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

Facilitating Smooth Handovers with a Gesture-Control Glove for Assistive Robotic Manipulators

Dilara Yigit

S3336433

Department of Discrete Technology and Production Automation

Period: 15/04/2024- 01/07/2024

Bachelor's project

1st Examiner: Dr. Elisabeth Wilhelm, department of Discrete Technology and Production Automation

2nd Examiner: Dr. Ir. René Fluit, Faculty of Science and Engineering

ABSTRACT

Individuals who survive a stroke often require assistance to be able to perform basic tasks. Robotic aids, such as Assistive Robotic Manipulators (ARMs) can be used to help stroke survivors perform tasks, by grasping and handing over items to the patient. Robotic handover is an essential part of human-robot interaction (HRI). Implementing intuitive robot to human handover is still challenging to this day. In this project, a glove facilitating gesture control was designed to improve robot-to-user handover. Two MPU-6050 sensors and two flex sensors were embedded onto a glove to provide orientation data of the hand and sense bending of the fingers, respectively. Handover was controlled by movements of the hand wearing the glove. Data from the glove was sent to the ARM to move the manipulator between positions while independently operating the gripper. Testing was conducted by handing over five different items (cardboard coffee cup, half-filled bottle of water, empty bottle of water, cardboard box of tea, whiteboard eraser) twenty times each. Results showed an average handover time of 7.76 seconds with an error rate of 18.03%. There was a significant difference in handover times between tested items. Possible factors that impacted the results are testing environment, the same user performing all tests and hardware limitations. The design shows promise in being a fast and intuitive method at facilitating robotic handover. Future research is needed to improve success rate and determine compatibility with the target demographic.

Table of Contents

INTRODUCTION

In 2020, 38,500 individuals in the Netherlands experienced a stroke, equating to over a hundred people suffering from a stroke each day [1]. Over half of the patients who survive a stroke require long-term care to be able to perform basic tasks and activities in their daily lives [2]. To aid patients in this, assistive robotic manipulators (ARMs) can be used. ARMs are designed to help people with upper or lower limb disabilities, such as stroke survivors, by performing small but essential tasks [3]. These tasks include opening doors, assisting with eating, and grasping objects. To perform these tasks successfully, the human and ARM need to interact and work together. Studies show that mimicking human to human behaviours in human-robot interaction (HRI) allow for smoother and more intuitive actions [4]. This is still something ARMs struggle with, as they have difficulties replicating human communication necessary for these interactions [5]. This project aimed to develop a method for seamless HRI, with a focus on facilitating smooth robot-to-human handovers, specifically designed for stroke survivors.

Problem definition

A stroke is a medical emergency in which the blood flow to parts of the brain is blocked, or sudden bleeding occurs in parts of the brain [6]. Generally, only 10% of stroke patients make a full recovery, and over 50% of stroke survivors sustain impairments that require special or long-term care [2]. Stroke is the leading cause of adult disability worldwide, and can cause a variety of semi-permanent to permanent disabilities which can be in the form of physical, cognitive, psychological or social limitations [7]. Stroke is globally the largest contributor of neurological disability-adjusted life years (DALYs), and one of the largest contributors for general DALYs [8]. DALYs among stroke survivors increase with age, and is highest among the 60–80-year age group.

Permanent complications of stroke hinder a patient’s ability to perform activities of daily living (ADL). The presence of post-stroke depression (PSD) increases this impairment even further, and delays a patient’s rehabilitation [9,10]. Approximately 30% of all stroke survivors develop PSD within 5 years after experiencing a stroke [11]. Empowering a patient’s ability to perform basic ADL has a positive effect on their psychological and physical Health-Related Quality of Life (HRQoL) [12,13].

The inability of stroke survivors to perform ADL and other general tasks forces them to rely on people in their direct environment. This increased stress on caregivers, often direct family, partners or friends, can cause burden and considerable strain, leading to both physical and psychological health risks for the caregiver [14]. Caregiver burden correlates to the stroke patient’s mental wellbeing and their dependency on the caregiver, which has a negative impact on a caregiver’s QoL [15]. Increasing a patient’s independence could improve QoL for patient and caregiver alike.

Stroke survivors with muscle weakness can benefit from robotic assistive aids, such as ARMs, to perform small tasks and regain independence in their daily lives. These ARMs can grasp objects for the user and either transfer them to a designated location or hand them directly to the user. However, a challenge with current assistive robots is the intuitiveness of human-robot interaction (HRI), particularly during the handover of items between robot and human.

Robot to human handover has been extensively researched for decades [16], with findings indicating that mimicking human behaviours in robots facilitates smoother and more intuitive HRI [4]. For instance, modifying an assistive robot with algorithms that emulate human movement, such as employing a human-inspired velocity profile instead of a standard trapezoidal one during handover motions, results in faster and smoother interactions [17]. A thing robots struggle to mimic is the communication humans use, where eye contact, verbal cues, gestures and motion are used to show initiation of both giving and receiving during a handover [18]. During a human-to-human handover, the giver is able to predict how the receiver will reach for the object and gauge when they have grasped it, while the receiver is able to anticipate when the giver will release the object based on visual cues.

There have been attempts at mimicking human communication for assistive robotic manipulators. This involves implementing sensors that enable robots to interpret data such as human vision [19,20,21], vocal commands [22,23,24] or gestures of different parts of the body [25,26] to enhance communication and handover control. However, this often also adds an element that makes the interaction less habitable for the human due to a high interaction time [23,24]. Developing smooth and intuitive handover for an ARM has to consider both user safety and user experience.

State-of-the-art

A wide variety of ARMs are available today, from innovative conceptual designs to market-ready products, all designed to aid users in performing Activities of Daily Living (ADL). This market review will specifically highlight the current handover designs in ARMs, focusing on how these devices transfer items between the ARM and the user during various tasks while ensuring the user maintains full control over the ARM [27].

Robot handover is defined by the robot moving within proximity to transfer the object, followed by the act of the transfer guided by both robot and user together [28]. This interaction can be autonomous, with the robot performing all steps, or by the user guiding the process. One of the challenges in handover in assistive aids is determining the precise moment for the ARM to release the object. Premature release could result in the object falling, whereas delayed release could result in user frustration. In general, robot to human handover has the most success if the process is closely resembling human-to-human interactions [29].

Recent studies have successfully implemented a Voice-User Interface (VUI) that allows the user to vocally let the robot know when they are ready for the robot to let go of the item [23,24]. The handover is guided through speech from both the robot and user. The robot utilises speech to inform the user of the current state of the handover, while the user gives vocal commands to the robot to initiate the steps. The studies showed a high success rate in handovers, both over 95%, and high user appreciation. However, both studies mentioned a long average duration time of over a minute for the handover task.

For full autonomous handover, a study used an RGB camera combined with an object detection algorithm to allow the ARM to both pick-and-place items and handover items to a bystander [30]. This study worked with the user, the receiver, holding out their hand and waiting for the robot to drop the item into their hand, rather than grasping the item while the robot was still holding it. The study showed a high accuracy in detecting objects and human hands. However, the method has only been tested on round fruits, limiting the scope of the design.

To utilise the movements humans make during grasping tasks, one study designed a grip-state indicator that integrates both gripping strength and gestures through the use of whole-hand tactile sensors [31]. This approach uses several different inputs to facilitate handover. Results showed fast handover times, but accuracy rates for unseen grip settings ranged from 75% to 89%. Additionally, new objects require a 25 second demonstration for the robot to adapt to the item. This approach has currently only been tested in industrial settings.

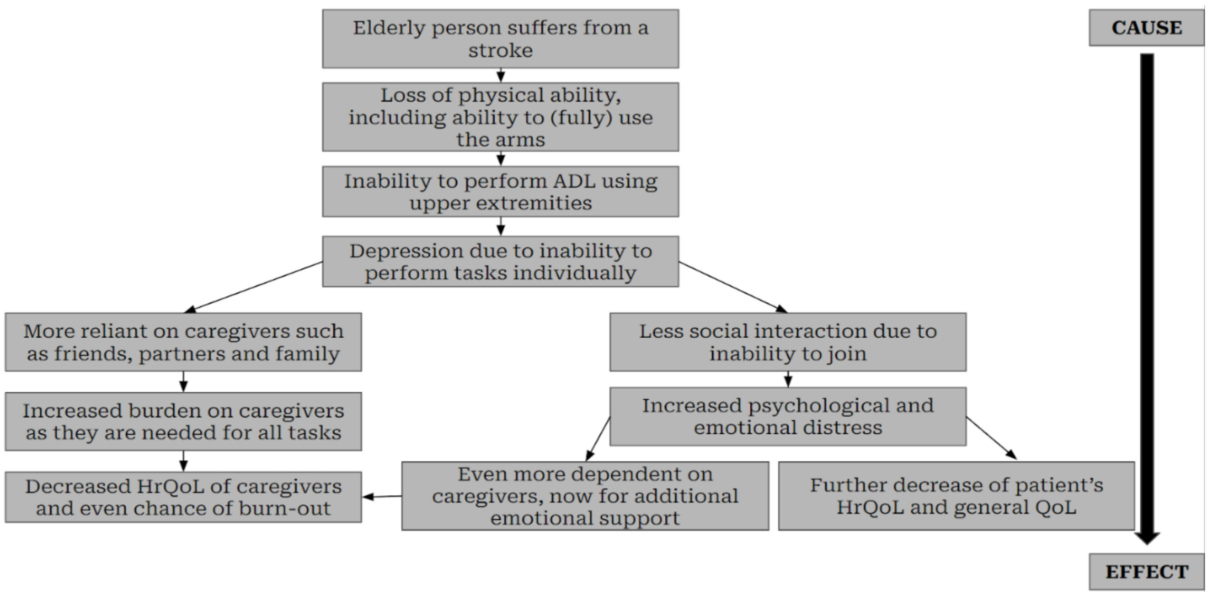
Objectives

This study aimed to develop a control method for an assistive robotic manipulator to improve the intuitiveness and efficiency of object handovers, by utilising a sensor-equipped glove. The goal was to create a more user-friendly and effective assistive device using gestures performed during handovers to guide the ARM.

State-of-the-art handover designs have shown that interaction speed is currently the biggest factor that inhabits handover intuitiveness. A glove design was chosen for this project to reduce handover duration. Performing gestures while wearing the glove takes little time while still empowering the user to control the ARM. Since the glove serves as a wearable remote-control rather than an additional input device, it enhances user comfort during interactions. The adaptability of the design and the low-cost, readily available components necessary to realise the project also contributed to the decision.

In figure 1 a cause-effect diagram of the effects of a stroke is depicted, focusing on the consequences muscle weakness has for the patient. In figure 2, it is shown that restoring the ability to perform basic tasks for the patient has a positive effect on the events that follow in the diagram. With the ARM making it possible for the patient to perform small tasks on their own, other problems created as a result of the lack of independence of the patient can be solved.

As there is already a wide variety of ARMs designed and manufactured, this project focused specifically on creating a way to achieve smooth and intuitive handover between the ARM and the user. To reach the goal, a DYNAMIXEL OpenMANIPULATOR-X RM-X52-TNM robot arm provided by the lab of the research group of Discrete Technology and Production Automation (DTPA) at the University of Groningen was altered in software so that it could read data sent by additional hardware. The hardware consisted of inertial measurement units (IMUs) and flex sensors affixed on a glove.



Elderly person suffers from a stroke
Loss of physical ability, including ability to (fully) use the arms
Inability to perform ADL using upper extremities
Depression due to inability to perform tasks individually
More reliant on caregivers such as friends, partners and family
Increased burden on caregivers as they are needed for all tasks
Decreased HrQoL of caregivers and even chance of burn-out
Less social interaction due to inability to join
Increased psychological and emotional distress
Even more dependent on caregivers, now for additional emotional support
Further decrease of patient's HrQoL and general QoL
CAUSE
EFFECT

Figure 1. Cause-effect diagram of the effects of a stroke pertaining to inability to use the upper extremities.

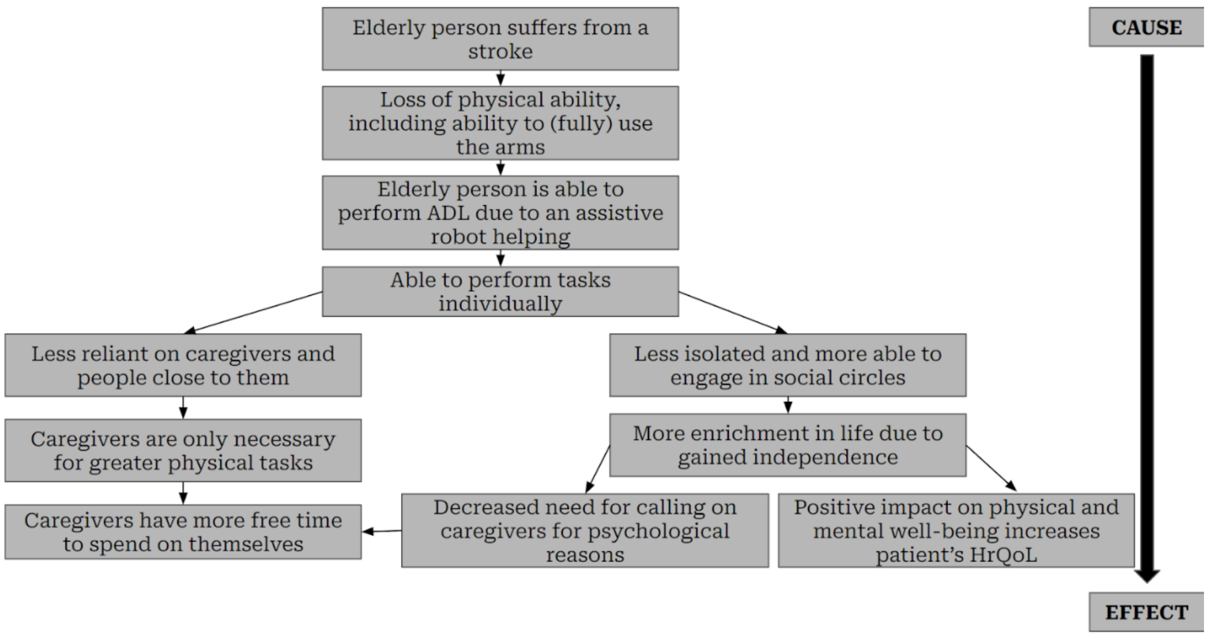

CAUSE
Elderly person suffers from a stroke
Loss of physical ability, including ability to (fully) use the arms
Elderly person is able to perform ADL due to an assistive robot helping
Able to perform tasks individually
Less reliant on caregivers and people close to them
Less isolated and more able to engage in social circles
Caregivers are only necessary for greater physical tasks
More enrichment in life due to gained independence
Caregivers have more free time to spend on themselves
Decreased need for calling on caregivers for psychological reasons
Positive impact on physical and mental well-being increases patient's HrQoL
EFFECT

Figure 2. Cause-effect diagram of reaching the goal of allowing the patient to perform basic tasks post-stroke.

As there is already a wide variety of ARMs designed and manufactured, this project focused specifically on creating a way to achieve smooth and intuitive handover between the ARM and the user. To reach the goal, a DYNAMIXEL OpenMANIPULATOR-X RM-X52-TNM robot arm provided by the lab of the research group of Discrete Technology and Production Automation (DTPA) at the University of Groningen was altered in software so that it could read data sent by additional hardware. The hardware consisted of inertial measurement units (IMUs) and flex sensors affixed on a glove.

The provided DYNAMIXEL OpenMANIPULATOR-X RM-X52-TNM robot arm has 5 degrees of freedom (DoF) and is able to move using five XM430-W350-T servomotors [32]. The robot arm was already fully assembled and operating, and it has open-source code available on multiple platforms that allows for full control [33]. The costs of the robot arm itself are approximately €1350 [34]. This does not include the OpenCR control board and the power supply, which would add another estimated €200 [35,36]. A successful, working prototype had to be realised within a timeframe of ten weeks. Due to time and cost limitations, the design had to be created with sensors that were already available in the electronics lab, or sensors that were affordable and could be shipped and received within a reasonable time. Given the variety of sensors in both their function and their availability, as well as the nature of the problem, a number of concepts could be created and the probability of creating a successful prototype was high. The prototype was created in the electronics lab of the DTPA, located at Nijenborgh 4, 5117.0205, under the guidance of supervisor Dr. Elisabeth Wilhelm.

By creating a method for the ARM to be able to hand over small items in a way that is seamless, intuitive, and easy for the user to operate, the patient will be able to receive items the ARM has grasped in the vicinity independently without needing the aid of another person. Achieving this main goal will allow the patient to perform small tasks on their own which will increase their independence and ultimately improve their mental state and HrQoL. It will also lessen the burden on the caregivers of the patient due to the patient being able to perform small tasks on their own and not needing to call over caregivers when they want to reach for an item. This will decrease the strain on said caregivers and improve both their physical and mental health.

Societal relevance

Improving the intuitiveness of HRI in ARMs can significantly enhance the independence of stroke survivors, reducing the burden on caregivers and improving the overall quality of life for both parties. Multiple other stakeholders are also likely to benefit from these advancements, as shown in table 1.

Table 1. Stakeholder analysis for the assistive robotic manipulator with gripper.

Stakeholder	**Characteristics**	**Expectations**	**Potentials and deficiencies**	**Implications and conclusions**
Stroke patient	Patients experience pain, discomfort and physical and social limitations. Patients are unable to perform basic ADL, further lowering their QoL.	They expect to regain the ability to reach for and grab items using the ARM and be independently able to perform small tasks.	The patient likely has a lack of grip strength. As the age group of the patients is around 60-80 years, mode of control of the ARM cannot be too complicated due to possible lack of technical knowledge.	The main demographic for the ARM. It needs to be designed with them in mind, which limits the possible complexity of the mode of control of the design.
Stroke patient caregivers	Often family, partners or close friends of the patient. Spends a long portion of their time with the patient.	They expect an increase in the independence of the patient so that the burden of care is lessened.	They are close to the patient and know the patient’s wishes. They might be too attached to let the patient try the assistive arm on their own.	They see the patient daily and their wishes as well as the stress put on them should be considered heavily.
Family and friends	Even when not directly the main caregiver, they often still aid the patient. They are personally impacted by the well-being of the patient.	They expect an improvement in both the physical and mental well-being of the patient, even if slightly.	They are very close to the patient and know whether a certain treatment approach or assistive device would work or not.	They are often a direct line to the patient that has known the patient since pre-stroke.
Primary care team	Focusses on the overall health, survival and well-being of the patient over all else.	They expect a product the patient can use on their own without it being harmful to the patient in any way.	They are experts in the medical field but generally lack technical knowledge.	They may not be interested in working with new assistive aids if they don’t believe it is an improvement over the current situation health-wise.

Occupational therapist	Assists the patient in rehabilitation with a focus on relearning how to perform basic ADL. Aids the patient in gaining independence in daily life.	They want to improve the independence of the patient. They expect a device that is able to aid the patient with small basic tasks.	They work directly with the patient to enhance their fine motor skills and cognitive abilities, and are an expert on rehabilitation with regards to performing ADL.	They can recommend the assistive aid to the patient if they believe in its capabilities and if the aid adds to a patient's independence.
Healthcare insurance	Aims to provide optimal care at the lowest costs.	They want to keep the costs as low as possible.	Their ability to cover the expenses of the assistive aid depends on the costs of the aid itself as well as the patient's specific insurance.	Given the expectedly high costs of the product, wishes of the insurance should be considered as the patient and/or care facility might need them for funds.
Assistive aids industry	Always interested in new products, preferably at low costs.	They want to run a profitable business.	They are more interested in products easy to manufacture, which an assistive robot arm is not.	They have knowledge of the market, the demand and the selling potential of the aid.

Demarcation

The primary focus of the project was on the design, development, and evaluation of a control mode that ensures smooth handover between the robotic manipulator and the user. To ensure successful completion of the project, clear boundaries had to be set to realise the design within the time limit.

To facilitate intuitive handover, the ARM is operated by flexion and extension of the hand as well as bending of the thumb and index finger, requiring the user to be able to move their hand and fingers. The method requires instructional training before it can be used. The handover method was designed specifically for stroke survivors with lower limb disabilities or general muscle weakness who are in need of assistance when performing small tasks and basic ADL. This strategy cannot be employed for patients who have extensive upper limb disabilities or cognitive impairments which makes them unable to utilise the glove design. Individuals other than stroke survivors can also make use of the designed control method, if they are able to perform the necessary actions for control. The research focused on robotic handover rather than comprehensive rehabilitation protocols.

Due to the low technology readiness level of the design, it could not be tested with the actual target group. Testing was instead performed by the developer of the design, who is able-bodied. To assess whether or not the design is suitable for the target demographic, separate testing needs to be performed, but this is not feasible within the time period for the project.

To distinguish the design from the various ARMs already on the market, the final operating system was created out of low-cost materials and sensors. The final control method is able to use the data it receives from the sensor on the glove to move the ARM between two positions while also independently operating the gripper. It is unable to rotate the ARM and the ARM cannot deviate from alternating between two positions, but the positions can be changed manually in the code. The ARM can operate on a delay that can be manually changed, but once the delay has passed it will perform the actions it received input for and this cannot be cancelled. The design was created for one type of manipulator, the DYNAMIXEL OpenMANIPULATOR-X RM-X52-TNM.

The gripper can close completely or close around an item to hold it, stopping gripper movement when it notices a current spike. This was a simple thresholding method to be able to test the design's functionality. The force the gripper enacts on the items was not measured. The control of the force the gripper exerts is outside of the scope of this project.

The DYNAMIXEL robot arm is 380mm long and has a payload of 500 grams. The gripper stroke ranges from 20mm to 70mm. This limited the weight and the size of the items that were used for testing. The final design was tested on handover speed and handover success rate.

The current design was created using the hardware and software available at the robotics lab of the Engineering and Technology Institute Groningen (ENTEG), constraining the possible options for the overall set-up. The glove was created for the right hand and the sensor placements as well as the movement positions the ARM reacts to are only based on right hand interactions. The set-up does not have a safety stop button implemented and the ARM does not stop its movement during an action if it encounters an obstacle. Using the glove requires a computer device with two USB ports.

MATERIALS & METHODS

This section outlines the procedures used in the development and evaluation of the glove control method for the DYNAMIXEL OpenMANIPULATOR-X RM-X52-TNM. The design was created to make the ARM perform its actions as shown in figure 3. Control to facilitate handover was established by allowing the Arduino board to read movement data from the MPU-6050 sensors and Spectra Symbol 2.2 flex sensors.

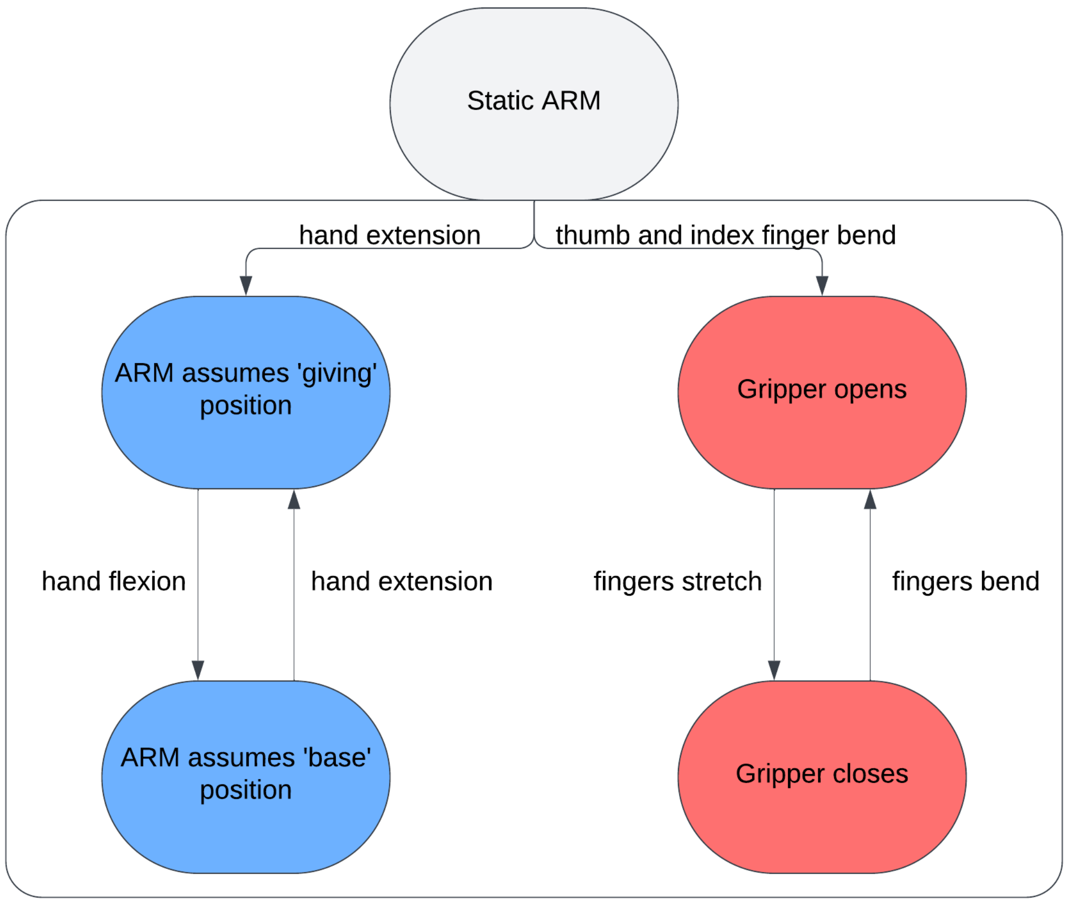


Figure 3. State-machine diagram of the two independent state changes the ARM can go through to facilitate handover.

System description

The idea behind the glove is to control the movements of the ARM, both in moving towards and away from the user as well as opening and closing the gripper, by moving the hand wearing the glove. To make this concept an intuitive method of control, the movements to facilitate control were to be based on movements a person generally performs when reaching for an item in a handover.

Hardware components

The final design of the glove consists of the following materials:

- Garden glove purchased from Wibra, €1.49
- 2 MPU-6050 Accelerometer and Gyroscope 3-Axis Module 3.3V-5V purchased from RS Components Europe, €8.50 per sensor
- 24 jumper wires

- 30-input breadboard
- 2 Spectra Symbol 2.2 flex sensors purchased from RS Components Europe, €12 per sensor
- 2 fixed resistors, approximately 8.2 kilohms each
- Arduino MKR Wi-Fi 1010 board with USB cable, €33.50

Any Arduino board with SCL and SDA inputs for I2C communication can be used. The resistance value of the resistor can be different, as different resistor values can give usable readings with the flex sensors. For flex sensor readings to be usable, the change of output value needs to be approximately the same for repeated sensor bending. To connect everything to the glove, self-adhesive Velcro, duct tape, 3D-printed holders for the MPU sensors and a soldering iron were used. The full physical setup of the glove can be seen in figure 4 and 5. All sensors used an input voltage of 5 volts.

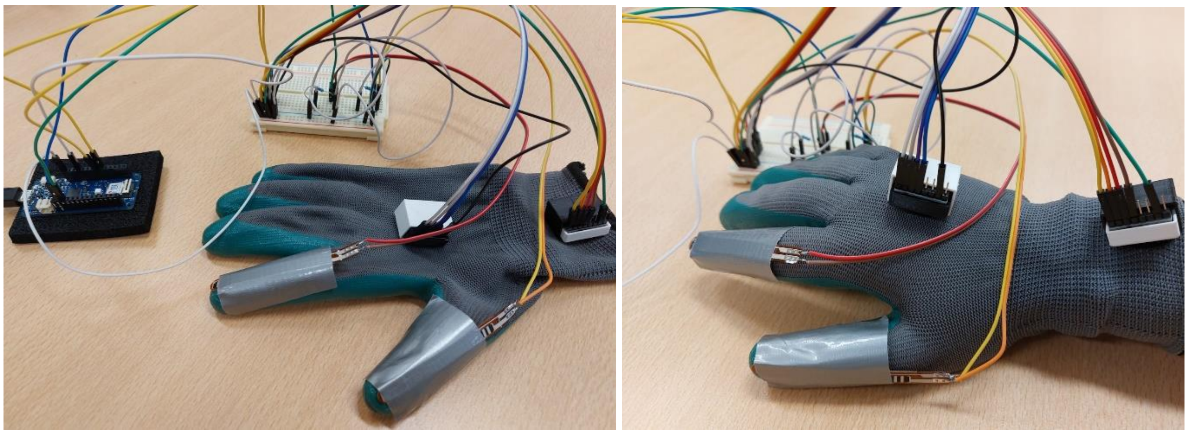


Figure 4 and 5. Setup of the glove with the circuits on the breadboard and the connections to the Arduino MKR (left), and the orientation of all sensors when the glove is worn (right).

Software components

The following software was used to read, transfer or find data:

- Arduino IDE 2.3.2
- Python ver. 3.12
- Dynamixel Wizard 2.0

The glove works by combining the various readings it receives from the different sensors and sending this data to the OpenCR board of the ARM, which reads the data and uses it to manipulate the ARM accordingly. To send the data from the glove to the ARM, serial port communication through Python was used. All code used can be found in the Appendix. The *glove_control* script runs on the Arduino board and is used to receive the data from readings of the sensors on the glove. The *serial_ports* script runs on Python and allows the data from the Arduino board serial bus to be sent to the OpenCR board serial bus. The *robot_control* script contains the code the OpenCR board uses to manipulate the ARM by reading the data it received from the Python script.

Glove set-up

The design utilises two MPU-6050 sensors to track the orientation of the hand and the forearm, and two flex sensors to detect bending of the fingers. The sensors were placed as shown in figure 6. MPU sensor 1 was placed in the middle of the back of the hand, with the Y-axis running vertically along the fingers and arm, and the X axis placed horizontally across the hand, perpendicular to the Y axis. The sensor was mounted flat on the hand with the pins oriented upwards, such that the Z-axis extended perpendicularly from the hand's surface. X, Y and Z axis orientation of the MPU-6050 sensor were found in the sensor datasheet [37].

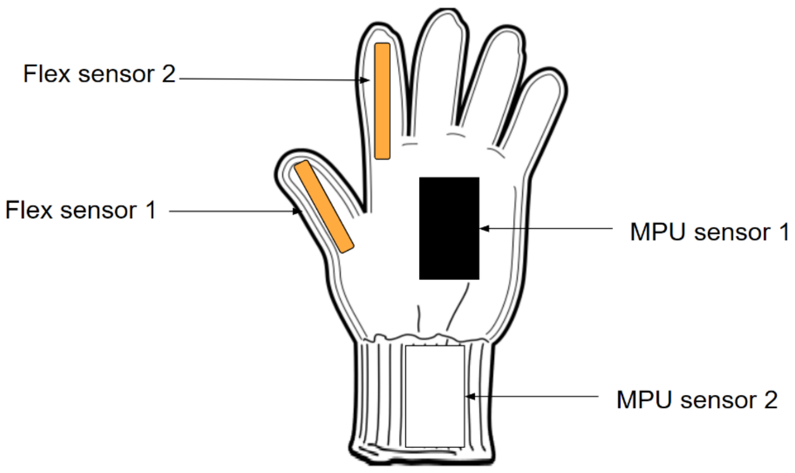


Figure 6: Schematic of the glove and the attached sensors with their significant numbering. Glove template taken from OpenClipart [38]

MPU-6050 sensors details

The MPU-6050 is a type of IMU sensor that integrates a 3-axis gyroscope and a 3-axis accelerometer, providing 6-axis motion sensing capability [37]. It allows for motion tracking by capturing both linear and rotational movements in the X, Y and Z axis. The accelerometer measures linear acceleration and the gyroscope measures angular rotation around the axes. To create motion tracking, sensor fusion algorithms can be used to find the acceleration and rotation data by combining accelerometer and gyroscope data.

The gyroscope has selectable ranges of ±250, ±500, ±1000, and ±2000 degrees per second, and the accelerometer has selectable ranges of ±2g, ±4g, ±8g, and ±16g [37]. This design utilises the default condition '**FS_SEL=0'** for the gyroscope, setting it to 250 degrees per second. The accelerometer was set to the default condition '**AFS_SEL=0'**, setting it to 2g. The default values for both were chosen as they provide sufficient readings for the application in the design while offering the highest accuracy.

Using the MPU-6050, the Euler angles roll, pitch and yaw can be detected on the location the sensor is implemented. Euler angles are a set of three angles that describe the orientation and rotation of any point relative to a fixed coordinate system [39]. Roll describes the X axis, pitch the Y axis and yaw the Z axis. To calculate roll and pitch, accelerometer data was used. For yaw, both accelerometer data and gyroscope data are required. To be able to detect the difference in orientation between the two MPU sensors, the roll and pitch values are subtracted from each other to provide delta values. These delta angles were used to determine the motion and orientation of the hand relative to the forearm, moving the ARM to a base or giving position depending on hand flexion or extension.

MPU sensor 2 was placed on the back of the forearm, below the wrist joint to minimize potential sensor movement during independent hand motions. MPU sensor 2 was oriented with the same axes alignment as sensor 1. Placing the sensors in these specific locations gives the microcontroller the ability to read orientation data of the hand and forearm separately. MPU sensor 1 was in a black casing with an AD0 connection to ground, and sensor 2 was in a white casing with an AD0 connection to the voltage input. Without these specific AD0 connections, I2C cannot read different addresses for the sensors.

To provide the roll and pitch data, acceleration values from all three axes were used. The roll angle ϕ in degrees is defined as:

$$\varphi = \arctan\left(\frac{-a.x}{\sqrt{a.y^2 + a.z^2}}\right) \cdot \frac{180}{\pi}$$

(1)

In equation (1), *a.x* is the acceleration on the X axis, *a.y* is the acceleration on the Y axis and *a.z* is the acceleration on the Z axis. The formula also converts the data from radians per second to degrees. Likewise, pitch angle θ was calculated as shown in equation (2):

$$\theta = \arctan\left(\frac{a.y}{a.z}\right) \cdot \frac{180}{\pi}$$

(2)

By using these formulas for the sensors, the microcontroller is able to read when the user moves their hand up, down, to the left or to the right in comparison to their arm.

To prevent outside forces from affecting the readings, such as pulling of the jumper wires, the sensors were secured with self-adhesive Velcro pasted on the glove and the 3D-printed sensor encasings. The MPU-6050 casings used in this study were created and provided by Paul Fetchenhauer. Drawings of the casings can be found in the Appendix.

The code for the sensor setup and calibration were based on the Adafruit MPU6050 guide for Arduino [40]. To get readings from two sensors at nearly the same time, different I2C addresses were given in the function setup, 0x68 and 0x69 to sensor 1 and sensor 2. This I2C bus communication reads sensors one after the other rather than simultaneously, but the delay is insignificant and negligible for the overall readings. To calibrate the sensors for first time use, the code reads the values of roll and pitch in base position an N number of times in the *calibrateSensors* function. It uses equations (1) to determine *baseRoll1* and *basePitch1* for MPU sensor 1, and equation (2) to determine *baseRoll2* and *basePitch2* for MPU sensor 2. Base position is defined as the user wearing the glove and having their hand in a resting position laying on a surface. In the code used, N = 100 for determining the base roll and pitch values. For these values, calibration was measured with a stopwatch to take about 6 seconds before the sensors start giving data. Roll and pitch values were calculated in the main *loop* function using equations (1) and (2), subtracted by the base values found during the calibration. To calculate the differences between the same angles of the two MPU sensors, the roll and pitch angle values of sensor 1 were subtracted from the values of sensor 2. The difference in angle positions were called *deltaRoll* and *deltaPitch*.

Flex sensor details

The glove has two flex sensors connected to it, one attached to the thumb and one to the index finger, with the metal pads of the flex sensor on top. The thumb and index finger were chosen because they are the most commonly used when grabbing an item during handover. Flex sensors are designed to detect and measure bending on their surface by measuring angular displacement [41]. This changes their resistance. To allow for measurements, the sensors were connected to custom voltage divider circuits on a breadboard. Both voltage divider circuits use a fixed 8.2 kilohms resistor and an input voltage of 5 volts.

The sensor operates by varying its resistance in response to bending, which alters the voltage output across the divider circuit. The flex sensors are connected to analogue-to-digital converter pins on the Arduino board and give an analogue output between 0 and 1023. Flex sensors work as a resistor in a voltage divider circuit, so an increase in resistance changes the output voltage. The sensitivity and the baseline value of the output can be altered by changing the fixed resistor in the circuit. The output value of the flex sensors was used to determine the state of the gripper, opening below a set value and closing again if it rises above this value.

The flex sensors have an offset value in relaxed state that decreases when the sensor is bent. Despite using two fixed resistors both labelled 8.2 kilohms, flex sensor 1 and flex sensor 2 have a different offset value measurement in relaxed state. This was accounted for during design. The output of the flex sensors was used to both open and close the gripper. The flex sensors were connected to the breadboard by soldering them to cut jumper wires, and they were attached to the glove with duct tape. As the entirety of the flex sensor needs to be stuck to the glove to notice bending of the fingers, duct tape was chosen rather than sewing the flex sensors onto the glove or using self-adhesive Velcro.

For a flex sensor to give values when connected to an Arduino microcontroller, it needs to be implemented into a voltage divider circuit. It is then able to give an output using the following equation:

$$Vout = Vin \cdot \frac{Rfixed}{Rfixed + Rflex}$$

(3)

Equation (3) is the standard equation for a voltage divider. It uses 5V for the input voltage V_{in}, and approximately 8.2 kilohms for the fixed resistors R_{fixed}. R_{flex} is the flex sensor in the voltage divider, and V_{out} is the output value that is ultimately converted from analog to digital. In this design, the fixed resistor is positioned between the input voltage and the flex sensor, with the flex sensor connecting the fixed resistor to ground. This causes the output voltage to decrease when the flex sensors are bent.

The code for the flex sensors was combined with the code for the MPU sensors. In the *loop* function, the code first reads data from flex sensor 1, then flex sensor 2, MPU 1 and finally MPU 2. The time between the readings of the different sensors is negligible. Flex sensor 1 is connected to A0 and flex sensor 2 is connected to A1. To use the data from the readings for all sensors, the output values *deltaRoll, deltaPitch, flexValue1,* and *flexValue2* were converted to string format. This data string format was then printed to Python for serial communication between the ports.

Python serial communication

The sensors on the glove and the ARM are not directly connected in hardware, but through port-to-port communication facilitated by Python using two USB ports on a computer device. By naming the two communication ports and setting the baud rate to the rate the Arduino board operates at (115200), the data can be sent from the Arduino port to the OpenCR port. The Python monitor prints the data the OpenCR board receives, that being the changes in roll and pitch angles, and the values the flex sensors read.

OpenCR commands

The code for the OpenCR board was written using data retrieved from Dynamixel Wizard 2.0. By using the example sketch *usb_to_dxl* for the OpenCR board [42], the ARM could be connected to the Wizard and manually moved into the desired positions. The wizard also showed the IDs for all servo motors, and allowed for movement of each servo motor separately while showing the exact angles. The control tables for the manipulator and each servo motor could also be found using the Wizard, which showed the locations of the addresses for values such as 'present current', 'current limit' and 'goal position'.

The OpenCR board connected to the ARM receives the data string from Python and uses this data in an Arduino code to move the 5 servo motors it consists of. Before reading any data, the code activates the torque for the servo motors so that the ARM remains stable. In the setup function, it sets the baud rate to the baud rate of the glove data, 115200.

In *moveServosToPosition*, the servo motors of the ARM are set to respond to their functions. This is not the case for servo motor 15, as ID 15 corresponds to the gripper, which is moved by the flex sensors rather than the MPUs. The gripper is defined in the function *closeGripper*, where it is set to a full closing position at 2700 when the flex sensors are relaxed. In the Dynamixel Wizard the angle for a full closing position was found to be 238 degrees. To convert the servomotor angles to encoder counts the OpenCR board can read, the following formula was used:

$$EC = \frac{\alpha \cdot 4096}{360} \tag{4}$$

In the equation (4), α is the angle of the servomotor in degrees and EC is the number of encoder counts. This equation had to be used to move the servomotor in the correct position, as it has a rotation of 360 degrees in which it can assume 4096 positions as specified by the XM430-W350 datasheet [43]. With this formula, the angle in degrees is converted to encoder counts. When the gripper is holding an item, it cannot fully assume the closing position of 238 degrees. To prevent the gripper from becoming unresponsive due to its inability to fully achieve the set position, the bool function *isGripperHoldingItem* checks to see if the current of the gripper servomotor is increased during closing, which happens when it squeezes an item it holds. If the specified current threshold is reached, it stops attempting to close and allows for commands again.

In the main *loop* function, the board checks if serial communication is established. If there is, it receives the data string from the port and reads the values. The servomotors are moved into position based on the deltaPitch value. If it is below a certain value, it will move into the 'giving' position. If it is above a certain value, it will move into the 'base' position. The manipulator does not move if deltaPitch has a value between these two limits. The values necessary for the limits of deltaPitch were determined by testing the glove on its own and reading the data output in the serial monitor of Arduino. The 'base' and 'giving' positions can be seen in figure 7 and 8 and are achieved by changing the angles of the motors that need to be adjusted in OpenCR. The desired positions and their corresponding angles were found using the Dynamixel Wizard. The *loop* function also opens the gripper based on the flex sensor values. If either of the flex sensors drops below a certain value, the gripper moves to a fully open position. The angle for a full open gripper is 110 degrees, converted to 1252 using equation (4). The values for the flex sensors were found using the serial monitor output of separate glove testing. If the flex sensors raise above the limit again, the gripper acts according to the *closeGripper* function, which closes the gripper either fully or until the current limit is reached. All values read by the OpenCR board are also printed in the Python monitor to check if the values are as expected and read correctly by the board.

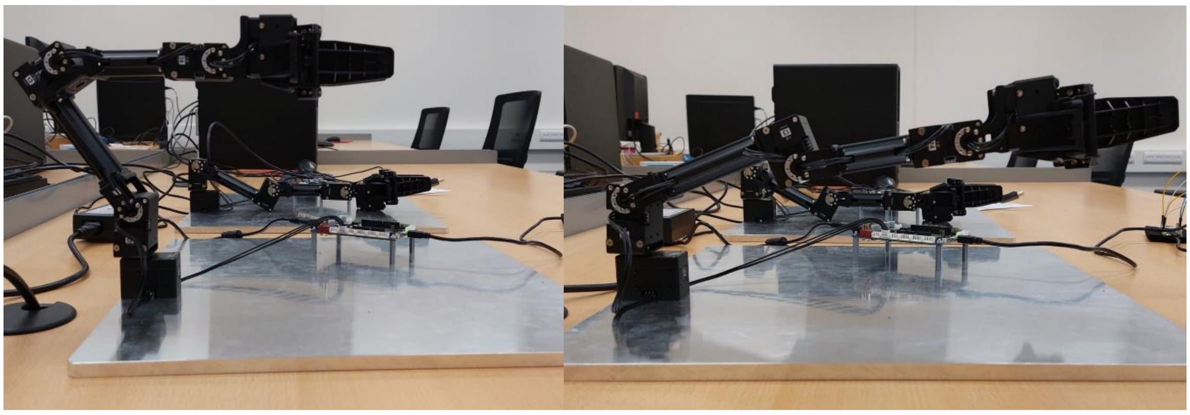


Figure 7 and 8. The gripper in its base (left) and giving (right) positions.

Test protocol

The experiment was conducted over the course of two weeks in the electronics lab. The design was tested by one person, the developer, using five items. The items were successfully handed over from ARM to the user twenty times. A 'handover' is defined as the robot, already holding the item and in 'base' position', moving into the 'giving' position towards the user, releasing the item when the user has grasped it, and moving back to 'base' position once the handover is finished. Before testing was conducted, the user followed guided training to familiarise with the method and improve consistency. The ARM was tested on its ability to correctly hand over an item by the user sitting in front of it at a desk. Handovers were timed separately on a stopwatch, meaning the times below have an error rate of 0 to 2 seconds, as the operator was also managing the stopwatch. Handovers were performed consecutively per item.

Table 2. Specifics of the items used during testing of the glove control.

Item number	**Item description**	**Length (mm)**	**Width or diameter (mm)**	**Weight (g)**
1	Empty cardboard coffee cup	90	55-80 (varies among length)	18.8
2	500ml bottle of water, half filled	230	64	302.7
3	500ml bottle of water, empty	230	64	41.2
4	Cardboard box of tea	120	80	40.3
5	Whiteboard eraser	143	50	84.7

With the codes uploaded to their respective boards and the serial communication being established through Python, the glove can be connected to the ARM for gesture control. The MPU sensors were calibrated while the glove was in resting position on the hand before testing was performed. Timing the handover started when the ARM was in base position holding the item, and stopped when the ARM returned to base position after it handed over the item. Specifics of the items used during testing can be found in table 2. Different shapes, dimensions and weights were used for testing. Handover times were noted of the successful handovers. If a handover failed, no time was noted and it was counted as a failed handover instead. A failed handover is defined as the user unable to take over the item from the ARM due to failure of the design or the ARM. If the ARM let go of the item before the user could grasp it, or if the ARM retracted before the handover was performed and without the user intending for the ARM to do so, it would count as a failure. The total handovers performed per item were the twenty successful handovers plus the number of failed handovers. Error rate was calculated in percentages by dividing the number of failed handovers by the number of total handovers performed.

To interpret the data, IBM SPSS Statistics 29.0.2.0 was used. Handover time was plotted in histogram and Q-Q plot form to analyse the data and determine if it has a normal distribution. A Shapiro-Wilk test was performed to fully ascertain that the handover time was normally distributed. These tests were performed with all hundred readings grouped together.

A one-way ANOVA test was performed to analyse the handover times of all items. The items served as independent variables in the test, and their handover times were used as dependent variables. The purpose of the analysis was to determine whether there were significant differences in handover times among the items. As the tests were performed using the same test person for every item, the items are not completely independent variables. Classical ANOVA assumes independent variables. This discrepancy can potentially influence the statistical analysis and interpretation of results.

RESULTS

Testing of the handover method was performed according to the test protocol. A full list of times for every item can be found in the Appendix. In table 3, the overall results per item can be found.

Table 3. Handover results and statistics per item, approximated to two decimal places.

Item	**Average time (s)**	**Total number of handovers**	**Standard deviation (s)**	**Range of time (s)**	**Failed handovers**	**Error rate**
Empty cardboard coffee cup	9.08	23	1.78	6.79 - 13.81	3	13.04%
500ml bottle of water, half filled	8.84	27	2.04	6.64 - 13.41	7	25.93%
500ml bottle of water, empty	8.53	25	1.45	6.50 - 12.47	5	20.00%
Cardboard box of tea	5.99	23	1.59	3.97 - 10.15	3	13.04%
Whiteboard eraser	6.36	24	1.57	4.23 - 10.08	4	16.67%

Handovers were performed a total of 122 times, of which 22 attempts failed, giving an overall error rate of approximately 18.03%. The overall mean time for all 100 successful handovers is approximately 7.76 seconds, with an overall handover time range from 3.97 seconds to 13.81 seconds. The overall standard deviation over all handovers was 2.15 seconds.

The Shapiro-Wilk test resulted in a p-value of 0.007, indicating a normal distribution in handover time. An ANOVA test was performed with all 5 items as separate groups for which a p-value of below 0.001 was found. Full results of the statistical tests and graphs can be found in the Appendix.

DISCUSSION

Interpretation of the results

The average handover time of the tested method was 7.76 seconds with a success rate of 81.97%. The error rate is high in comparison with other studies that tested specifically on handover, which usually range between 1% to 10% [38, 39]. Handover time is very fast in comparison to other designs. The aforementioned studies both had an average handover time of over a minute for solely a handover task, but also had a much more complex set-up. Future advancements should focus on improving success rate of the handover, which may cause an increase in handover time.

Both graphs and the Shapiro-Wilk test showed a normal distribution of the time of all handovers. The ANOVA test showed a p-value of below 0.001 for the times of the five items. It should be noted that all handovers were performed by one person, making the variables not truly independent and affecting the ANOVA data analysis. This lack of independence could potentially bias the results, as the variability in handover times may not fully reflect real-world scenarios where multiple individuals might be involved. In addition to this, the last two items to be tested were the items with the fastest handover times. Multiple handovers by the user could have improved the user's skill during testing, leading to faster times for the last items.

In real-world scenarios, it is not realistic that the user is always sitting right in front of the ARM, in the exact location the giving position moves to. In those scenarios, the ARM also has to grasp an item first before it can hand it over to the user, rather than it already being positioned with the item like in the tests. The handover is also designed for stroke survivors, who may have less upper body mobility. Testing was done by the able-bodied developer of the design, who is very familiar with the set-up and knows exactly how it works. It is expected that real-world handover times and error rate will be higher than the results found in this project.

Problems during development and testing

The design started with the idea to have the ARM move freely with movement of the glove, rather than moving between set positions. For this, the roll, pitch and yaw values of both sensors would be used, as well as differences in these values between the sensors. During development, it became clear that this necessitates the creation of a complex kinematic model capable of mapping hand movements to the motion of the robotic setup. To achieve the project goals, it was decided to reduce the model complexity and limit the motion of the robot accordingly. The ARM now moves between two positions depending on movement of the hand. It has no rotation, which was initially included in the design.

The gripper rarely responds to the bool function *isGripperHoldingItem*. This function should allow the gripper to stop the attempt at closing when it detects it is holding an item. This causes a current spike, which should stop the gripper’s movements and allow for new commands, such as the command to open again. Regardless of what value the current threshold is set to, it only sporadically performs the correct actions for this function. Attempts were made at solving the issue by making it a closed *loop* function, but this did not show different results. To work around this, the angle the gripper closes at was manually changed to a little below the width of each item the tests were performed with, to allow the gripper to close and still take commands. This was done by changing the encoder counts in the code every time a new item was tested. This workaround might have impacted how well the gripper could hold each item, as it could not ‘squeeze’ items.

The Dynamixel manipulator also slows down significantly in performing actions after an unspecified number of commands, and eventually completely stops responding. When this happens, the Arduino code needs to be reuploaded to the OpenCR board before it responds again. This locking up seems to be affected by object weight, and is also the reason why objects heavier than 302.7 grams could not be tested despite the ARM's payload of 500 grams. It can be seen from the results that items with a lighter weight have better handover times and a higher success rate. The weight of an item might also have impacted the results.

In the giving position, the gripper of the ARM is 12.5 cm above ground. During movement from base to giving position, the ARM moves its gripper down before it moves up to assume the giving position, leaving less room between gripper and ground. This means that items held by the gripper could not stick out below the gripper much, or else they would hit the ground during movement. This creates an imbalance in the item while the gripper is holding it, which may have affected handover of long items.

Possible hardware improvements

The most restricting problem currently is the movement limitations while wearing the glove due to the wires and the boards it is connected to. Even when the wiring is bundled together, or changed in direction or orientation, it sometimes obstructs hand movement. This includes the wiring from the breadboard to the Arduino MKR. Additionally, the breadboard that necessitates the dual clock line for the MPUs and allows for two voltage divider circuits for the flex sensors unintentionally functions as an anchor and obstructs movement, making it difficult to fully stretch out the arm without messing up the wiring

There are multiple ways to fix this, such as using longer wire, bundling them together and sewing these bundles onto the glove itself to restrict their movement. Another option is using a battery-powered Arduino board with Bluetooth connection, and integrating both the Arduino board and the breadboard with the circuits onto the glove. This would however make the glove heavier and require it to be longer. Even so, redesigning the current set-up so that the wires and boards no longer obstruct the movement of the user would be a huge improvement.

Another aspect that can be improved is the connection of all the sensors to the glove. If they are not well secured, it will directly impact the readings of the sensors. To make sure the flex sensors sense every bending movement of the fingers, they were attached with duct tape to the glove, but this is not a good permanent solution as duct tape loses its adhesive strength over time. The MPU sensors are inserted in cases and attached to the glove with self-adhesive Velcro. While the cases are a snug fit and do not allow for any movement of the MPU sensors within them, the Velcro is not very secure and prone to being moved by the pulling of the jumper wires. To improve MPU readings, the sensors should be attached directly onto the glove, either by sewing them on directly or creating sensor casings with holes on the side that can be sewn onto the glove.

Possible software improvements

Currently, the gripper responds to the inputs of either flex sensor dropping below a set value. This means that bending either the pointer finger or the thumb opens the gripper. The gripper can also be set to open only when values for both flex sensors are dropped below their threshold. While this threshold can be adjusted so that light bending of the fingers does not open the gripper, every time the fingers fully bend the gripper opens. A way to improve this would be to add an on and off button to the glove part of the operating system, so that the user can turn the glove manipulator on when they need to move the robot arm, and off when they want to use the hand without moving the robot arm and without taking off the glove. This functionality should be added without undoing the initial calibration of the glove.

The number of readings taken for the initial calibration of the sensors on the glove can be changed. During this project, 100 readings were taken, which takes about 6 seconds for calibration before data can be used. This is a significant amount of time, and the user should be able to see when the calibration is finished and they can move their hand. Currently, the serial output in Python displays this, but for user convenience haptic feedback, such as a light lighting up when calibration is complete, should be implemented.

Despite being able to calculate the roll and yaw values, the final operating system does not actually use these values due to forced simplification of the final design. This is also because the roll and yaw values are less reliable than the pitch value. Pitch measures the flexion and extension of the hand, which is easy to perform, but roll measures the radial and ulnar deviation of the hand, which is an uncomfortable gesture to make. Yaw measures the rotation of the hand, but when the hand rotates the forearm automatically rotates as well, meaning only overall yaw values can be used. Because of the unreliability of these values, only pitch angles were used to move the ARM, but the unused angles can be used to for example allow rotation, something it cannot currently do.

Future use of the glove

The operating system designed for the specific DYNAMIXEL OpenMANIPULATOR-X RM-X52-TNM robot arm can be used for different manipulators. Different manipulators use different joint motors, so the code for the robot arm will have to be adjusted depending on what type is used, but the code for the sensors and the serial communication can be used as is as an operating system for any manipulator arm. As the glove system is completely separated from the manipulator, it can easily be attached, detached, personalised and adapted if necessary.

The glove operating system has shown to be an effective method of human to robot handover, with intuitive movements and short handover time. If the previously described improvements are implemented, it will be a more comfortable method of handover, created with readily available and affordable components. Future studies are necessary to determine if the method of handover described in this project is intuitive in use for the demographic it was designed for, as it has only been tested so far by the developer.

CONCLUSION

This project evaluated the current state of robotic handovers in assistive aids and designed a method of handover for an ARM with the objective of achieving seamless user interaction. The aim was to develop a handover method specifically designed for stroke survivors that improves efficiency and mode of control for an overall smoother HRI. To do so, a glove was designed with MPU-6050 and flex sensors to give it the ability to read hand and finger movements. This allowed for gesture control of the ARM.

The tests performed showed that compared to peers, the error rate was high at 18.03%, however average handover times were very fast at 7.76 seconds. Average handover time differed significantly between the five types of items tested. The handover times show promise of a fast method of facilitating handover from robot to human while improving handover intuitiveness.

Improvements are necessary to increase the design's success rate and overall performance. Future research should focus on reducing the error rate of handovers to make it a more eligible method for ARM control. Additionally, testing should be performed by the target demographic to determine whether it is a suitable method of control. Functions of the design that were ultimately dropped due to limitations, such as making the ARM rotate, should also be researched to see if implementation is possible.

REFERENCES

1. Koop, Y., Wimmers, R. H., Vaartjes, I., & Bots, M. L. (Eds.). (2021). *Hart- en vaatziekten in Nederland, 2021*. Den Haag: Hartstichting.
2. Northwestern Medicine. (n.d.). *Life after stroke*. Retrieved 24-06-2024 from https://www.nm.org/conditions-and-care-areas/neurosciences/comprehensive-stroke-centers/life-after-stroke
3. Mohebbi, A. (2020). Human-Robot Interaction in Rehabilitation and Assistance: a Review. *Curr Robot Rep, 1*, 131–144. https://doi.org/10.1007/s43154-020-00015-4
4. Basili, P., Huber, M., Brandt, T., Hirche, S., & Glasauer, S. (2009). Investigating Human-Human Approach and Hand-Over. In H. Ritter, G. Sagerer, R. Dillmann, & M. Buss (Eds.), *Human Centered Robot Systems* (pp. 151–160). Springer. https://doi.org/10.1007/978-3-642-10403-9_16
5. Ortenzi, V., Cosgun, A., Pardi, T., Chan, W. P., Croft, E. A., & Kulić, D. (2021). Object Handovers: A review for Robotics. *IEEE Transactions on Robotics, 37*(6), 1855–1873. https://doi.org/10.1109/tro.2021.3075365
6. National Heart, Lung, and Blood Institute. (2023, May 26). *What is a stroke? | NHLBI, NIH*. Retrieved 24-06-2024 from https://www.nhlbi.nih.gov/health/stroke
7. Chohan, S. A., Venkatesh, P. K., & How, C. H. (2019). Long-term complications of stroke and secondary prevention: an overview for primary care physicians. *Singapore Medical Journal, 60*(12), 616–620.
8. Feigin, V. L., Abajobir, A. A., Abate, K. H., Abd-Allah, F., Abdulle, A., Abera, S. F., Abyu, G. Y., Ahmed, M. B., Aichour, A. N., Aichour, I., Aichour, M. T. E., Akinyemi, R., Alabed, S., Al-Raddadi, R., Alvis-Guzmán, N., Amare, A. T., Ansari, H., Anwari, P., Ärnlöv, J., ... Vos, T. (2019). Global, regional, and national burden of neurological disorders, 1990–2016: a systematic analysis for the Global Burden of Disease Study 2016. *Lancet Neurology, 18*(5), 459–480. https://doi.org/10.1016/s1474-4422(18)30499-x
9. Park, G. Y., Im, S., Lee, S. J., & Pae, C. U. (2016). The Association between Post-Stroke Depression and the Activities of Daily Living/Gait Balance in Patients with First-Onset Stroke Patients. *Psychiatry Investig, 13*(6), 659-664. https://doi.org/10.4306/pi.2016.13.6.659
10. Robinson, R. G., & Jorge, R. E. (2016). Post-Stroke Depression: A Review. *The American Journal of Psychiatry, 173*(3), 221–231. https://doi.org/10.1176/appi.ajp.2015.15030363
11. Hackett, M. L., & Pickles, K. (2014). Part I: Frequency of Depression after Stroke: An Updated Systematic Review and Meta-Analysis of Observational Studies. *International Journal of Stroke, 9*(8), 1017-1025. https://doi.org/10.1111/ijs.12357
12. Kwok, T., Lo, R. C., Wong, E., Tang, W. K., Mok, V., & Kai-Sing, W. (2006). Quality of Life of Stroke Survivors: A 1-Year Follow-Up Study. *Archives of Physical Medicine and Rehabilitation, 87*(9), 1177–1182. https://doi.org/10.1016/j.apmr.2006.05.015
13. Li, J., Yang, L., Lv, R., et al. (2023). Mediating effect of post-stroke depression between activities of daily living and health-related quality of life: meta-analytic structural equation modeling. *Quality Life Research, 32*, 331–338. https://doi.org/10.1007/s11136-022-03225-9
14. McCullagh, E., Brigstocke, G. H., Donaldson, N., & Kalra, L. (2005). Determinants of caregiving burden and quality of life in caregivers of stroke patients. *Stroke, 36*(10), 2181–2186. https://doi.org/10.1161/01.str.0000181755.23914.53

15. Tsai, P. C., Yip, P. K., Tai, J. J., & Lou, M. F. (2015). Needs of family caregivers of stroke patients: a longitudinal study of caregivers' perspectives. *Patient Preference and Adherence, 9*, 449–457. https://doi.org/10.2147/PPA.S77713
16. Shibata, S., Sahbi, B. M., Tanaka, K., & Shimizu, A. (1997). An analysis of the process of handing over an object and its application to robot motions. *IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*. https://doi.org/10.1109/ICSMC.1997.625724
17. Huber, M., Lenz, C., Rickert, M., Knoll, A., Brandt, T., & Glasauer, S. (2008). Human Preferences in Industrial Human-Robot Interactions. In *International Workshop on Cognition for Technical Systems*.
18. Ortenzi, V., Cosgun, A., Pardi, T., Chan, W. P., Croft, E. A., & Kulić, D. (2021). Object Handovers: A review for Robotics. *IEEE Transactions on Robotics, 37*(6), 1855–1873. https://doi.org/10.1109/tro.2021.3075365
19. Shahria, M. T., Ghommam, J., Fareh, R., & Rahman, M. H. (2024). Vision-Based object manipulation for activities of daily living assistance using assistive robot. *Automation, 5*(2), 68–89. https://doi.org/10.3390/automation5020006
20. Leroux, M., Raison, M., Adadja, T., & Achiche, S. (2015). Combination of eyetracking and computer vision for robotics control. In *Technologies for Practical Robot Applications*. https://doi.org/10.1109/tepra.2015.7219692
21. Ding, D., Styler, B., Chung, C., & Houriet, A. (2022). Development of a Vision-Guided Shared-Control system for assistive robotic manipulators. *Sensors, 22*(12), 4351. https://doi.org/10.3390/s22124351
22. Poirier, S., Routhier, F., & Campeau-Lecours, A. (2019). Voice Control Interface Prototype for Assistive Robots for People Living with Upper Limb Disabilities. In *International Conference on Rehabilitation Robotics*. https://doi.org/10.1109/icorr.2019.8779524
23. Langer, D., Legler, F., Kotsch, P., Dettmann, A., & Bullinger, A. C. (2022). I Let Go Now! Towards a Voice-User Interface for Handovers between Robots and Users with Full and Impaired Sight. *Robotics, 11*(5), 112. https://doi.org/10.3390/robotics11050112
24. Langer, D., Legler, F., Diekmann, P., Dettmann, A., Glende, S., & Bullinger, A. C. (2024). Got it? Comparative Ergonomic evaluation of robotic object handover for visually impaired and sighted users. *Robotics, 13*(3), 43. https://doi.org/10.3390/robotics13030043
25. Haseeb, M. A., Kyrarini, M., Jiang, S., Ristic-Durrant, D., & Gräser, A. (2018). Head Gesture-based Control for Assistive Robots. In *Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference (PETRA '18)* (pp. 379–383). Association for Computing Machinery. https://doi.org/10.1145/3197768.3201574
26. Canal, G., Escalera, S., & Angulo, C. (2016). A real-time Human-Robot Interaction system based on gestures for assistive scenarios. *Computer Vision and Image Understanding, 149*, 65-77. https://doi.org/10.1016/j.cviu.2016.03.004
27. Mohebbi, A. (2020). Human-Robot Interaction in Rehabilitation and Assistance: a Review. *Current Robotics Reports, 1*(3), 131–144. https://doi.org/10.1007/s43154-020-00015-4
28. Strabala, K. W., Lee, M. K., Dragan, A. D., Forlizzi, J. L., Srinivasa, S., Cakmak, M., & Micelli, V. (2013). Towards seamless Human-Robot handovers. *Journal of Human-robot Interaction, 2*(1), 112–132. https://doi.org/10.5898/jhri.2.1.strabala

29. Castro, A., Silva, F., & Santos, V. (2021). Trends of Human-Robot Collaboration in industry contexts: handover, learning, and metrics. *Sensors, 21*(12), 4113. https://doi.org/10.3390/s21124113
30. Chen, Q., Wan, L., & Pan, Y. (2023). Robotic pick-and-handover maneuvers with camera-based intelligent object detection and impedance control. *Transactions of the Canadian Society for Mechanical Engineering, 47*(4), 486–496. https://doi.org/10.1139/tcsme-2022-0176
31. Yu, H., Kamat, V. R., Menassa, C. C., McGee, W., Guo, Y., & Lee, H. (2023). Mutual physical state-aware object handover in full-contact collaborative human-robot construction work. *Automation in Construction, 150*, 104829. https://doi.org/10.1016/j.autcon.2023.104829
32. ROBOTIS. (2024). OpenManipulator-X. *ROBOTIS e-Manual*. Retrieved 24-06-2024 from https://emanual.robotis.com/docs/en/platform/openmanipulator_x/overview/
33. Robotis. (n.d.). Chain example from the OpenManipulator [Computer software]. *Robotis*. Retrieved 24-06-2024 from https://emanual.robotis.com/docs/en/platform/openmanipulator_x/quick_start_guide/
34. RM-X52-TNM. (n.d.). *ROBOTIS*. Retrieved 24-06-2024 from https://en.robotis.com/shop_en/item.php?it_id=905-0024-000
35. OpenCR1.0. (n.d.). *ROBOTIS*. Retrieved 24-06-2024 from https://en.robotis.com/shop_en/item.php?it_id=903-0257-000
36. SMPS 12V 5A PS-10 [US-110V]. (n.d.). *ROBOTIS*. Retrieved 24-06-2024 from https://en.robotis.com/shop_en/item.php?it_id=903-0126-00
37. InvenSense. (2013). MPU-6000 and MPU-6050 Product Specification Revision 3.4. Retrieved 24-06-2024 from https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf
38. JicJac. (2006, November 30). Various Clothing. *OpenClipart*. Retrieved 24-06-2024 from https://www.openclipart.org/detail/11225/various-clothing
39. Weisstein, E. W. (n.d.). Euler Angles. In *MathWorld--A Wolfram Web Resource*. Retrieved 24-06-2024 from https://mathworld.wolfram.com/EulerAngles.html
40. Adafruit. (2024). MPU6050 6-DoF Accelerometer and Gyro Guide: Arduino. *Adafruit*. Retrieved 24-06-2024 from https://learn.adafruit.com/mpu6050-6-dof-accelerometer-and-gyro/arduino
41. Spectra Symbol. (2014). Flex Sensor [PDF]. Retrieved 24-06-2024 from https://cdn.sparkfun.com/assets/9/5/b/f/7/FLEX_SENSOR_-_SPECIAL_EDITION_DATA_SHEET_v2019__Rev_A_.pdf
42. ROBOTIS. (2024). DYNAMIXEL Workbench - USB to DXL Example. *ROBOTIS e-Manual*. Retrieved 24-06-2024 from https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_workbench/
43. ROBOTIS. (2024). XM430-W350. *ROBOTIS e-Manual*. Retrieved 24-06-2024 from https://emanual.robotis.com/docs/en/dxl/x/xm430-w350/

APPENDIX A: CODE SCRIPTS

Arduino board script: glove_control

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

const int flexSensorPin1 = A0;  // Analog pin A0 to which the first flex
sensor is connected
const int flexSensorPin2 = A1;  // Analog pin A1 to which the second flex
sensor is connected

Adafruit_MPU6050 mpu1;
Adafruit_MPU6050 mpu2;

#define NUM_READINGS 100  // Amount of calibration readings

float basePitch1[NUM_READINGS], baseRoll1[NUM_READINGS];
float basePitch2[NUM_READINGS], baseRoll2[NUM_READINGS];
float avgPitch1, avgRoll1;
float avgPitch2, avgRoll2;
unsigned long lastTime;

void setup(void) {
  Serial.begin(115200);   // Sets baud rate
  while (!Serial) delay(10);

  Serial.begin(9600);  // Initialise serial communication at 9600 bps

  if (!mpu1.begin(0x68)) {
    Serial.println("Failed to find MPU6050 chip 1");
    while (1) delay(10);
  }

  if (!mpu2.begin(0x69)) {
    Serial.println("Failed to find MPU6050 chip 2");
    while (1) delay(10);
  }

  Serial.println("MPU6050 sensors initialized. Calibrating sensors...");  //
Calibration message, user should not move hand when message is displayed

  calibrateSensors();
  lastTime = millis();
}

void calibrateSensors() {
  for (int i = 0; i < NUM_READINGS; i++) {
    sensors_event_t a1, g1, temp1;
    sensors_event_t a2, g2, temp2;

    mpu1.getEvent(&a1, &g1, &temp1);
    mpu2.getEvent(&a2, &g2, &temp2);

    basePitch1[i] = atan2(a1.acceleration.y, a1.acceleration.z) * 180 / PI;
// Pitch and roll are calculated according to equations (1) and (2)
    baseRoll1[i] = atan2(-a1.acceleration.x, sqrt(a1.acceleration.y *
a1.acceleration.y + a1.acceleration.z * a1.acceleration.z)) * 180 / PI;
    basePitch2[i] = atan2(a2.acceleration.y, a2.acceleration.z) * 180 / PI;
    baseRoll2[i] = atan2(-a2.acceleration.x, sqrt(a2.acceleration.y *
a2.acceleration.y + a2.acceleration.z * a2.acceleration.z)) * 180 / PI;

    delay(50);
  }

  float sumPitch1 = 0, sumRoll1 = 0;
  float sumPitch2 = 0, sumRoll2 = 0;

  for (int i = 0; i < NUM_READINGS; i++) {
    sumPitch1 += basePitch1[i];
    sumRoll1 += baseRoll1[i];
    sumPitch2 += basePitch2[i];
    sumRoll2 += baseRoll2[i];
  }

  avgPitch1 = sumPitch1 / NUM_READINGS;
  avgRoll1 = sumRoll1 / NUM_READINGS;
  avgPitch2 = sumPitch2 / NUM_READINGS;
  avgRoll2 = sumRoll2 / NUM_READINGS;

  Serial.println("Calibration complete.");    // Now user can move hand
  Serial.print("Base Pitch1: "); Serial.println(avgPitch1);
  Serial.print("Base Roll1: "); Serial.println(avgRoll1);
  Serial.print("Base Pitch2: "); Serial.println(avgPitch2);
  Serial.print("Base Roll2: "); Serial.println(avgRoll2);
}

void loop() {
  int flexValue1 = analogRead(flexSensorPin1);  // Read the analog value
from the first flex sensor
  int flexValue2 = analogRead(flexSensorPin2);  // Read the analog value
from the second flex sensor

  sensors_event_t a1, g1, temp1;
  sensors_event_t a2, g2, temp2;

  mpu1.getEvent(&a1, &g1, &temp1);
  mpu2.getEvent(&a2, &g2, &temp2);
  unsigned long currentTime = millis();
  float deltaTime = (currentTime - lastTime) / 1000.0;
  lastTime = currentTime;

  float pitch1 = atan2(a1.acceleration.y, a1.acceleration.z) * 180 / PI -
avgPitch1;    // Pitch and roll are calculated according to equations (1)
and (2)
  float roll1 = atan2(-a1.acceleration.x, sqrt(a1.acceleration.y *
a1.acceleration.y + a1.acceleration.z * a1.acceleration.z)) * 180 / PI -
avgRoll1;

  float pitch2 = atan2(a2.acceleration.y, a2.acceleration.z) * 180 / PI -
avgPitch2;
  float roll2 = atan2(-a2.acceleration.x, sqrt(a2.acceleration.y *
a2.acceleration.y + a2.acceleration.z * a2.acceleration.z)) * 180 / PI -
avgRoll2;

  float deltaPitch = pitch2 - pitch1;   // Gives the angles between the two
MPUs. Numbering is important, changes +/- in the values if switched
  float deltaRoll = roll2 - roll1;

  // Format data string
  String dataString = String(deltaPitch) + "," + String(deltaRoll) + "," +
String(flexValue1) + "," + String(flexValue2);
  Serial.println(dataString);

  delay(500);
}
```

Python script: serial_ports

```python
import serial
import threading
import time

mkr_port = 'COM9' # Adjust COM if necessary
opencr_port = 'COM5'
baud_rate = 115200 # Baud rate of the sensors in the Arduino code

def open_serial_port(port, baud_rate): # Defines opening of the ports
    try:
        ser = serial.Serial(port, baud_rate, timeout=2)
        print(f"Opened port: {port}")
        return ser
    except serial.SerialException as e:
        print(f"Error opening port {port}: {e}")
        return None

mkr_serial = open_serial_port(mkr_port, baud_rate)
opencr_serial = open_serial_port(opencr_port, baud_rate)

if not mkr_serial or not opencr_serial:
    print("Failed to open one or more serial ports. Exiting.")
    exit()

def relay_mkr_to_opencr(): # Establishes one way communication (Arduino to 
OpenCR)
    while True:
        try:
            if mkr_serial.in_waiting > 0:
                data = mkr_serial.readline().decode().strip()
                if data:
                    opencr_serial.write((data + "\n").encode())
                    print("Sent to COM5 (Robot Arm):", data)
        except serial.SerialException as e:
            print(f"Error reading from MKR: {e}")
            break

def relay_opencr_to_mkr(): # Prints received data, necessary for calibration
    while True:
        try:
            if opencr_serial.in_waiting > 0:
                data = opencr_serial.readline().decode().strip()
                if data:
                    mkr_serial.write((data + "\n").encode())
                    print("Received from COM5 (Robot Arm):", data)
        except serial.SerialException as e:
            print(f"Error reading from OpenCR: {e}")
            break
```

```python
thread_mkr_to_opencr = threading.Thread(target=relay_mkr_to_opencr)
thread_opencr_to_mkr = threading.Thread(target=relay_opencr_to_mkr)

thread_mkr_to_opencr.start()
thread_opencr_to_mkr.start()

try:
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    print("Exiting...")
finally:
    if mkr_serial.is_open:
        mkr_serial.close()
    if opencr_serial.is_open:
        opencr_serial.close()
    print("Serial ports closed.")
```

OpenCR board script: robot_control

```
#include <DynamixelSDK.h>

#define DEVICENAME "1"  // Adjust to the correct USB port
#define BAUDRATE 1000000
#define PROTOCOL_VERSION 2.0

#define DXL_ID_11 11    // Servo motor IDs
#define DXL_ID_12 12
#define DXL_ID_13 13
#define DXL_ID_14 14
#define DXL_ID_15 15

dynamixel::PortHandler *portHandler;
dynamixel::PacketHandler *packetHandler;

void setup() {
  Serial.begin(115200);   // Baud rate from the sensors

  portHandler = dynamixel::PortHandler::getPortHandler(DEVICENAME);
  packetHandler =
dynamixel::PacketHandler::getPacketHandler(PROTOCOL_VERSION);

  if (portHandler->openPort()) {
    Serial.println("Succeeded to open the port!");
  } else {
    Serial.println("Failed to open the port!");
    return;
  }

  if (portHandler->setBaudRate(BAUDRATE)) {
    Serial.println("Succeeded to change the baudrate!");
  } else {
    Serial.println("Failed to change the baudrate!");
    return;
  }

  // Enable Torque for all servos
  packetHandler->write1ByteTxRx(portHandler, DXL_ID_11, 64, 1);
  packetHandler->write1ByteTxRx(portHandler, DXL_ID_12, 64, 1);
  packetHandler->write1ByteTxRx(portHandler, DXL_ID_13, 64, 1);
  packetHandler->write1ByteTxRx(portHandler, DXL_ID_14, 64, 1);
  packetHandler->write1ByteTxRx(portHandler, DXL_ID_15, 64, 1);
}

void moveServosToPosition(int servo11, int servo12, int servo13, int
servo14, int servo15) {
  packetHandler->write4ByteTxRx(portHandler, DXL_ID_11, 116, servo11);
  packetHandler->write4ByteTxRx(portHandler, DXL_ID_12, 116, servo12);
  packetHandler->write4ByteTxRx(portHandler, DXL_ID_13, 116, servo13);
  packetHandler->write4ByteTxRx(portHandler, DXL_ID_14, 116, servo14);
  if (servo15 != -1) {
    packetHandler->write4ByteTxRx(portHandler, DXL_ID_15, 116, servo15);
  }
}

bool isGripperHoldingItem() {   // Should allow the gripper to open when
not fully closing due to holding an item
  uint16_t current;
  int dxl_comm_result = packetHandler->read2ByteTxRx(portHandler,
DXL_ID_15, 126, &current);  // 126 is the address for present current
  if (dxl_comm_result != COMM_SUCCESS) {
    Serial.println(packetHandler->getTxRxResult(dxl_comm_result));
  }
  return current > 5000;  // Current threshold for the gripper in mA,
adjust if necessary
}

void closeGripper() {
  int closePosition = 2700;  // Gripper close (238 degrees = ~2700 EC =
full close)
  int increment = 10;  // Small steps for closing

  while (true) {
    packetHandler->write4ByteTxRx(portHandler, DXL_ID_15, 116,
closePosition);
    delay(100);  // Small delay to allow the servo to move

    if (isGripperHoldingItem()) {
      break;  // Stop closing when item is held
    }
    closePosition -= increment;  // Incrementally close more
    if (closePosition <= 0) {
      break;  // Safety check to prevent over-closing
    }
  }
}

void loop() {
  if (Serial.available() > 0) {
    String data = Serial.readStringUntil('\n');
    if (data.length() > 0) {
      float deltaPitch, deltaRoll;
      int flexValue1, flexValue2;
      sscanf(data.c_str(), "%f,%f,%f,%d,%d", &deltaPitch, &deltaRoll,
&flexValue1, &flexValue2);

      // Move servos based on delta pitch value
      if (deltaPitch <= -15) {  // Adjust value if necessary. If MPU 1
and 2 are switched, this value becomes positive
        // Giving position
        moveServosToPosition(360 * 4096 / 360, 240 * 4096 / 360, 100 *
4096 / 360, 180 * 4096 / 360, -1);   // Servomotor ID 12 and 13 move and
decide position
      } else if (deltaPitch >= 25) {  // Adjust value if necessary. If
MPU 1 and 2 are switched, this value becomes negative
        // Base position
        moveServosToPosition(360 * 4096 / 360, 140 * 4096 / 360, 210 *
4096 / 360, 180 * 4096 / 360, -1);
      }

      // Control gripper based on flex sensor values
      if (flexValue1 < 300 || flexValue2 < 400) {   // Adjust if
necessary. May change if flex sensor position in voltage divider circuit
is swapped, or resistor value differs
        // Open gripper
        packetHandler->write4ByteTxRx(portHandler, DXL_ID_15, 116,
1252); // Gripper open at 110 degrees (110/360 * 4096)
      } else {
        // Close gripper
        closeGripper();
      }

      // Print out the received data for debugging, shown through Python
      Serial.print(" Pitch: "); Serial.print(deltaPitch);
      Serial.print(" Roll: "); Serial.print(deltaRoll);
      Serial.print(" Flex Sensor 1: "); Serial.print(flexValue1);
      Serial.print(" Flex Sensor 2: "); Serial.println(flexValue2);
    }
  }
}
```

APPENDIX B: MPU-6050 CASING

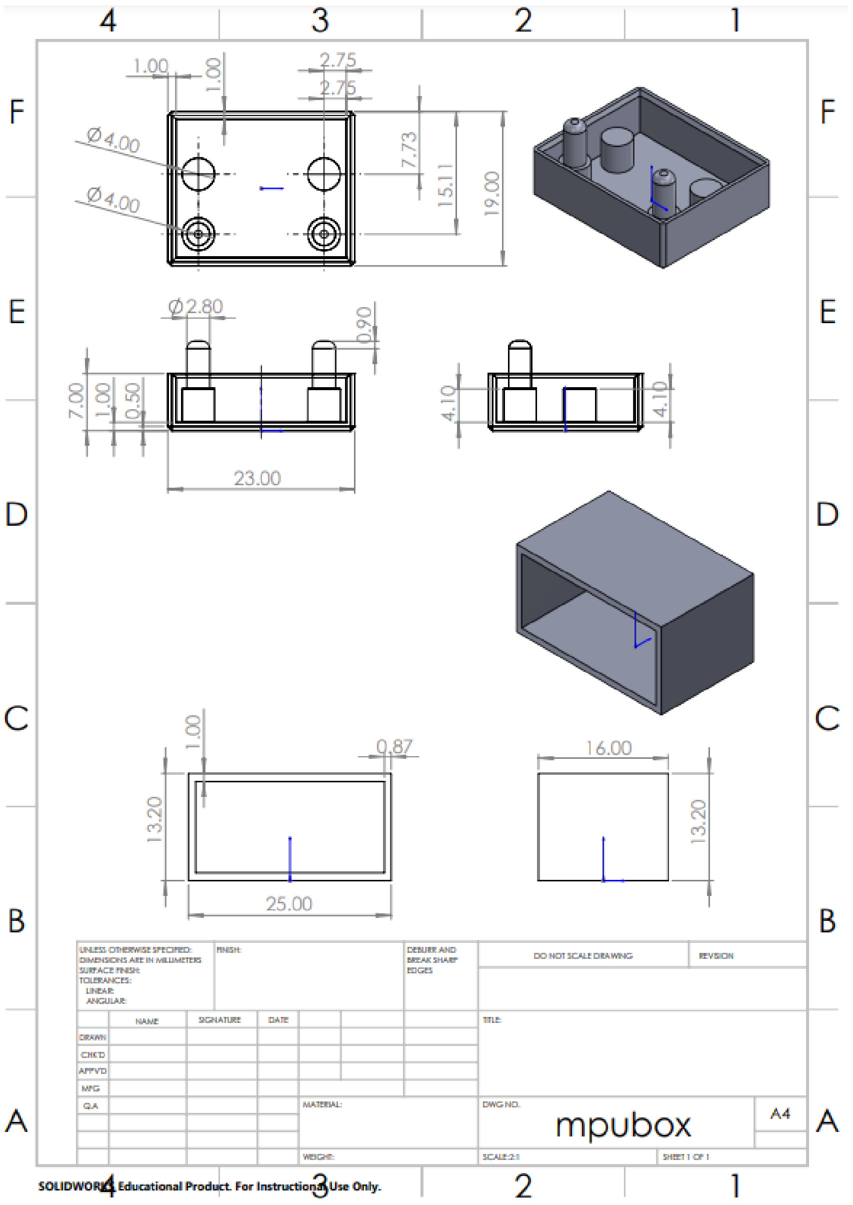


Figure 9: Full drawing of the 3D-printed casings used for the MPU-6050 sensors. Drawing and casings created and provided by Paul Fetchenhauer, used with permission.

APPENDIX C: FULL DATA AND RESULTS

Table 4: Individual handover times per item.

Handover attempt	Time for item 1 (s)	Time for item 2 (s)	Time for item 3 (s)	Time for item 4 (s)	Time for item 5 (s)
1	10.47	13.41	7.69	4.34	4.72
2	11.76	11.84	6.50	10.15	6.10
3	13.81	7.77	9.13	6.38	9.97
4	12.72	9.53	6.89	6.16	5.90
5	8.96	6.64	8.12	6.04	5.84
6	9.43	7.27	7.48	7.05	4.50
7	8.56	6.84	8.94	5.99	5.38
8	9.17	8.57	6.92	3.97	6.74
9	8.23	11.86	9.93	7.25	5.81
10	7.92	7.91	8.16	4.84	6.08
11	8.64	8.90	12.47	5.76	4.92
12	8.35	8.98	8.61	4.51	8.07
13	7.83	7.42	7.45	8.31	6.65
14	8.98	8.36	7.76	6.36	7.74
15	9.49	13.26	8.18	4.17	6.98
16	8.13	7.87	10.72	8.59	10.08
17	7.22	7.55	9.68	4.75	5.19
18	6.79	6.87	7.61	5.16	6.83
19	7.38	8.24	10.38	4.78	5.54
20	7.84	7.62	7.92	5.24	4.23

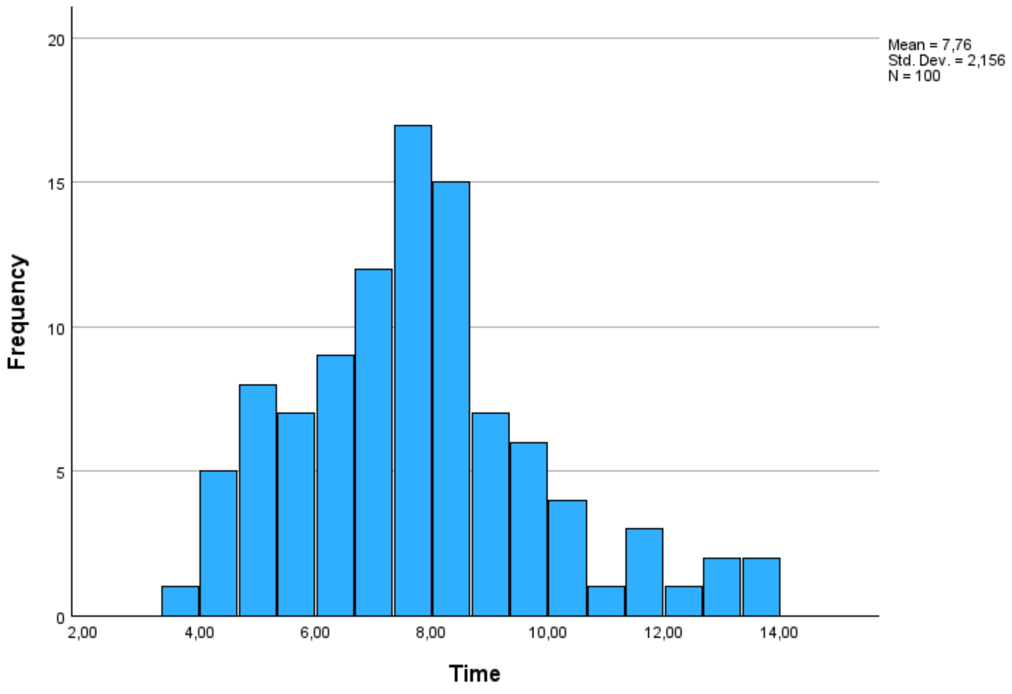


Figure 10: Histogram graph of the frequency distribution of handover time.

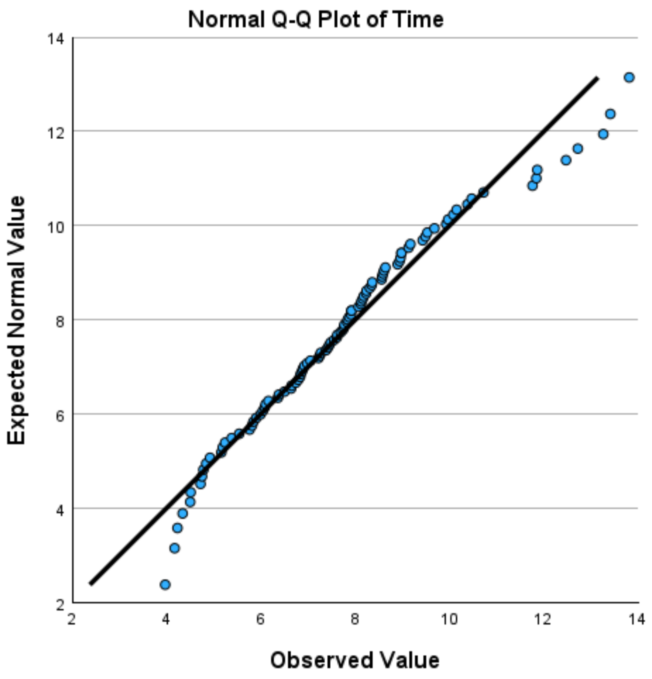
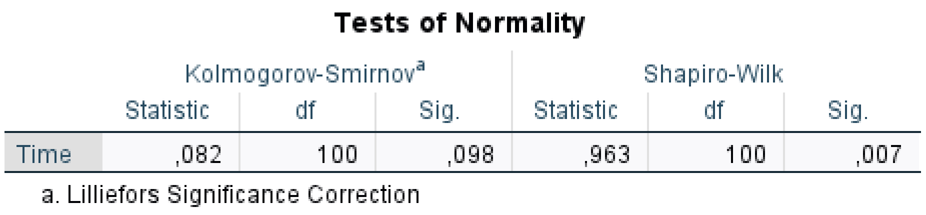


Figure 11: Q-Q plot distribution of the handover time.

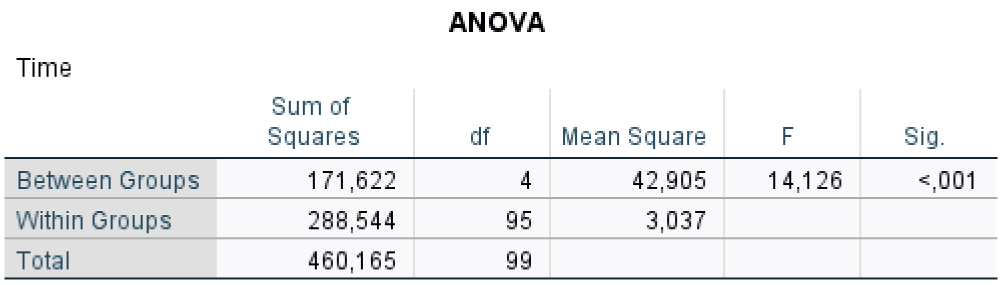


Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Time	,082	100	,098	,963	100	,007

a. Lilliefors Significance Correction

Figure 12: Full results of the performed Shapiro-Wilk test in SPSS.



ANOVA

Time

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	171,622	4	42,905	14,126	<,001
Within Groups	288,544	95	3,037		
Total	460,165	99			

Figure 13: Full results of the performed ANOVA test in SPSS