MECHATRONIC SYSTEMS

4MECHMA032

Project Report

Group 8

Mostafa Ahmed Ismail Bayoumy Kashaf

Kothalawala Hewage Randima Fernando

Guegang Vanes Loic Gnoitik

Naresh Chand

Raihan Ahamed Rafeeq

1. Introduction

This report contains details regarding the methodology followed, the theory implemented, the calculations made, and the tests run by Group 8 in the completion of the Mechatronic Systems Project. This project involved the use of the Lego Mindstorm EV3 system alongside Matlab to systematically control a robot arm to pick, place and move between different heights. While the first few labs involved getting used to the robot system and programming basic motions, over the course of the project timeline, the Group worked on programming maneuvers based on directly adjusting angles, implementing PID Control Loops to these maneuvers to make them smoother and more precise, and finally using inverse kinematics functions to allow the program to read Cartesian inputs for arm movement. All requirements of the project were successfully completed and the theory and .

2. Theory

2.1. Inverse Kinematics

A vital part of the program involved the use of Inverse Kinematics to obtain the joint angles of the robot. Implementation of a function for inverse kinematics allowed the program to directly take a Cartesian input to maneuver the robot arm.

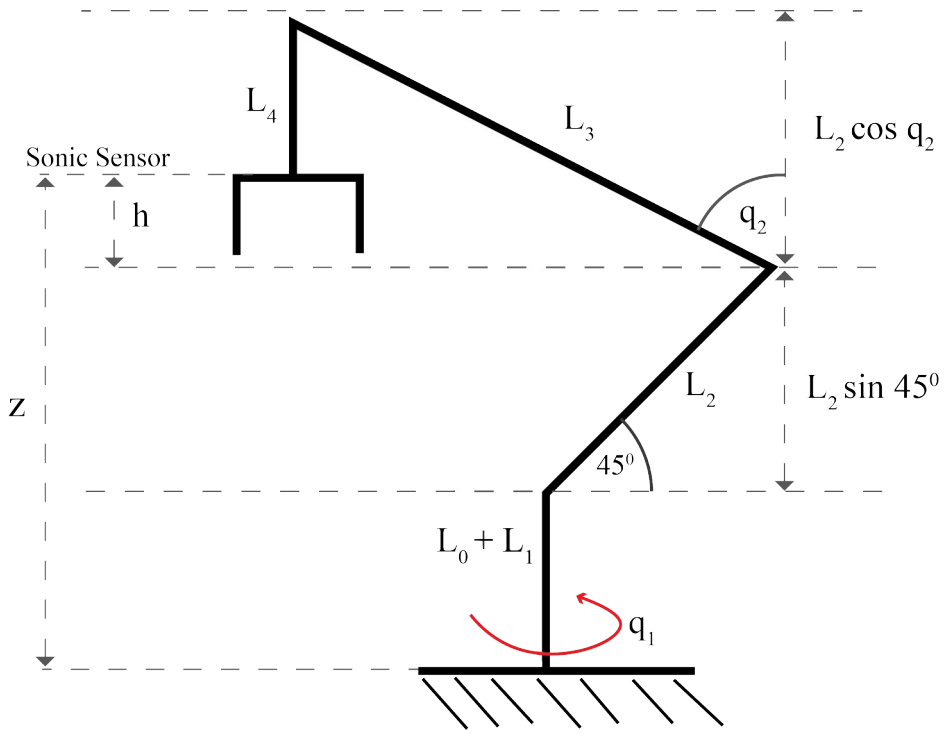

Figure 1: 2D Geometric Model of Robot

Considering Figure 1, the following data is available as measurements for the Links.

L_0 = 8mm L_1 = 80mm L_2 = 95mm L_3 = 185mm L_4= 60mm

Observing the geometry of Figure 1, the following trigonometric relation could be derived via the triangle comprising L_3 and (L_4 + h).

$$(L_4+h)=L_3\cos(q_2)$$

$$q_2=\arccos\left(\frac{h+L_4}{L_3}\right)$$

Observing the placement of **z** in Figure 1, the following relationship could be constructed.

$$z=L_0+L_1+L_2\sin(45^0)+h$$

$$h=z-[L_0+L_1+L_2\sin(45^0)]$$

Applying this to the expression for q_2 we may obtain a formula for this joint angle based on **z.**

$$q_2=\arccos\left(\frac{z+L_4-[L_0+L_1+L_2\sin(45^0)]}{L_3}\right)$$

Based on measurements and calculations it was determined that the value of the q_2 angle at its minimum (limited by the touchsensor on arm) is 43^0. For ease of implementation in the program, this lower boundary was taken as reference for q_2 making q_2 = 0 at the above joint angle. The formula above could be modified as follows to reflect this.

$$q_2=\arccos\left(\frac{z+L_4-[L_0+L_1+L_2\sin(45^0)]}{L_3}\right)-43^0$$

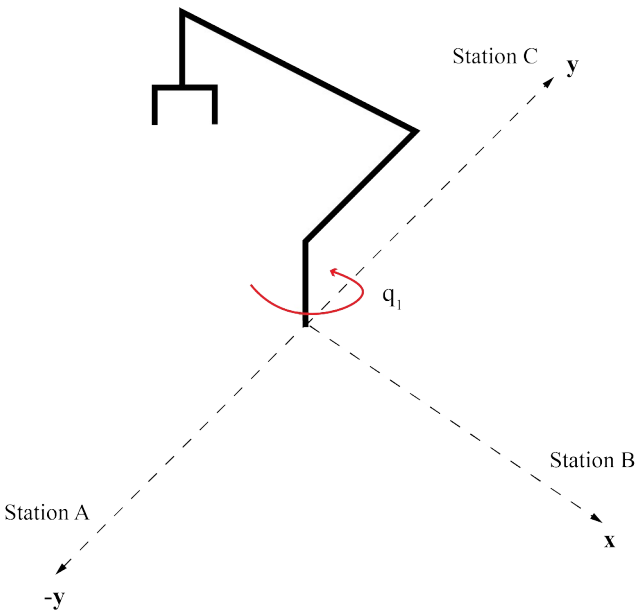

Figure 2: 3D Geometric model with Cartesian Coordinate System

From Figure 2 it could be observed that the angle q_1 can be directly expressed as a function involving the x and y coordinate inputs. Hence, the following relationship could be obtained.

$$q_1 = \arctan\left(\frac{x}{-y}\right)$$

The above formulae were used in a separate function to obtain the q_1 and q_2 angles required for a particular movement.

2.2. PID Tuning

Implementing a PID Control Loop into the program allowed the movements to be done more precisely, smoothly, and with no overshoot. A basic representation of the system could be given as follows.

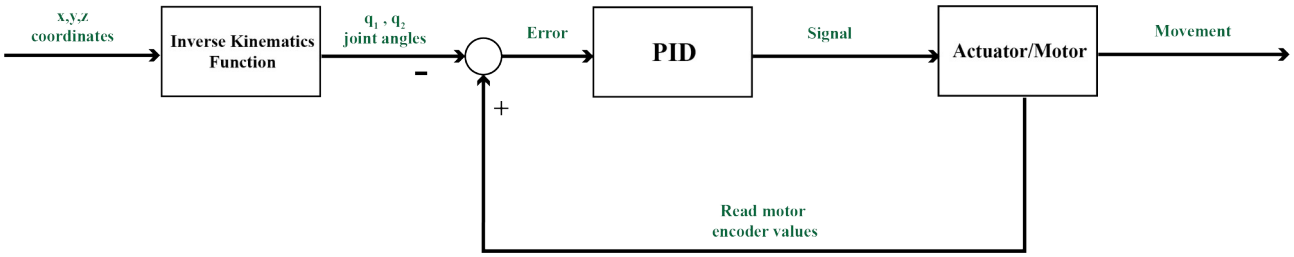

Figure 3: Operation of the system

PID stands for Proportional, Integral, Differential Control. These are simply various ways of treating the Error term obtained (difference between required output and encoder reading). If the error is given as follows,

E = Required q value – q value read via encoder

The signal produced by the PID can be mathematically represented as follows.

$$PID = K_P E + K_I \int E\,dt + K_D \frac{dE}{dt}$$

The K_p, K_I and K_D values are gain values which need to be tuned to obtain the acceptable behavior from Robot.

Note: The Derivative of the above function was not implemented to the loop as it has a higher tendency to damage the motors. Hence, K_D was assigned 0.

Before the PID tuning began, Clamping of the Output signal and a Wind-up Prevention logic was implemented within the PID function.

2.2.1. Clamping

To prevent an exponentially large signal being provided to the motors and the subsequent jerking of the robot arm, the output signal from the PID was clamped within a certain region. Therefore, despite the magnitude of the input signal to the motors, the rotational speed of the motors were limited to between 25 and -25 units.

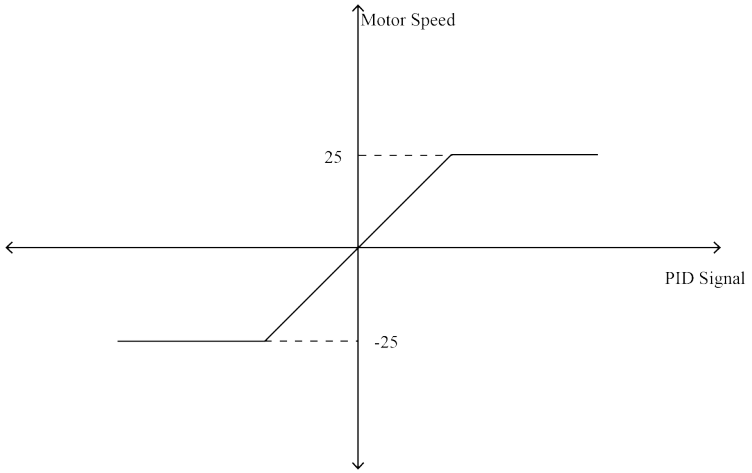

Figure 4: Clamping

2.2.2. Wind-up Prevention

The wind-up of the integral occurs in a situation where for some reason the integral keeps increasing even after the output signal to the motor has reached saturation. In such an instance, the PID signal increases exponentially, resulting in the PID signal possessing a large magnitude even when the error term reaches 0, causing the Robot arm to overshoot. To prevent this, a logic must be implemented to the program to cut-off the integral when the Motor Speed signal is saturated and the signs of the error term and the integral term outputs are equal.

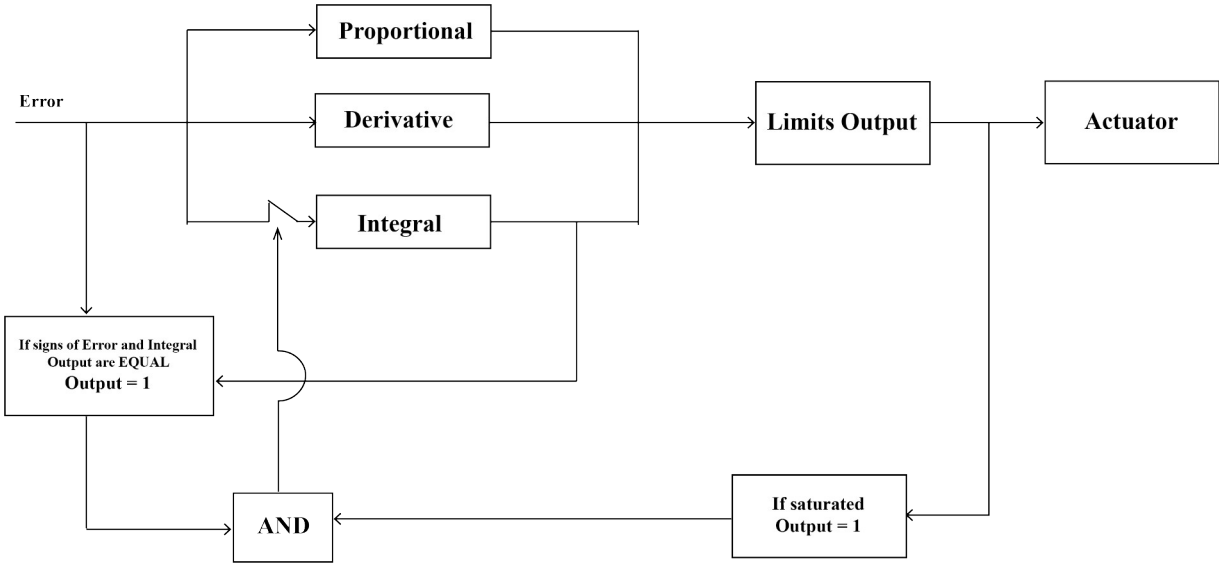

Figure 5: Wind-up Prevention by cutting off integral

3. Methodology

3.1. Tuning

As a clear mathematical model or a transfer function was not available for the system, the PID tuning had to be done manually via trial and error. A representation of the workflow involved in tuning the PID values is given below.

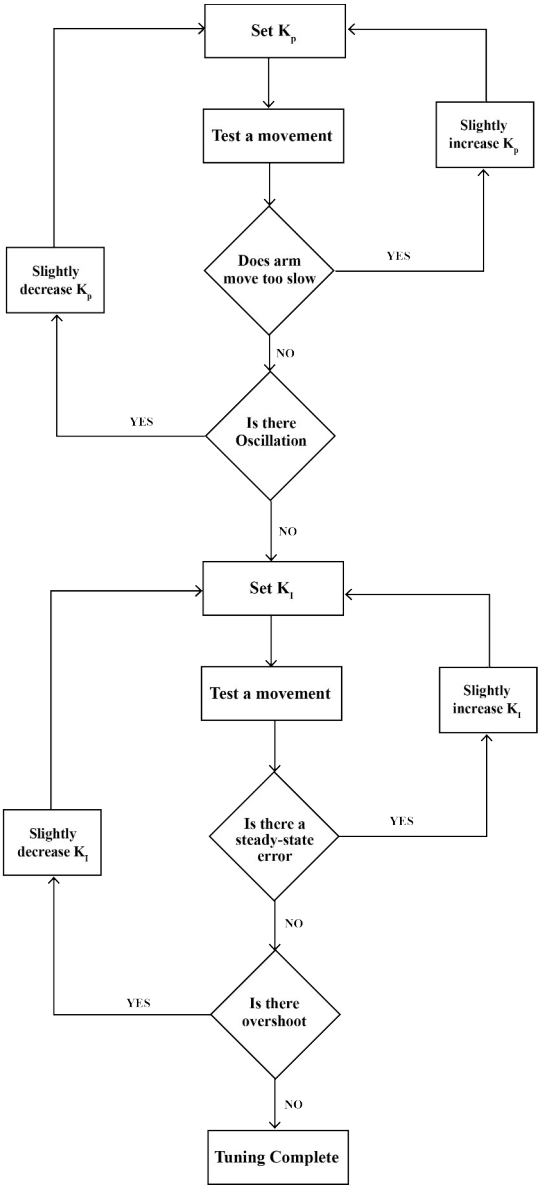

Figure 6: PID Tuning Workflow

3.1.1. Tuning Results

The following plots were obtained from the tuning of the PID.

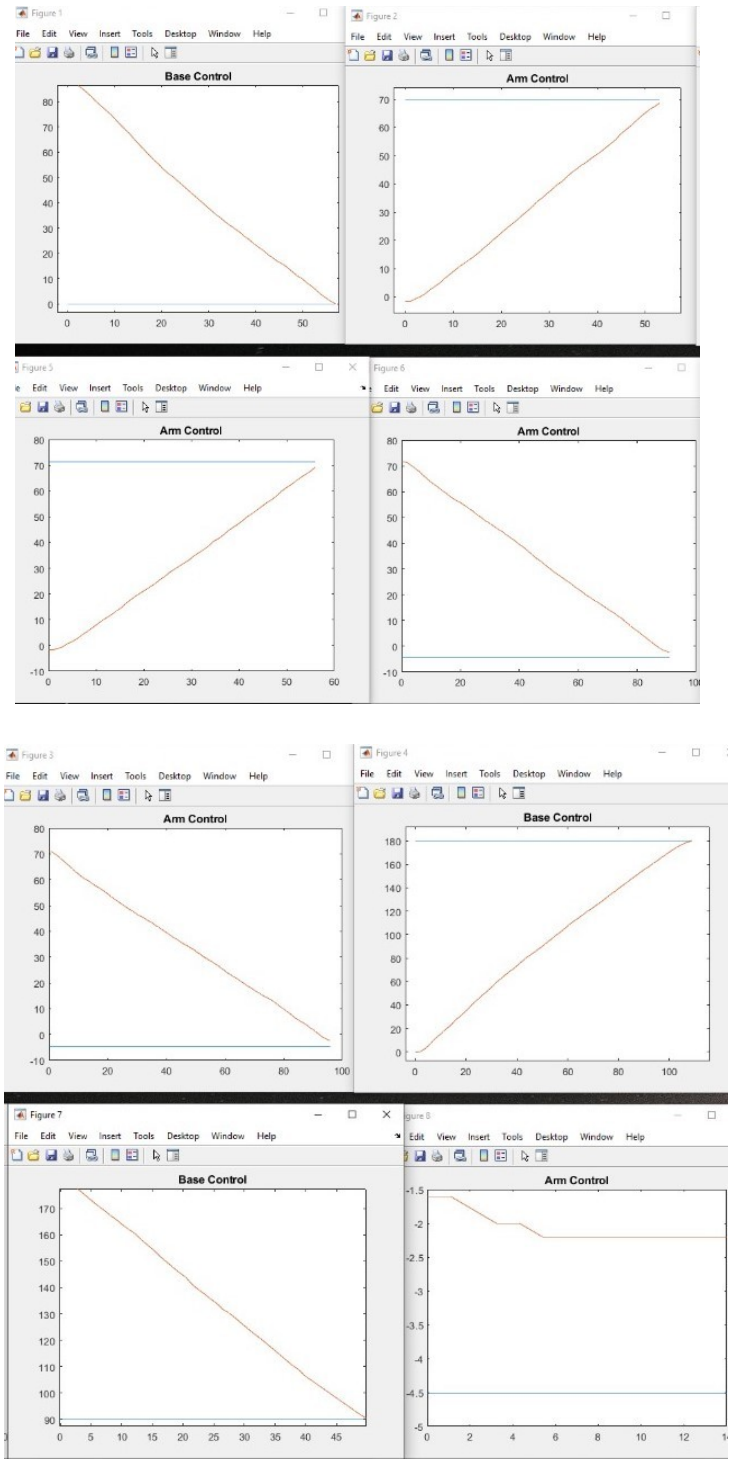

Figure 7: The variation of q1 (base) and q2 (arm) against the required values.

The above plots were obtained during PID tuning. These were used to observe speed, saturation, overshoot and steady state errors in the system.

3.2. Task 5 – Homing

The Homing function in the program returns the arm to the topmost position in Station B. All complicated maneuvers start and end in this position. The Homing position has the joint angles q_1 (base angle) and q_2 (arm angle) at 90^0 and 43^0 (note that this is the minimum angle for q_2 due to touch sensor, hence the code treats this angle as 0) respectively. The logic implemented in this function behaves as follows.

Note : The following flowchart represents the basic mathematical logic implemented in the program. In the robot however the base angle was measured in the clockwise direction by the motor encoder (whereas the kinematics defined q_1 in the counterclockwise direction) and the arm motor angle was measured so that a downwards motion would be positive. These variations were suitably accommodated in the code so that the following mathematical logic holds true.

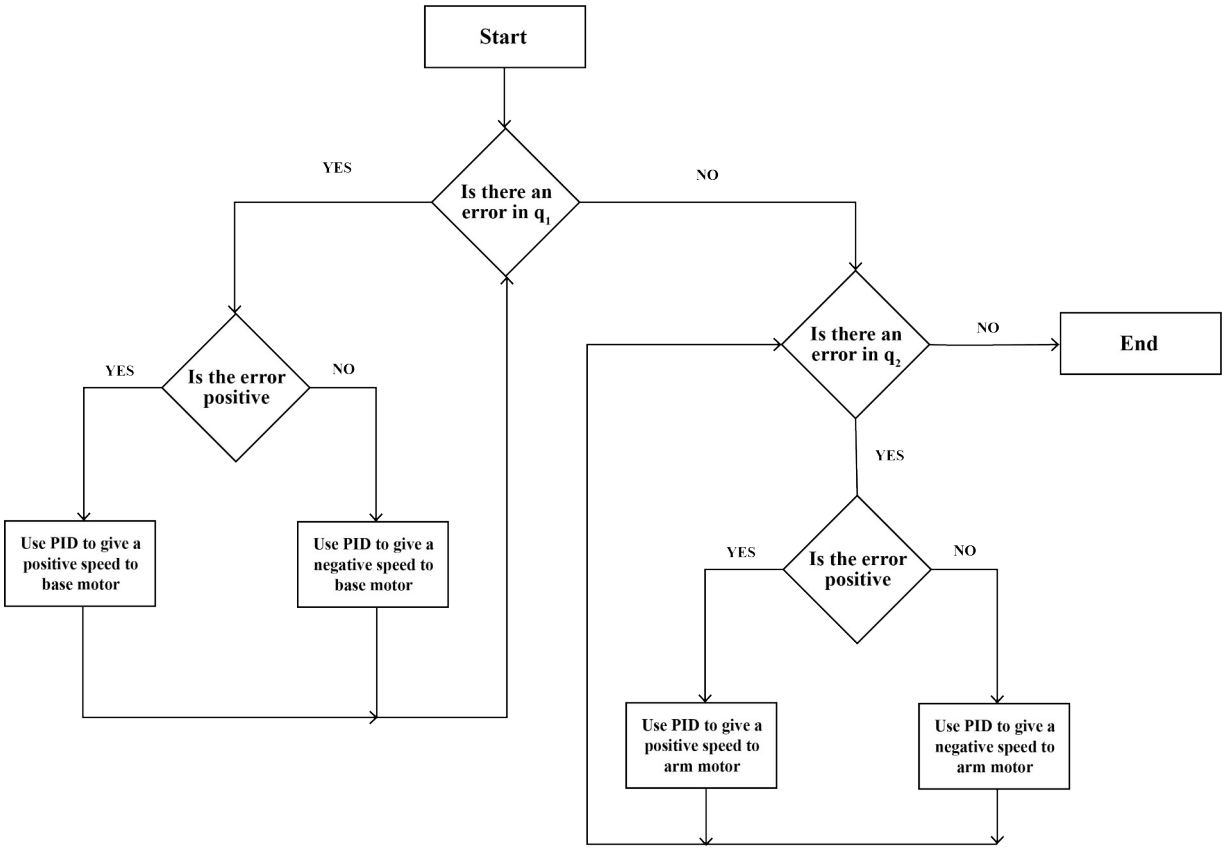

Figure 8: Task 5 Implemented Logic

3.3. Task 6 – Pick from B, Place at C, and Return Home

This procedure starts from home position, checks height of Station B and implements a Pick operation, carries the object to station C, checks height of Station C and implements a Place operation, and returns to home position. Throughout this operation the program uses PID control to move the robot arm to its various positions. Further, the Inverse Kinematics formulae prepared previously are used to obtain the q_1 angles for different Stations and the q_2 angles for different Station heights. Station heights are measured using the sonicsensor.

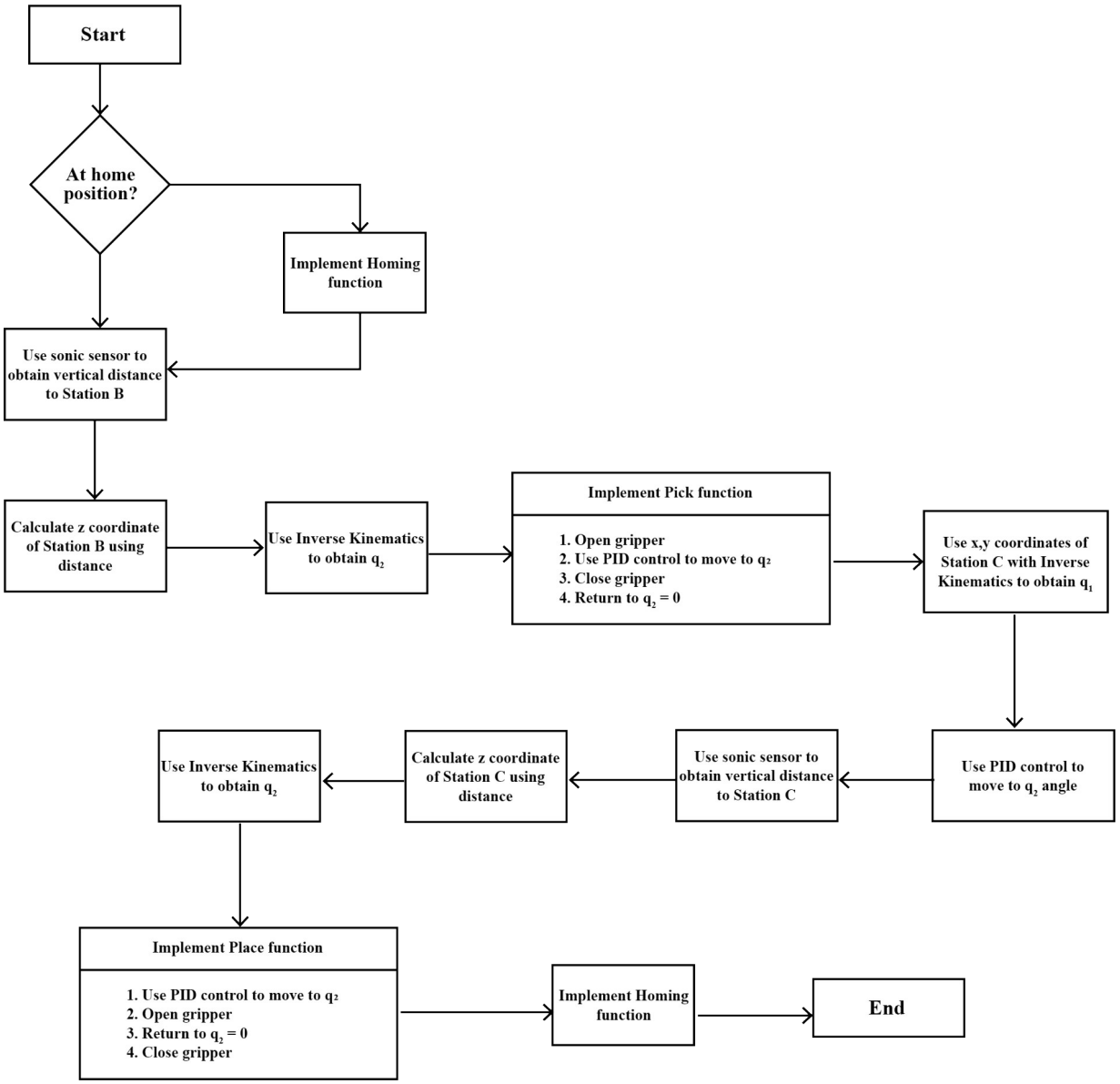

Figure 9: Task 6 - Implemented Logic