UNIVERSITY OF TECHNOLOGY SYDNEY

ASSIGNMENT COVERSHEET

UTS: ENGINEERING & INFORMATION TECHNOLOGY

SUBJECT NUMBER & NAME	NAME OF STUDENT(s) (PRINT CLEARLY)	STUDENT ID(s)
42003 – Engineering Graduate Project	Udhaya Kumar Parameswaran	24603437

STUDENT EMAIL	STUDENT CONTACT NUMBER
UdhayaKumar.Parameswaran@student.uts.edu.au	0435545307

NAME OF TUTOR	TUTORIAL GROUP	DUE DATE

ASSESSMENT ITEM NUMBER & TITLE
Final Report

☐ I confirm that I have read, understood and followed the guidelines for assignment submission and presentation on page 2 of this cover sheet.
☐ I confirm that I have read, understood and followed the advice in the Subject Outline about assessment requirements.
☐ I understand that if this assignment is submitted after the due date it may incur a penalty for lateness unless I have previously had an extension of time approved and have attached the written confirmation of this extension.

Declaration of originality: The work contained in this assignment, other than that specifically attributed to another source, is that of the author(s) and has not been previously submitted for assessment. I understand that, should this declaration be found to be false, disciplinary action could be taken and penalties imposed in accordance with University policy and rules. In the statement below, I have indicated the extent to which I have collaborated with others, whom I have named.

Statement of collaboration:

Signature of student(s) **Udhaya Kumar Parameswaran** **Date** **26-May-2024**

✂- -

ASSIGNMENT RECEIPT

To be completed by the student if a receipt is required

SUBJECT NUMBER & NAME	NAME OF TUTOR

SIGNATURE OF TUTOR	RECEIVED DATE

Table of Contents

Final Report

Project Title

Control of SCARA robot system using Inverse Kinematics for remote monitoring in digital agriculture.

Research Supervisor

Dr Daniel Franklin, Senior Lecturer, School of Electrical and Data Engineering.

Introduction

Robots have a wide range of applications, from the simplest tasks to the most complex challenges in fields such as medicine, manufacturing, film technology, and beyond. The complexity of these robots can vary greatly depending on the application, encompassing designs such as simple 2-axis robots, Gantry robots, SCARA robots, and sophisticated 6-axis industrial robots. In this project, we employ a PRR (Prismatic-Rotational-Rotational) configured SCARA robot to monitor an agricultural setup remotely. Typically, SCARA robots are configured in an RRP (Rotational-Rotational-Prismatic) arrangement, but our implementation of the PRR configuration maintains similar functionality and system implementation.

Requirements

The requirements set for the project are as follows:

Mechanical-
1. Design and integrate pan motion for the end-effector.
2. Design and integrate tilt motion for the end-effector.
3. Integrate a camera mount as the end-effector.

Electrical-
1. Integration of stepper motors for the Pan and Tilt movements.
2. Design a single driver board to interface between Raspberry Pi 4b and the physical setup.
3. Integrate limit switches for end/ home position detection.

Software-
1. Develop software to control stepper and servo motors via the drivers and Raspberry Pi.
2. Enable control of individual joints using command line inputs.
3. Implement inverse kinematics to control the Tool Centre Point (TCP) precisely.
4. Create a forward kinematics function to determine the end-effector's position.
5. Incorporate passive feedback mechanisms using software variables.
6. Implement a home position feature for consistent passive feedback.

Hardware and Software Used

Hardware-
1. pyBot Robotic Arm- jjrobots.
2. A4988- Stepper motor driver.
3. Raspberry Pi 4B.
4. SG90- Servo motors.
5. LM7805- Linear Voltage Regulator.
6. SPDT micro switch with lever (Limit switch).

Software-

1. Language- Python3
2. Libraries-
    a. RPi
    b. pigpio
    c. pygame
    d. math

Literature Review/ Related Works

2-axis planar robots, commonly used in various industrial and research applications, are fundamental robotic systems that provide a basis for understanding more complex kinematic and inverse kinematic problems. This review synthesises insights from several key sources to provide a comprehensive overview of the kinematics and inverse kinematics of 2-axis planar robots.

Kinematics of 2-Axis Planar Robots

Kinematics is the study of motion without considering the forces that cause it. For 2-axis planar robots, kinematics involves determining the position and orientation of the end-effector based on given joint parameters. The primary resources for understanding these principles include the following:

Daslhub on 2-Link Kinematics- This source explains the kinematic equations governing a 2-link planar robot. The forward kinematics equations are derived using trigonometric functions to relate joint angles to the end-effector's position. The Cartesian coordinates (x, y) of the end-effector are expressed as:

$$x = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2)$$

$$y = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2)$$

Fig. 1 Forward kinematics equation

where L_1 and L_2 are the lengths of the robot's links, and Θ_1 and Θ2 are the joint angles (Daslhub, n.d.).

MIT OpenCourseWare on Robotics- This resource expands on the fundamental principles of kinematics for planar robots, discussing both forward and differential kinematics. It emphasises the importance of Jacobian matrices in relating joint velocities to end-effector velocities, providing a basis for understanding dynamic behaviours and control strategies in robotic systems (MIT OpenCourseWare, 2005).

Robot Academy on Robotic Arms and Forward Kinematics- This lesson explores forward kinematics for robotic arms, including 2-axis planar robots. It emphasises the geometric interpretation of kinematic equations and offers visual aids to enhance understanding. The material demonstrates how changes in joint angles affect the end-effector's position in a 2D plane (Robot Academy, n.d.).

Inverse Kinematics of 2-Axis Planar Robots

Inverse kinematics involves determining the necessary joint parameters to achieve the desired position and orientation of the end-effector. This is often more complex than forward kinematics due to the potential for multiple solutions and the need to handle singularities and joint limits.

Mathematical Analysis of Kinematics- This paper provides a thorough mathematical framework for solving inverse kinematics problems. For 2-axis planar robots, the inverse kinematics can be expressed as:

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2}\right)$$

$$\theta_1 = \arctan 2(y, x) - \arctan 2\left(\frac{L_2 \sin(\theta_2)}{L_1 + L_2 \cos(\theta_2)}\right)$$

Fig.2 Inverse kinematics equation

These equations provide a way to compute joint angles Θ_1 and Θ_2 for a given end-effector position (x, y), considering the robot's geometry (Hawkins, 1999).

Core Kinematics Documentation- This document delves into the practical implementation of inverse kinematics algorithms, highlighting numerical methods and iterative approaches for solving the equations. It discusses challenges such as convergence and computational efficiency, crucial for real-time robotic applications (Chang & Dubey, 1986).

Robot Academy on Inverse Kinematics and Robot Motion- This lesson explores the concepts of inverse kinematics through interactive examples and visualisations. It addresses the multiple solutions problem and demonstrates techniques for selecting the most appropriate solution based on the robot's configuration and task requirements. This resource is beneficial for understanding the practical implications of inverse kinematics in robotic motion planning (Robot Academy, n.d.).

Mechanical Design of Pan-Tilt Mount

The mechanical design of pan-tilt mounts is critical for ensuring stability, smooth movement, and precise control of camera systems. The Mini Pan-Tilt Kit from The Pi Hut was the primary inspiration for the design (The Pi Hut, n.d.).

Methodology

Mechanical

The primary mechanical tasks in this project involved creating a pan-tilt motion mount using SG90 servo motors to support the Raspberry Pi High Quality (HQ) Camera. This design was inspired by the Mini Pan-Tilt Kit from The Pi Hut, as referenced in the related works section.

The setup includes a slot for an SG90 servo motor on the second link of the planar section (arm 3), serving as the contact point between the robot arm and the mount. The first servo motor provides pan motion, while the second handles tilt motion. Both motors are secured with screws, and the horns are supplied with the servos. The housing for the tilt motor is designed in two parts for easier installation. The 3D models of the designed components are presented below.

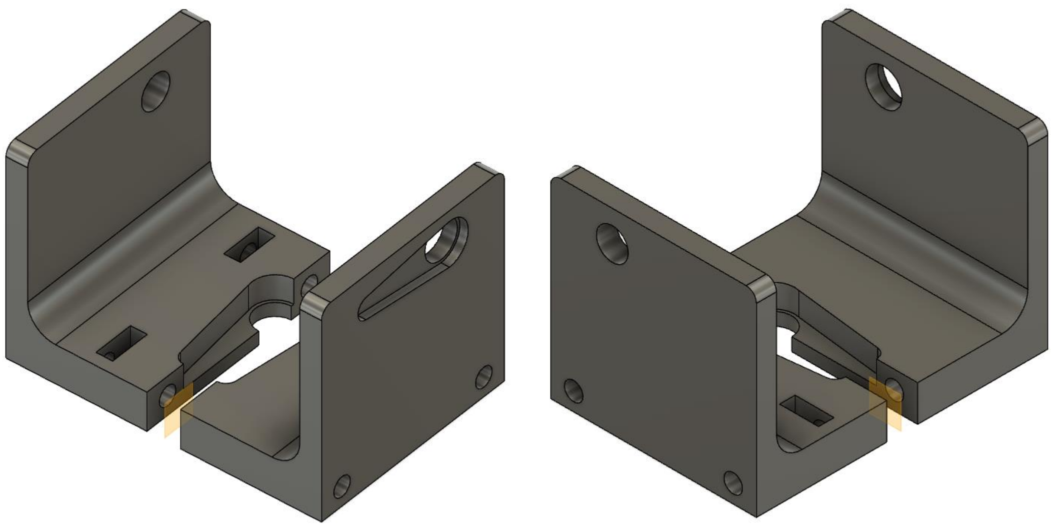


Fig. 3 Pan section of the mount

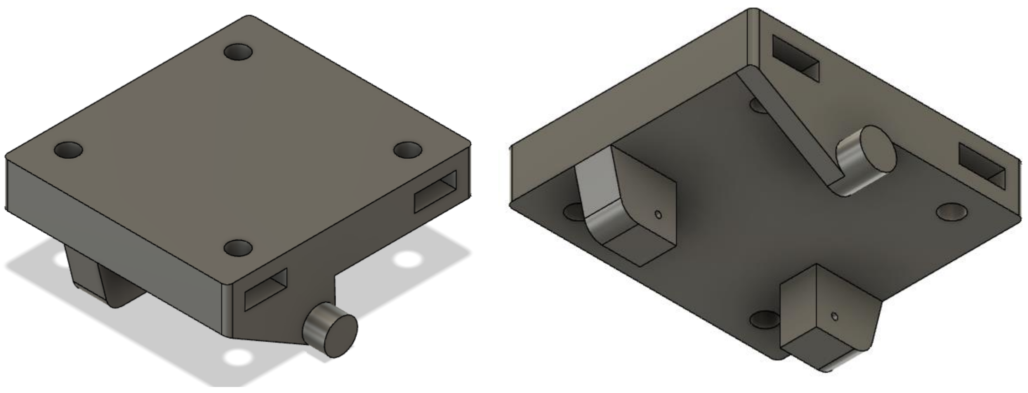


Fig. 4 Tilt section of the mount

Electrical

On the electrical side, the key goals were powering and controlling the three stepper motors and two servo motors and interfacing the limit switches. To achieve this, a dedicated driver and interfacing board was designed.

Power Delivery- The stepper motors operate at 12V and are controlled by 5V control lines via the A4899 driver. The servo motors require a 5V input. Therefore, a 12V input through a barrel connector and a 5V linear regulator were used to meet the power requirements. The circuit design is detailed below.

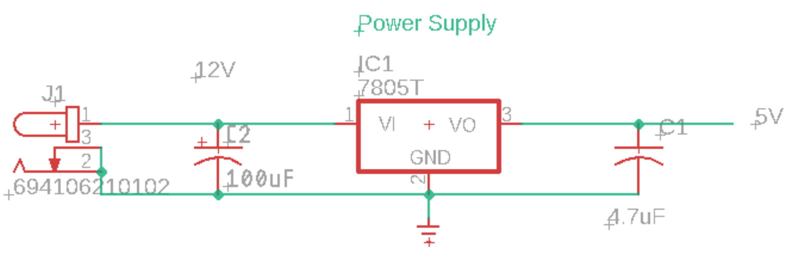


Fig. 5 Power Circuit

Stepper Driver- Each stepper motor utilised in the setup necessitates an individual A4988 driver. These drivers are controlled via five pins: MS1, MS2, MS3, STEP, and DIR. The STEP pin determines the motor's rotational direction, while the STEP pin regulates the revolutions per minute based on the frequency.

Additionally, micro-stepping, which controls the number of steps in a single revolution, is determined by the states of the MS1, MS2, and MS3 pins. This configuration connects MS1 and MS2 pins to the 5V lines to set the resolution to 1/8 steps. The ENABLE' pin is pulled high when not in operation and low through a transistor when the drivers are used. The schematic for the stepper drivers is given below.

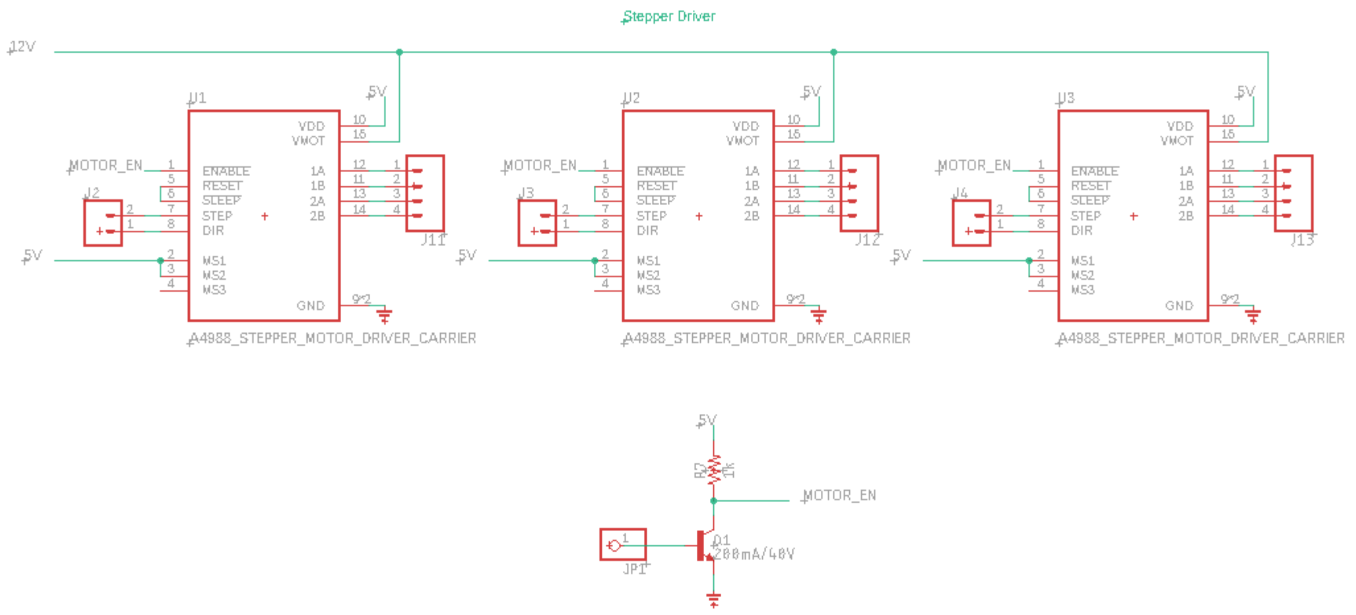


Fig. 6 Stepper Driver

Servo Circuit- The SG90 servo motors do not require specialised drivers and can be directly controlled by setting the frequency at the control pin. The linear voltage regulator provides the supply voltage. The circuit configuration is depicted below.

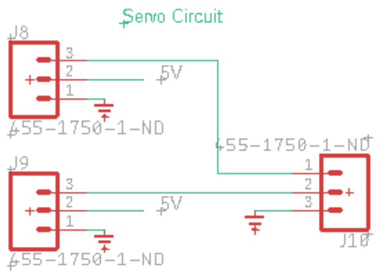


Fig. 7 Servo Circuit

Limit Switch Circuit- The limit switches serve the purpose of locating the robot's home position. Given that the Raspberry Pi operates on a 3.3V control line, the supply provided to the limit switch must be limited to 3.3V as well. Consequently, the supply is sourced from the Pi to the limit switch, and feedback within the same voltage range is acquired through a combination of resistors. The circuit layout is illustrated below.

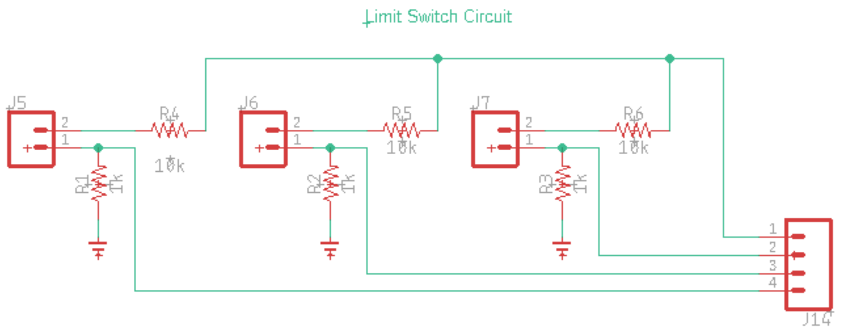


Fig. 8 Limit switch circuit

Software

This Python code controls a robotic arm composed of servo motors (SG90) and stepper motors (NEMA17). It utilises several libraries, such as pigpio for servo motor control, RPi.GPIO for general GPIO control, datetime for time-related functions, time for sleep, pygame for a graphical user interface and event handling, copy for duplicating objects, and math for mathematical computations. The code includes classes and functions to handle the initialisation, control, and movement of these motors, along with kinematics calculations to determine the position and movement of the robotic arm.

Class Descriptions-

class servoSG90()- Handles the operations of an SG90 servo motor.

- __init__(self, pinNo: int): Initializes the servo motor on a specified GPIO pin, sets the mode to output, sets the PWM frequency to 50Hz, and sets the initial position to the middle (90 degrees).
- find_pulseWidth(self, angle: int): Calculates the pulse width for a given angle to control the servo position.
- set_angle_easing(self, angle, time): Gradually moves the servo to the target angle over a specified time using easing (smooth transition).
- set_angle(self, angle: int): Sets the servo to the specified angle directly.
- close(self): Stops the PWM signal to the servo motor.

class stepperNEMA17()-

Handles the operations of a NEMA17 stepper motor.

- __init__(self, name: str, directionPin: int, stepPin: int, stepsPerUnit: int, minTravel: int, maxTravel: int, pulseDelay: float): Initializes the stepper motor with given parameters including its name, GPIO pins for direction and step control, steps per unit, travel limits, and pulse delay.
- continuous_rotate(self, direction): Rotates the motor continuously in the specified direction until stopped.
- move_relative(self, angle: float): Moves the motor by a relative angle from its current position.
- motor_stop(self): Stops the motor by setting a stop flag.

Function Descriptions-

goToHomePosition ()- Moves the robotic arm to its home position.

- Description: Sets servos to the middle position, enables motors, and moves stepper motors to their home switches.

inverseKinematics (x: float, y: float)- Calculates the joint angles required for the robotic arm to reach a given (x, y) position.

- Parameters: x, y - target coordinates.
- Returns: Angles q1, q2 in degrees and a success flag.

forwardKinematics (q1Deg: float, q2Deg: float)- Calculates the (x, y) position of the robotic arm's end effector based on the given joint angles.

- Parameters: q1Deg, q2Deg - joint angles in degrees.
- Returns: Coordinates x, y.

moveRelativePosition (x: float, y: float)- Moves the robotic arm's end effector by a relative (x, y) distance.

- Parameters: x, y - relative distances to move.
- Description: Calculates the target joint angles and moves the motors accordingly.

moveToPosition (x: int, y: int, z: int)- Moves the robotic arm's end effector to a specified (x, y, z) position.

- Parameters: x, y, z - target coordinates.
- Description: Uses inverse kinematics to find target angles and moves the motors to the desired position.

Main Function and Setup-

The main function initialises the GPIO settings, creates motor objects, and offers a user interface for teleoperation using pygame. It provides different modes for individual motor control, TCP teleoperation, and moving to a specific point.

main():
Runs the main program loop, allowing the user to choose between different control modes and handle the motor operations accordingly.
Modes:

- Individual Motor Control: Allows manual control of each motor using keyboard inputs.
- TCP Teleoperation: Allows control of the end effector's position in the x, y, and z planes.
- Go to Point: Moves the end effector to a user-specified position.

Exception Handling:
Ensures that motors are stopped and GPIO is cleaned up if the program is interrupted.

if __name__ == "__main__":
The entry point of the program that calls the main() function within a try-except block to handle keyboard interrupts gracefully.

Pin and Variable Declarations

- GPIO Pins: Specifies the GPIO pins used for motor control, home switches, and servos.
- Robot Parameters: Defines physical parameters of the robotic arm, including lengths and angles.
- Motor Parameters: Specifies the steps per unit for each motor axis.
- Motor and Servo Initialization: Creates instances of the stepper motors and servos with specified parameters.

This structured approach allows the robotic arm to be controlled flexibly and interactively, providing various methods for precise movements and adjustments.

Realtime Implementation and Results

Mechanical

The 3D-printed parts for the pan and tilt motion, as printed and after assembly, are given below.

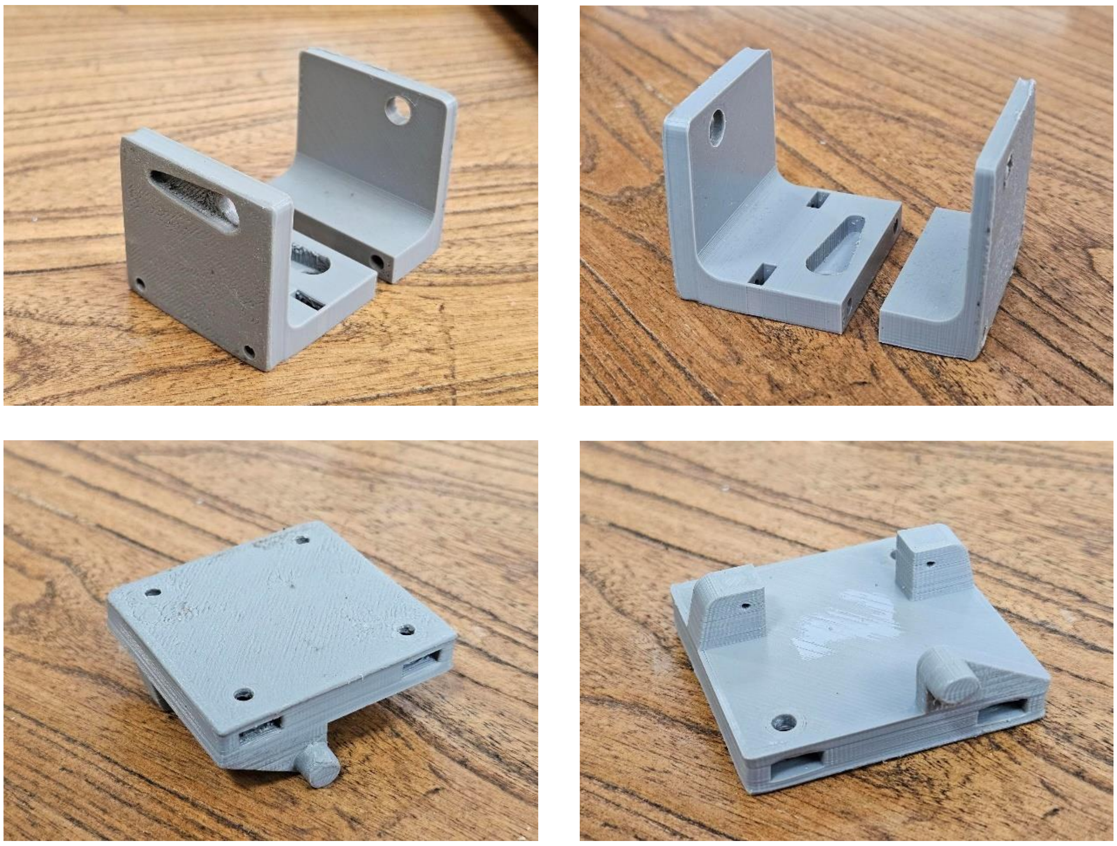


Fig. 9 3D-printed parts before assembly

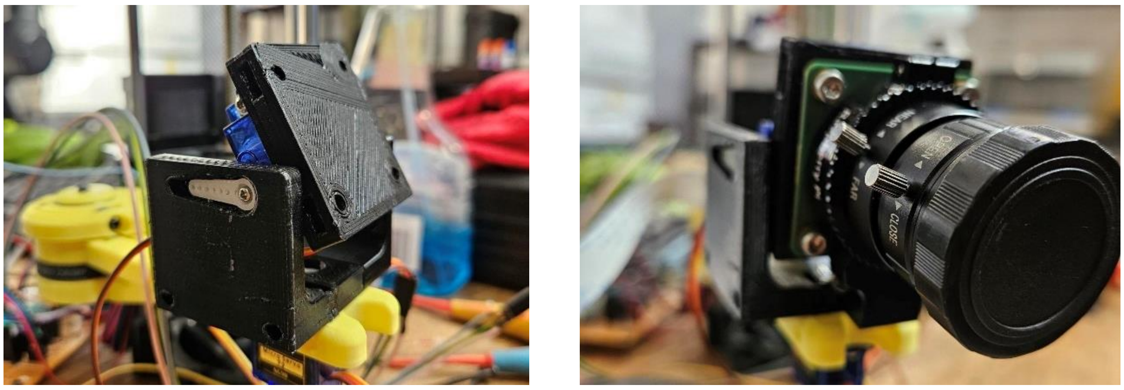


Fig. 10 3D-printed parts after assembly

Three limit switches were introduced to set the home position of the robot, which helps calculate the position of the robot links as per the revolutions of the motors. The placement of these switches is shown in the images below.

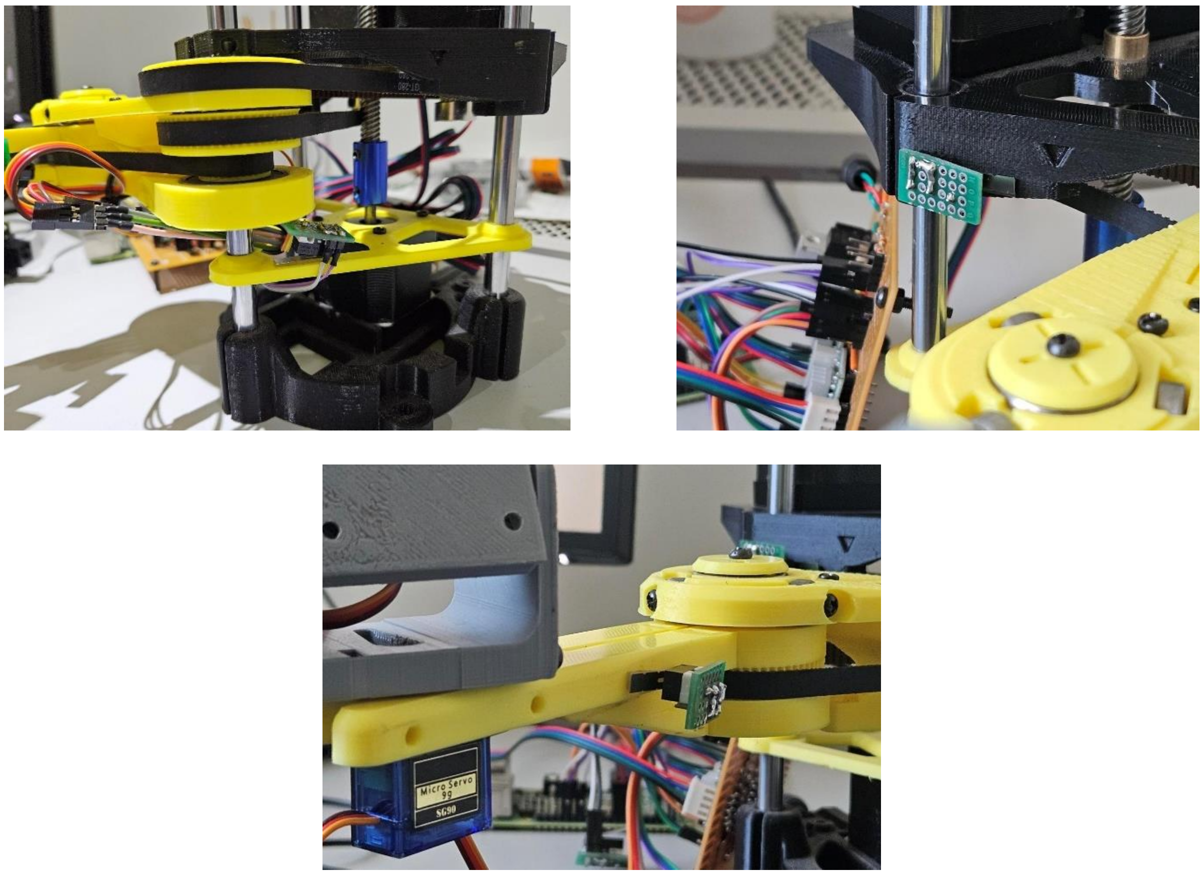
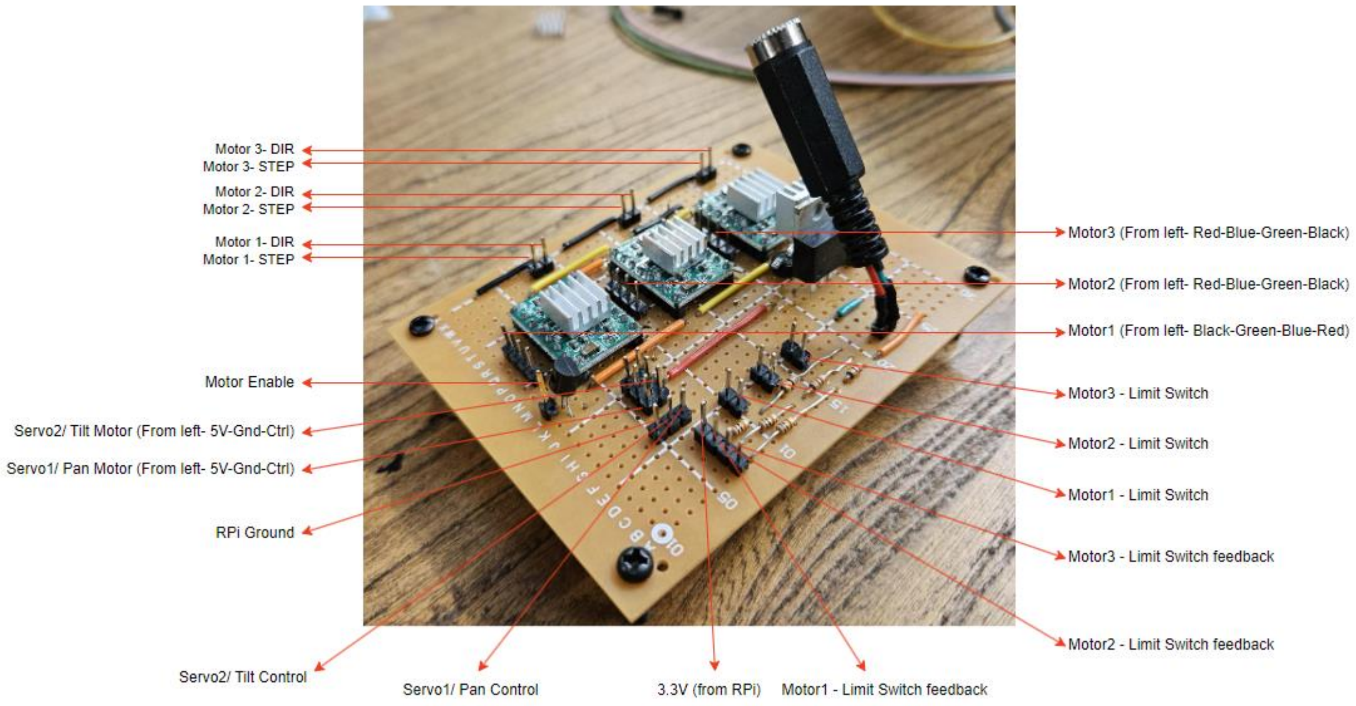


Fig. 11 Limit Switch Implementation

Electrical

The interface board between the motors and the Raspberry Pi was hand-soldered on a general-purpose (GP) board. The completed board with pin details to be connected to the Raspberry Pi 4b is given below.



Motor 3- DIR
Motor 3- STEP
Motor 2- DIR
Motor 2- STEP
Motor 1- DIR
Motor 1- STEP
Motor Enable
Servo2/ Tilt Motor (From left- 5V-Gnd-Ctrl)
Servo1/ Pan Motor (From left- 5V-Gnd-Ctrl)
RPi Ground
Servo2/ Tilt Control
Servo1/ Pan Control
3.3V (from RPi)
Motor1 - Limit Switch feedback
Motor3 (From left- Red-Blue-Green-Black)
Motor2 (From left- Red-Blue-Green-Black)
Motor1 (From left- Black-Green-Blue-Red)
Motor3 - Limit Switch
Motor2 - Limit Switch
Motor1 - Limit Switch
Motor3 - Limit Switch feedback
Motor2 - Limit Switch feedback

Fig. 12 Interface Board

#	Interface Board	Raspberry Pi (GPIO)	Comments
1	Motor 1- DIR	27	
2	Motor 1- STEP	17	
3	Motor 2- DIR	5	
4	Motor 2- STEP	6	
5	Motor 3- DIR	23	
6	Motor 3- STEP	24	
7	Motor Enable	22	
8	Servo1/ Pan Motor	-	From left- 5V-Gnd-Ctrl
9	Servo2/ Tilt Motor	-	From left- 5V-Gnd-Ctrl
10	RPi Ground	Pin 6	Loop completion
11	Servo1/ Pan Control	12	
12	Servo2/ Tilt Control	13	
13	3.3V	Pin 1	Since limit switches are used for feedback, 3.3V is used
14	Motor1- Limit Switch feedback	1	
15	Motor2- Limit Switch feedback	7	
16	Motor3- Limit Switch feedback	8	
17	Motor1- Limit Switch	-	
18	Motor2- Limit Switch	-	
19	Motor3- Limit Switch	-	
20	Motor1	-	From left- Black-Green-Blue-Red
21	Motor2	-	From left- Red-Blue-Green-Black
22	Motor3	-	From left- Red-Blue-Green-Black

Table 1 – Pin Connections

Software

The software (python code) used for the project is given in Appendix 2.

Limitations

The project encountered several mechanical and electrical limitations:

1. Interference in Arm Movement- Arm 2 rotates when Arm 1 rotates due to the series connection of motor belts. Motor 2 is connected to Arm 1 via a single belt drive, while Motor 3 is connected to Arm 2 using two belt drives. This configuration causes Arm 2 and Arm 1 to move together, independent of

Motor 3. For instance, a 90° turn in Motor 2 requires a compensatory 55° turn of Motor 3 in the same direction.

2. Belt Drive Accuracy- The belt drives responsible for rotating the arms result in inaccuracies in the number of steps needed for the motors to rotate 1°. Theoretical calculations suggest:
   - Motor 2: Steps per rev = 1600, Gear ratio = 4.5 --> 1600/360 * 4.5 = 20 steps/degree
   - Motor 3: Steps per rev = 1600, Gear ratio = 7.2803 --> 1600/360 * 7.2803 = 32.357 steps/degree

   However, practical testing showed:
   - Motor 2: 16 steps/degree
   - Motor 3: 25 steps/degree

3. Lack of Feedback from Stepper Motors- The absence of feedback from the stepper motors affects system accuracy. The motors' angles are estimated based on initial values set in the software rather than actual feedback, leading to potential positional errors.

4. Load Capacity of Servo Motors- The servo motor used for tilt motion is significantly affected by the load of the camera and lens mounted at the end of the arm. The total load exerts a force in the direction of gravity, impacting the motor's rotation even when a dedicated power supply is provided.

5. Cantilever Structure- The cantilever design of Arms 1 and 2, where the load acts at the end of Arm 2, results in vibrations during the tilt motion. This affects the stability and precision of the tilt stepper motor.

6. Limit Switch Placement- Positioning the limit switches to detect the home position is challenging due to the need for additional mounting provisions. The angles at which the arms meet the limit switches are not ideal, complicating the detection process.

Recommendations

Based on the findings and limitations of the project, the following practical recommendations are proposed to enhance the performance and reliability of the SCARA robot system for remote monitoring in digital agriculture:

1. Implement Custom PCBA- Replace the hand-soldered driver and interfacing board with a custom Printed Circuit Board Assembly (PCBA). This will improve the reliability and durability of the electrical connections, reduce potential errors, and provide a cleaner, more professional setup.

2. Upgrade Servo Motors- Replace the current SG90 servo motors with more powerful alternatives to better handle the load of the camera and other end-effector components. This will ensure smoother and more accurate movements, especially under the weight of the mounted equipment.

3. Redesign Limit Switch Mounts- Develop and 3D print more effective mounts for the limit switches. Improved mounting solutions will ensure consistent and accurate home position detection, enhancing the overall precision and reliability of the robotic system.

The SCARA robot system can achieve greater accuracy, stability, and functionality by addressing these recommendations, making it more effective for remote monitoring applications in digital agriculture.

Conclusion

The project "Control of SCARA Robot System using Inverse Kinematics for Remote Monitoring in Digital Agriculture" successfully demonstrates the integration of a PRR-configured SCARA robot for precise monitoring and control in an agricultural setup. Through meticulous design and implementation, the project achieved significant milestones in mechanical, electrical, and software domains.

Mechanically, the pan-tilt motion system was designed using SG90 servo motors, ensuring smooth and accurate movement of the camera for remote monitoring. Integrating 3D-printed parts and the innovative design inspired by the Mini Pan-Tilt Kit from The Pi Hut provided a robust and reliable setup. The challenges of mechanical inaccuracies due to belt drives and the cantilever structure were identified, highlighting areas for future improvement.

Electrically, the project successfully interfaced stepper motors and servos with a Raspberry Pi 4B, utilising a custom-designed driver and interfacing boards. Implementing limit switches for home position detection enhanced the system's reliability, although the mechanical placement of these switches posed challenges.

Software development was a critical aspect of this project, involving the control of motors through Python code and the implementation of inverse kinematics for precise end-effector positioning. The software effectively managed the coordination of movements and provided a user-friendly interface for teleoperation using pygame. Despite limitations in feedback from stepper motors and the load capacity of servo motors, the system demonstrated accurate control and functionality.

The literature review provided a solid foundation for understanding the kinematics and inverse kinematics of 2-axis planar robots, which was crucial for the project's success. The insights gained from sources like Daslhub, MIT OpenCourseWare, and Robot Academy guided the development of kinematic equations and control algorithms.

In conclusion, this project showcases the potential of using a SCARA robot for remote agricultural monitoring, emphasising the importance of precise control and real-time feedback. The limitations identified in mechanical accuracy, load capacity, and feedback mechanisms offer valuable lessons for future enhancements. This project lays a strong foundation for further research and development in applying robotic systems in digital agriculture.

Bibliography

Chang, T. S., & Dubey, R. V. (1986). A study of inverse kinematics solutions for robotic manipulators. Retrieved from https://core.ac.uk/download/pdf/80147945.pdf

Daslhub. (n.d.). 2-link kinematics. Retrieved from https://www.daslhub.org/unlv/wiki/doku.php?id=2_link_kinematics

Hawkins, R. P. (1999). Mathematical analysis of kinematics. Retrieved from https://www.math.lsu.edu/system/files/paper.pdf

MIT OpenCourseWare. (2005). Introduction to robotics. Retrieved from https://ocw.mit.edu/courses/2-12-introduction-to-robotics-fall-2005/c8828a16e71c246b78461dd0596b983f_chapter4.pdf

Robot Academy. (n.d.). Robotic arms and forward kinematics. Retrieved from https://robotacademy.net.au/masterclass/robotic-arms-and-forward-kinematics/?lesson=262

Robot Academy. (n.d.). Inverse kinematics and robot motion. Retrieved from https://robotacademy.net.au/masterclass/inverse-kinematics-and-robot-motion/?lesson=291

The Pi Hut. (n.d.). Mini Pan-Tilt Kit Assembled with Micro Servos. Retrieved from https://thepihut.com/products/mini-pan-tilt-kit-assembled-with-micro-servos

Appendix I: Project Communication Log

Project Title:	Control of SCARA robot system using Inverse Kinematics for remote monitoring in digital agriculture		
Student Name:	Udhaya Kumar Parameswaran	**Supervisor Name:**	Dr Daniel Franklin
Date	**Event**	**Topic of Communication**	**Outcome**
20-02-2024	CB11.08.205	Initial discussion about the project topic	Project topic finalisation
06-03-2024	CB11.08.205	Project scope discussion	Project scope declaration
19-03-2024	Teams chat	Initial pan and tilt motion 3d printed components	-
28-03-2024	Teams chat	Interfacing of pan and tilt servos and 3d printed parts in the robot setup	-
15-05-2024	Teams chat	Robot modes discussion, Project report discussion	Project report structure finalisation
23-05-2024	Teams chat	Project code	-

Appendix II: GitHub Repo link

https://github.com/UdhayROB/pyBotControl