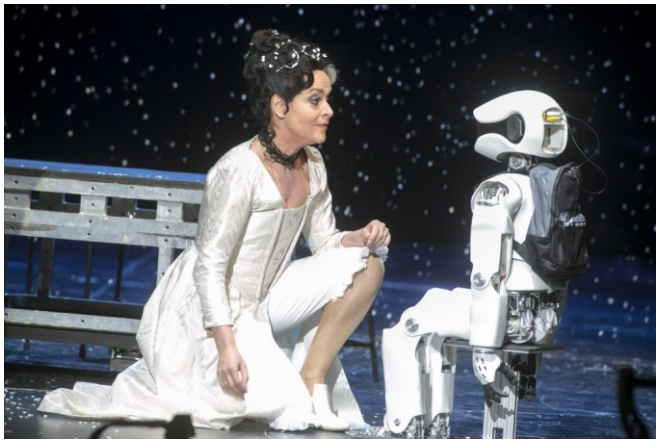


Figure 2 Myon on "My Square Lady" [3]

The Copernicus Science Centre in Poland has already taken it a step further and created a fully robotic theatre where no humans can be found on stage [5]. 2 to 3 robots are fully controlled by one easily accessible operating system, allowing not only mechanics, but almost everyone at the museum to change the shows by making pre-programming sequences. As Engineered Arts Director Will Jackson mentioned, “Reason for the robots, is to try and bring some of the more abstract ideas to people in a very accessible way.” [6, p. 1:04] [7] The robotics theatre is located in a museum together with many more different robotics which makes the transition from robotics to theatre more accessible.

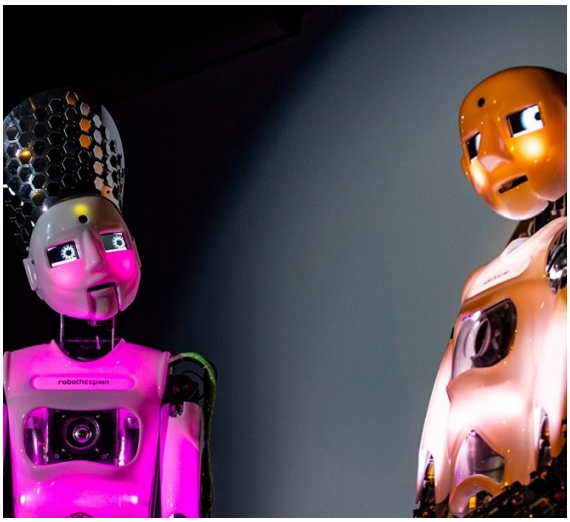


Figure 3 Robotic Theatre | Centrum Nauki Kopernik [5]

Robotics on stage do not stop at small robots. For 12 years, starting from 2007, producers "Global Creatures" [8] and dino-creators "Creature Technology Company" [9] travelled the world with their, based on the BBC Documentary "Walking with dinosaurs", life-sized dinosaurs, where movements is created with robotic parts. Where the small dinosaurs were full scale body suites, the bigger dinosaurs were controlled by up to 3 operators. Divided in 1 person hiding within the dinosaur, and 2 using VooDoo controls to fully control the dinosaurs movement and sounds. The operators make use of a smaller robotic arm which they bend, press and rotate to manage the dinosaurs full body movement [10].

Figure 4 Creature Technology working on their controllable dinosaurs [9]

During the Consumer Electronic Show (CES) in 2018, British artist Giles Walker showcased one of his creations, the robotic pole dancers [11]. Robots made out of scrap metal, CCTV cameras and old mannequins were programmed with different sequences to perform fluid dances on a pole. The goal of the installation was, according to the creator as followed, "I wanted to do something about voyeurism and questions of power, how everyone is being watched." [12]. Giles Walker made the robot heads from old CCTV cameras to give the audience a feeling of being watched, just like they are watching others for pleasure. The installation of Walker was also seen as controversial and sparked a debate on the objectification of women and the ethics of using technology in this manner.

Figure 5 The pole dancing robots created by Giles Walker [12]

2.1.2 What is the current state of autonomous robots on stage?

The next section will continue the investigation into robot-actors on stage. We will look into how autonomy is achieved on stage, as well as how puppeteered and pre-programmed robots are created.

2.1.2.1 Description of the key words

To start with, three important key words have to be understood correctly. The three words "puppeteered", "pre-programmed" and "autonomous" are all related to the control of a system, but they differ in the level of control and the extend of which the entity is able to act independently and react on situational changes. Each of these words will be given a explanation below of how the three key words are used in this paper.

Puppeteering in short, given by Cambridge Dictionary is the following: "a person who makes or uses puppets" [13]. In this paper, the puppets will be the robotics acting as actors on stage. When referring to puppeteering in this paper, the meaning is as followed: a system that is fully controlled by a operator and it has no mind of its own, it can only perform actions directly commanded by the operator.

Pre-programmed in short, given by the dictionary [14] is: "to program in advance" which is already a good explanation. What is meant in this paper when talking about pre-programmed robots, we imply that a system is designed with a set of sequences that is will follow automatically. This includes fully pre-programmed robots such as the Copernicus Science Centre [5] mentioned under section 2.1.1, or partially pre-programmed robots such as Pendulum Choir [2] robots mentioned under section 2.1.1, where different sequences can be selected according to manual real-time input

"An autonomous robot is a robot that acts without recourse to human control" [15]. A robot can have autonomy, while not being autonomous. In this paper we will talk about autonomous behaviour when a robot has certain aspect of independence, where no control or guidance is needed. Decision are based on internal programming algorithms.

In summary, all three terms relate to the control and behaviour of a system, they differ in level of independence. The level ranges from completely puppeteered to completely autonomous, where many robots on stage currently have mixed controls.

2.1.2.2 Deeper dive in the technicalities behind robots on stage

For the second part of researching the current state of autonomous robots on stage, we start by taking a better look on how the different robotics work internally. In section 2.1.2.1 the definition of the three keywords that are often used are explained, with these terms we can sort different robot under certain terms. Starting off with the robot that this whole research is about, called Ravi.

Theater Sonnevanck used the robot Ravi in 2022 for their theatre performance “Ravi de Robot” where Ravi played a stubborn robot [16]. In a collaboration with the University of Twente Ravi was created to be part of the play for about 2 months time. At that time, the robot was fully puppeteered during the play by a person controlling it in real-time through a game console controller. Each button on the controller was bound to certain movement of the robot. Parameters such as the speed of movements of each joint is adjustable on the controller as well. Once Ravi was not performing on stage anymore experiments started how to implement some kind of autonomy. One of these experiments was implementing another type of control to Ravi, where a performer on stage can grab the hand and guide the robot around in this way, where the robots arm acts as a kind of joystick.

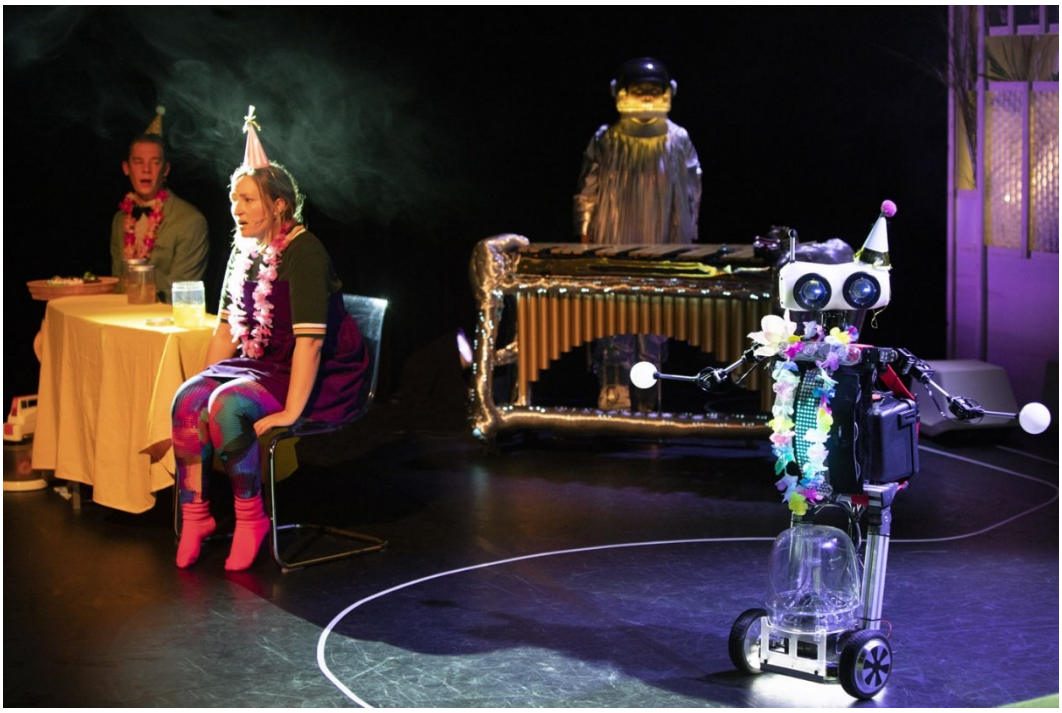


Figure 6 Ravi the robot during the theatre play [16]

Walking with dinosaurs, mentioned in section 2.1.1 is completely puppeteered as well. Although it was fully controlled by 2 operators, it is already a way more advanced way of operating. A combination of a mechanical arm, seen in figure 7, joysticks, buttons and a keyboard for sound effects

made it possible to make realistic dinosaur movements during the show. No autonomy or pre-programmed sequences were used.

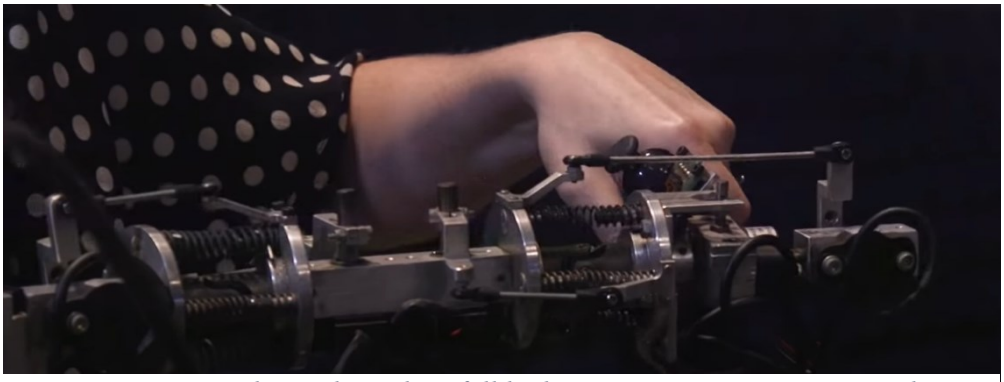


Figure 7 Mechanical arm how full body movements were puppeteered

Mentioned before is the fact that many robots on stage currently have a mixed control input. Where Bartneck mentioned in his research on robots in theatre and media:

"The Locus of Control factor describes where the control unit for the robot is located, and can either be inside the robot or outside of it. The Control Entity factor can either be human or the robot itself." [17, p. 67]

Bartneck mentioned that the control entity is either a human or a robot, without any mixed possibilities. Figure 8 shows how he categorized all theatre robots into four categories. In reality this seems to not be the case. Often there is a mix where certain part might be puppeteered and others pre-programmed. For this reason we will sort the robots researched in this paper into a point graph as shown in figure 9 to give a quick overview of the way most robots are currently controlled in theatre play.

	Locus Of Control	
Control Entity	Inside	Outside
Human	Guy in a suit	Puppeteering
Robot	Autonomous	Robotdrone

Figure 8 Bartneck's classification system organizes robotic behavior into four distinct categories [17]

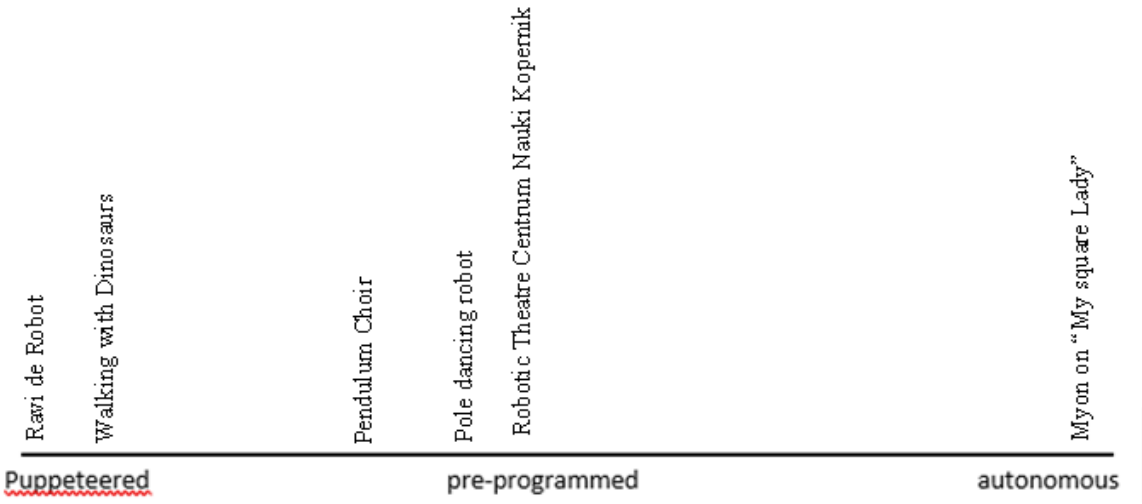


Figure 9 point graph of how robots described in this paper are controlled

To get a more comprehensive overview we dive further into researches and research papers on robotics on stage. Julian M. Angel F described how it is necessary to make a distinction between works where the robot interacts with the audience, and those without any audience interaction [18]. Depending on whether there is audience interaction, different sensors or algorithms need to be used, more on this in section 2.1.3.

A more interactive form of theatre play is stand-up comedian, where the comedian decides his jokes upon the reaction of the audience. There have been a few interesting works on these kind of robots [19] [20]. One of these is the RoboThespian [19] experiment, which is a humanoid robot designed for interaction in public places. The robot is programmed with a marked-up JSON script of jokes and punchlines, able to speak with custom voices with the use of the Acapela Text-To-Speech engine. During the experiment SHORE™ (Sophisticated High-speed Object Recognition Engine) was found to be able to detect audience facial expressions under relatively low lighting conditions in real time. Depending on which of the 4 different emotions "Happiness", "Anger", "Surprise", and "Sadness" was detected, the robot would make different movements. The camera detecting the audience was not implemented in the robot itself, it was put in the air on stage, preferable unseen by the audience. Communication wise, "All communication between the three systems was made through local wired connections using the Transmission Control Protocol (TCP)." This made sure all programs were able to run in real time.

Another great example of more autonomy on stage is Jon the robot [21]. Jon is a small NAO robot that performs pre-programmed jokes to the audience, but responds in an interactive way to the

audience. The noise and laughter of the audience is recorded with the on board microphone and based on those responses, the robot comedian choses a pre-programmed follow up joke. By experimenting they came up with the following settings: "*at least 20 sounds is positive (e.g., hearty laughter), fewer than 9 sounds is negative (e.g., silence), and anything in-between is unclear (e.g., lukewarm chuckles).*" [21]. By setting internal thresholds it was decided whether a joke was received as funny or not, based on the response, a follow up joke was choses. This is a good example of how pre-programmed and autonomous behaviour can work together.

Precise indoor localization tends to be a very difficult topic to tackle for robotics on stage. Petrovic D. et al. [22]. mentioned in his research and experiment on autonomous robot actors that, when trying to create autonomous movement in theatre, a lot of different obstacles have to be overcome. First and foremost, the floor must be flat so no wheels can get stuck behind objects. This can already be a challenge since there often are a lot of light and sound systems on stage that require cables. For performances where the stages are not dynamic, a layout map can be made beforehand and the robot can plan it's path using algorithms (e.g. A*) and avoid all static objects. To avoid dynamic objects such as other humans or robots, adequate dynamic avoidance strategies need to be implemented, while always keeping a safe distance to the podiums edge and performers. Humans might not only be the obstacle to avoid, but also the target for a robot's interaction.

In the case of "Ravi de Robot" [16], the problems mentioned also partly intervene in the development of implementing autonomy. Currently it is the only robot on stage during the performance, which avoids problems such as multi-managing. Cappo et al. [23] showed in his research on multi-robot trajectory generation in online context how multiple robotics, in this case drones (quadcopters), can be managed on stage. The main way to direct the drones is by gesture based input, where a motion capture system tracks reflective markers on the performer and this motion data is passed to a gesture recognition system to identify gestured descriptors [23, p. 10]. This is an interesting way of combining autonomy with performers on stage.

With the given information in this section, we can update figure 9, where the updated figure is shown in figure 10.

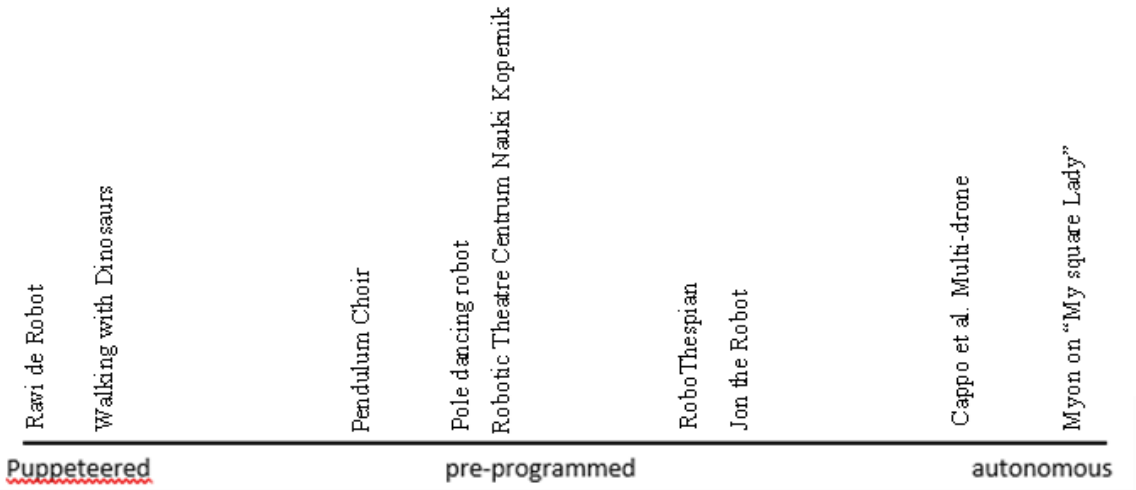


Figure 10 updated point graph of how robots described in this paper are controlled

What figure 10 shows is how puppeteering, pre-programming and autonomous behaviour can be mixed together in objects to create an immersive experience for the audience. What can seem like a rather small autonomous behaviour for the audience can create a completely different dynamic in a performance in theatre.

In summary, there are a lot of different techniques and ways how robotics are used on stage in theatre plays. It varies from completely puppeteered or pre-programmed to complete autonomy where the actors need to find their way around the robot in real-time. There are numerous occasions where a combination of control elements are used.

2.1.3 Which sensors are mostly used to help a robot achieve autonomous behaviour that could be of interest for a theatrical robot?

Robots on and off stage require sensors to perceive their environment and make decisions based on the received input. Sensors can be categorized into two different types: proprioceptive sensors and exteroceptive sensors [24]. In the following sections multiple interesting sensors of both categories will be investigated and it appliance to a theatrical robot.

2.1.3.1 Proprioceptive sensors

Proprioceptive sensors measure the internal state of the robot itself, such as the joint angle and motor positions. The main use of these sensors is to track the robots position and orientation of any rotating parts. We will take a closer look at a few sensors, selected based on which sensors could be interesting for a theatrical robot.

Encoders:

Optical incremental or absolute encoders are on of the most used sensors to measure rotation and angular speed within a motor drive or shaft of a robot's joint. Absolute encoders can provide the exact rotational position as well as eliminate the need of origin alignment, while incremental encoders can only provide a change in position, but tend to have higher resolution [25] [26].

Internal Measurement Units (IMUs):

IMUs typically consist of gyroscopes and accelerometers [26]. Gyroscopes are sensors used to determine orientation and inclination. Measuring inclination of a robot could be useful when working with uneven terrains, which in the case of "Ravi de Robot" [16] is currently not a obstacle to tackle. Another use of the gyroscope is to monitor the direction of travel, it can detect any direction change and give feedback on whether the robot is traveling in a straight line or if turns are accurately performed.

Accelerometers are primarily used in robotics with the aim to measure the robot's acceleration regarding the rest pose. The sensor senses the body acceleration, which can be integrated to determine the robots velocity, which than can locate the position in comparison to it's starting position.

The measured acceleration of the accelerometers alone generally do reflect on the actual robot speed when moving on horizontal surfaces. Incline surfaces can cause the sensor to not reflect the

correct acceleration due to forces such as gravity [27]. This is the reason why often gyroscopes and accelerometers are used together to primarily measure location and placement of the robot as accurately as possible. By the use of multiple gyroscopes and accelerometers omnidirectional orientation and acceleration can be determined. IMU's can after a long period of use start do drift, which causes the reflection of the sensors to be inaccurate. Often exteroceptive sensors are used in combination to verify the robots position [26].

Torque sensors:

Joint torque sensors measure the force that is applied on certain robotic joints. Applications including these kind of sensors can be measuring how much force is applied when touching or picking up objects, environment interactions.

2.1.3.2 Exteroceptive sensors

Exteroceptive sensors measure the external environment of the robot, such as the distance to obstacles, sound or temperature. Overall, there is a lot more variety in exteroceptive sensors than in proprioceptive sensors. We will take a closer look at a few sensors, selected based on which sensors could be interesting for a theatrical robot.

Active-Ranging Sensors:

Active-Ranging sensors are a very common used sensor for indoor applications. The most popular type of these sensors depends on time-of-flight (TOF) active ranging [28]. Electromagnetic or sounds pulses are fired and depending on the time which it takes for the reflection to come back to the sensor is used to calculate the distance. Ultrasonic, laser rangefinders and TOF cameras are commonly used active-ranging sensors

Ultrasonic distance sensor work by emitting sound waves at certain intervals, while receiving the reflections to measure the distance to objects. Besides the fact that ultrasonic sensors often become less accurate at greater distances, U. Grimaldi et al. [29] mentioned in a research that the accuracy of these sensors can suffer greatly from loud acoustic noise, which is often the case in theatre plays. Better ultrasonic sensors have been made since Grimaldi's research which reduced this interference, but external noise with the same frequency as the ultrasonic sensor can still affect the accuracy of the sensor.

Laser rangefinders, also called lidar, which stands for light detection and ranging, is a very often used sensor for mobile robotics. It uses laser beams to scan distances to objects. Lidar sensors tend to be very accurate and able to map environments accurately in 3D.

TOF cameras are used to form depth maps, able to capture a whole 3D scene [28]. TOF camera's tend to still provide high-resolution data maps while working in low-light environments.

Lidar sensors and TOF cameras are used in many different fields. Besides being used for 3D mapping for mobile robots, they are also already included in the newest smartphones. TOF camera's can judge the depth and distance in photo's which makes it possible to focus on something in front and blur out the background in real time [30].

(Omnidirectional) Camera & AI:

Cameras play a crucial role nowadays in all kind of robotics. They allow for visual data to be captured which can be processed by AI algorithms to enable various functions.

All sort of cameras are used to detect the environment around a robot. Cameras with a wide FOV (field of view), often over 180 degrees are omnidirectional cameras. Shaped lenses or mirrors are used to create a wider FOV. Wide FOV provides the opportunity to capture data in a wider range without the use of multiple camera's, enabling to track more objects at the same time. Up to 360 degrees is possible with the use of para- or hyperbolic mirrors [28].

Cameras alone do not have the ability to recognize anything within a picture, this is where AI comes in play. AI algorithms can be trained on large datasets to recognize items in its environment, label it and provide a robot with the vision needed to interact with it, if there is possibility to. Besides object detection, AI algorithms can be used for many other purposes such as facial/body/expression recognition, as well as anomaly detection or saliency mapping, which is a method that shows how certain items are detected by providing heatmaps and highlighting features of the input of an AI model [31] [32].

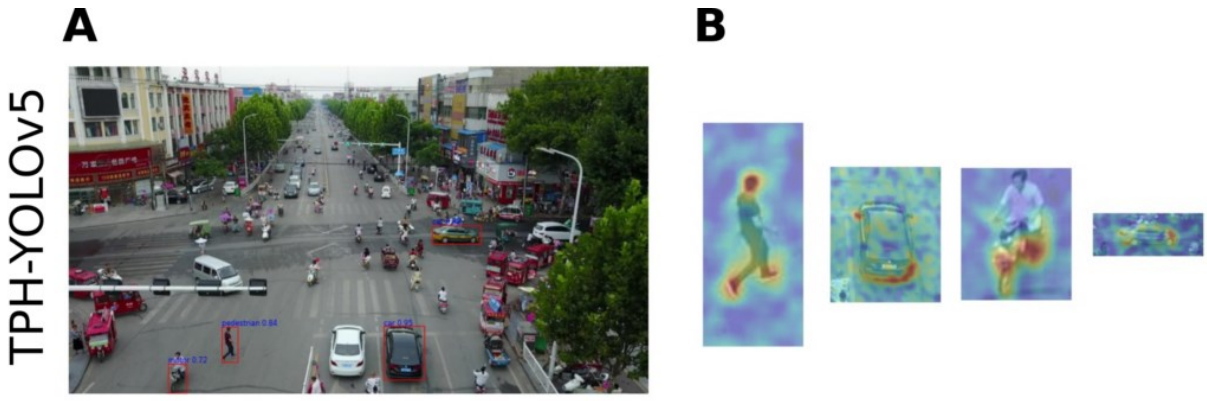


Figure 11 (A) YOLOv5 object detection model (B) Saliency map showing the input features how objects are detected [32]

Audio sensors:

Audio sensors are commonly used in robotics to detect sound. Combining audio sensors with AI algorithms can perform tasks such as noise monitoring and speech recognition. In theatrical environments such sensors can be implemented in a robot itself or it can be input from a actors microphone. Implementation of audio sensors can provide the ability to create interaction with the actors on stage, as well as the audience [28].

2.1.3.3 Ravi the robot current sensor situation

At present, Ravi lacks sensors that can directly contribute towards autonomous behaviour. Ravi primarily consists of servos, LED lights and control hardware, such as an Arduino, which facilitates the option to control the robot. However, there are no integrated sensors that enable Ravi to autonomously interact with its environment or other actors on stage.

2.2 What kind of autonomy is feasible in a theatre on stage?

The use of autonomous robots on stage has the potential to change the way people experience theatre. However, there are a few important limiting factors when creating autonomy on stage.

One factor that could be limiting the autonomous behaviour in theatrical environments is visual perspectives such as camera's. Theatre productions often have low lighting settings to create an atmosphere, which can create difficulties for visual sensors to operate accurately. Interactions with

actors could be difficult under these circumstances, but mainly interactions with the audience since the lighting on stage is often a lot brighter, while lighting in the audience is often limited to minimal.

Another factor that can limit the level of autonomy is the presence of physical obstacles and cables. Cables used for microphones, speakers and lights could interfere with the robot and limit it's ability to move autonomously. Possibilities of the robot getting stuck or even falling over might be at play, which makes it a safety risk for the robot and performers on stage. Ignoring the fact of cables being a possible problem, stages are often dynamic environments during theatre plays. Obstacles and actors themselves move around which causes the need for active object detection, and for path finding algorithms to recalculate a possible path to the designated location. For performances on smaller stages this can create problems that possible paths are blocked by moving actors or objects which makes the robot recalculate its path each time, resulting in a robot not acting smoothly or even failing on stage.

Another factor to keep in mind when using audio sensors is that surrounding sounds might exceed the threshold of the audio sensor since audio in theatre is often loud. When implementing audio sensors in the robot itself together with AI for audio recognition, it could receive input audio with a lot of noise when standing to close to a speaker or when thresholds are not correctly set. Higher quality audio input sensors can severely lower these problems.

In conclusion, the level of autonomy that is achievable in theatrical environment may be limited by factors such as low lighting conditions and obstacles. These problems can be worked around but require more knowledge, time and money than available for this solo thesis project. There are still many other autonomous possibilities available to implement to Ravi. Using a combination of different sensors can lead to interesting and innovative autonomy on stage.

Chapter 3: Ideation & specification

3.1 Methods and Techniques

With the completion of state-of-the-art research, we now proceed to the ideation phase. We have identified the factors that can limit autonomy on stage so first a brainstorm session started to gather as many idea's as possible. The brains storm session started with identifying all the different technically possible options within the given time limit. Once identified, various concepts were made how autonomy could bring an interesting aspect to the theatrical play of Ravi, these concepts can be found in appendix A. The outcomes of this brainstorming session were subsequently discussed with the supervisor, narrowing down the selection to the most intriguing and viable ideas. Following the initial session, sketches were made, shown in appendix B, incorporating the most interesting idea's based on preliminary research and conversations with the supervisor.

The second objective of the design process involves the creation of a testable prototype. The prototype will be developed and tested within a controlled environment rather than an actual theatre. Once a functional prototype is successfully produced, it can be presented to the supervisor, who himself is one of the key stakeholder in this project since he is directly involved in the making and controlling of Ravi.

The third objective in the design process entails a meeting with the client, in this case "Theater Sonnevanck" to present the selected ideas and gather their opinion and feedback on it, as well as hearing their opinion on the prototype and explore potential enhancements that could be interesting to implement. Keeping the limitations in mind during the whole process, and streamline the clients expectations to make sure they know what is and isn't feasible for this graduation project. This conversation with the client will also serve as validation for the project.

Lastly, our goal is to refine the prototype based on the feedback given by the stakeholders to a finalized product to be presented to all parties involved. This includes integrating all the information gathered throughout the other design stages to make a more polished prototype/product.

3.2 Ideation

In the ideation section of the report there will be a showcase of the design process. Starting at idea creation in the form of brainstorming, to testing and experimenting with different setups.

3.2.1 Brainstorm session setup

The brainstorming session was set up as followed. First of all technically possible idea's where gathered and distributed amongst which part of the robot it would influence, such as full body, arm, legs or head. With this in mind, possible interesting options based on the preliminary research were written down. This resulted in a wide range of ideas to discuss with supervisor Edwin Dertien, shown in appendix A.

3.2.1.1 Brainstorm session 1 & findings

The first brainstorm sessions with Edwin Dertien took place online on a video conferencing tool. The different ideas were presented and discussed. Some important design criteria were found. In this case Edwin Dertien, is currently the person controlling Ravi throughout the whole theatrical play, so he was able to provide a good insight on which implementations could improve the play. One thing that was noticed is that the person controlling the robot had to be actively involved in the movement of the robot during the entire play. Reacting on movement from actors, music and lighting. This can be a demanding job for the operator to be fully focused for the entire play. For this reason, more focus was moved towards an implementation that would not only be interesting for the audience, but as well as make the operator less actively involved during the entire play. The advantages of this is that, with the use of sensors and data, reactions of the robot can be quicker and more precise than when controlled by an operator. With this in mind, sketches were made and a second appointment in person was arranged to present the new and improved ideas.

3.2.1.2 Brainstorm session 2 & findings

The second brainstorm session with Edwin Dertien took place in person. Sketches were made , shown in appendix B, an presented. After some more discussion we came to the conclusion that implementing a fully autonomous solution would not be best suited for Ravi, since going from fully controlled to fully autonomous is a big gap to fill. Ideas including facial expressions, reaction to music and active motion control were the ones that came forward as the most interesting options for this project. Looking into the technical aspect of these implementations as well as considering the limitations of a theatrical environment, the idea's shifted from external sensing to internal sensing.

With the use of internal measuring units, you are able to gather a lot of data for interesting interactions.

3.2.1.3 Conclusion of the brainstorm sessions

To conclude the brainstorm sessions, the decision was made to take a deeper look into the use of internal measurement methods and technologies, with the idea to let an actor interact with the robot on stage through their own body movement.

3.2.2 The first experiment

In the first experiment, the focus lays on how to gather data from an actors body. The main question of his experiment is *"Which IMU sensors are suitable to use on actors to create a interaction between Ravi and the actor?"*. The experiment was designed in multiple aspects. Firstly, what type of IMU sensor do we need to gather the data needed. This brings us back to the preliminary research and shows the relevance of it, where we investigated some different proprioceptive sensors. Proprioceptive sensors measure the internal state of the robot itself, such as the joint angle. In this case, we do not want to measure the joint angle of the robot, but of the actor. With the use of a gyroscope we are able to write a code to extract data to determine a specific angle. The second aspect is to test different gyroscopes to see how fluently data can be extracted, as well as how long it takes to extract to extract it. For this, we chose 2 small and popular gyroscopes with online available libraries to work with, and 1 all-in-one board from Seeed.

3.2.2.1 Experimental setup

For the experiment we used the following equipment:

- ESP32 LOLIN32 with micro USB cable
- Arduino Nano Board with USB-C cable
- Seeed studio xiao nrf52840 sense
- Jump wires
- Breadboard
- Gyroscope
    - MPU-6050 Accelerometer & Gyroscope 3-Axis
    - GY-85 IMU 9DOF Sensor
- Arduino IDE Software
- Libraries

- MPU6050.h, for the MPU6050 Accelerometer & Gyroscope
- GY_85.h, for the GY-85 IMU

The first experiment was done with the Seeed studio xiao nrf52840 sense sensor. This board is a small but advanced piece of hardware, with onboard Bluetooth 5.0, IMU and PDM control [33]. Very quickly after starting to use this board, errors occurred where uploading or reading data was not possible due to an unresolved issue, and there were struggles working with the embedded Bluetooth module. Documentation about this specific board is hard to find and not many solutions for this problem were known. After discussion with Edwin Dertien, the decision was made not to continue working with this board and focus on the remaining 2 IMU's.

The experiment was set up as followed. The MPU6050 was connected with the Arduino Nano Board, and the board was connected in serial to the laptop to read the extracted data. 5V was used to power the IMU, and pin 3 & 4 were used to gather the data. Figure 12 shows the setup for the first testing. With the use of the MPU6050.h library and the Arduino IDE software we were able to write a code to extract the raw data and map it. With the built in function of the Arduino IDE to record time stamps, the duration of each time the data was sent could be gathered. The exact code used to extract the data can be found in appendix C.

The with use of Excel, the data was transformed to a readable graph and the average response time is calculated by using the following command in Excel:

'=AVERAGE(data)'

This resulted in the following data, shown in figure 13.

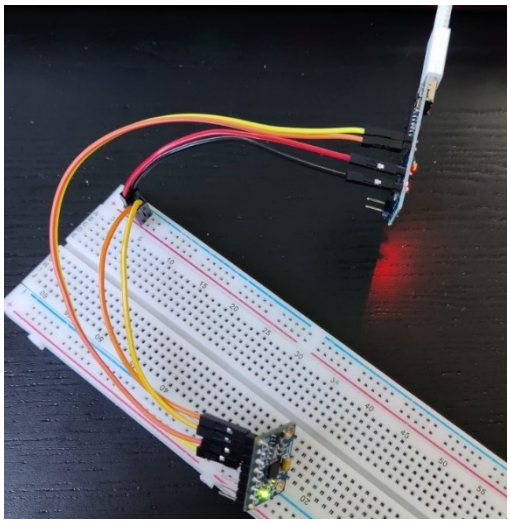


Figure 12 The setup used for experiment one containing the Arduino nano + MPU-6050 gyroscope

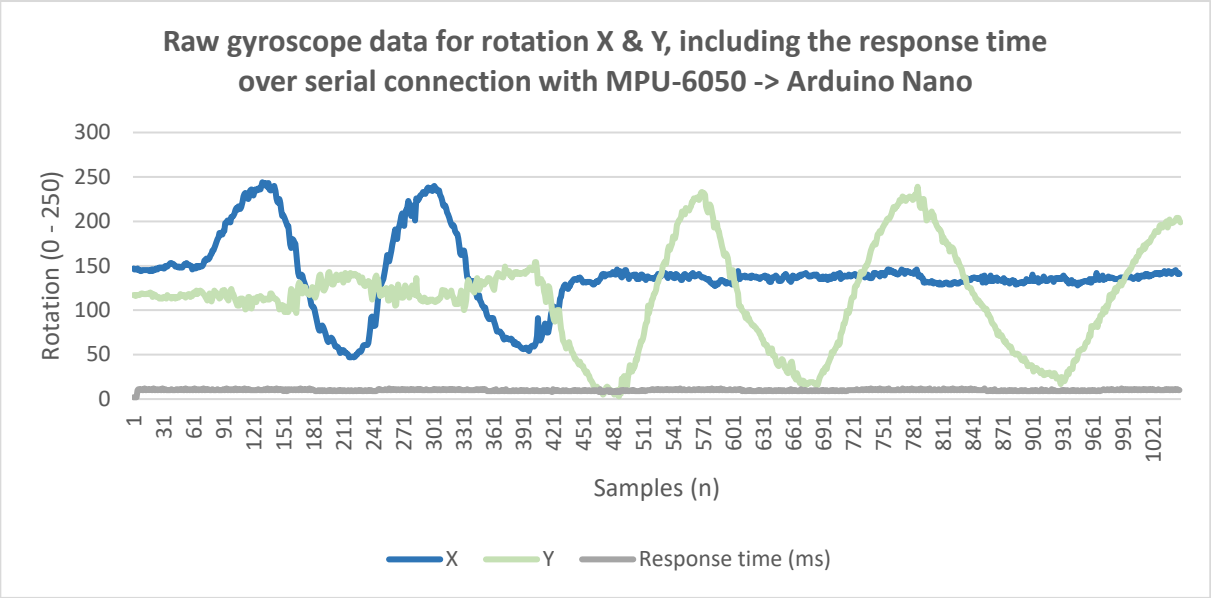


Figure 13 Raw gyroscope data for X & Y, including the response time over serial connection with the Arduino Nano and MPU-6050 sensor.

The average response time in milliseconds came down to ± 9.93 ms (9.92295 ms to be exact).

The same experiment was now done but this time with the use of the ESP32 LOLIN32 board and the GY-85 IMU 9DOF Sensor. This combination of technology has been chosen due to a project that Edwin Dertien did 2 years ago, where a combination of these 2 electrical components, together with a small battery to power it, was used to read data from the swinging of a dog's tail [34]. The compact setup can be seen in figure 14, the code used to gather the required data can be found in appendix D. The advantage of the ESP32 board is that it has an build in Wi-Fi and Bluetooth component.

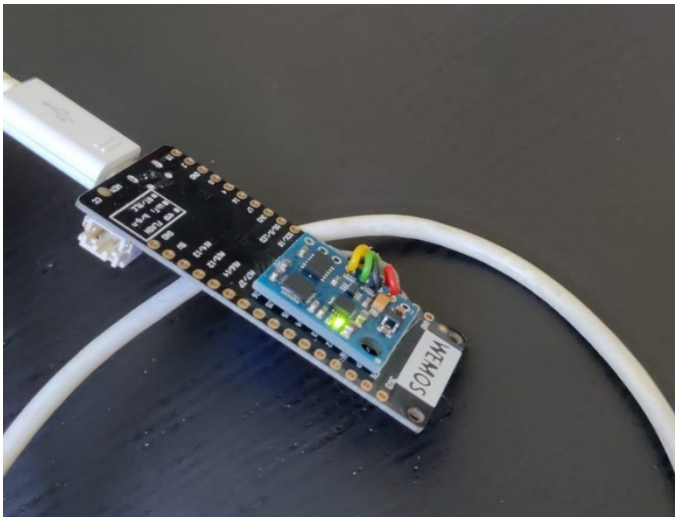


Figure 14 The setup used for experiment one containing the ESP32 and the GY-85

Once again with the use of Excel, the data was transformed to a readable graph and the average response time is calculated by using the following command in Excel:

'=AVERAGE(data)'

The average response time in milliseconds came down to ± 3.43 ms (3.429796356 ms to be exact). Figure 15 shows the rotational values against the amount of samples taken for the second setup.

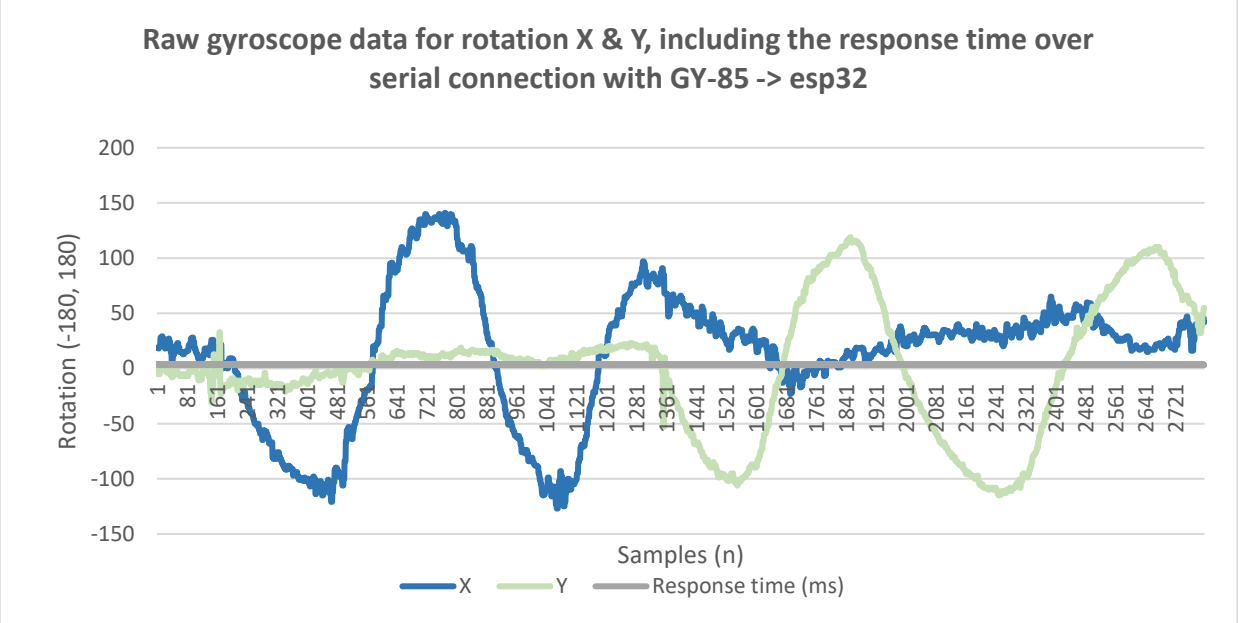


Figure 15 Raw gyroscope data for X & Y, including the response time over serial connection with the ESP32 and the GY-85 IMU

3.2.2.2 Finding during the first experiment

After the first experiment we are not yet able to answer the question which IMU sensor would be the best option to continue with, due to a few things that we came across during the experiment. First we will take a closer look at the response time for both gyroscopes. Figure 16 shows a graph of the response time of the MPU-6050 besides the response time of the GY-85. As calculated before, there is a significant difference between the 2 sensor connections and their response time. From our findings, we can conclude that the MPU-6050 sensor connected to an Arduino Nano is 2.9 times slower in it's response time than the GY-85 connected to the ESP32. Another thing we notice from the graphs is that he raw MPU-6050 data sometimes has a deviation of ± 2 ms in either direction, while the GY-85 has a more steady deviation of ± 0.5 ms, which shows that it is more consistent in it's output.

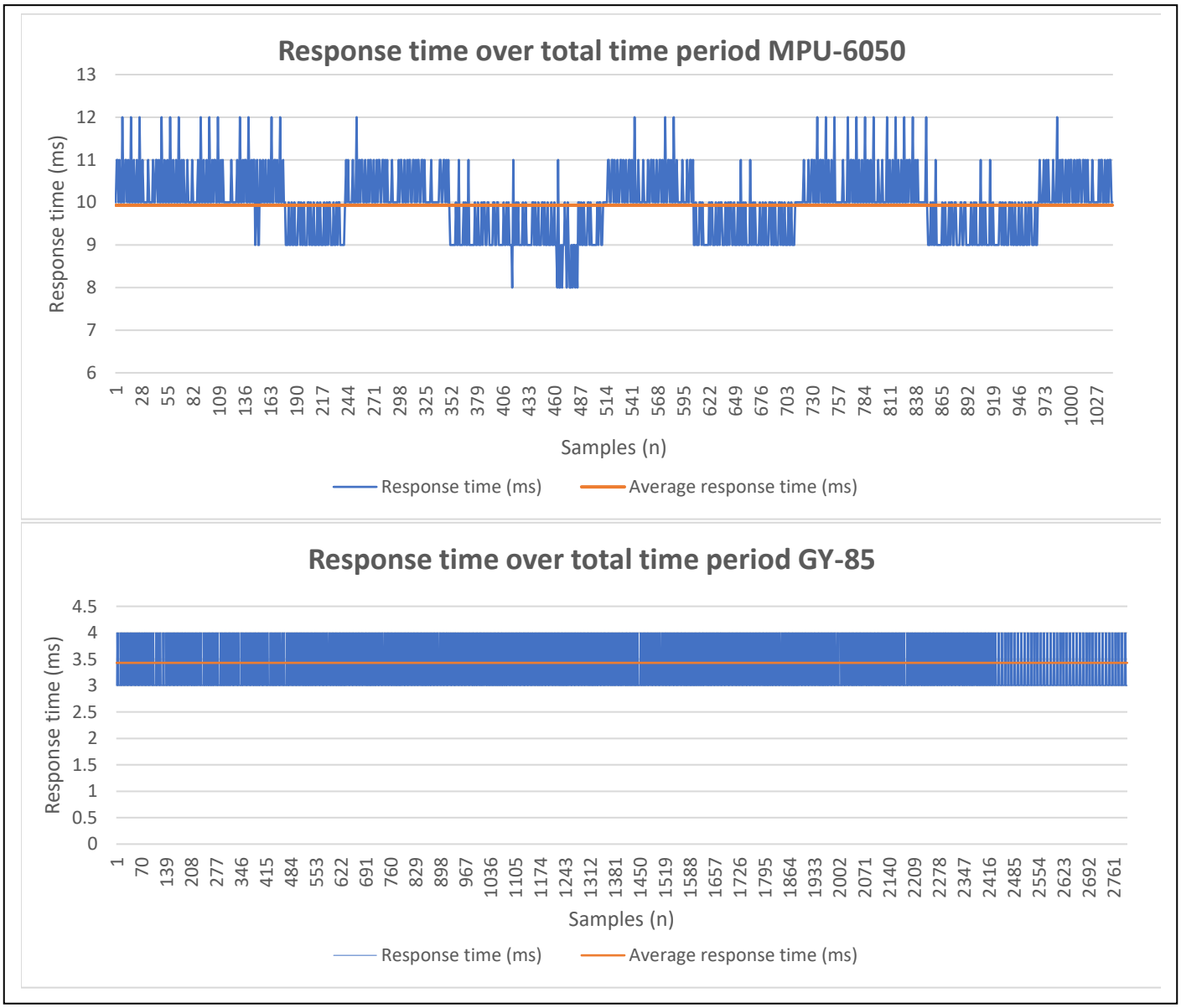


Figure 16 Graphs of the response time of both the MPU-6050 and the GY-85

Another detail that we noticed during this experiment while taking a look at figure 13 and figure 15, we can see that the raw data for the MPU-6050 is slightly more smooth than the data from the GY-85. During the experiment, both sensors got rotated in the same way and with approximately the same speed. The raw output data of the GY-85 sensor shows more shaking behavior during angular changes than the MPU-6050.

Both IMU sensors have their own advantages, but overall we can conclude that both would fulfill our project requirements. It is worth noting that the MPU-6050 has a response time that is 2.9 times slower, however, considering our project's specification, an angular detection change every ± 10 milliseconds is sufficient.

3.2.3 The second experiment

In the second experiment, we incorporated the two setups used in the first experiment, along with an additional Bluetooth feature. In order to enable an actor to control a robot through their motion, it was of great benefit to the actor to eliminate the requirement of a serial connection between the actor and the robot. This approach offers increased efficiency by granting the actor a greater freedom of movement, while also eliminating the limitations on other actors on stage due to obstructive cables. The goal of this experiment was to see how the delay in data would increase or decrease through the usage of Bluetooth communication instead of serial communication, and if this would effect which IMU sensor we would move forward with.

3.2.3.1 Experimental setup

For the experiment we used the following equipment:

- ESP32 LOLIN32 with micro USB cable
- Arduino Nano Board with USB-C cable
- HC-05 Bluetooth module
- Jump wires
- Breadboard
- Gyroscope
  - MPU-6050 Accelerometer & Gyroscope 3-Axis
  - GY-85 IMU 9DOF Sensor
- Arduino IDE Software
- Libraries
  - MPU6050.h, for the MPU6050 Accelerometer & Gyroscope
  - GY_85.h, for the GY-85 IMU

The experiment was set up in the almost exactly the same way as the first experiment for both of the setups. For the setup including the Arduino Nano + MPU-6050, a HC-05 Bluetooth module was connected to the circuit since the Arduino Nano used did not have an embedded Bluetooth function. Figure 17 shows a picture of both of the setups used to gather the data.

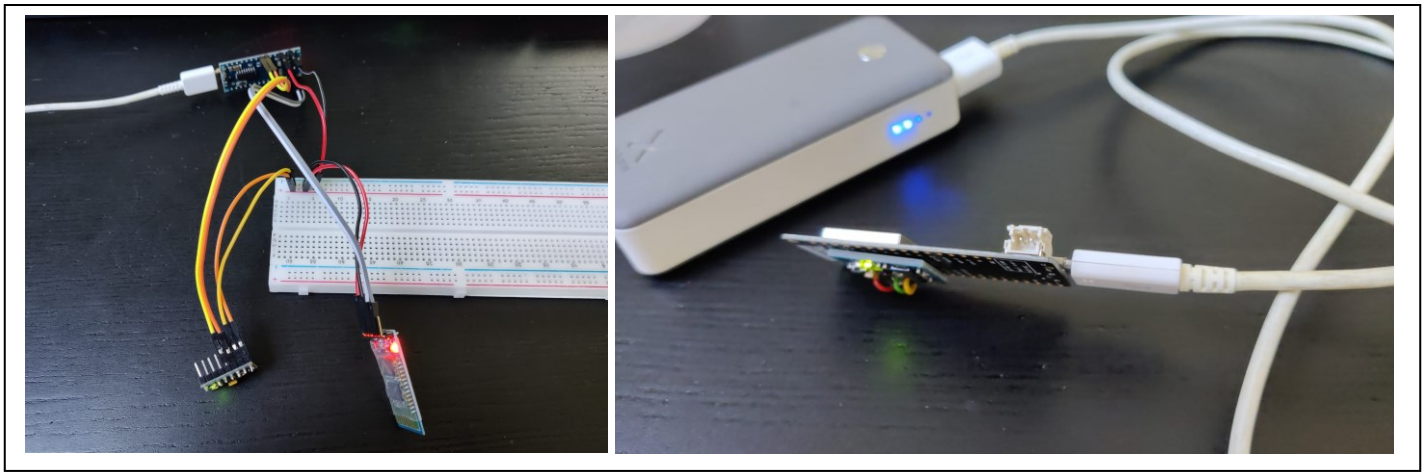


Figure 17 The left picture shows the experimental setup with the Arduino Nano + MPU-6050 + HC-05 Bluetooth module, the right picture shows the experimental setup for the ESP32 + GY-85

The code used to gather the required data of the setup for the MPU-6050 can be found in appendix E, and for the GY-85 setup you can find the code in appendix F.

Both setups made a Bluetooth connection to the laptop to send data. With this raw data, the following graphs in figure 18 were made for the MPU-6050 + Arduino Nano + HC-05 experimental setup. With the use of Excel, we found the average response time to be ± 9.97 milliseconds (9.970882621 milliseconds to be exact), and again there is a maximum deviation of ± 2 milliseconds.

The second setup, including the ESP32 board and the GY-85 IMU, resulted in the graphs given in figure 19. Again with the use of Excel, we found the average response time to be ± 3.88 milliseconds (3.875658588 milliseconds to be exact), and the deviation has slightly increased to 1 milliseconds.

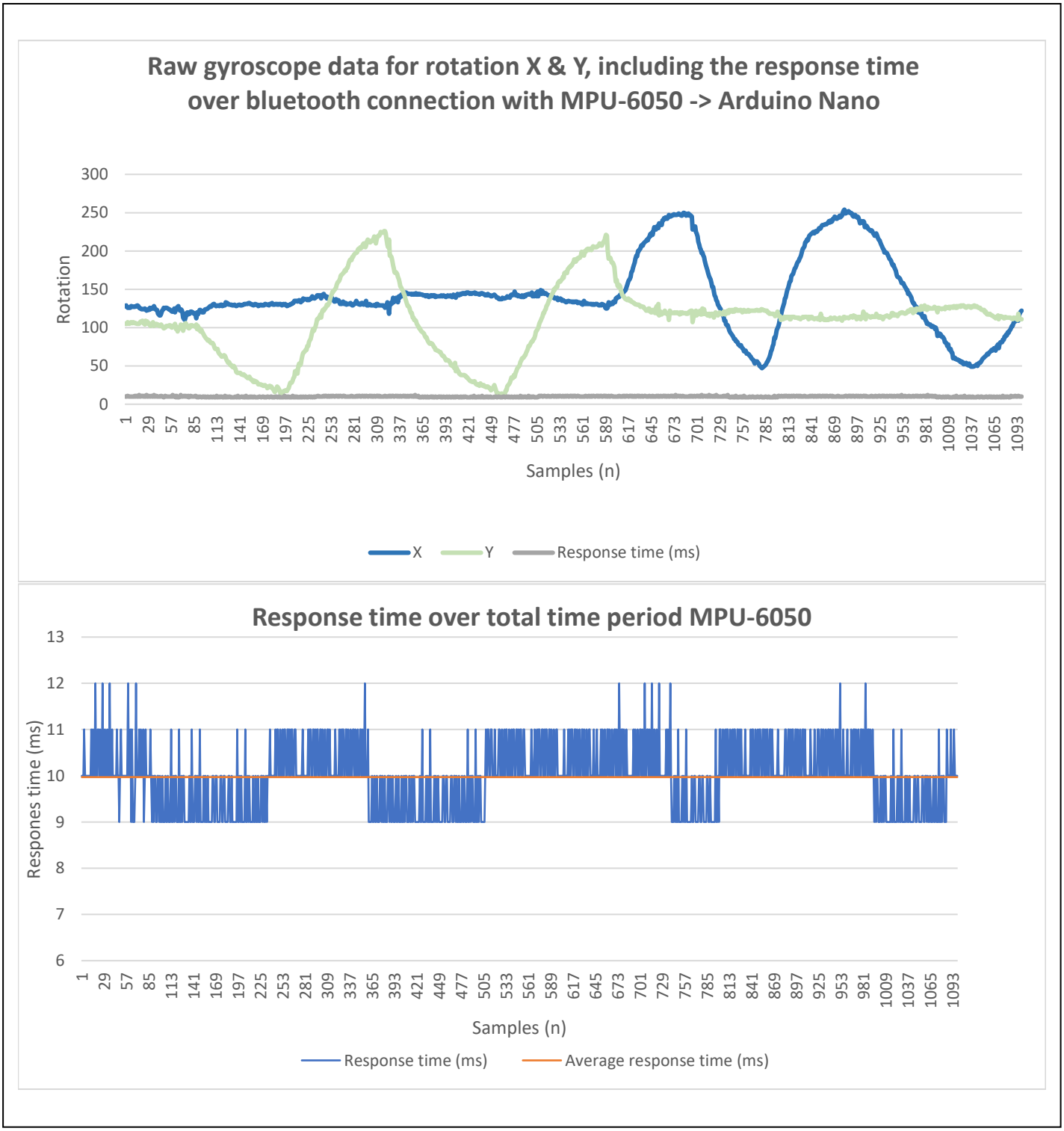


Figure 18 The top graph shows the raw MPU-6050 data transferred over Bluetooth connection. The bottom graph shows the time it took between requesting the data and receiving it on the desktop

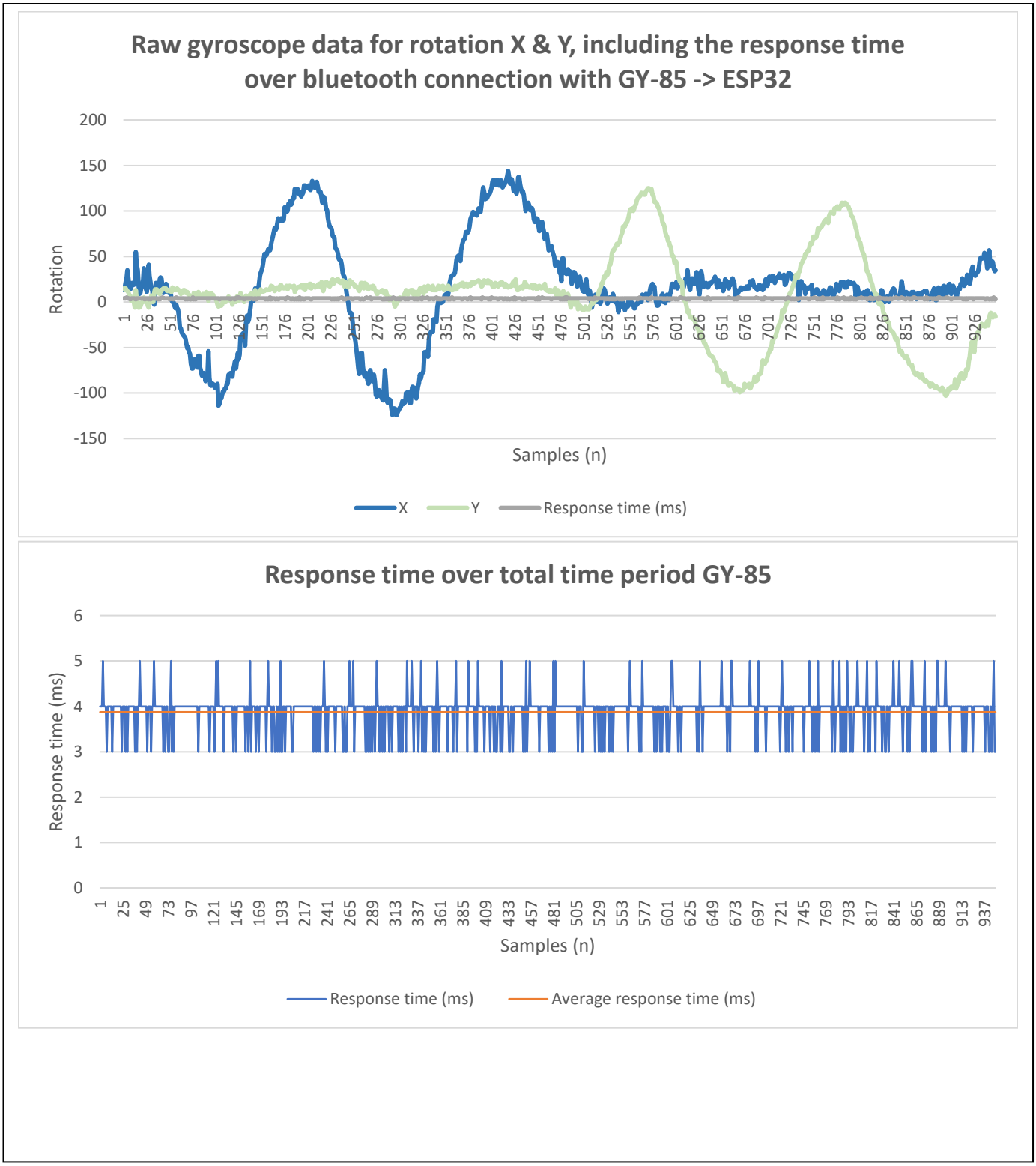


Figure 19 The top graph shows the raw GY-85 data transferred over Bluetooth connection. The bottom graph shows the time it took between requesting the data and receiving it on the desktop

3.2.3.2 Finding during the second experiment

After the second experiment we can conclude which sensor we would like to use for the remaining of the project. There are a few things we noticed during the experiment, starting with the increase in response time on both sensors with the use of Bluetooth. The difference between the response time for the MPU-6050 in serial or Bluetooth is 9.97 – 9.93 = 0.04 milliseconds, and for the GY-85 the difference is 3.86 – 3.43 = 0.42 milliseconds. The changed in response time are this small that they can be neglected for our project. This does provide us with the information that it is possible to receive data from either IMU every 10 milliseconds, which is sufficient for the project.

Another noticeable thing, just like in the first experiment, is that the data provided by the GY-85 + ESP32 combi is less smooth than the data output from the MPU-6050. This is not something that can not be fixed by writing a formula to smoothen out the data, but it is a noticeable difference.

All the raw data from the experiments can be found in the attached directory.

3.2.4 Prototyping

After finding some appropriate hardware to continue the project, the first prototype is made. We are unable to work on the actual Ravi robot itself since the hardware used in the robot is of a higher price range and wrongfully testing new applications can cause damage, which is not wanted. Due to this, another way of implementing and testing needs to be created. Together with Edwin Dertien it was decided to implement the feature of mimicking an actor for the head of the robot, since this would be a good starting point and the head of Ravi should be possible to recreate with smaller, less strong servo's.

With this in mind, we came across S. de Jong, who made a robotic head for the University of Twente Interaction Lab [35]. With the use of a Grove shield attached on an Arduino, he was able to create a robot head with multiple functions. One of the features of the small GroveBot, as it is called, is able to recognize a face and adjust it's servo motors in such a way that the robot is tracking the face. It is also able to show different kind of emotions through 2 hexagonal LED matrices. The basis of this GroveBot is used as the basis for this project. Figure 20 shows the assembled GroveBot with it's multiple functions.

Figure 20 GroveBot made by S. de Jong [35]

3.2.4.1 Setup prototype

The components from the GroveBot that we use for the project are the following:

- 2x MG996R servo
- Pan-Tilt neck setup
- Arduino Uno with USB cable
- Grove Shield
- Grove connection cables
- 2x M5 hexagonal neopixel display

The components added to this list to create the prototype are the following:

- Adjusted power supply input
- NI-MH power supply
- MPU-6050 IMU
- Adjusted GroveBot head
- HC-05 Bluetooth module
- Arduino Nano with battery

3.2.4.2 GroveBot testing and adjustments

During the initial setup of the GroveBot, there quickly came some problems to the light that needed to be fixed in order to reach the desired output. The first problem that occurred was the fact that the power supply at hand, which was a regular power bank with 5V and 2.5A output, was not able to power the MG996R servo's. The moment the power bank was turned on, it immediately shut off and ended up not working anymore. After some research in the lab to read the peak current draw and discussion with technical staff at the University of Twente, who wishes to remain anonymous, we came to the conclusion that the problem was most likely a big draw in current when big movements where made with a servo. The GroveBot servo's were only able to be powered through a custom made USB connector. For this reason, a new style connector had to be made to supply enough voltage and current to the MG996R servo's. A common connector for battery packs is a Tamiya connection, so for this, a Xcell Ni-MH 7.2V – 3000 mAh battery pack with a Tamiya output plug got used. A Tamiya input cable got soldered on the already existing supply connector. By doing this there were now 2 ways of powering the servos if ever needed.

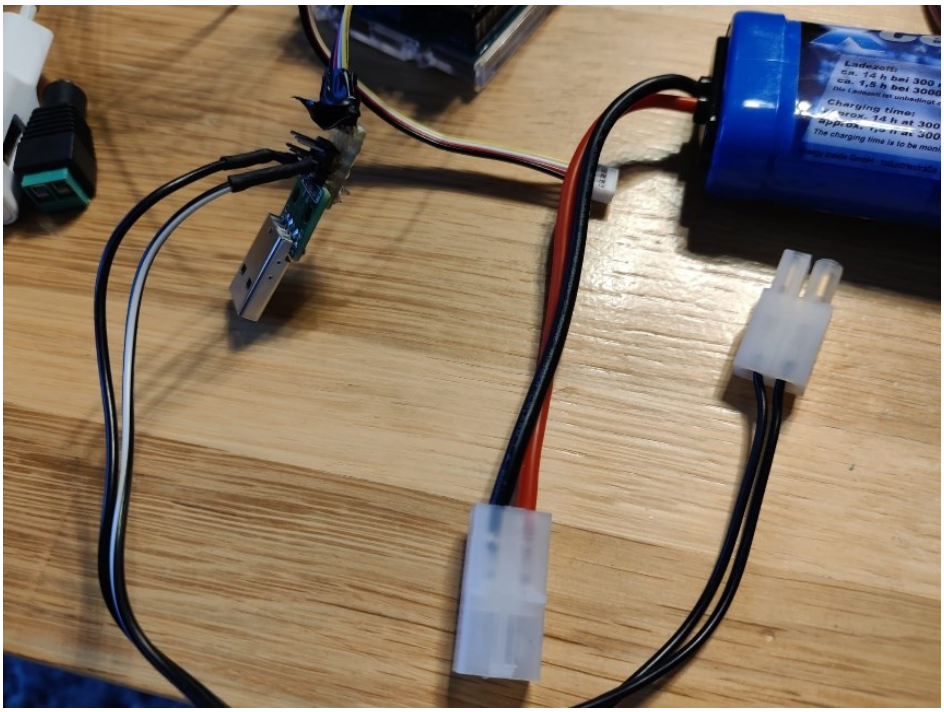


Figure 21 Custom Tamiya connection with a Xcell Ni-MH 7.2V - 3000 mAh battery pack

After adjusting the power supply, 1 of the 2 servo's was still not running properly and was shaking heavily, even while no input was given. Eventually it ended up overheating and not working at all anymore. After extensive research on the MG996R servo's and disassembling the servo's, we found out that one of the MG996R servo was made by Tower Pro, while the other servo was made by Tiankongrc. See appendix G, the first photograph shows the 2 different servo's.

The specifications of the two MG996R servo motors are as followed:

Table 1 Specifications of the 2 different MG996R servos

	Tiankongrc	Tower Pro
Minimum supply voltage [V DC]	4.8V	4.8V
Maximum supply voltage [V DC]	7.2V	6.6V
Minimum recommended supply current [A]	2.5A	2.5A

The MG996R servo's had different maximum supply voltages, which caused the Tower Pro servo's to overheat. After ordering a few new Tiankongrc MG996R servo's and implementing them into the setup, the servo's where running properly but not smoothly. After several different attempt with different power supplies and circuit setups the servo's were still not moving as smoothly as intended.

Besides the MG996R by Tiankongrc, attempts have been made with the Micro Servo MG90s 180º by TowerPro and the Micro Servo SG90 by TowerPro. The same code and setup was used and the servo's were able to move smoother than the MG996R servo. This experiment showed that the code and setup was correct but that the MG996R servo's were not as accurate as aimed for. Unfortunately we were not able to replace the servo's by any of the smoother servo's since they were a lot smaller and not able to hold the head of the robot steady while on an incline angle. Neither of these servo's had a position feedback integrated which gave limited options on how to know where the servo's actually was positioned at, and correct if it over rotated or was not able to hold it's weight in an inclined position. The second picture within appendix G shows the MG996R servo's which where strong enough to hold the head of the robot, shown in picture 3.

One of the mayor issues was the fact that when the MPU-6050 was directly connected to the Arduino Uno, the servo's would move in an acceptable smoothness. Once the MPU-6050 was attached to the Arduino Nano and the data was sent over a Bluetooth communication to the Arduino Uno with the use of 2 HC-05 Bluetooth modules, the servo's would behave in very unpredictable and stuttering way. The way the Arduino Uno received the data is as followed: X and Y data of the IMU is send in a string through Bluetooth communication, and split in 2 integer value's stored as the target value for the servo's. The following code shows how this is done within the Arduino IDE software:

```
data = bluetooth.readStringUntil('\n');
   servoX_Target_Int = bluetooth.readStringUntil(',').toInt(); // Get X position
   servoY_Target_Int = bluetooth.readStringUntil('\n').toInt(); // Get Y position
```

The way the incoming data is stored is the same way the data is stored when the MPU-6050 is serial connected. As discussed earlier in the IMU selection section of this documentation, the data received over Bluetooth only had a very small increase in response time, but the data itself seems to not be affected. After discussion with Edwin Dertien and multiple other Creative Technology students we were unable to resolve the stuttering problem over Bluetooth communication. Due to this, we would set this function aside for the final product even thought it could have a great impact on the usage of the function.

3.2.4.3 Conclusion

During the prototype phase of the project, a lot of unexpected problems came along the process. After solving a majority of the problems we ended up with a working prototype that has an acceptable way of moving. Together with 1 of the stakeholders, Edwin Dertien, we found a number of different criteria that got implemented within the prototype, and other criteria were set aside to make sure to stick within the given time limit for this project.

3.2.5 Discussion with the stakeholders

In the third phase of the design process, we arranged a meeting with the key stakeholders involved in this project, namely Daniel van Klaveren (director, writer, artistic director) and Michiel Bijmans (dramaturge) from theater Sonnevanck. Both individuals expressed enthusiasm and eagerness to discuss the project and were particularly intrigued by the prototype that had been developed. The primary objective of this conversation was to engage in an open conversation about the prototype that has been made and gather their opinions, as well as a validation of the project that all stakeholders involved were pleased with the progress made within the graduation project.

Prior to initiating any conversation with these stakeholders, it was necessary for them to fill in a consent form, which got approved by the EEMCS Ethics Committee. The consent form can be found in appendix H.

During the conversation, several video's were shown of the prototype in action. A few interesting points came into discussion and the conversation lead to the following 5 main questions.

"Is it limited to head movement or could it be implemented on other parts of the body"

For this project, we are limited to head movement only. This is the case since we are not able to work with the actual Ravi robot itself. Currently the head is fully controlled by the movement of the actor. However, once the communication method between data and servo movement is established, it becomes a straightforward process to expand the system by incorporating multiple IMUs on the actor. This expansion would enable tracking of shoulder and arm movement, translating into corresponding movements of Ravi on stage.

"In what way is the communication between the robot and actor done?"

Currently the way of communication is done by wire since we were unable to solve the problem with the way the servo's reacted on the data received by Bluetooth. Preferably there would be a wireless connection between actor and robot with the use of Bluetooth or Wi-Fi.

"Is there a possibility to switch it on an off during a theatre play"

Currently, the robot operates under complete control of the operator. However, by providing the operator with the option to activate or deactivate the mode where it reacts to the extracted data, the robot can be switched between modes where it responds to data or remains under manual control.

"Are there multiple modes available with the same sensors"

This question sparked a extensive and interesting conversation about all the possibilities with the current prototype. As of now, the data is used to track the head movement and respond accordingly. The stakeholders would be very interested in seeing different options such as doing the opposite as what the actor does.

"Is it possible that the robot is controlled by multiple people"

While this question falls beyond the project's scope, it sparked an intriguing discussion. With the use of Wi-Fi you are able to connect to multiple devices and extract data from it. This way it is possible to

have multiple actors equipped with IMU's to track body movement and an operator can switch between different set's of IMU's, depending on who intends to control the robot during a particular stage performance.

During the conversation we came up with a lot of interesting idea's how this project could result in future work that could lead to amazing theatrical possibilities. The stakeholders of theater Sonnevanck were pleased with the process and intrigued to see the end results.

Chapter 4: Realization

The realization of this project is the creation of the final prototype of the robot, with which the designer can play around and provide the stakeholders with a good overview of the possibilities of the work. The prototype aims to initiate the implementation of increased autonomy for Ravi, serving as a foundation upon which future projects can build and enhance autonomy further. The initial section of this documentation segment will discuss the reasoning behind selecting specific components. The second section will go more in depth of the utilized code.

4.1 Physical components

4.1.1 Size

The size of the prototype has been chosen due to 3 practical reasons. The first reason is that we came across the already existing GroveBot which gave a great example of which components we could use to make sure all movements needed are possible. The second reason why we chose to stay with this design is because eventually, the goal is that this function will be implemented in the actual Ravi robot. The hardware in Ravi is more advanced and expensive, but the idea and way of functioning is the same. The third reason why this size has been chosen is because it is a right size for the chosen servos. Testing has been done with several servos, but all where smaller than the MG996R servo. Some of these servos had better performances but were not able to stably move the robots head due to their smaller size. The chosen size is a sufficient size to show all possibilities that come with the result of this project.

Figure 22 On the left the design of the prototype, on the right the design of Ravi

4.1.2 IMU

Overall, both IMU's work as intended and were suitable for the remaining of the project. After the 2 experiments, the decision has been made to continue working with the MPU-6050 sensor for the following reasons. Both sensors use a different library within the Arduino IDE software. The MPU-6050 sensor is a more often used IMU sensor than the GY-85. For this reason, more documentation and information is available online to back up on if needed during the rest of the project. This, in combination with the smoother output data and a sufficient response time makes it an excellent sensor to work with.

4.1.3 Arduino

The robot is controlled with an Arduino Uno with a Grove Shield attached on top. Grove provides an easy way to connect sensors and other components without the use of messy breadboard wiring. Not all components are connected to the Grove Shield since there were only limited Grove connection cables available. Arduino has a wide range of options and a quick way for rapid testing and prototyping.

4.1.4 Servos

One of the most important decisions within the project is which servos to use for the prototype. There were several requirements that the servo had to meet. Firstly, the requirement that the servo needs to be able to make rapid movements if the actors decides to make rapid head movements as well. The MG996R servo has a no-load speed of 0.14 sec/60º on a 6.0V input, which makes it possible to move the head from left to right (0 - 180º) 2 times within a second. This speed is sufficient for this project since an actor would, under normal circumstances, not rotate his head 180 degrees twice within 1 second. Furthermore, for the MG996R there are DOF brackets available that suite the servo's for a stable setup. Another benefit of using the MG996R servos is the fact that they are a lot cheaper than some other servo's of this size. The more expensive servos that are included in Ravi have higher precision, so the movement will only be more precise when implementing it on the actual robot.

4.2 The code

The robot is fully controlled through an Arduino code with the use of the Servo [36] library to manage the servos, the Adafruit Neopixel [37] library to manage the LED screens and the MPU6050 library [38] to extract the gyroscope data from the MPU-6050 IMU.

4.2.1 MPU6050

The MPU6050 library has been chosen because it easy to use, free and a lot of accessible information can be found online. Under the section getIMU(), shown in appendix I, the raw gyroscope is extracted and mapped into to values between 0 and 180 degrees since this are the limits of the servos. Noticed during all the testing is that occasionally errors occur where the value ramps up to either 0 or 180 for one sample, or even past 180 even though the constrain function within the Arduino IDE is used to constrain the data between 0 and 180. To prevent the servos to react on that and make a big jump, a smoothing equation has been added to smooth out the data that is going towards the servos. Figure 23 shows how the smoothened data looks compare to the raw data.

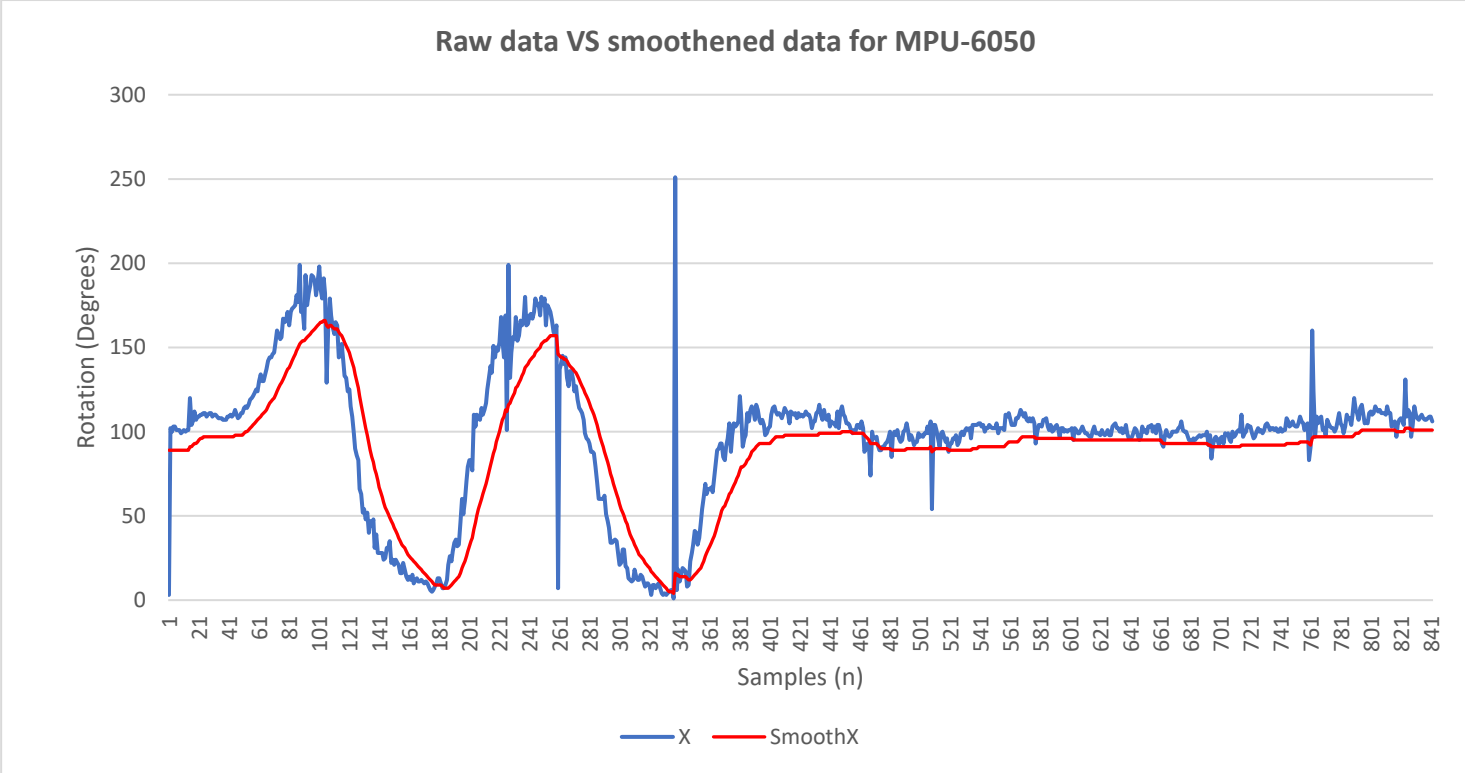


Figure 23 Raw data VS smoothened data by applying a formula over the collecting data

The smoothened data is created by storing the previous value:

```
servoX_Smooth_Int = (data.X * 0.07) + (servoX_PrevPos * 0.93);
```

Increasing the value that is multiplied by data.X, which is the newest received value, will lead to less smoothing of the data. By smoothening the data the head movement get's slightly delayed compared

to when you use the raw data. This slight delay of movement when looking at what we gain is an acceptable sacrifice. Moving from 0 to 180 degrees with the current smoothness applied takes ± 400 milliseconds. Due to the fact that movement from left to right is based on the tilt of the actors head, where an average person is not able to tilt it's head in such an angle, the maximum delay will always be less than 400 milliseconds

4.2.2 Servo

With the data provided by the MPU-6050, the servo can be moved accordingly. For this, there are 5 different cases. As mentioned during the conversation with the stakeholders of Sonnevanck, they would like to have an option to change the way the robot behaves depending on their movement. Each case shows how the robot would react on a movement of the actor.

Table 2 The head movement of the actor compare to the head movement of the robot for case 2, 3 & 4

Case 2			**Case 3**			**Case 4**	
Actor	**Robot**		**Actor**	**Robot**		**Actor**	**Robot**
Left	Left		Left	Right		Left	Right
Right	Right		Right	Left		Right	Left
Up	Up		Up	Up		Up	Down
Down	Down		Down	Down		Down	Up

Case 1 is where the operator is in full control of the robot, so the current situation. Table 2 above shows how the robotic head moves in comparison to the actors head for case 2, 3 and 4. Case 5 uses the same movement method as case 2, but it has a slight delay of 800 milliseconds. By doing this the head moves slowly behind the head of the actor which creates another type of experience for the actors to play with.

After the data has been gathered and adjusted depending on the case needed, the data then gets sent to the servos using the servo.h library on the Arduino IDE, which is an easily accessible library to control all kinds of servos.

4.2.3 hexagonal neopixel display

The hexagonal LED matrices are built upon the WS2812B LEDs. The Adafruit Neopixel library is an library that is suitable to use for these kind of LEDs, with many options to create graphic. The code used to create these visualizations is based on the code provides by S. de Jong [35]. He wrote a code

for this exact setup and how to control each individual LED or row of LEDs. Another tool from S. de Jong is used to create the visualizations of the eyes patterns [39]. The outcome of these visualizations are stored in byte value to be called later by the display function of the eyes.

They eye color is based on HSV, this is a method where colors are measured in degrees. Based on the data received from the MPU-6050, the color of the eyes are mapped between green and red, this results in 0 and 180 being red, and 90 being green. Anything in between those values is a graduate change in color. This gives a very smooth transition of lighting when the robotic head is moving. This feature is added to show how the use of IMU data can also be used to create automated lighting effects. The code is as follows:

```
int hueMapRight, hueMapLeft;
hueMapRight = map(data.X, 0, 90, 0, 21845);
hueMapLeft = map(data.X, 90, 180, 21845, 0);

if (data.X > 90) {
  display_eyes(slight, hueMapLeft);
}
if (data.X <= 90) {
  display_eyes(slight, hueMapRight);
}
```

The data.X is split between a right and left side, this to make it possible for both eyes to be mapped from red to green on the HSV circle.

In the scaling of HSV coloring, an angle of 90 degrees is equal to 21845. By mapping the gyroscope data to match those values you get a graduate change in color while moving left and right. Figure 24 shows how the color scaling works with the HSV method, and how the color will gradually change from red to green and green to red.

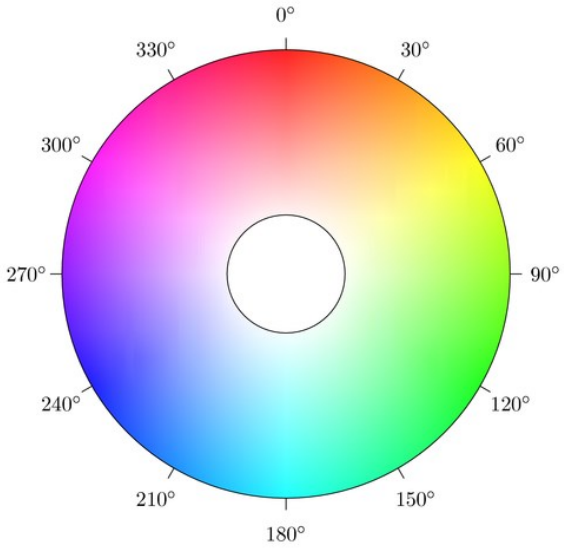


Figure 24 HSV with cylindrical geometries to determine color based on degrees.

Chapter 5: Evaluation

To validate the realization we needed to see whether we reached the requirements set by the stakeholders of Sonnevanck. The goal was to set up a final meeting to show the prototype and it's possibilities, as well as give the actors themselves the chance to play around with the final version. Unfortunately there was no possibility to set up a new meeting within the timeframe of this project due to their full summer schedule.

Some of the validation was already concluded during the meeting that is described in section 3.2.5 of this thesis. This validation was based on the existing prototype at that time, but after this meeting new features were implemented which needed validation. In order to do this, 4 video's were made displaying the features of the robotic head. Figure 25-28 show pictures of the recording, which can be found in the attached directory, that was send to the stakeholder in order to give them a clear view of the capabilities of the finalized prototype.

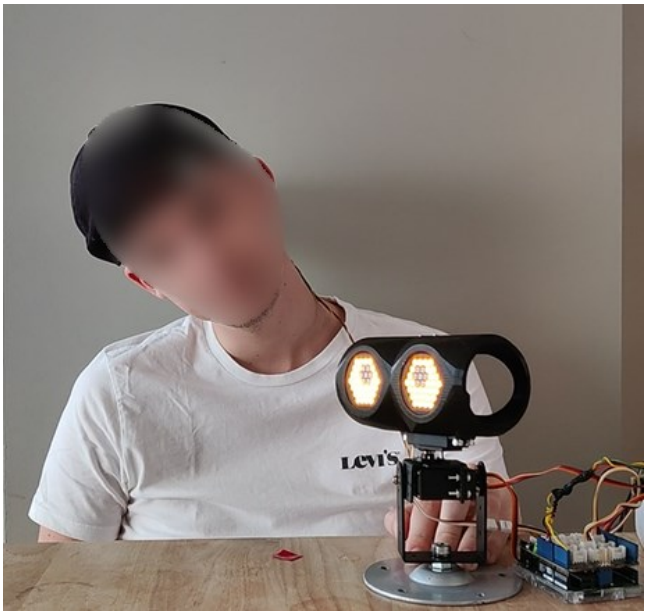


Figure 25 Showcase of case 2 where the head of the actor is the same as the head of the robot

Figure 26 Showcase of case 3 where the left/right movement of the robot head is opposite of the head movement of the actor

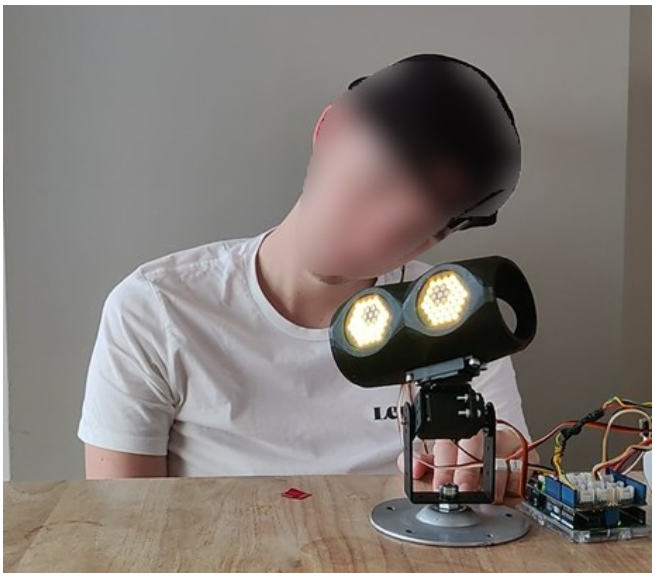


Figure 27 Showcase of case 4 where the movement of the robot head is the complete opposite of the actors head movement

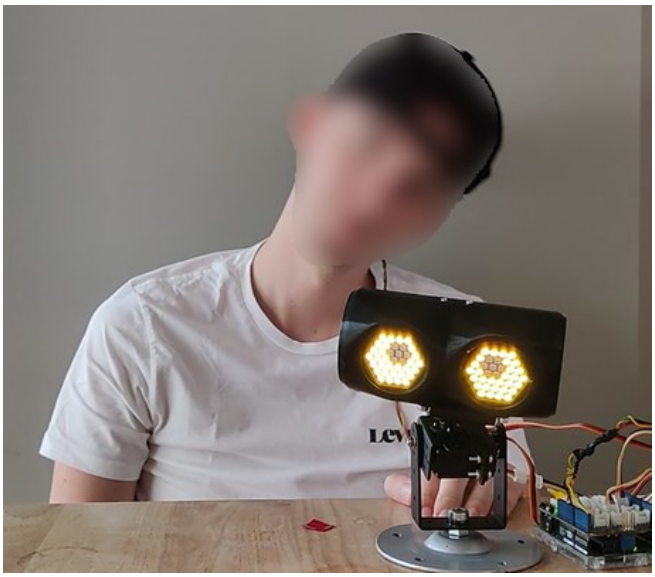


Figure 28 Showcase of case 5 where the movement of the robot head is delayed compare to the head movement of the actor

Based on the response of Daniel, one of the stakeholders of Sonnevanck, we can validate that the final prototype does meet almost all of the requirements. A part of the response is as follows:

"We fantasize about a new project with Ravi and this opportunity would be great to include in it. Offers many possibilities, also for the actor to be able to improvise."

Implementation of the suggestions/requirements given during the previous meeting made it possible for the actors to create different scenarios to play with, as well as improvise on stage cause the actions of the robot reflect directly of the movement of the actor. One of the requests of the stakeholders was the fact that they would prefer the connection to be wireless. This requirements was not met due to the issues with the twisting MG996R servos.

Chapter 6: Discussion & Future Work

The results of this graduation project are a solid basis for future work. The research demonstrates that there is a huge opportunity to integrate more autonomous robots into the theatrical environment. One of the key elements discovered within this research is that many robotics used in theatre are still manually operated or pre-programmed sequences are used to pretend that the robot is autonomous. This can be due to the fact that there are many limiting factors when designing autonomy for robots on stage. One of the best examples of those limitations found within this research is “My Square Lady” [3], where a fully autonomous robot operated based on what it learned from it’s surroundings. The actors got challenged to engage as emotionally as possible to capture the attention of Myon, with the possibility that while doing so, it still might not happen. This created a very dynamic play to work around the unpredictability of the robots autonomous behavior.

These research results were taken into account when considering the possibilities for this graduation project, which caused that during the design process the decision was made to not go for complete autonomy, but rather focus on how to create an interactive and engaging play between robot and actor with the use of sensor data.

There were a few limitations that impacted the creation of the final prototype. On of these is the fact that the connection between actor and robot is not wireless. During the testing of servos we noticed that the twisting happening, with Bluetooth connection and the MG996R servo, was not the case with the other servos, which shows that it is not an issue with the code. Regarding this issue, we can only assume that it is in internal communication issue between the Arduino and the specific servo in question. Further testing is needed on Ravi itself to see if the problem also occurs with the servos used for Ravi. Second limitation is that we were not able to implement the prototype on Ravi itself and expand it technology onto its arm & shoulders. This is beyond the scope of this project with the given time limit and is something future research could venture into.

Chapter 7: Conclusion

In this thesis, the primary research question explored the implementation of (partially) autonomous behavior for theatrical robot Ravi. The aim of this research was to identify how robots use autonomous behavior during theatrical performances, considering the problems and constraints that theatrical environments bring.

By researching the sub-questions, a comprehensive understanding of current and past robotic performances on stage, commonly used sensors for autonomous behavior, and the limitations of autonomy in a theatrical setting was obtained.

The wide exploration of the first sub-question, looking into already existing robot performances on stage, showed that a majority of them are controlled by operators or sequence programmed. Both of these approaches limit the ability for spontaneous interactions with actors.

Exploration of the seconds sub-question, the examination of sensors frequently used to create autonomous behavior. It was found that a combination of proprioceptive sensors such as IMU's and exteroceptive sensors such as camera's or lidar systems are commonly utilized to gather the data needed for autonomous behavior.

Regarding the third sub-question, the research identified various factors that can limit the feasibility of autonomy on theatrical stages. Limiting factors included low lighting conditions, dynamic environment and obstacles, and loud sounds that can exceed sensor thresholds.

Achieving full autonomy in a theatrical setting is a challenging project due to the limitations and constraints, but there are opportunities to create partial autonomy. With the use of the founded applicable sensors partial autonomy can enhance the overall theatrical play and performance.

The graduation work presented in this project holds significant relevance and demonstrates an application with potential widespread implications. By the creation of a robotic head where the movement is based on IMU sensor data captured from head movement offers a more engaging experience in theatrical play. It has the potential to create more natural and direct communication between human and robot. The base creation presented in this research creates a solid base to expand further up on, such as implementing the technology on other parts of the body to create a full robotic body movement based on a human body movement.

Overall, the graduation work presented in the project showcases the potential of implementing partial autonomous behavior on theatrical robots. Utilizing sensor data the capture head movement and translate it into real-time rotation of a robot opens up a lot of potential future research. The research contributes to the advancement of robotics technology in theatrical environments.

References

[1] H. Christiansen and A. M. Lindelof, “Robots on stage,” Roskilde, 2020.

[2] P. Choir, “Cod.Act,” 25 April 2019. [Online]. Available: https://codact.ch/works/pendulum-choir-2/. [Accessed 16 February 2023].

[3] M. S. Lady, “My Square Lady - Gob Squad,” 2015. [Online]. Available: https://www.gobsquad.com/projects/my-square-lady/. [Accessed 22 Februari 2023].

[4] M. E. S. T. C. Publications, “The (Dis-)Enchantment of the Theatre: Gob Squad’s My Square Lady at the Komische Oper,” 2016. [Online]. Available: https://europeanstages.org/2016/04/14/the-dis-enchantment-of-the-theatre-gob-squads-my-square-lady-at-the-komische-oper/. [Accessed 22 February 2023].

[5] “Copernicus Science Centre,” Centrum Nauki Kopernik, 7 September 2020. [Online]. Available: https://www.kopernik.org.pl/en/exhibitions/robotic-theatre. [Accessed 16 February 2023].

[6] W. Jackson, “Robotic Theatre: All in One Theatre By Engineered Arts,” Copernicus Science Centre, 2018. [Online]. Available: https://www.youtube.com/watch?v=t2t4BALBmxg. [Accessed 18 Februari 2023].

[7] R. Vos, “The beginning of a robot actor,” July 2021. [Online]. Available: https://essay.utwente.nl/87775/. [Accessed 18 February 2023].

[8] global-creatures, “WALKING WITH DINOSAURS – THE ARENA SPECTACULAR,” global-creatures, 2020. [Online]. Available: https://global-creatures.com/productions/walking-with-dinosaurs/. [Accessed 23 February 2023].

[9] T. C. T. C. P. LTD, “Creature Technology,” 2021. [Online]. Available: https://www.creaturetechnology.com/projects/walking-with-dinosaurs/. [Accessed 22 Februari 2023].

[10] I. Tech, “Behind The Scenes Of 'Walking With Dinosaurs' Show,” 20 August 2018. [Online]. Available: https://www.youtube.com/watch?v=WSTmvCS-FlA. [Accessed 3 March 2023].

[11] K. Wagner, “Here’s the story behind the pole-dancing robots everyone’s talking about at CES,” 9 January 2018. [Online]. Available: https://www.vox.com/2018/1/9/16870894/ces-2018-pole-dancing-robots-giles-walker-strip-club-las-vegas]. . [Accessed 22 March 2023].

[12] M. Blunden, “Brixton artist shows off robot pole-dancers at CES 2018,” 9 January 2018. [Online]. Available: https://www.standard.co.uk/tech/brixton-artist-shows-off-robot-poledancers-at-ces-2018-a3735671.html. [Accessed 22 March 2023].

[13] C. A. C. Dictionary, “Cambride Dictionary,” Cambridge University Press, [Online]. Available: https://dictionary.cambridge.org/dictionary/english/puppeteer. [Accessed 24 March 2013].

[14] “Dictionary.com,” [Online]. Available: https://www.dictionary.com/browse/preprogram. [Accessed 24 March 2023].

[15] W. contributors, “Autonomous robot,” WikipediaWikipedia, The Free Encyclopedia., 29 May 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Autonomous_robot&oldid=1157616640.

[16] “THEATER SONNEVANCK,” [Online]. Available: https://www.sonnevanck.nl/voorstelling/ravi-de-robot. [Accessed 25 March 2023].

[17] C. Bartneck, “Robots in the Theatre and the Media,” HIT Lab NZ, University of Canterbury Private Bag, Christchurch, 2013.

[18] J. M. Angel Fernandez and A. Bonarini, “Towards an Autonomous Theatrical Robot,” Humaine Association Conference on Affective Computing and Intelligent Interaction, Geneva, 2013.

[19] K. Katevas, P. G. Healey and M. T. Harris, “Robot Comedy Lab: experimenting with the social dynamics of live performance,” Queen Mary University of London, London, 2015.

[20] “Robot Stand-up: Engineering a Comic Performance,” 2015.

[21] J. Vilk and N. T. Fitter, “Comedians in Cafes Getting Data: Evaluating Timing and Adaptivity in Real-World Robot Comedy Performance,” International Conference on Human-Robot Interaction, 2020.

[22] D. Petrović, L. Kićinbaći, F. Petric and Z. Kovačić, “Autonomous Robots as Actors in Robotics Theatre - Tribute to the Centenary of R.U.R.,” 2019 European Conference on Mobile Robots (ECMR), Prague, 2019.

[23] E. A. Cappo, A. Desai, M. Collins and N. Michael, “Online planning for human–multi-robot interactive theatrical performance,” Auton Robot, 2018.

[24] J. Kelly and G. S. Sukhatme, “A General Framework for Temporal Calibration of Multiple Proprioceptive and Exteroceptive Sensors,” Berlin, 2014.

[25] “Meltec,” MELTEC Corporation, 2023. [Online]. Available: https://etching-meltec.com/optical-encoder/. [Accessed 2 April 2023].

[26] Editorial, “roboticsbiz,” November 2022. [Online]. Available: https://roboticsbiz.com/types-of-proprioceptive-sensors-for-mobile-robots/. [Accessed 2 April 2023].

[27] J. R. Nistler and M. F. Selekwa, “Gravity compensation in accelerometer measurements for robot navigation on inclined surfaces,” Department of Mechanical Engineering, Dakota, 2011.

[28] Editorial, “Roboticsbiz,” Editorial, 2022. [Online]. Available: https://roboticsbiz.com/types-of-exteroceptive-sensors-for-mobile-robots/. [Accessed 2 April 2023].

[29] G. U. and M. Parvis, “Noise-tolerant ultrasonic distance sensor based on a multiple driving approach,” Dipartimento di Elettronica, Torino, 1999.

[30] S. Support, “Samsung,” 23 November 2023. [Online]. Available: https://www.samsung.com/nz/support/mobile-devices/globalgalaxywhat-istof-camera/. [Accessed 2023 April 6].

[31] R. Reilink, “Realtime Stereo Vision Processing,” University of Twente, Twente, 2008.

[32] B. Hu, P. Tunison and S. Schmitt, “Kitware,” 27 September 2022. [Online]. Available: https://www.kitware.com/understanding-ai-with-saliency-maps/. [Accessed 17 April 2023].

[33] L. Seeed Technology Co., “seeed studio,” 2023. [Online]. Available: https://www.seeedstudio.com/Seeed-XIAO-BLE-Sense-nRF52840-p-5253.html.

[34] E. Dertien, “tail2speech,” 14 10 2021. [Online]. Available: https://github.com/edwindertien/tail2speech.

[35] S. d. Jong, “Building the Grovebot,” 12 9 2022. [Online]. Available: https://github.com/utwente-interaction-lab/interaction-lab/wiki/Building-the-GroveBot.

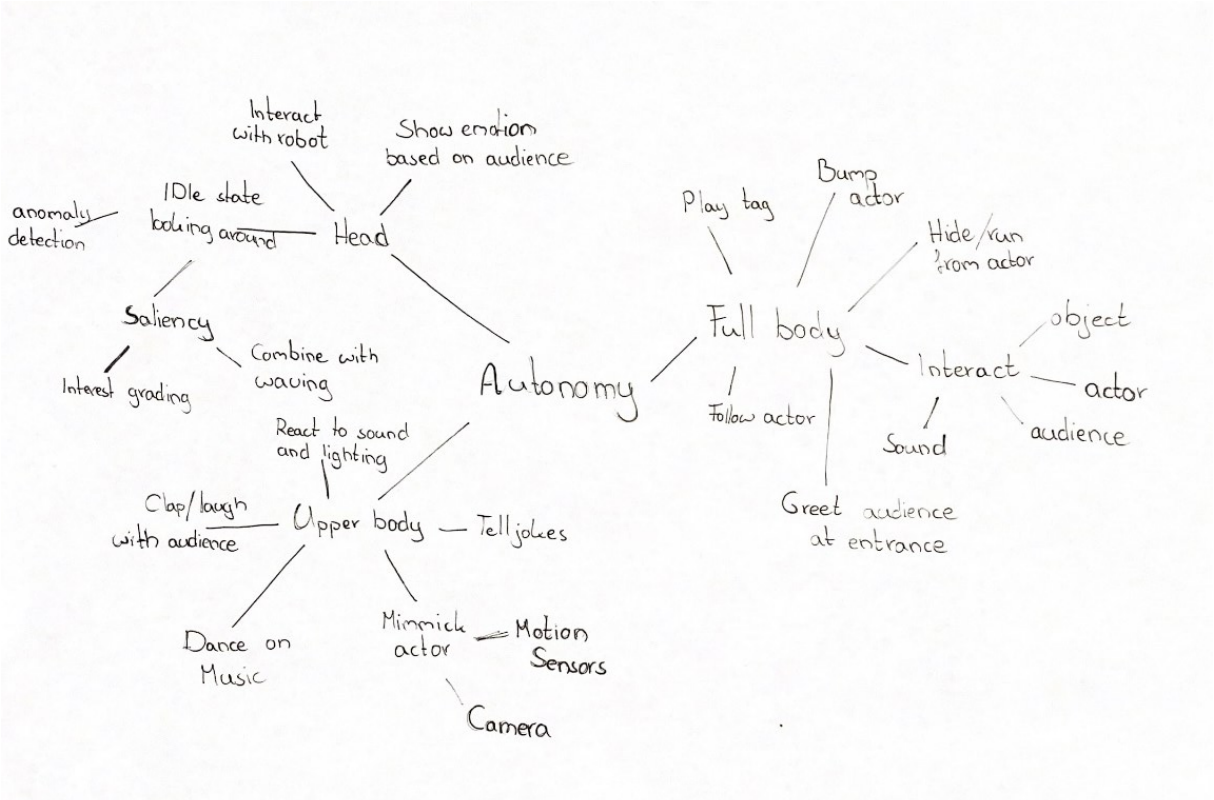
[36] Arduino, “Servo,” [Online]. Available: https://www.arduino.cc/reference/en/libraries/servo/.

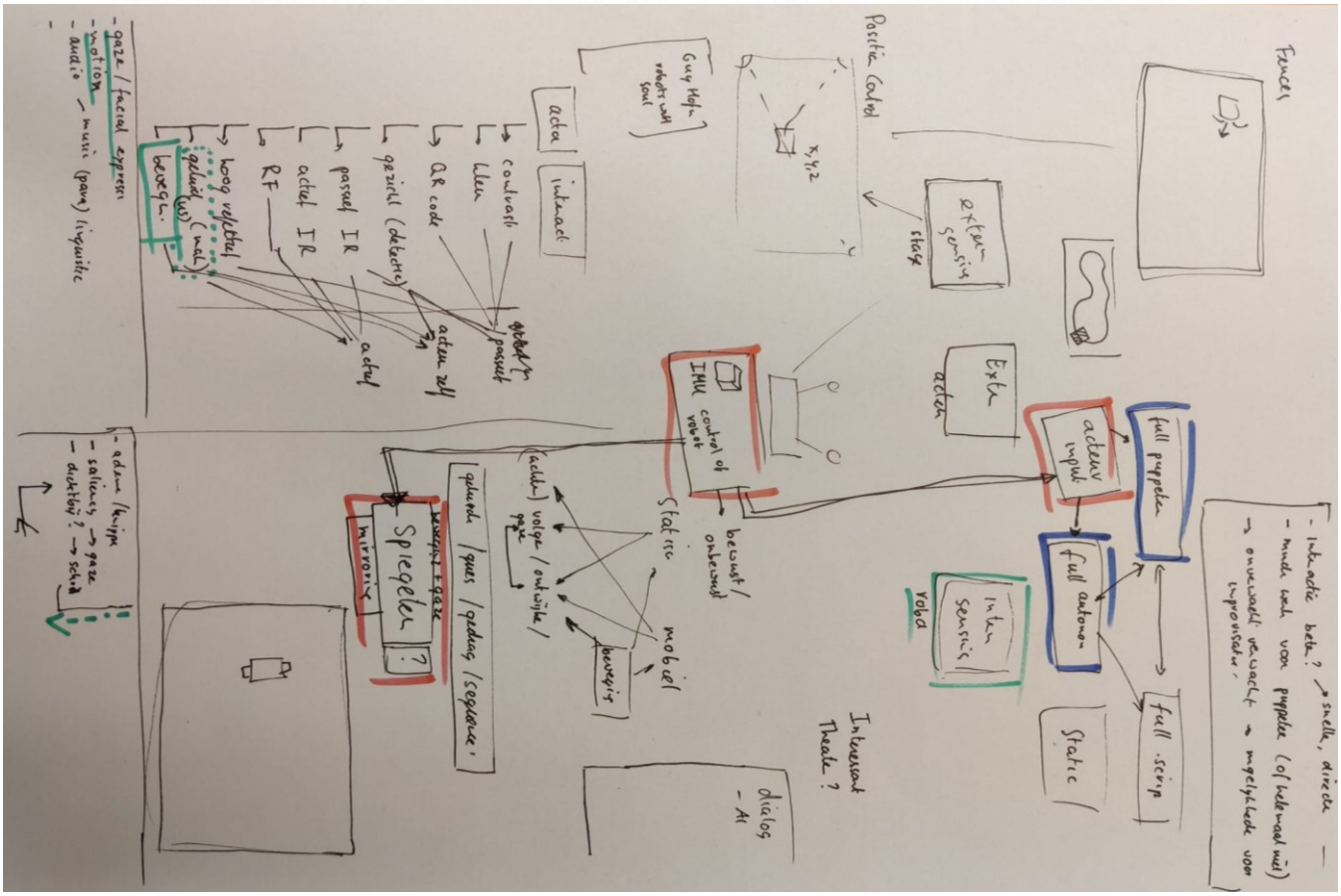
[37] A. Industries, “Adafruit_NeoPixel,” Adafruit Industries, [Online]. Available: https://github.com/adafruit/Adafruit_NeoPixel.

[38] AdaFruit, “Adafruit_MPU6050,” Adafruit Industries, [Online]. Available: https://github.com/adafruit/Adafruit_MPU6050.

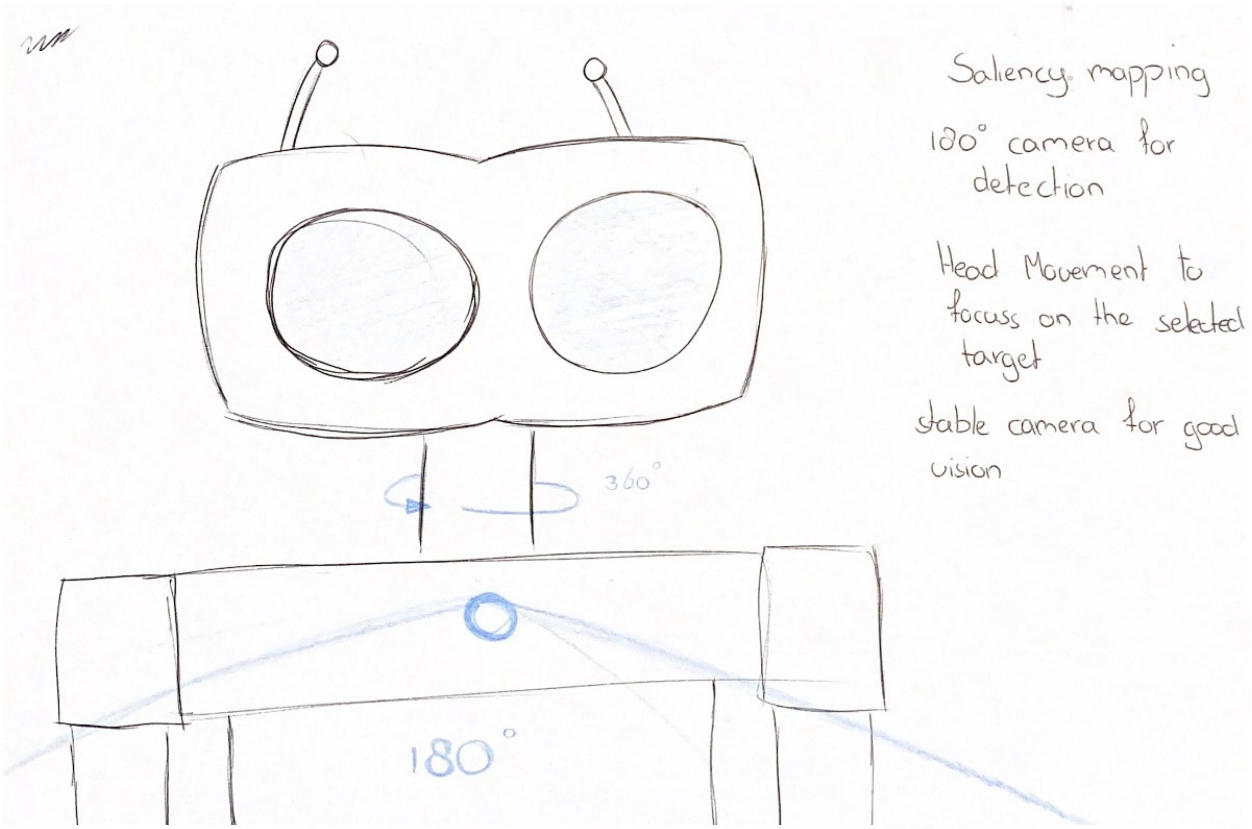
[39] S. d. Jong, “eyes,” [Online]. Available: https://sjoerd.tech/eyes/.

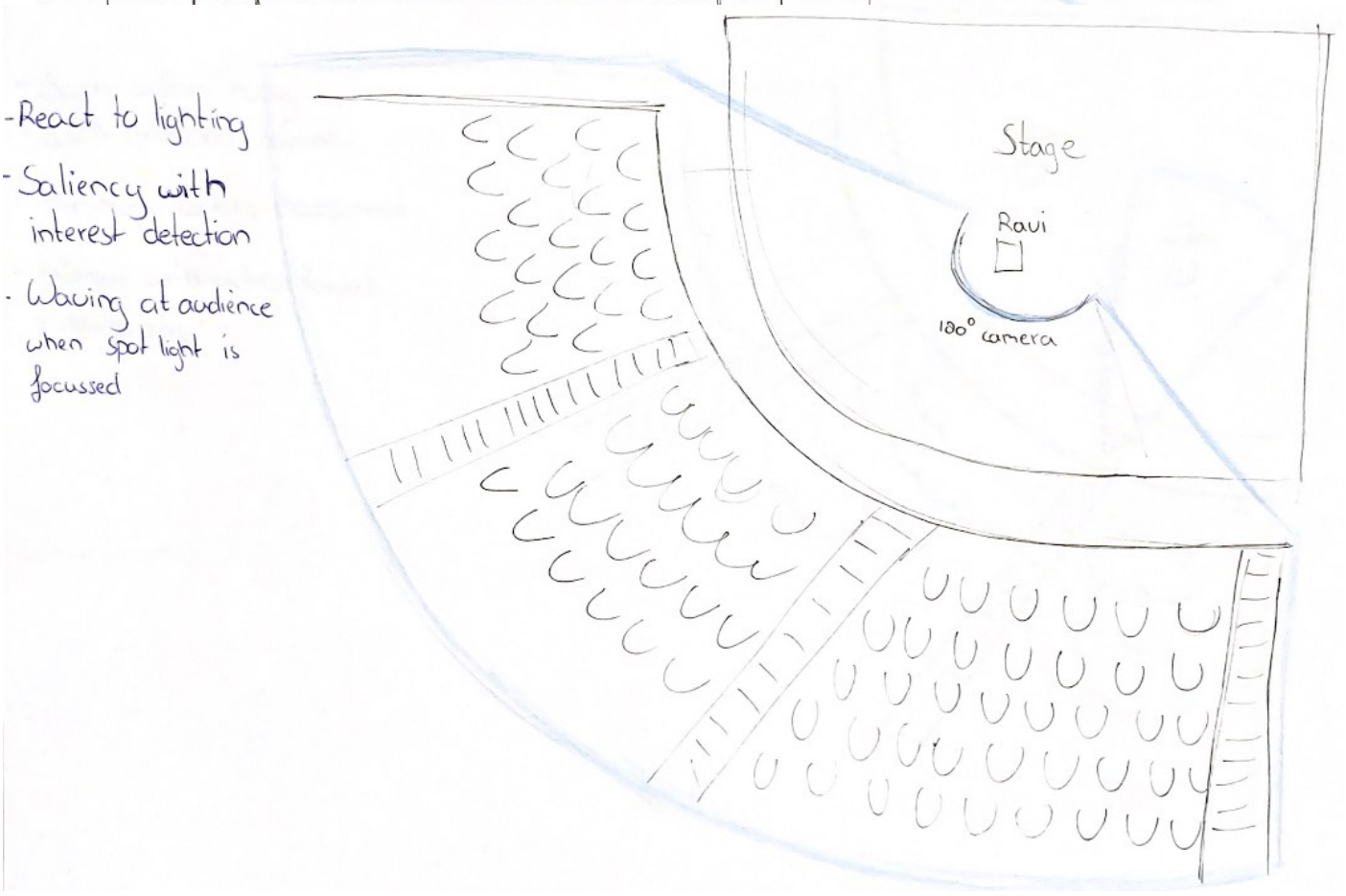
APPENDIX A


Interact with robot
Show emotion based on audience
Idle state looking around
anomaly detection
Head
Saliency
Interest grading
Combine with waving
Autonomy
React to sound and lighting
Clap/laugh with audience
Upper body
Tell jokes
Dance on Music
Mimmick actor
Motion Sensors
Camera
Play tag
Bump actor
Hide/run from actor
Full body
Follow actor
Greet audience at entrance
Interact
object
actor
audience
Sound

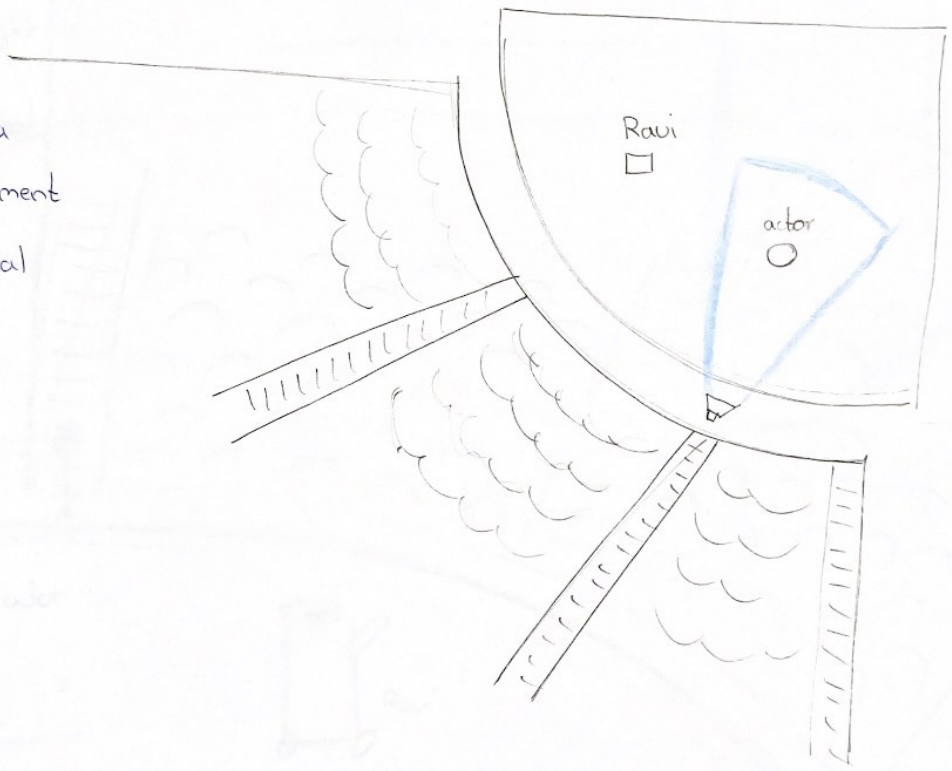


APPENDIX B

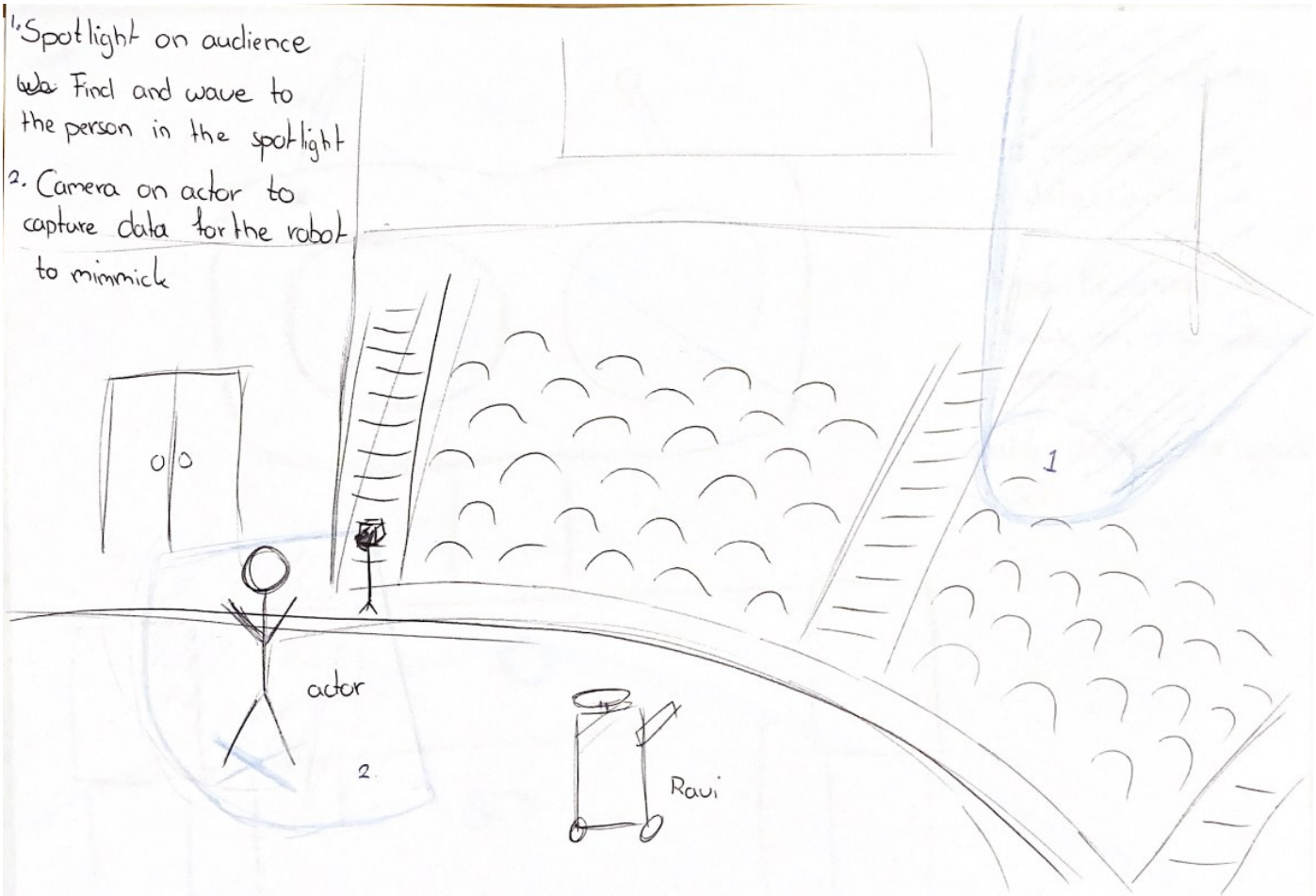

Saliency mapping
100° camera for detection
Head Movement to focuss on the selected target
stable camera for good vision
360°
180°


-React to lighting
- Saliency with interest detection
- Waving at audience when spot light is focussed
Stage
Ravi
180° camera

- Scan actors body with external camera
- Mimmick actors movement
- Interact with actors facial expressions



1. Spotlight on audience
2. Find and wave to the person in the spotlight
3. Camera on actor to capture data for the robot to mimmick



APPENDIX C

```
#include "MPU6050.h"

MPU6050 mpu;

int16_t ax, ay, az;
int16_t gx, gy, gz;

struct MyData {
  byte X;
  byte Y;
  byte Z;
};

MyData data;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  mpu.initialize();
}

void loop() {
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  data.X = map(ax, -17000, 17000, 0, 250); // X axis data
  data.Y = map(ay, -17000, 17000, 0, 250);
  //delay(500);
  Serial.print(data.X);
  Serial.print(",");
  Serial.println(data.Y);
}
```

APPENDIX D

```
#include "GY_85.h"

GY_85 GY85;

void setup() {
  Serial.begin(115200);  //115200
  GY85.init();
  delay(10);
}

void loop() {
  int ax = GY85.accelerometer_x(GY85.readFromAccelerometer());
  int ay = GY85.accelerometer_y(GY85.readFromAccelerometer());
  int az = GY85.accelerometer_z(GY85.readFromAccelerometer());
  if (ax > 50000) ax = ax - 65535;
  if (ay > 50000) ay = ay - 65535;
  if (az > 50000) az = az - 65535;

  ax = map(ax, -138, 140, -180, 180);
  ay = map(ay, -131, 145, -180, 180);

  Serial.print(ax);
  Serial.print(",");
  Serial.println(ay);
}
```

APPENDIX E

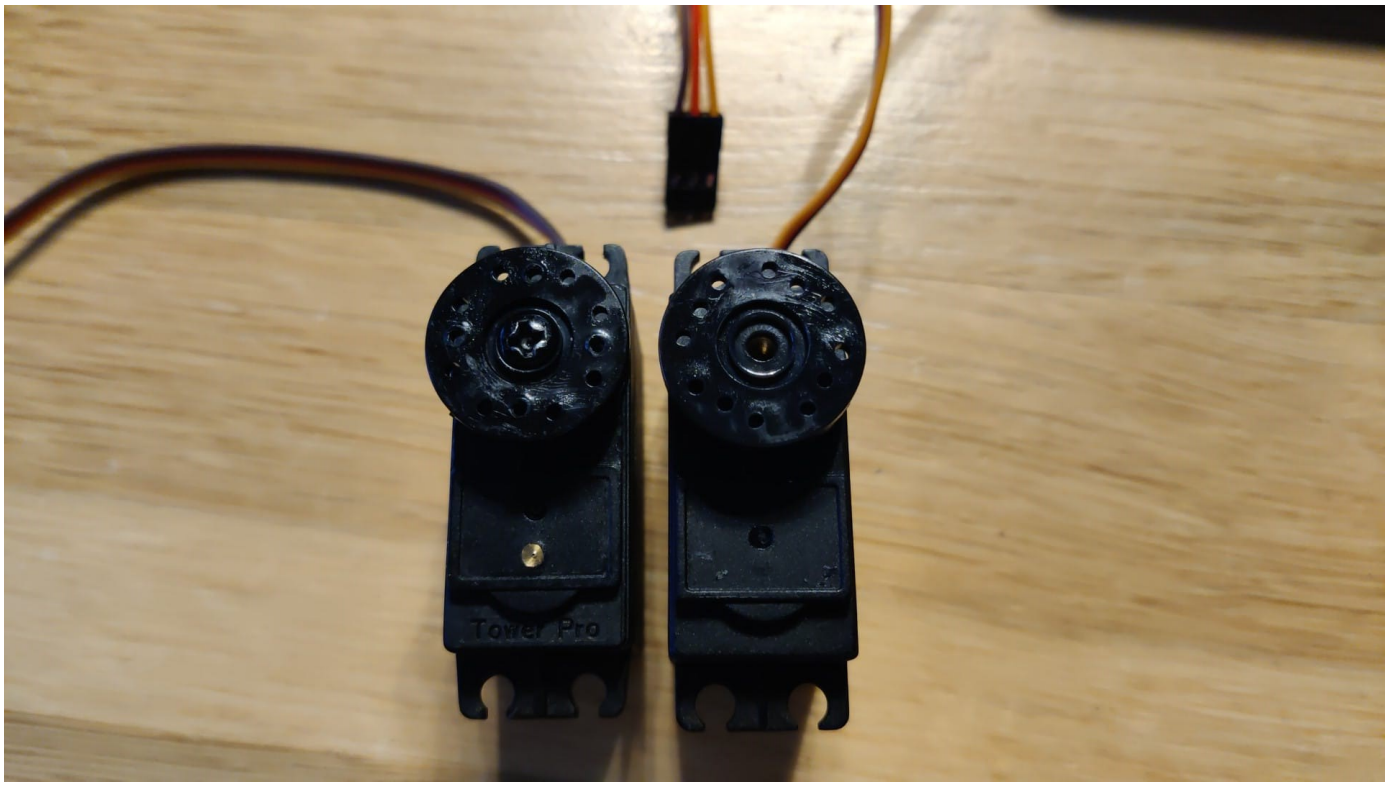
```
#include <SoftwareSerial.h>
#include "MPU6050.h"

SoftwareSerial bluetooth(10, 11); //RX, TX
int input;
MPU6050 mpu;

int16_t ax, ay, az;
int16_t gx, gy, gz;

struct MyData {
  byte X;
  byte Y;
  byte Z;
};

MyData data;

void setup()
{
  Serial.begin(9600);
  bluetooth.begin(9600);
  pinMode(device, INPUT);
}

void loop()
{
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  data.X = map(ax, -17000, 17000, 0, 180);  // X axis data
  data.Y = map(ay, -17000, 17000, 0, 180);
  data.Z = map(az, -17000, 17000, 0, 180);

  data.X = constrain(data.X, 0, 180);
  data.Y = constrain(data.Y, 0, 180);


  bluetooth.print(data.X);
  bluetooth.print(",");
  bluetooth.println(data.Y);

}
```

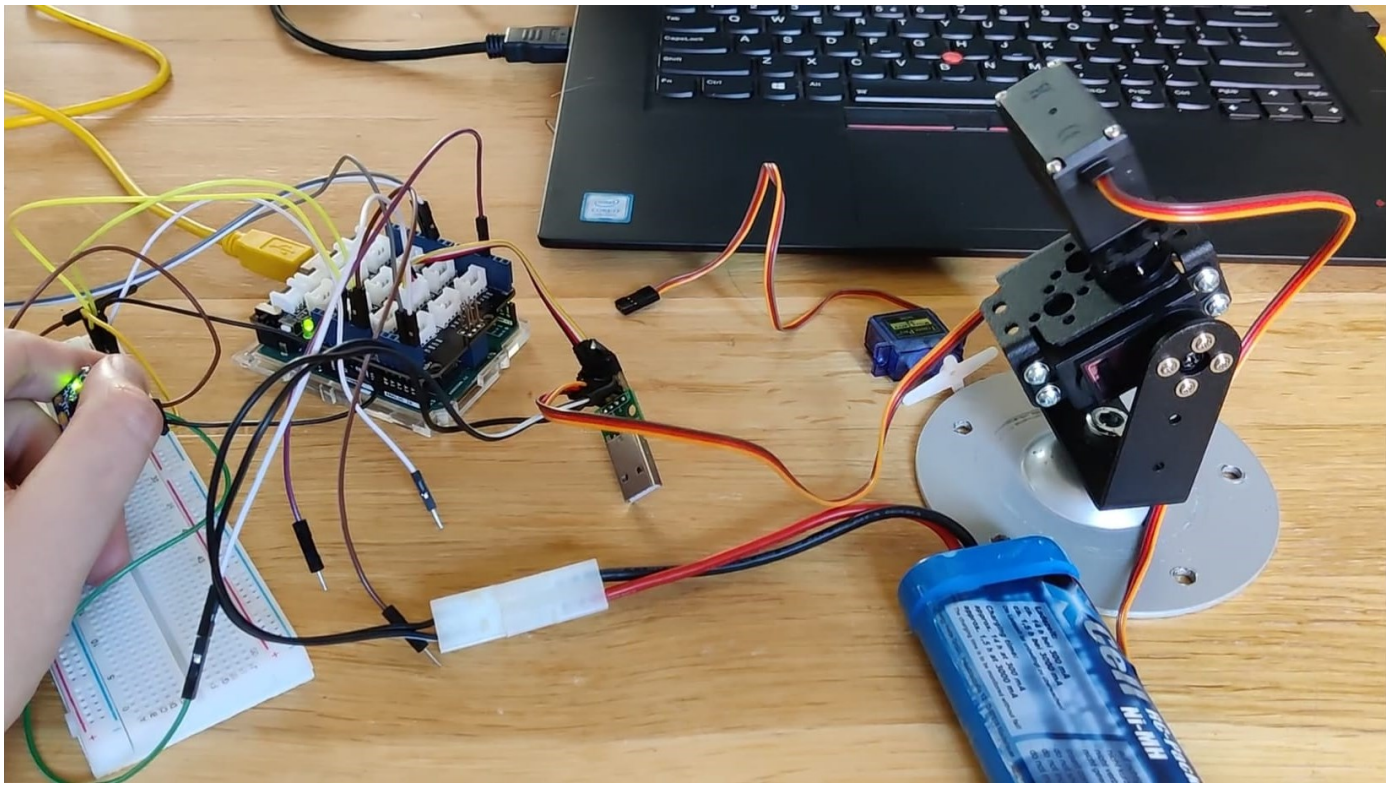
APPENDIX F

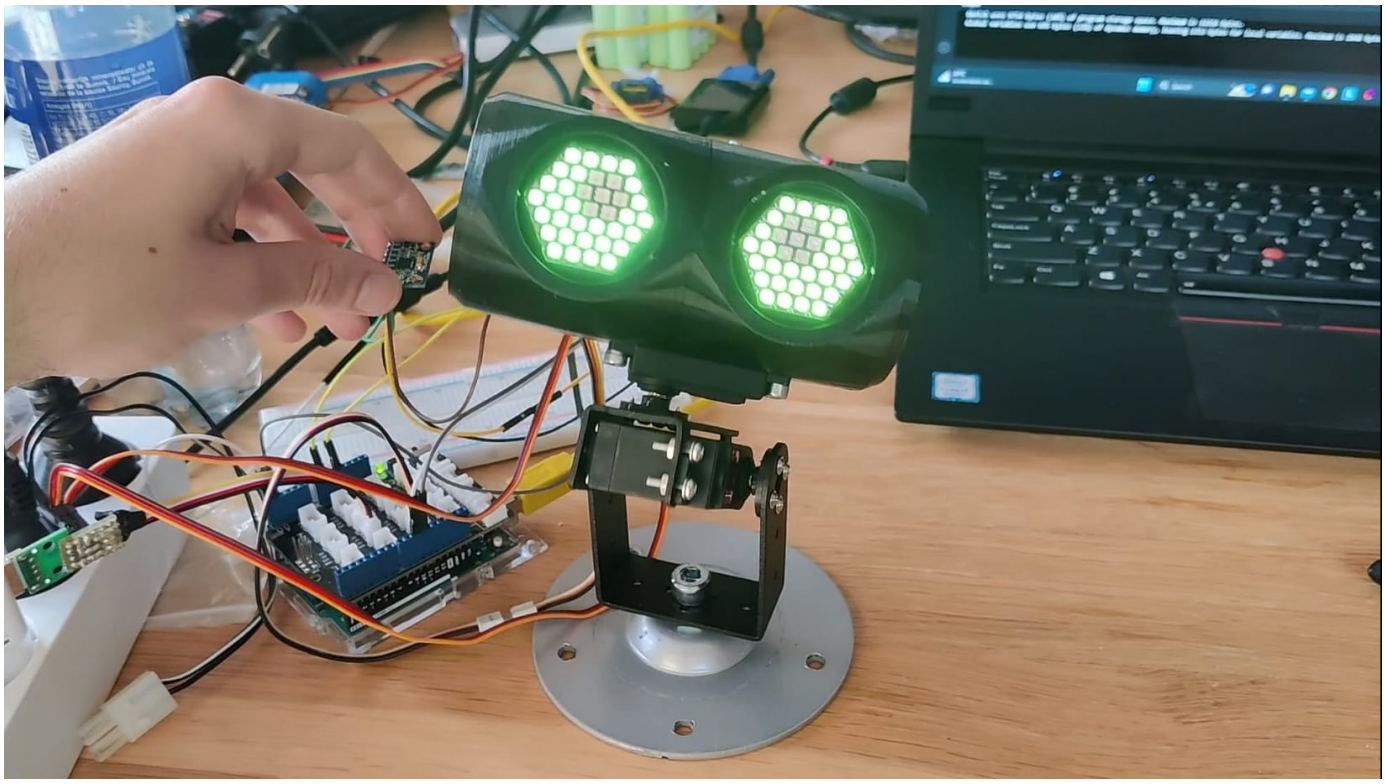
```
#include "GY_85.h"
#include <Wire.h>
GY_85 GY85;  //create the object

#include "BluetoothSerial.h"
BluetoothSerial bluetooth;

#define LED_STATE_BLUE 2  //Check if bluetooth is connected

// Mac address HC-05 of arduino shield
// HIDDEN IN DOCUMENTATION

// Mac address laptop
// HIDDEN IN DOCUMENTATION

// On the dots of the mac address put the address of your target device

uint8_t mac_address[] = { 0x.., 0x.., 0x.., 0x.., 0x., 0x.. };
long timer1;

int servoX_Target_Int, servoY_Target_Int;

void setup() {
  Serial.begin(115200);  //115200

  bluetooth.begin("IMU-sensor", true);  //Bluetooth device name
  bluetooth.connect(mac_address);

  pinMode(LED_STATE_BLUE, OUTPUT);

  Wire.begin();
  delay(10);
  GY85.init();
  delay(10);
}

unsigned long looptime;
void loop() {

  if (bluetooth.hasClient() == true) {
    digitalWrite(LED_STATE_BLUE, HIGH);

    int ax = GY85.accelerometer_x(GY85.readFromAccelerometer());
    int ay = GY85.accelerometer_y(GY85.readFromAccelerometer());
```

```
    int az = GY85.accelerometer_z(GY85.readFromAccelerometer());
    if (ax > 50000) ax = ax - 65535;
    if (ay > 50000) ay = ay - 65535;
    if (az > 50000) az = az - 65535;

  if (millis() > looptime + 10) {
    looptime = millis();
    bluetooth.print(servoX_Target_Int);
    bluetooth.print(",");
    bluetooth.println(servoY_Target_Int);

    Serial.print(servoX_Target_Int);
    Serial.print(",");
    Serial.println(servoY_Target_Int);
  }
  }
}


//98D3:71:F6AE33
```

APPENDIX G







APPENDIX H

Information letter for research with human participants for graduation project “More autonomy for Ravi”

Authors: David Lammers (based on template by BMS EC)

Last edited: 19-04-2023

The purpose of the this research is to explore the opportunities of autonomy in theatrical environments. The project will be based on an already existing, currently puppeteered robot called ‘Ravi’. The robot Ravi has been used previously by the theater group ‘Theater Sonnevanck’. Currently the robot has no autonomous behavior so the goal is to explore the use of autonomous behavior on the robot Ravi, highlighting benefits, limitations and also the possibilities that it will create for the future of performing art on stage. The goal of the conversation is to gain an insight on the opinion of experts on the currents stage of the project as well as provide insides for the researcher.

During this session, the researcher (in this case David Lammers, student at the University of Twente) will have a conversation with professionals in the space of theatre and interaction. The duration of the session will be dependent on the conversation. As participant in this study you are allowed to refuse answering questions and you are allowed to withdraw from the research or conversation at any given point without having to give a reason.

There are no risks involved in participating in this research, as the participant you are able to choose where and how to meet the researcher, either physical or online. If given consent, the conversation will be recorded to be reviewed later by the researcher for research and documentational purposes. The recording will be destroyed after the research has been completed. If given consent, personal information including full name, job and experience will be used in the final documentation to provide proof and validation of the conversation that took place between the participant and researcher.

If you have questions regarding the research, or wish to obtain more information, please contact the researcher David Lammers: d.n.b.lammers@student.utwente.nl or the supervisor of the project Edwin Dertien: e.dertien@utwente.nl

If you have questions about your rights as a research participant, or wish to obtain information, ask questions, or discuss any concerns about this study with someone other than the researcher(s), please contact the Secretary of the Ethics Committee Information & Computer Science: ethicscommittee-CIS@utwente.nl

Consent Form for research more autonomy for Ravi

YOU WILL BE GIVEN A COPY OF THIS INFORMED CONSENT FORM

Please tick the appropriate boxes	Yes	No
Taking part in the study		
I have read and understood the study information dated [19-04-2023], 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.	□	□
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.	□	□
I understand that taking part in the study involves having a conversation with the researcher about the use of autonomous robots in theatrical environments. The conversation will be recorded as an audio recording and the recording will be destroyed once the research is completed.	□	□
Use of the information in the study		
I understand that information I provide will be used in the final documentation of this graduation project.	□	□
I understand that personal information collected about me that can identify me, such as name or job, will not be shared beyond the study team without consent.	□	□
I agree that my information can be quoted in research outputs	□	□
I agree that my real name can be used for quotes	□	□
I agree to be audio recorded.	□	□

Future use and reuse of the information by others

I give permission for the information that I provide to be archived in the final documentation so it can be used for future research and learning. □ □

Signatures

_____ _____

Name of participant [printed] Signature Date

I 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.

David Lammers _____ _

_____

Researcher name [printed] Signature Date

Study contact details for further information:

Researcher: David Lammers, d.n.b.lammers@student.utwente.nl

Supervisor: Edwin Dertien, e.dertien@utwente.nl

APPENDIX I

```
#include <SoftwareSerial.h>
#include <Servo.h>
#include <Adafruit_NeoPixel.h>
#include "MPU6050.h"
#include "Wire.h"

//  Global constants
#define RX 10  // RX on arduino UNO to TX on HC-05 module
#define TX 11  // TX on arduino UNO to RX on HC-05 module
#define SERVO_PIN_1 6
#define SERVO_PIN_2 7
#define LED_PIN 4
#define NUMPIXELS 74

MPU6050 mpu;
Adafruit_NeoPixel pixels(NUMPIXELS, LED_PIN);

int LED_BRIGHTNESS = 20;

// enum Emotion {STILL, SLIGHT, HEAVY, ANGRY, SAD};
// Emotion emotion = NEUTRAL;

int16_t ax, ay, az;
int16_t gx, gy, gz;

struct MyData {
  byte X;
  byte Y;
  byte Z;
};

MyData data;
Servo servo1, servo2;

// And the rest
SoftwareSerial bluetooth(RX, TX);

int servoX_Target = 90, servoY_Target = 90;
int servoX_Target_Int = 90, servoY_Target_Int = 90;
int servoX_Smooth_Int = 90, servoY_Smooth_Int = 90;
int servoX_PrevPos, servoY_PrevPos;
int var = 1, storeVar = 1;

long timer1, timer2;

// ---------------------------------------------------------------------------
------ //
```

```
// ---------------------------------- EYE PATTERNS ---------------------------
------ //
// ---------------------------------------------------------------------------
------ //
byte still[]{
  B0110,
  B01110,
  B011110,
  B0111110,
  B011110,
  B01010,
  B0000
};

byte slight[]{
  B1111,
  B11111,
  B111111,
  B1100111,
  B100011,
  B10011,
  B1111
};

byte heavy[]{
  B1001,
  B01010,
  B001100,
  B0001000,
  B001100,
  B01010,
  B1001
};

byte angry[] = {
  B1100,
  B10100,
  B100100,
  B0100100,
  B101110,
  B10001,
  B1110
};

byte sad[] = {
  B0000,
  B00000,
  B000110,
  B0011111,
```

```
  B011111,
  B11111,
  B1110
};



void setup() {
  Serial.begin(9600);
  Serial.setTimeout(1);
  // bluetooth.begin(9600);

  // Start IMU
  Wire.begin();
  mpu.initialize();

  // Set servos
  servo1.attach(SERVO_PIN_1);  // TOP
  servo2.attach(SERVO_PIN_2);  // BOTTOM
  servo1.write(90);
  delay(100);
  servo2.write(90);
  delay(1000);

  // Initialize leds
  pixels.begin();
}

void loop() {
  //getIMU();

  move_servos();
  run_eyes();
}

void receiveData() {
  String data = "";
  if (bluetooth.available()) {
    data = bluetooth.readStringUntil('\n');
    servoX_Target_Int = bluetooth.readStringUntil(',').toInt();   // Get X
position
    servoY_Target_Int = bluetooth.readStringUntil('\n').toInt();  // Get Y
position

    servoX_Smooth_Int = (servoX_Target_Int * 0.05) + (servoX_PrevPos * 0.95);
    servoY_Smooth_Int = (servoY_Target_Int * 0.05) + (servoY_PrevPos * 0.95);

    servoX_PrevPos = servoX_Smooth_Int;
    servoY_PrevPos = servoY_Smooth_Int;
```

```
  }
}

// ------------------------------------------------------------------------
------ //
// ------------------------------------ IMU -------------------------------
----- //
// ------------------------------------------------------------------------
------ //
void getIMU() {
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  data.X = map(ax, -17000, 17000, 0, 180);  // X axis data
  data.Y = map(ay, -17000, 17000, 0, 180);
  data.Z = map(az, -17000, 17000, 0, 180);  // Y axis data

  data.X = constrain(data.X, 0, 180);
  data.Y = constrain(data.Y, 0, 180);
}


// ------------------------------------------------------------------------
------ //
// ---------------------------------- SERVO -------------------------------
------- //
// ------------------------------------------------------------------------
------ //
void move_servos() {

  var = Serial.parseInt();
  if (var != 0) {
    storeVar = var;
  }
  Serial.println(storeVar);

  switch (storeVar) {
    case 1:  // Reset
      {
        getIMU();
        servo1.write(90);
        servo2.write(90);
      }
      break;

    case 2:  // Regular setting where the head follows the actors head
      {

        while (true) {
          getIMU();
```

```
          servoX_Smooth_Int = (data.X * 0.07) + (servoX_PrevPos * 0.93);
          servoY_Smooth_Int = (data.Y * 0.07) + (servoY_PrevPos * 0.93);

          servoX_PrevPos = servoX_Smooth_Int;
          servoY_PrevPos = servoY_Smooth_Int;
          servo1.write(servoY_Smooth_Int);
          servo2.write(servoX_Smooth_Int);

          var = Serial.parseInt();
          if (var != 2) {
            break;
          }
        }
      }
      break;
    case 3:  // Setting where the upper servo rotates in the opposite
direction of the actors head
      {

        while (true) {
          getIMU();

          data.X = 180 - data.X;

          servoX_Smooth_Int = (data.X * 0.07) + (servoX_PrevPos * 0.93);
          servoY_Smooth_Int = (data.Y * 0.07) + (servoY_PrevPos * 0.93);

          servoX_PrevPos = servoX_Smooth_Int;
          servoY_PrevPos = servoY_Smooth_Int;
          servo1.write(servoY_Smooth_Int);
          servo2.write(servoX_Smooth_Int);

          var = Serial.parseInt();
          if (var != 3) {
            break;
          }
        }
      }
      break;
    case 4:  // Setting where the upper AND lower servo rotate in the opposite
direction of the actors head
      {
        while (true) {
          getIMU();

          data.X = 180 - data.X;
          data.Y = 180 - data.Y;

          servoX_Smooth_Int = (data.X * 0.07) + (servoX_PrevPos * 0.93);
```

```
          servoY_Smooth_Int = (data.Y * 0.07) + (servoY_PrevPos * 0.93);

          servoX_PrevPos = servoX_Smooth_Int;
          servoY_PrevPos = servoY_Smooth_Int;
          servo1.write(servoY_Smooth_Int);
          servo2.write(servoX_Smooth_Int);

          var = Serial.parseInt();
          if (var != 4) {
            break;
          }
        }
      }
      break;
    case 5:  // Setting with extreme slowness
      {
        while (true) {
          getIMU();

          servoX_Smooth_Int = (data.X * 0.02) + (servoX_PrevPos * 0.98);
          servoY_Smooth_Int = (data.Y * 0.02) + (servoY_PrevPos * 0.98);

          servoX_PrevPos = servoX_Smooth_Int;
          servoY_PrevPos = servoY_Smooth_Int;
          servo1.write(servoY_Smooth_Int);
          servo2.write(servoX_Smooth_Int);

          var = Serial.parseInt();
          if (var != 5) {
            break;
          }
        }
      }
      break;
  }
}

// ------------------------------------------------------------------------
------ //
// ------------------------------------ EYES ------------------------------
------ //
// ------------------------------------------------------------------------
------ //
void display_eyes(byte arr[], int hue) {
  display_eye(arr, hue, true);
  display_eye(arr, hue, false);
}

void display_eye(byte arr[], int hue, bool left) {
```

```
  // We will draw a circle on the display
  // It is a hexagonal matrix, which means we have to do some math to know
where each pixel is on the screen

  int rows[] = { 4, 5, 6, 7, 6, 5, 4 };  // The matrix has 4, 5, 6, 7, 6, 5, 4
rows.
  int NUM_COLUMNS = 7;                   // There are 7 columns
  int index = (left) ? 0 : 37;           // If we draw the left eye, we have
to add an offset of 37 (4+5+6+7+6=5+4)
  for (int i = 0; i < NUM_COLUMNS; i++) {
    for (int j = 0; j < rows[i]; j++) {
      int brightness = LED_BRIGHTNESS * bitRead(arr[i], (left) ? rows[i] - 1 -
j : j);
      pixels.setPixelColor(index, pixels.ColorHSV(hue, 255, brightness));
      // pixels.setPixelColor(index, r, g, b);
      // pixels.setBrightness(brightness)
      index++;
    }
  }
}

void run_eyes() {
  pixels.clear();

  int hueMapRight, hueMapLeft;
  hueMapRight = map(data.X, 0, 90, 0, 21845);
  hueMapLeft = map(data.X, 90, 180, 21845, 0);

  if (data.X > 90) {
    display_eyes(slight, hueMapLeft);
  }
  if (data.X <= 90) {
    display_eyes(slight, hueMapRight);
  }

  // if (data.X > 90) {
  //   display_eyes(still, hueMapLeft);
  //   if (data.X > 60 && data.X < 150) {
  //     display_eyes(slight, hueMapLeft);
  //   } else if (data.X > 150) {
  //     display_eyes(heavy, hueMapLeft);
  //   }
  // }
  // if (data.X <= 90) {
  //   display_eyes(still, hueMapRight);
  //   if (data.X > 30 && data.X < 60) {
  //     display_eyes(slight, hueMapRight);
  //   } else if (data.X < 30) {
  //     display_eyes(heavy, hueMapRight);
```

```
  //   }
  // }

  pixels.show();
}
```