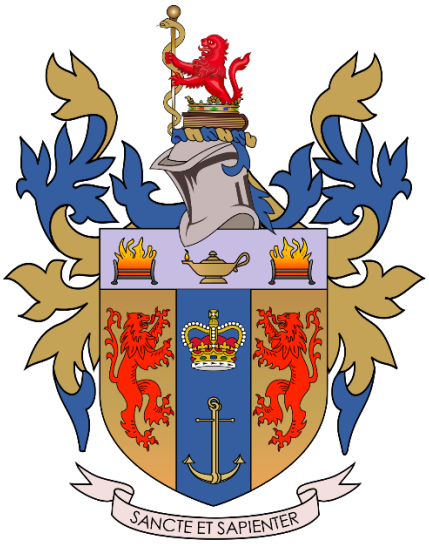
KING'S COLLEGE LONDON

FACULTY OF LIFE SCIENCE AND MEDICINE

7MRI0060: Applied Medical Robotics Report

Author: The Hung Dang

Student Number: 23133931



Submitted in partial fulfilment of the requirements for the Applied Medical Robotics Module as part of the Master of Science in Healthcare Technologies at King's College London University

2nd January 2024

Contents

1. Introduction

Humans are complex and mysterious machines with many undiscovered aspects (1). Not only medicine is a demanding profession, but the lack of workforce also intensify the stress experienced by healthcare provider, these conditions were greatly exaggerated and became more apparent during the COVID-19 pandemic (2, 3). Robotics offers opportunities to address challenging medical problems as well as the potential to alleviate the workload (4, 5). For example, robotic-assisted endoscopy allow doctor more control to explore the hard-to-visual organ (7) or neuro surgical robots could aid in meticulous action (8). Utilization of robotics in medicine can be traced back over half a century ago (9). The development of robotics theory, control theory, sensor technology, actuators, and computational hardware allowed for precise real-time control of robots, laying the groundwork for many robotics systems in medicine (10-13).

Before building highly functional robots, the first step in constructing a robot is making it more precisely to a single point, which is why we perform this project. The project aims to construct planar two degree of freedom robotic arms for drawing. We proposed three objectives: firstly, the robot must have two arms, each is 78mm length with a mechanism for holding a pen at the end effector; secondly, in the joint space, the arms must move to the desired position with the error less than 0.5 degrees for each joint while exhibiting minimal and acceptable overshoot; last but not least, some features are incorporated into the robots for easier and safter control including: homing routine, object detection, intuitive user interface, and pathway generation. We successfully built a two degree of freedom robot, being able to draw some simple shapes, with overshoot less than 2 degree and a precision of 0.5 degree. Some features were successfully tested and incorporated.

2. Methodology

2.1. Computer-Aided Design, 3D printing and hardware integration

2.1.1. Computer-Aided Design (CAD)

The robot was designed using cloud-based CAD websites (14). The designs were planned in advance and shared for editing between group members. Different versions of the design were stored in separate files. The design criteria are:

- There are three separate main parts: the base, the first arm, and the second arm.
- The base can hold the Arduino board and H-bridge.
- The two arms are exactly 78mm in length.
- There are holders for motors in the base or the arms.
- There are mechanisms for connecting parts using two bearings: 10x19x5mm and 6x10x3mm (inside diameter x outside diameter x race width).
- There is space for wiring and maintaining the hardware.
- There is a mechanism for holding pens of different sizes and limiting all six degrees of freedom of the pens.
- There are detachable walls for the homing routine (moving the arm back to original position).

2.1.2. 3D printing

The CAD files were exported in STL format, with a “fine” resolution in Onshape. The file was later sliced using UltiMaker Cura (15). 3D printings were performed using Flsun QQ-S or Flsun Super Racer (16) and Polylactic Acid Plus (PLA) or PLA+ filaments. Regarding the setting, 0.2-0.3mm precision was used for printing test versions, and 0.1mm precision was used for the final product. Support was turned on, and different infill levels were tested. Other settings were set as UltiMaker Cura recommended.

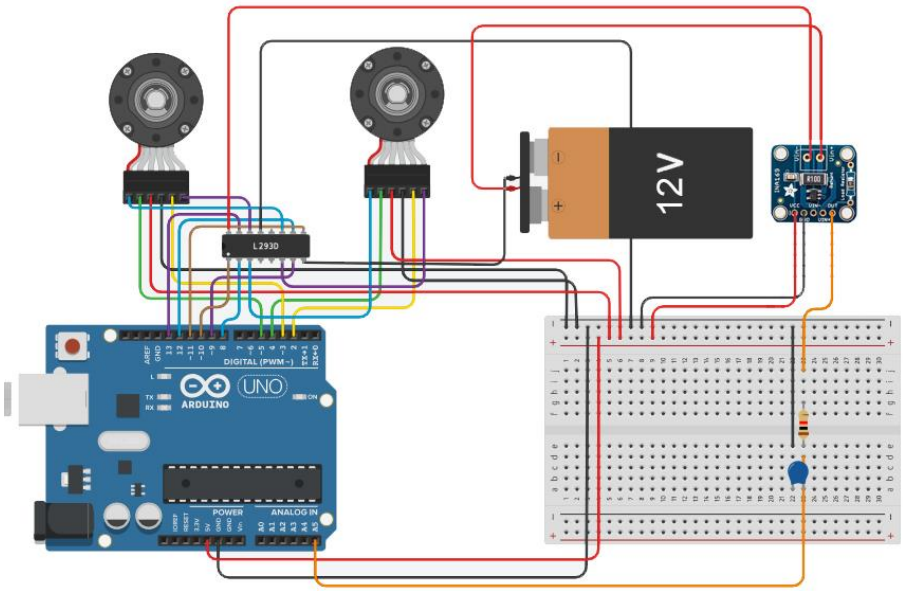
2.1.3. Hardware Components

Two motors were used (Premium N20 Gear Motor 100:1 Ratio, model NFP-JGA12-N20-12200), the encoder incorporated in the motor was incremental optical encoder, with pulses per revolution (PPR) of outer shaft is 82.77 and a gear ratio of 100 (17). The velocities of the motors were controlled using 4tronix L298N Dual H-Bridge Motor Driver Module (18), powered by a 12-V power adapter. A low-pass filter and current sensor (Adafruit 1164) were used for measuring the current output from the H-bridge supporting the homing routine. The highest frequency components of the current were determined using Nyquist theorem (19). Appropriate values for capacitance C and resistance R were then chosen for an appropriate cut-off frequency f_{cutoff} of the low-pass filter using the formula:

$$f_{cutoff} = \frac{1}{2\pi RC}$$

All off the system were controlled by Arduino Uno Rev3 (20). The diagram of interconnection is presented in *Figure 1*.

Figure 1. Diagram of interconnection of the hardware



2.2. Low-level control

2.2.1. Homing routine

Homing routines were run once at the beginning of the code (in the *setup()* function). After an arm rotating at a constant voltage hits an object, the current will spike, signaling the Arduino board to stop the motors. The algorithm was applied to the second arm first, and then to the first arm (Supplementary pseudocode 1). The cutoffs for the currents were tested separately for each arm. It was chosen as the smallest value where the arm can rotate without triggering an automatic stop.

2.2.2. Encoder reading

The encoders are read using Encoder library (21). The values of motor angles are converted to degrees using the following formula:

$$\frac{Encoder\_value \cdot 360°}{PPR \cdot GR}$$

Where $Encoder\_value$ are values obtained from the encoders, pulses per revolution (PPR) of outer shaft is 82.77, gear ratio (GR) is 100. When the angles are less than $-180°$ degrees or greater than $180°$ degrees, the encoder values are reset within $(-180°, 180°)$ range

2.2.3. Motor control

The voltage input to the motor is controlled through pulse width modulation duty cycle value (PWM). Our project tested two control systems: position control (main system) and velocity control. A proportional-integral-derivative (PID) control was used (22). Later, integral clamping was added as an anti-windup technique, derivative filtering was used to smoothen the signal from the derivatives part of PID controls.

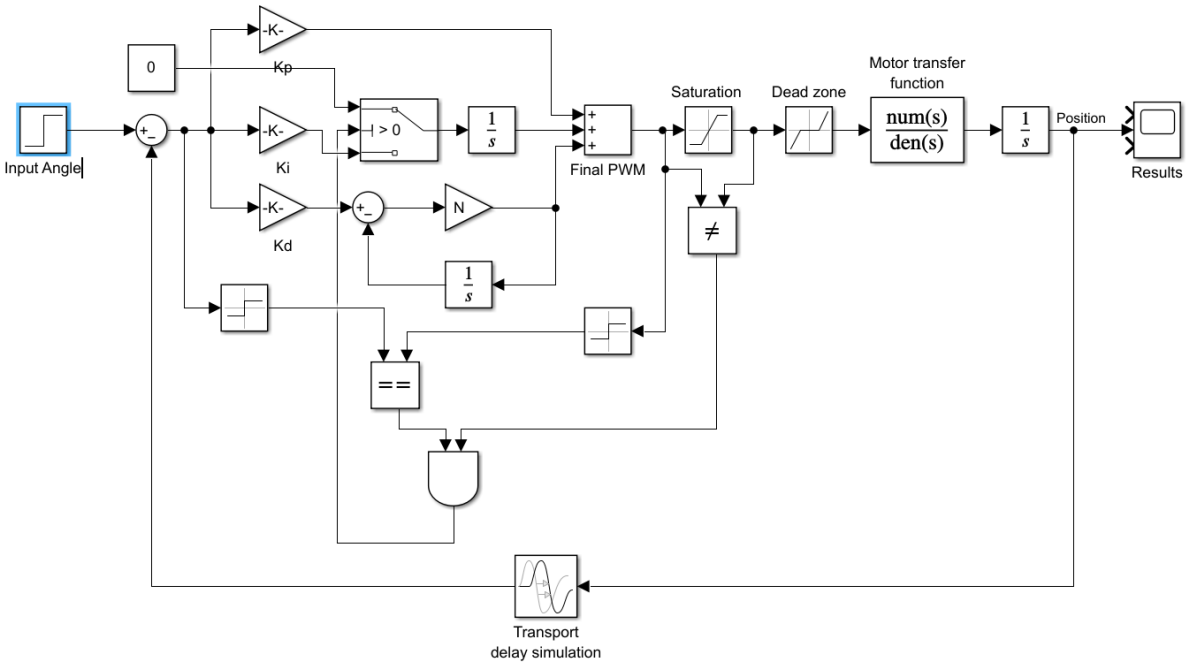
Mathematical models of two motors were identified. Motor 1 and 2 were run at constant PWM values (35 and 30, respectively) until reaching a constant speed. Second-order transfer functions with two poles and no zero were used to fit the data. The models were validated with the data collected at PWM values of 30/40 for motor 1, and 25/35 for motor 2.

Simulations were performed in Simulink (Figure 2) (23). Errors between input signal and current state go through a PID control with gains of Kp, Ki, and Kd values for P, I, and D controls. Integral clamping is performed for integral signal. A derivative filter at frequency N were implemented for the D signals. The output PWM values are reset within a saturation zone (signals outside this zone are larger than the maximal signals) and outside a dead zone (signals within this zone are not strong enough to rotate the motor); the feedback signals are delayed by a

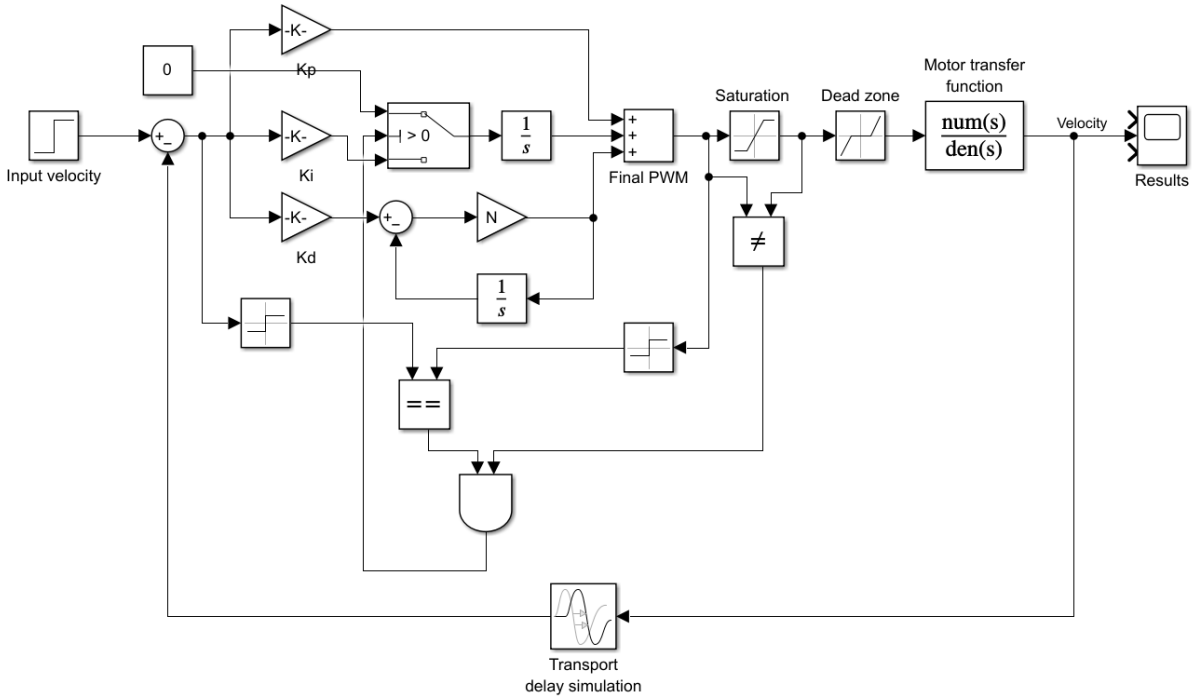
period equal to the sampling time. The simulated PID values obtained were later used and tuned again in the physical system.

Figure 2. Simulation models for position control (a) and velocity control systems (b)

a)



b)



2.2.4. Object detection

Object detections were incorporated into the main code. When the voltage is higher than a voltage cutoff, but the velocity is smaller than a velocity cutoff over a certain number of loops, the machine will stop in 3 seconds (Supplementary pseudocode 2).

2.3. Mathematical formula

2.3.1. Forward kinematics

Forward kinematics was performed using the following formula:

Arm 1:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} r_1 cos(\theta_1) \\ r_1 \dot{s}in(\theta_1) \end{bmatrix}$$

Arm 2:

$$\begin{bmatrix} \mathrm{x}_2 \\ \mathrm{y}_2 \end{bmatrix} = \begin{bmatrix} \mathrm{r}_2\cos(\theta_1 + \theta_2) + \mathrm{r}_1\cos(\theta_1) \\ \mathrm{r}_2\sin(\theta_1 + \theta_2) + \mathrm{r}_1\sin(\theta_1) \end{bmatrix}$$

Where x_1, y_1, x_2, y_2 is the x, y coordinate of the tips of arm 1 and arm 2; θ_1 and θ_2 are the angle from motor 1 and motor 2 shaft. The length of each arm is $r_1 = r_2 = 78mm$.

2.3.2. Inverse kinematics

2.3.2.1. Analytical inverse kinematics

Inverse kinematics was calculated analytically using the following formula:

$$cos(\theta_2) = \frac{x^2 + y^2 - r_1^2 - r_2^2}{2r_1 r_2}$$

$$sin(\theta_1) = \pm\sqrt{1 - cos(\theta_2)^2}$$

$$\theta_2 = atan2\big(sin(\theta_2), cos(\theta_2)\big)$$

$$(Only\ the\ positive\ value\ of\ \sin(\theta 2)\ was\ used)$$

$$\theta_1 = atan2(y, x) - atan2\big(r_2 sin(\theta_2), r_1 + r_2 cos(\theta_2)\big)$$

Where x and y are (x, y) coordinate of the end effector, $r_1 = r_2 = 78mm$.

2.3.2.2. Calculating inverse kinematics using differential method

Another approach to solve inverse kinematics was tested. By using the Jacobian matrix, the joint angles for the target $x_{target},\ y_{target}$ coordinate in the task space were approximated by constructing 2 series of $\theta_{1_n}\ and\ \theta_{2_n}$ that convert to $\theta_1\ and\ \theta_2$. Jacobian matrix J is calculated as:

$$J = \begin{bmatrix} -r_1 sin(\theta_{1_n}) - r_2 sin(\theta_{1_n} + \theta_{2_n}) & -r_2 sin(\theta_{1_n} + \theta_{2_n}) \\ +r_1 cos(\theta_{1_n}) + r_2 cos(\theta_{1_n} + \theta_{2_n}) & +r_2 cos(\theta_{1_n} + \theta_{2_n}) \end{bmatrix}$$

Where $r_1 = r_2 = 78mm$ are the length of each arm. Forward kinematics are used to calculate the $n^{th}\ x,\ y$ coordinate:

$$\begin{bmatrix} x_{\mathrm{n}} \\ y_{\mathrm{n}} \end{bmatrix} = \begin{bmatrix} r_2 cos(\theta_{1\mathrm{n}} + \theta_{2\mathrm{n}}) + r_1 cos(\theta_{1\mathrm{n}}) \\ r_2 sin(\theta_{1\mathrm{n}} + \theta_{2\mathrm{n}}) + r_1 sin(\theta_{1\mathrm{n}}) \end{bmatrix}$$

The difference $\triangle = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$ between $x_{target},\ y_{target}$ and $x_{\mathrm{n}}, y_{\mathrm{n}}$ of the end effector are calculated:

$$\triangle = \begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} x_{target} \\ y_{target} \end{bmatrix} - \begin{bmatrix} x_{\mathrm{n}} \\ y_{\mathrm{n}} \end{bmatrix}$$

The difference $\triangle$ is clamped down if its length is greater than a clamping distance d_{max}:

$$\begin{bmatrix} d_x \\ d_y \end{bmatrix} = \triangle = \begin{cases} \triangle, & \text{if } || \triangle || < d_{max} \\ \dfrac{\triangle}{|| \triangle ||} d_{max}, & if\ || \triangle || \geq d_{max} \end{cases}$$

Changes in joint angles leading to such $\triangle$ are calculated using a damping factor k to avoid singularity.

$$\begin{bmatrix}\dot{\theta_1}\\ \dot{\theta_2}\end{bmatrix} = J^T(JJ^T + k^2I)^{-1} \cdot \begin{bmatrix}d_x\\ d_y\end{bmatrix}$$

The $(n+1)^{th}$ joint angles are then updated:

$$\begin{bmatrix}\theta_{1_{n+1}}\\ \theta_{2_{n+1}}\end{bmatrix} = \begin{bmatrix}\theta_{1_n}\\ \theta_{2_n}\end{bmatrix} + \begin{bmatrix}\dot{\theta_1}\\ \dot{\theta_2}\end{bmatrix}$$

$\theta_{1_{n+1}}\ and\ \theta_{2_{n+1}}$ are updated until $|| \triangle || \leq 0.001$.

2.3.2.3. Using Jacobian matrix to control the system via velocity control

In another approach, the robot was moved to target position by controlling its velocity. Similar steps as differential method were performed. However, the $\dot{\theta_1}\ and\ \dot{\theta_2}$ were input directly to Arduino for velocity control, instead of $\theta_1\ and\ \theta_2$.

2.3.3. Manipulability and manipulability ellipsoid

Manipulability μ of current positions are calculated using the Jacobian matrix J:

$$\mu = \sqrt{det(JJ^T)}$$

Manipulability ellipsoids of current positions of end effector are calculated as:

$$U\Sigma V^T = svd(JJ^T)$$

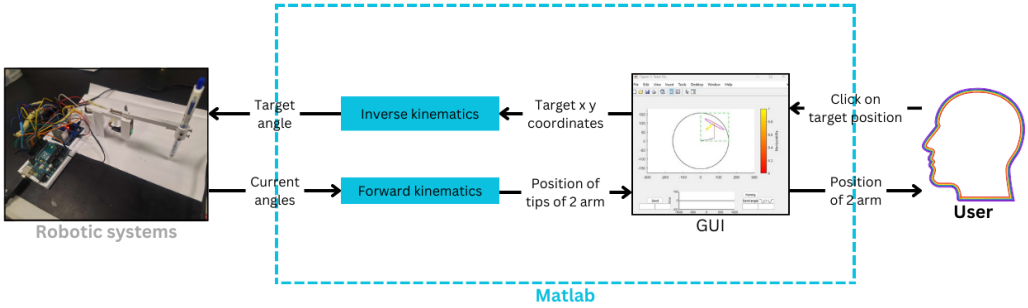
Where the direction and magnitude of ellipse axis are derived from U and Σ, respectively.

2.4. High-level control

2.4.1. Communication

For every loop in Arduino, the current angle of two motors are read and sent to MATLAB. MATLAB uses forward kinematics to calculate the positions of two arms and plot on graphical user interface (GUI). The obtained x and y coordinates from user are used to calculate the angles needed for the arms to be at desired position (target angles). Arduino board uses PID control rotating the motor shaft to those angles.

Figure 3. Communication between MATLAB and Arduino



Function *write(device,data,datatype)* is used to send a string structured as "C$a1, a2$;" to Arduino ($a1, a2$ is the target angles of motor 1 and 2). On the other side, Arduino print a string structured as "c:a1-a2-e1-e2-t" to the serial buffer (a1 and a2 are the current angle 1 and 2, e1 and e2 are errors between angle 1 and 2 with the target angles, t is the current time in Arduino). MATLAB read the string using *configureCallback(device,"terminator",callbackFcn)* function.

2.4.2. Graphical user interface (GUI)

The GUI are designed with the following criteria:

- There are intuitive methods for inputting target coordinates.
- Arms' positions can be easily visualized in real time.
- The errors (in joints space) are visualized in real time.
- The homing routine only starts if the user controls it from MATLAB.
- The code can be reset without closing the GUI.
- Closing the GUI shut down and reset all system and codes.

2.4.3. Trajectory Generation

As our system controls one variable, we only tested path generation for a circle. The coordinate of n points on the circle with a radius r are generated as:

$$\left(x_{center} + r \cdot cos\left(\frac{2i\pi}{n}\right),\ y_{center} + r \cdot sin\left(\frac{2i\pi}{n}\right)\right), \mathrm{i} \in \{0, 1, 2 \ldots, \mathrm{n} - 1\}$$

2.5. Risk assessment

Failure Mode and Effects Analysis Model (FMEA) were used to assess the risk of the system (24).

2.6. Validation and verification of experiments

The systems were tested by moving the end effector to a single position and a series of positions. Homing and object detection routines were also implemented.

3. Results

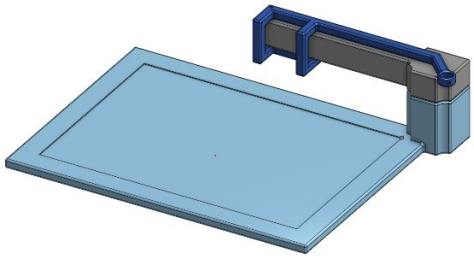
3.1. Computer-Aided Design, 3D printing and hardware integration

3.1.1. Computer-Aided Design

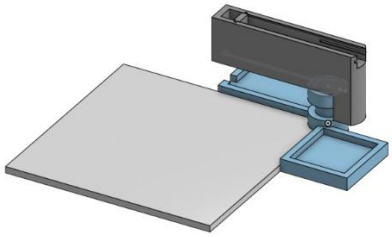
We have gone through 6 iterations of CAD designs (Figure 4). Problems with each version and solution could be found in Table 1.

Figure 4. Different versions of CAD designs

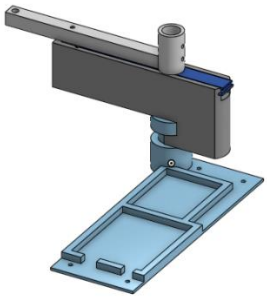
Version 1



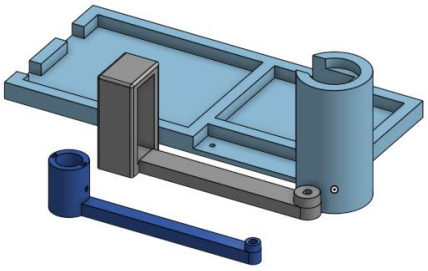
Version 2



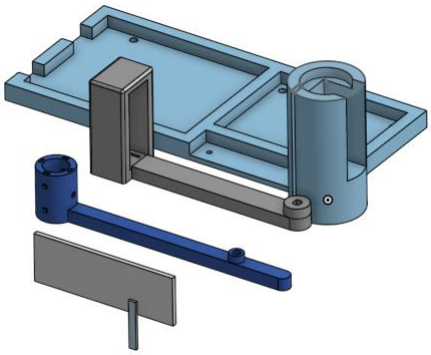
Version 3



Version 4



Version 5



Version 6

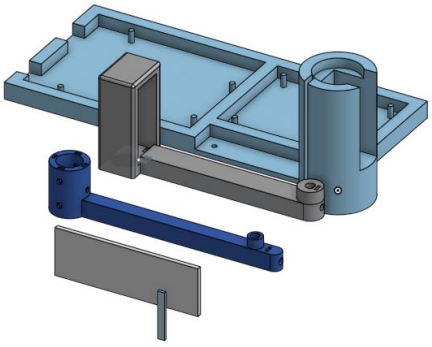


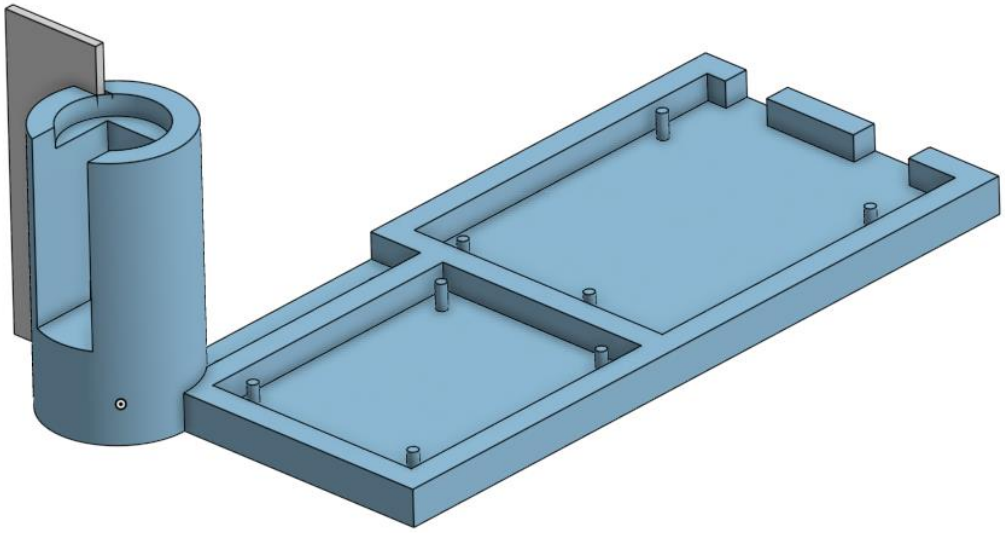
Table 1: Summary of changes of different designs iterations

Iteration	Updates	Reason
1	Two degree of freedom robots comprising three parts: the base, arm 1, and arm 2 with a pen holder mechanism	
2	Adding holders for motors, routes for wiring in arm 1	Adding spaces to put motors and wiring
	Adding holders for Arduino board and H-bridge in the base	Adding spaces to put Arduino board and H-bridge
3	Removing the drawing table to put paper on	Redundancy. Reducing 3D printing materials needed
4	Reducing the size and redesigning two arms	Reducing 3D printing materials needed
	Moving the motor 1 holder from arm 1 to the base	No place for keeping 2 motors in a small arm-1 version
	Adding mechanism for connecting the part using bearings	Reducing friction, supporting the shaft, and distributing the load thanks to the bearings
5	Adding 4 more screw holes to the side of pencil holders	Limiting the lateral rotation of pens
	Adding screw holes in the base to fix the Arduino board and H-bridge	Preventing slippage of the items
	Adding removable walls for homing routine	Needs for obstacle when running the homing routine
	Relocating the motor 1 holder entrance from behind the axis to the front	Preventing motor from falling out due to the torque of the arms
6	Add screw hole for fixing the motor shafts	Adding a screw help fixing the motor shaft when shaft holes suffer wear and tear over time
	Changing screw holes in base to poles for fixing the Arduino and H-bridge	Screwing the items down is unnecessary

3.1.1.1. The base

The base comprises of two main parts: the axis (where motor 1 is held) and the electrical items holder. There is a groove for a removable wall on the side of the axis. A mechanism for grasping the bearing from the outside was added at the top of the axis (Figure 5).

Figure 5. CAD design of the base

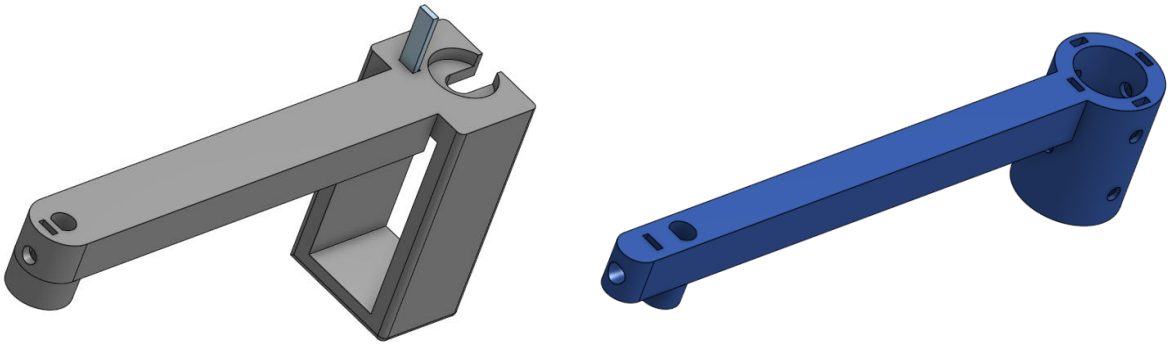


3.1.1.2. Arm 1 and arm 2

At the beginning of arm 1, a small cylinder protrudes out to keep the bearing from the insides. The motor shaft will thread through the cylinder. Motor 2 holder is designed at the end of arm 1 with a small groove for another removable wall. There is a similar mechanism for connecting arm 1 and arm 2 through a second bearing. The pen holder is placed at the end of arm 2 with 7 screw holes (2 on the left, 2 on the right, 2 on the front, and 1 on the back). The skew holes near the motor shafts help tighten the motors when they are loose (Figure 6).

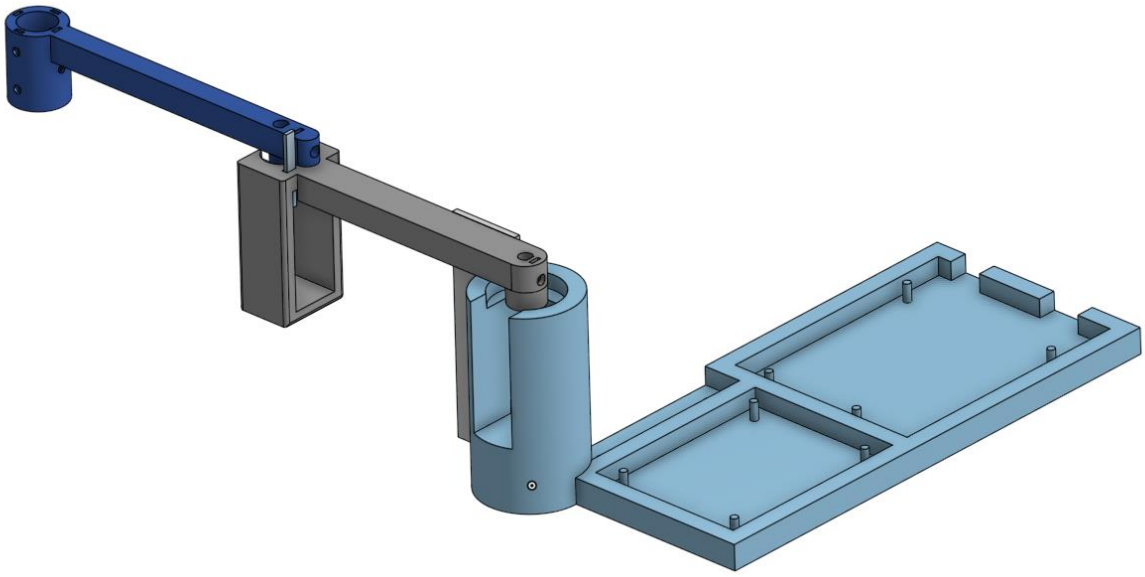
Figure 6. CAD design of two arm 1 (a) and arm 2 (b)

a) b)



The final assembly is presented in figure 7.

Figure 7. CAD assembly



3.1.2. 3D printing

Smaller test pieces were printed to find the best measurements for CAD designs. Differences up to 0.5mm between Flsun QQ-S and Flsun Super Racer and up to 0.2mm between different Flsun QQ-S were found when using similar setting. Consequently, the final printings were performed using only one Flsun QQ-S machine with PLA+ filaments. Infill percentage of 0%, 10%, and 20% were tested. As some samples with 0% is broken, the final devices used a 10% infill (Figure 8).

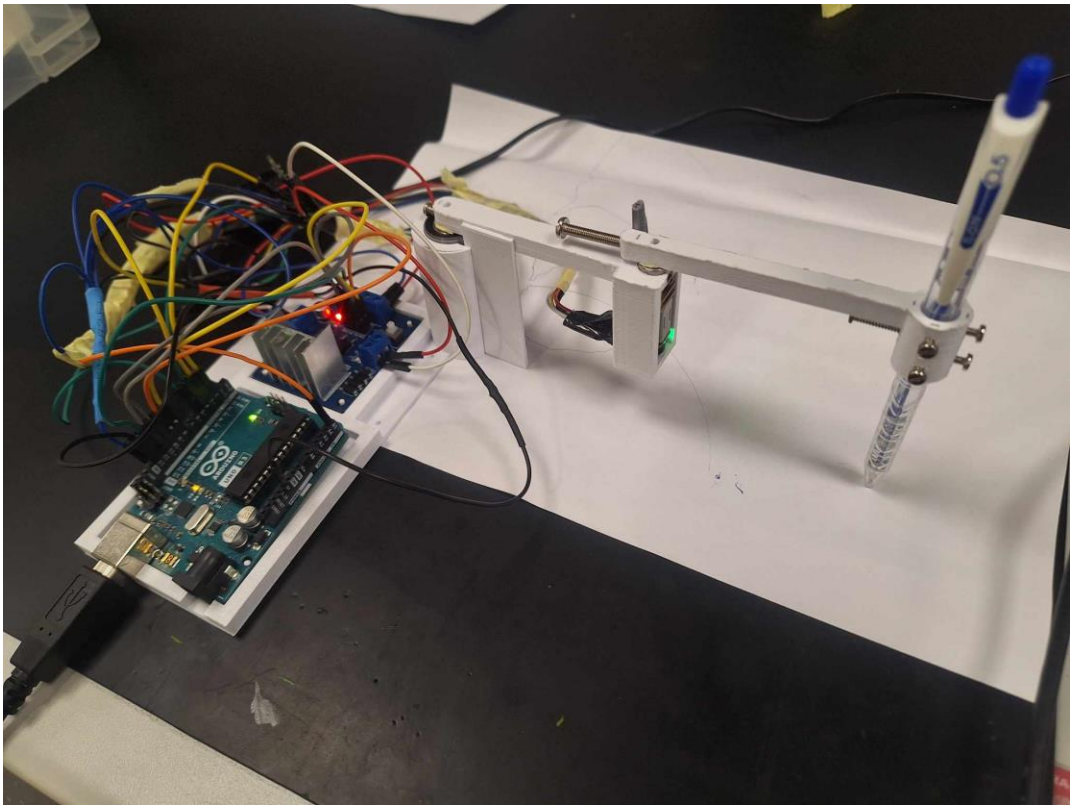
Figure 8. Small test pieces (a) and prototypes (b)

a) b)

3.1.3. Hardware Integration

The breadboard was removed after testing, all wires were soldered to secure electrical connections. Longer wires were used to increase flexibility for motor 2 movement. Integration of the system is presented in figure 9.

Figure 9. Hardware integration of the system.



3.2. Low-level control

3.2.1. Homing routine

The sampling time is 70ms, corresponding to a sampling rate of 14 hertz. A current sensor was added together with a low-pass filter using a 40-kiloohm resistor and a 2-microfarad capacitor attenuating signal component with frequency higher than 2 hertz. The details are presented in the experiment section.

$$2 \approx \frac{1}{2\pi \cdot 40000 \cdot 0.000002}$$

3.2.2. PID control

The models return an R^2 of 94.16% for training data of motor 1, and 96.04% for motor 2. However, when validating the model at different PWM values, the models exhibited significantly lower performance. For motor 1, the model achieved R^2 values of 64.12% (PWM=40) and

39.68% (PWM=30). For motor 2, the validation R^2 values were 78.57% (PWM=35) and 49.29% (PWM=25). However, the obtained models were still used for simulation.

Additionally, although the maximum PWM value is 255, at PWM values of 35 for motor 1 and 30 for motor 2, the maximum speed were $260°/second\ and\ 190°/second$, respectively. To further increase the stability the maximum PWM value are limited to 35 for motor 1 and 30 for motor 2 after the PID control.

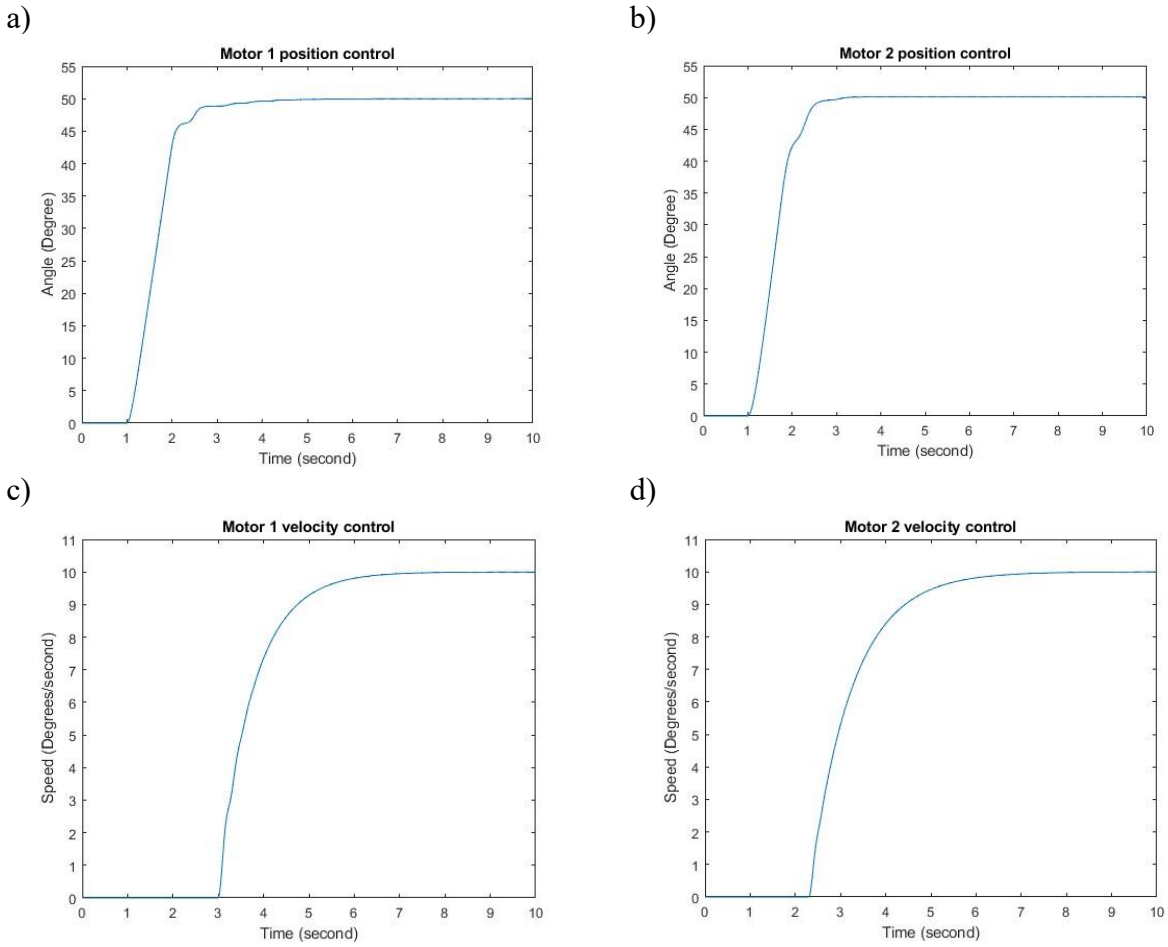
An N=10 is used as cutoff value for derivative filters in both simulated and real systems. Although PID value for position control system do not differ much between simulation and real system, the PID value for velocity control system underwent a lot of changes (Table 2).

Table 2: Summary of PID values tested

Data are presented as (Kp, Ki, Kd)		Simulated PID values	Real PID values
Position control system	Motor 1	(5, 5, 0.5)	(5, 5, 0.5)
	Motor 2	(3, 3, 0.5)	(3, 4, 0.5)
Velocity control system	Motor 1	(0.5, 1, 0)	(0.5, 0.3, 0)
	Motor 2	(0.7, 1, 0)	(0.3, 0.5, 0)

The simulated signals are presented in figure 10.

Figure 10. Simulated signals of motor 1 (a) and motor 2 (b) position control in Simulink. Another velocity control system was also tested (c), (d).



3.2.3. Object detection

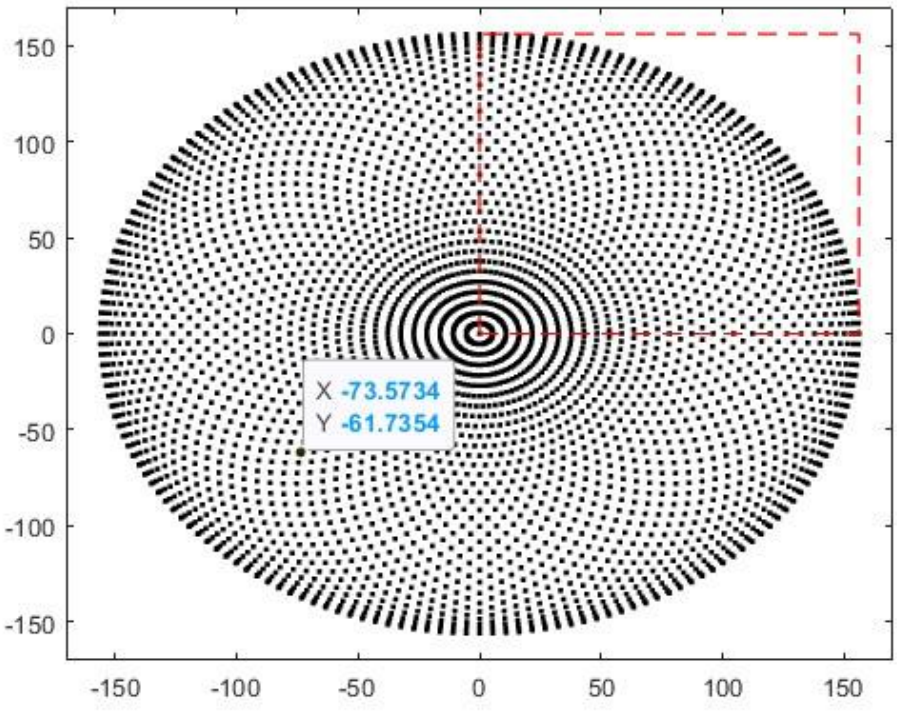
Object detections were successfully incorporated. More details could be found in the experiment section.

3.3. Mathematical formula

3.3.1. Forward kinematics

To identify the workspace of robot, forward kinematics were used to calculate and plot the position of the end effector for each angle combinations of arm 1 and arm 2 within $(-180°, 180°)$ with a step of 2°. Theoretically, the arms can reach any point within a circle with a radius of 156mm with a center coincide with the axis of rotation of motor 1. Our workspace is limited to the right upper half of the circle (Figure 11).

Figure 11. Identify the workspace of robot



3.3.2. Inverse kinematics

3.3.2.1. Analytical inverse kinematics

Analytical inverse kinematics were successfully implemented.

3.3.2.2. Differential inverse kinematics

A damping factor $k = 1$ and a clamping distance $d_{max} = 20mm$ were used. Inverse kinematics were successfully approximated with an error less than 0.002 compared to analytical inverse kinematics. However, the differential approach ran much slower than the analytical one (Table 3).

Table 3: Comparison between two approaches for solving inverse kinematics.

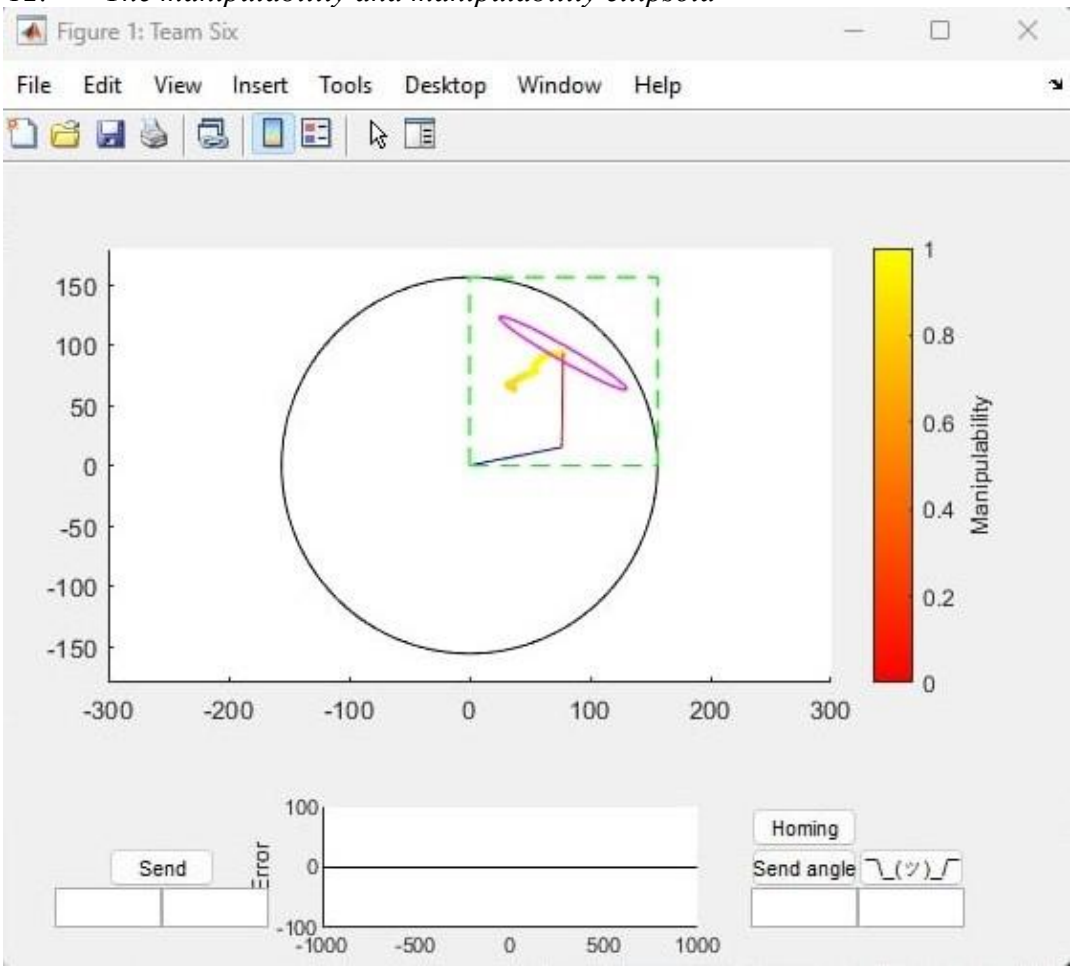
Task space $[x, y]$	Analytical inverse kinematics $[\theta 1, \theta 2]$	Runtime (milliseconds)	Differential inverse kinematics $[\theta 1, \theta 2]$	Runtime (milliseconds)
[151, 5]	[-12.5274, 28.8479]	0.11200	[-12.5275, 28.848]	0.41850
[36, 125]	[40.43, 67.0072]	0.01820	[40.4296, 67.0083]	0.15460
[76, 15]	[-49.0613, 120.4523]	0.01120	[-49.0615, 120.4525]	0.09210
[97, 41]	[-24.6285, 95.0827]	0.01540	[-24.6286, 95.0835]	0.08570
[62, 106]	[21.5995, 76.1538	0.02110	[21.5994, 76.1543]	0.086

**Five random [x, y] values were generated to compare two approaches.*

3.3.3. Manipulability and manipulability ellipsoid

Manipulability and manipulability ellipsoid were successfully integrated to GUI. The manipulability were normalized to a range of $(0,1)$ before plotting ($\overline{\mu} = \frac{\mu}{78 \cdot 78}$).

Figure 12. The manipulability and manipulability ellipsoid



The manipulability is lowest when the end effector lies on the circle, where the ellipsoid become a straight line tangent to the circle. This occurs because θ_2 become 0, and the Jacobian matrix lose one rank:

$$J = \begin{bmatrix} -156 \cdot sin(\theta_1) & -78 \cdot sin(\theta_1) \\ +156 \cdot cos(\theta_1) & +78 \cdot cos(\theta_1) \end{bmatrix}$$

3.4. High-level control

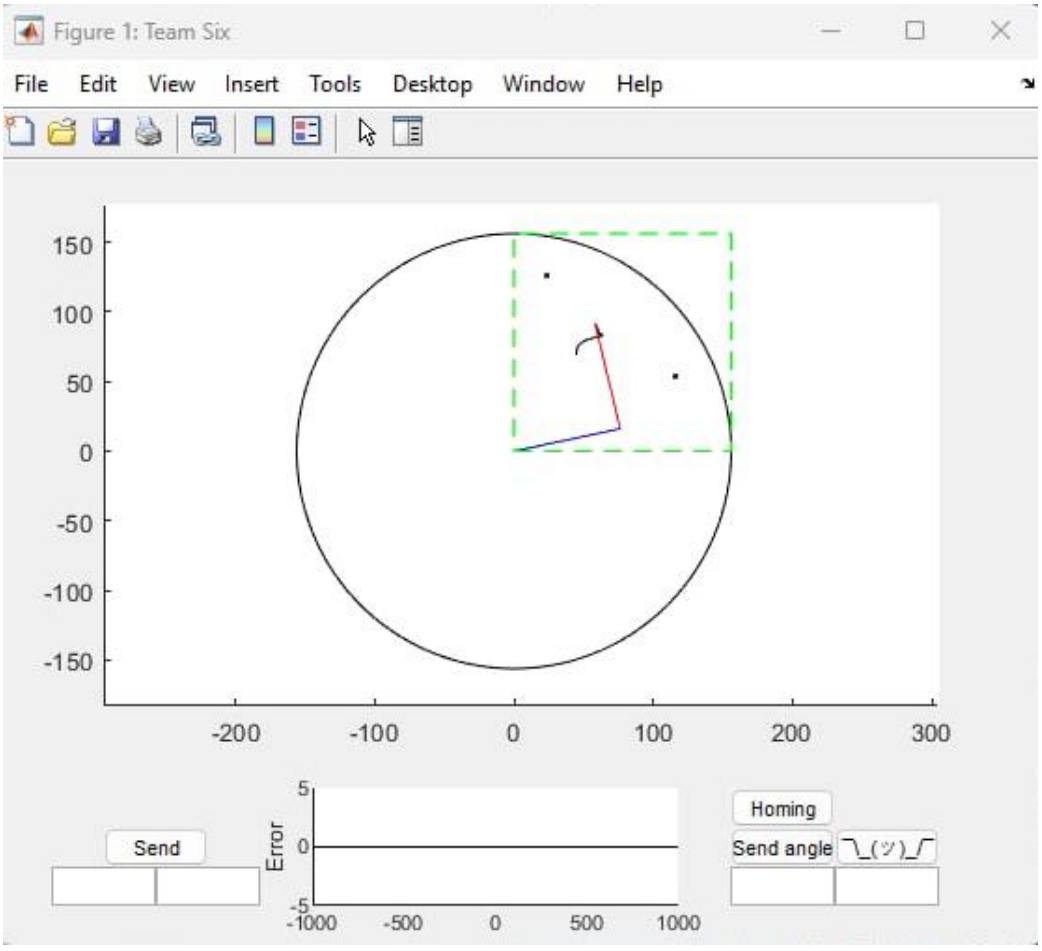
3.4.1. Communication

The target angles could be sent to Arduino multiple times in one run. The code *t = timer(Name,Value)* was first used for reading the data from Arduino; however, it was difficult for real time plotting. Later *configureCallback(device,"terminator",callbackFcn)* was used to read the Arduino serial buffer whenever a terminator appeared, which resulted in real time plotting in MATLAB. Visual inspection showed a one-to-one correspondence between the physical arms and GUI plotting.

3.4.2. Graphical user interface

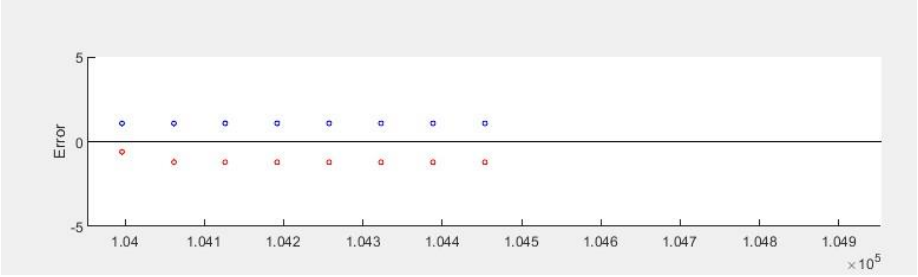
GUI were designed with real-time plotting of two arms represented as two different colored segments. User can instruct the robot to move to a position by clicking a point within the robot workspace or entering the coordinate manually. The homing routine only starts after use click the "Homing" button on the lower right. To reset the code, the user first click the reset button on Arduino board and button with a smile face on the lower right of MATLAB GUI. Closing the GUI leads to shut down and reset all codes (Figure 13).

Figure 13. GUI



The graph for plotting the error could be turned on or off (Figure 14).

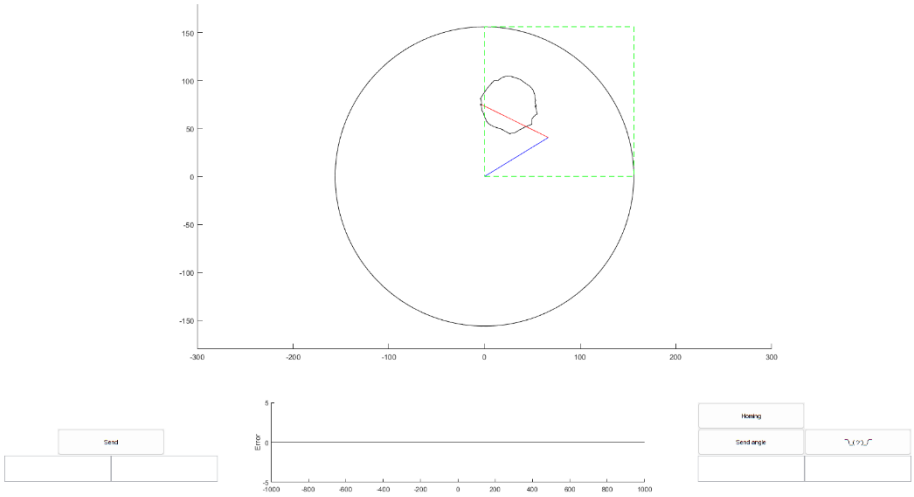
Figure 14. Error plot



3.4.3. Trajectory Generation

Pathway for a circle with a radius of 30mm were generated at 32 points and input to the system at once. The result drawing the experiments was not smooth as robot's motions show lots of jerking (Figure 15).

Figure 15. Drawing a circle



3.5. Risk assessment

The FMEA was performed across three areas: hardware and designs, Arduino control, and MATLAB control. The hazardous situation details and their recommended actions and mitigations are given in table 4.

Table 4: Hazardous situations and recommendations

ID	Item / Function	Potential Failure Mode	Potential Cause(s)/Mechanism(s) of Failure	Hazardous Situation	Recommended Action / mitigation	Notes
H01	Hardware	Wire got disconnected	Transporting equipment	Delaying the work	Replacing/ soldering the hardware again	
H02	Design	Pen falls off	Mechanical failure	Unintended pen trace on paper	Tightening the pen holder screws	
H03	Homing routine	Arms cannot get to starting position after homing routine	Wear and tear of systems leading to changes in dynamics	Incorrect starting position	Resetting the system. Changing the homing routine cutoff	
H04	Low-level control	There are lots of jerking in drawing	Poor control system	Serrulate drawing	Constructing a different control system where position, velocity, and acceleration are controlled at the same time	The current system could provide simple drawing as instructed; however, it is not precise enough for practical use.
H06	Object detection	Nothing is draw, uneven drawing	Higher load on pens than object detection cut-off	No movement	Changing the object detection cutoff	
H07	High-level control	No signal is received when commanded	Connection problems	Delaying the work	Checking connection between computer and Arduino board	

H08	High-level control	Performance is slowed down	Computational power problem	Delaying the work, stop the work in the middle	Reducing the input frequency, resetting the machine	

The occurrence and severity of each situation were estimated (Table 5). Although the criticality is "Low" for nearly all situations. The consequences of jerking due to incomplete control system is high, which make it impractical to use. To solve this problem a control system where position, velocity, and acceleration are controlled was suggested.

Table 5: Scoring the severities for each situation.

ID	Occurrence (P1)	Severity	Occurrence (P2)	Score	Criticality
H01	Remote	Minor	Frequent	4	Low
H02	Remote	Serious	Occasional	6	Low
H03	Occasional	Minor	Frequent	6	Low
H04	Frequent	Catastrophic	Frequent	25	High
H06	Remote	Serious	Frequent	6	Low
H07	Remote	Minor	Frequent	4	Low
H08	Occasional	Minor	Frequent	6	Low

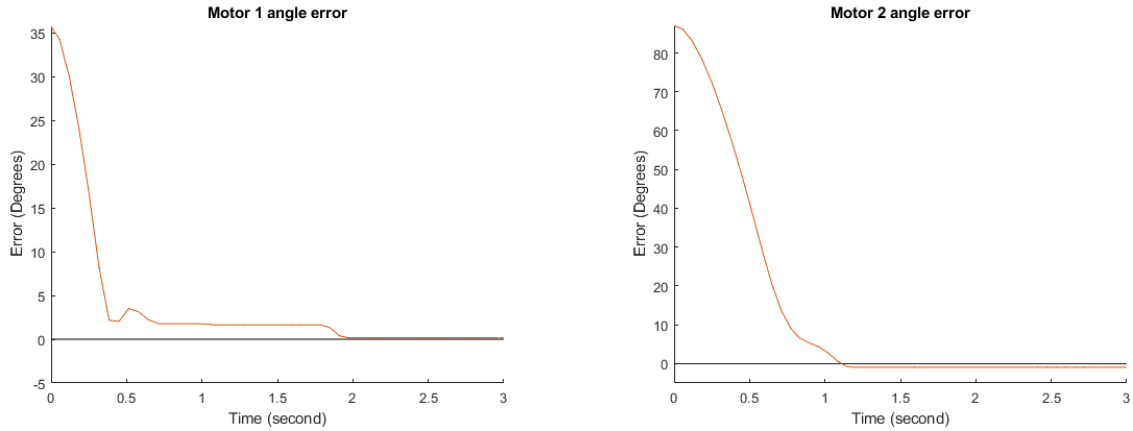
3.6. Validation and verification of experiments

3.6.1. Move to a specific point.

The end effector can move to the desired positions within 3 seconds with less than 0.5° error (1.5mm maximal error in task space), and a maximum 2° overshoot every time.

Figure 16. Errors of motor 1 (a) and motor 2 (b) angles with reference to target angles

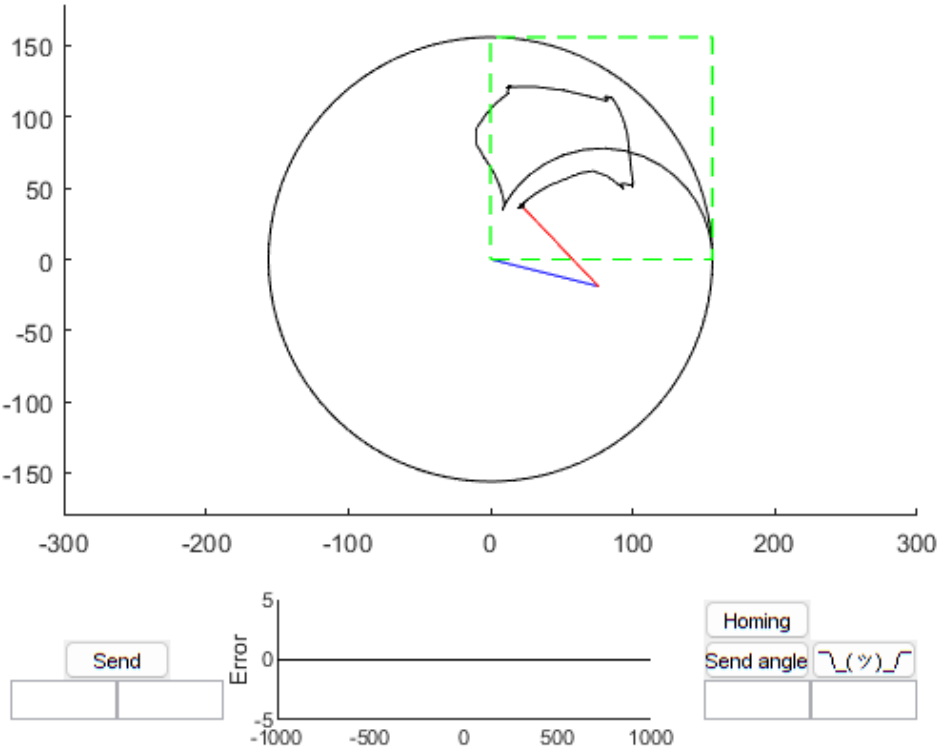
a) b)



3.6.2. Move through a set of points

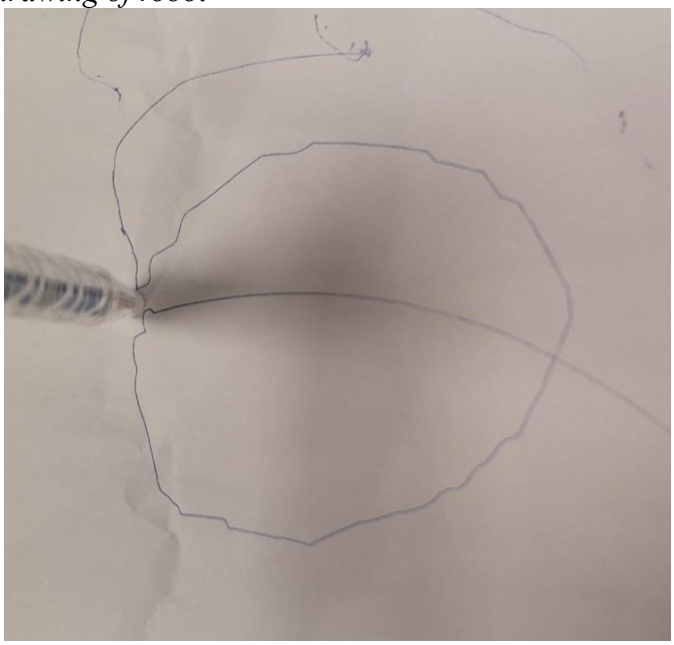
The robot can be instructed to move through a serial of points (Figure 17).

Figure 17. Moving the robot through 4 points



In pathway generation experiments, the movement of robot is serrulate (Figure 18)

Figure 18. Circle drawing of robot



3.6.3. Homing routine and obstacle detection

The motors were instructed to move using a constant PWM value of 30. It could get to homing position in nearly all of the time. However, the cutoffs for the currents were updated twice during the experiment, as the old cutoff became invalid due to dissembling and assembling the device. Raw analog signals instead of the processed current values were used for finding the cutoff. The cutoff value for motor 1 was 5.5, and motor 2 was 6.5 (Figure 19).

Figure 19. Cut off value for current sensor

```
// Cut off value for homing
volatile float motor_2_homing_limit = 5.5;
volatile float motor_1_homing_limit = 6.5;
```

For obstacle detection function, if PWM values are either higher than 25 for motor 1 or 20 for motor 2 while the speed was less than $5°/second$ for the corresponding motor over 7 iterations, both motors will stop. Our sampling time was 70ms, therefore, the motors will stop after $0.070 \cdot 7 \approx 0.5$ second after hitting and obstacle. In the experiments, the robot stop for 3 seconds and move again in all trials.

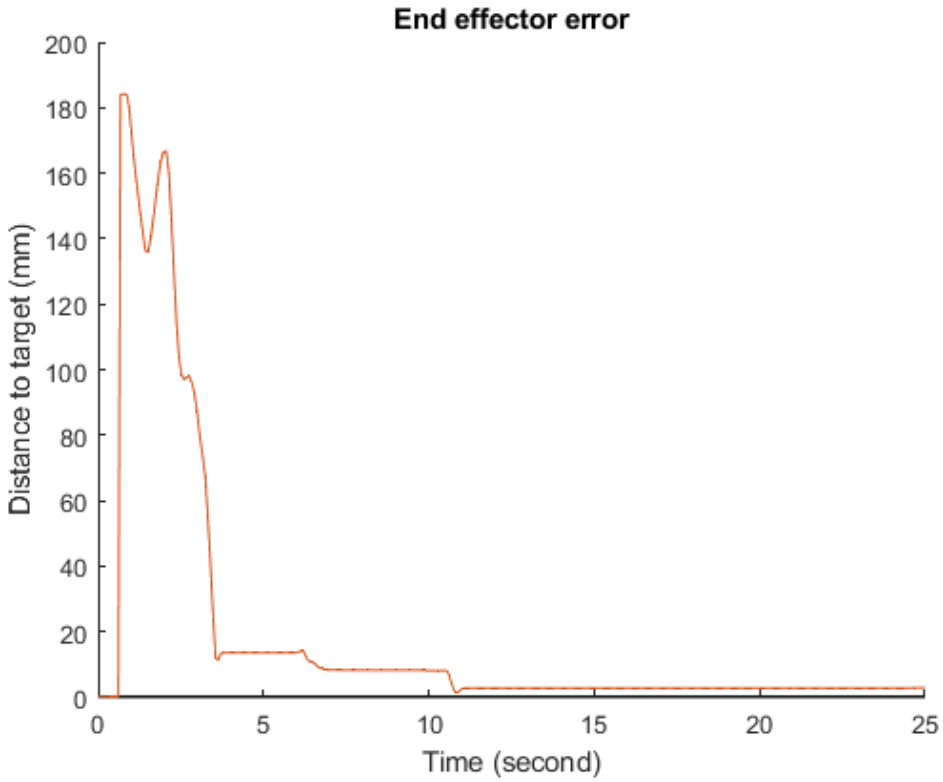
3.6.4. Solving the inverse kinematics using differential approach

Similar results when used for controlling the robot is found compared to analytical inverse kinematics.

3.6.5. Using Jacobian matrix to control the velocity of the motors

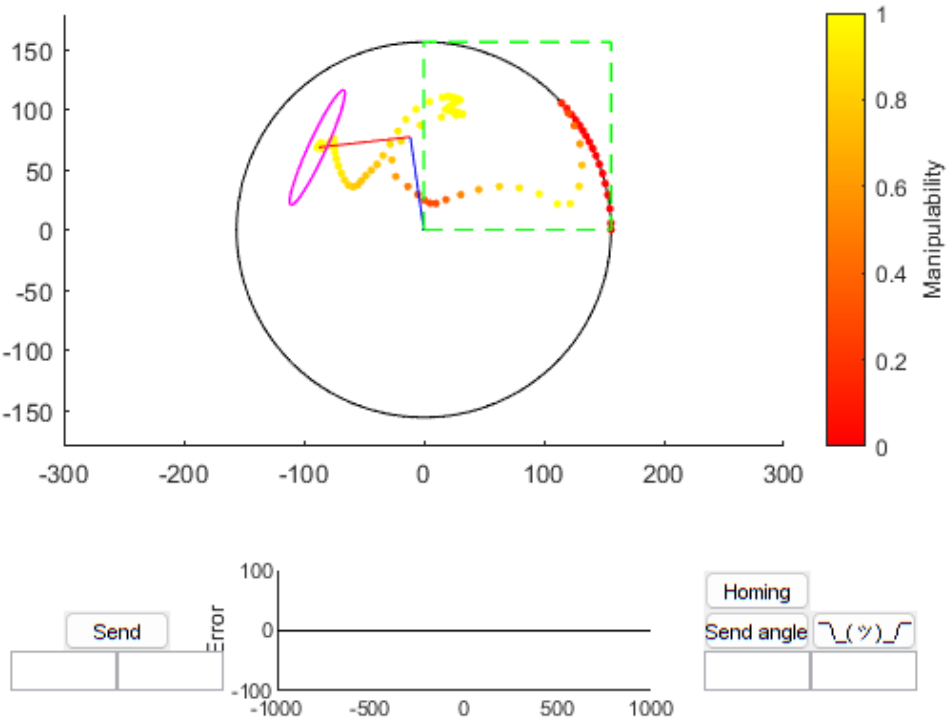
The Jacobian matrix is used to calculate the velocities in joint space. Arduino control the motors to achieve those velocities until reaching the target. This approach successfully moved the end effector to the target positions as near as 3mm, which is twice the error of the position control system (Figure 20).

Figure 20. Error when controlling the robot through the speed.



The pathways of end effector are curly as expected (Figure 21).

Figure 21. The pathway of robot when controlling its speed.



4. Discussion

4.1. CAD/3DP and hardware integration

The project successfully designed a two degree of freedom robot. On consistent problems across the projects was deterioration of shaft holes, leading to reprinting and replacing components. This, in turn, caused delays and unnecessary waste of materials. This problem was eventually solved by add a screw to fix the motor shaft, which could be tighten when wear and tear occurred. 3D printings have lots of potential in medicine, however, using it required a certain level of skills and experience (25, 26). The dimensions of designs was also a big problem, printing a whole system without testing the measurement of components made it impossible for connecting the part. Therefore, to avoid wasting material, smaller test pieces near the connection should be printed first to find the most suitable numbers. Additionally, using only one machine should be attempted for printing the project to avoid variance between machines. For the homing routine, removable walls is used to avoid limiting the ranges of motions for two arms

4.2. Low-level control

PID controls is good enough for controlling the robotics arms. The transfer equations for motor is a second order transfer function (27, 28) In our design, we limited the maximum voltage that can be input to the motor. This, although led to more stable movement, potentially impacted the PID control system as the motor had to work outside its linearity range (29). When fitting the transfer functions back to the motor, the models do not fit well will other voltage input,

indicating that the motors were not in its linear range. However, the experiment showed that, despite this limitation, the PID control still work better than expected. In future designs, non-linear controls system could be tested (30).

The biggest limitation that make our system impractical is jerking when moving through a series of points, which is because the robot stops whenever it reaches a point. This problem could be solved through trajectory generation (31), where both position and velocity are estimated in advance. However, our control system can only control one component at a time, which make it impossible to fully implement this. More advanced control technique can be tested in the future for solving this problem (32).

4.3. High-level control

Both forward kinematics and inverse kinematics were implemented in MATLAB. This emphasized the importance of computational power in building robotics systems. Moreover, inverse kinematics is not always solvable using geometric method, differential approach offer solutions for such complex cases (33), however, it requires high computational power for it to be implemented in real time. Development of hardware and computational power have opened a great opportunity for the development of robotics.

Active constraint is a great concept for controlling a robot system (34). As the human body has many important parts, adding active constraints on a high-level control add another layer of safety for operating on humans. The current computational power allow to do it, adding it to future MATLAB code allow further control of our system

4.4. Experiments

When controlling the position of motor shafts, we successfully moved the robot to the target position with errors less than 0.5 degree in joints spaces and an overshoot less than 2 degree. However, when controlling the velocity, it was harder to get similar precision, this could be explained as the closer the robot gets to the target position, the smaller the speed is required. As the speed get lower than a certain value, it is not possible for maintaining such speed constantly due to frictions and the discontinuity of output voltage. Although controlling the position offers a more precise control, moving through different points introduces lots of jerking as discuss above, therefore, speed control should be incorporated in future projects.

Homing and object detection functions worked well in our systems. However, updating the cutoff value over time requires some systematic strategy. Future projects could consider saving the current working data in a separate file and updating the parameters depending on the data. The parameters can be later uploaded from a high-level system to a low-level system at the beginning of the code.

4.5. Limitation

The biggest limitation of our system is the single variable control system of the robot, which was discussed above. Another limitation of our system is the materials used. This led to reprinting and error due to deterioration of the materials. After having a good control system, future projects should consider what materials for constructing a robot as a following step.

5. Conclusions and future work

In summary, we successfully designed two degrees of freedom drawing robot, a PID control system with precise movement and little overshoot, and an intuitive GUI. Our limitations include a single variable control system and unoptimized materials. Future projects are suggested to explore more advanced control systems and test different materials for the final products. Techniques in high-level control systems including active constraints or storing data and updating parameters over time could be tested.

6. Contribution

Ideas and designs are discussed by all team members. Dang The Hung (DTH) was responsible for the operation of the team at all steps. CAD design was mainly responsible by Huiluomin Shen (HS). Electrical assembly was mainly responsible by Mustafa Kafaji (MK). Arduino coding was mainly responsible by Yiqi Ma (YM). MATLAB coding was mainly responsible by DTH.

References

1. Motoc A, Motoc M, Bolintineanu S, Mușuroi C, Munteanu M. The construction of human body--from model to reality. Rom J Morphol Embryol. 2005;46(1):63-6.

2. Bagenal J. Health-care students: committed to improving health but frustrated. Lancet.

3. Džakula A, Relić D. Health workforce shortage - doing the right things or doing things right? Croat Med J. 2022;63(2):107-9.

4. Gyles C. Robots in medicine. Can Vet J. 2019;60(8):819-20.

5. Raje S, Reddy N, Jerbi H, Randhawa P, Tsaramirsis G, Shrivas NV, et al. Applications of Healthcare Robots in Combating the COVID-19 Pandemic. Appl Bionics Biomech. 2021;2021:7099510.

6. Kanakis GS, Rovithakis GA, editors. Improving Safety in Human-Robot Collaboration via Dynamic Active Constraints Enforcement. 2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN); 2021 8-12 Aug. 2021.

7. Kaan HL, Ho KY. Robot-Assisted Endoscopic Resection: Current Status and Future Directions. Gut and liver. 2020;14(2):150-2.

8. Cepolina F, Razzoli RP. An introductory review of robotically assisted surgical systems. Int J Med Robot. 2022;18(4):e2409.

9. George EI, Brand TC, LaPorta A, Marescaux J, Satava RM. Origins of Robotic Surgery: From Skepticism to Standard of Care. Jsls. 2018;22(4).

10. Industrial robotics: theory, modelling and control. Cubero S, editor. Mammendorf, Germany: Pro Literatur Verlag; 2006. 964 p.

11. Ngu JC, Tsang CB, Koh DC. The da Vinci Xi: a review of its capabilities, versatility, and potential role in robotic colorectal surgery. Robot Surg. 2017;4:77-85.

12. Kurup G. CyberKnife: A new paradigm in radiotherapy. J Med Phys. 2010;35(2):63-4.

13. Roche M. The MAKO robotic-arm knee arthroplasty system. Arch Orthop Trauma Surg. 2021;141(12):2043-7.

14. Onshape®. Onshape® 2023 [Available from: https://www.onshape.com/en/.

15. UltiMaker. UltiMaker Cura 2023 [Available from: https://ultimaker.com/software/ultimaker-cura/.

16. Zhengzhou Chaokuo Electronic Technology Co. L. FLSUN 2023 [Available from: https://flsun3d.com/.

17. NFP-Motor. 3V, 6V, 12V DC Micro Metal Gearmotor Model NFP-JGA12-N20 2023 [Available from: https://nfpmotor.com/3v-6v-12v-dc-micro-metal-gearmotor-model-nfp-jga12-n20.

18. 4tronix. L298N Dual H-Bridge Motor Driver Module 2023 [Available from: https://shop.4tronix.co.uk/products/l298n-dual-h-bridge-motor-driver-module?_pos=1&_sid=959f443a7&_ss=r.

19. Por E, van Kooten M, Sarkovic V. Nyquist–Shannon sampling theorem. Leiden University. 2019;1(1).

20. Arduino. Arduino Uno Rev3 2023 [Available from: https://store.arduino.cc/products/arduino-uno-rev3.

21. Stoffregen P. Encoder 1.4.4: Arduino; [cited 2023. Available from: https://www.arduino.cc/reference/en/libraries/encoder/.

22. Johnson MA, Moradi MH. PID control: Springer; 2005.

23. Documentation S. Simulation and Model-Based Design: MathWorks; 2020 [Available from: https://www.mathworks.com/products/simulink.html.

24. Dhillon BS. Failure modes and effects analysis — Bibliography. Microelectronics Reliability. 1992;32(5):719-31.

25. Hegedus T, Kreuter P, Kismarczi-Antalffy AA, Demeter T, Banyai D, Vegh A, et al. User Experience and Sustainability of 3D Printing in Dentistry. International Journal of Environmental Research and Public Health. 2022;19(4):1921.

26. Kusaka M, Sugimoto M, Fukami N, Sasaki H, Takenaka M, Anraku T, et al. Initial Experience With a Tailor-made Simulation and Navigation Program Using a 3-D Printer Model of Kidney Transplantation Surgery. Transplantation proceedings. 2015;47(3):596-9.

27. Tutunji TA, editor DC Motor Identification using Impulse Response Data. EUROCON 2005 - The International Conference on "Computer as a Tool"; 2005 21-24 Nov. 2005.

28. Lipo TA, Plunkett AB. A Novel Approach to Induction Motor Transfer Functions. IEEE Transactions on Power Apparatus and Systems. 1974;PAS-93(5):1410-8.

29. Li Z. Review of PID control design and tuning methods. Journal of Physics: Conference Series. 2023;2649(1):012009.

30. Taylor DG. Nonlinear control of electric machines: an overview. IEEE Control Systems Magazine. 1994;14(6):41-51.

31. Ata AA. Optimal trajectory planning of manipulators: a review. Journal of Engineering Science and technology. 2007;2(1):32-54.

32. Mutha PK, Sainburg RL. Control of velocity and position in single joint movements. Human Movement Science. 2007;26(6):808-23.

33. Rodriguez E, Saha BN, Romero-Hdz J, Ortega D. A multiobjective differential evolution algorithm for robot inverse kinematics. SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE). 2016;3(5).

34. Bowyer SA, Davies BL, Baena FRy. Active Constraints/Virtual Fixtures: A Survey. IEEE Transactions on Robotics. 2014;30(1):138-57.

Supplementary materials

Supplementary pseudocode 1: Homing routine

```
FUNCTION setup()
    Wait until users input something to serial input buffer.
    WHILE current is less than cutoff 2
        Rotate motor 2 clockwise using a constant voltage.
    END WHILE
    Stop rotating motor 2
    WHILE current is less than cutoff 1
        Rotate motor 1 clockwise using a constant voltage.
    END WHILE
    Stop rotating motor 1
END setup()

FUNCTION loop()
    Main code
END setup()
```

Supplementary pseudocode 2: Object detection

```
Declare Object_detected = 0
FUNCTION loop()
    IF Object_detected == 0
        Main code
        IF (Voltage is higher than a cutoff) and (Velocity is smaller than a cutoff)
            Increase Number_of_stop_loops by 1
            IF Number_of_stop_loops is greater than a certain number
                Set Object_detected = 1
            END IF
        ELSE
            Set Number_of_stop_loops = 0
        END IF
    ELSE
        Stop 2 motors
        Wait 3 seconds
        Set Object_detected = 0
    END IF
END loop()
```