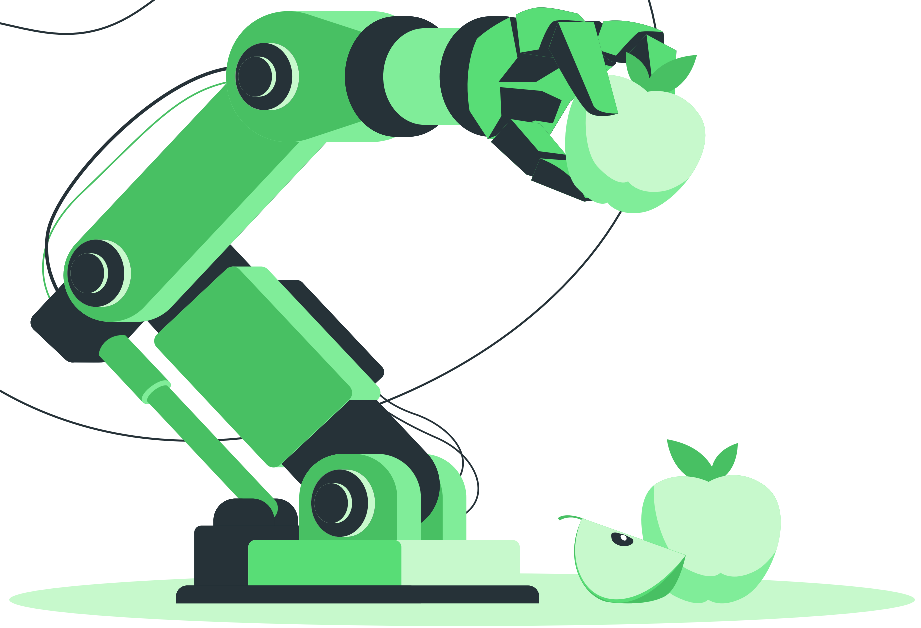


Robotic Arm & MATLAB Kinematics Interface

(Intro to Robotics)

6/9/2025

Presented To:

Sir Haroon

Presented By:

Group A

Work Division

Course Project A: Robotic Arm & MATLAB Kinematics Interface				
S#	Name	Reg Number	Role Assigned	Duties Assigned
1	Laiba	2021-BEE-008	CAD Design Lead Aroosa's Helper	- Lead CAD modeling of robotic arm
2	Aroosa	2021-BEE-003	Component Selection, BOM & Calibration Lead Palwasha's Helper	- Lead selection of components, BOM and calibration of mechanical and electronic parts
3	Palwasha	2021-BEE-015	Wiring Diagram & Physical Assembly Lead	- Lead development of wiring diagrams and physical assembly of electronics and wiring
4	Ayesha (Group Lead)	2021-BEE-004	Kinematics Implementation Lead (FK & IK) Shanza's & Muneeba's Helper	- Lead development and testing of forward and inverse kinematics - Compilation of Final Document, Poster and Presentation
5	Shanza	2021-BEE-016	MATLAB GUI Development Lead Muneeba's Helper	- Lead MATLAB GUI development (interface controls and 3D plotting)
6	Muneeba Bibi	2021-BEE-012	Integration Lead Lead Afsha's Helper	- Lead system integration
7	Afsha	2021-BEE-001	Testing Lead	- Lead system testing, debugging, and error analysis

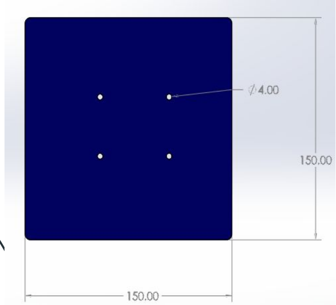
1.

CAD Design

Laiba Shahzad
2021-BEE-008

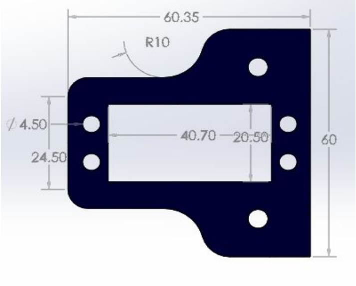
Parts of Robot

Base Plate



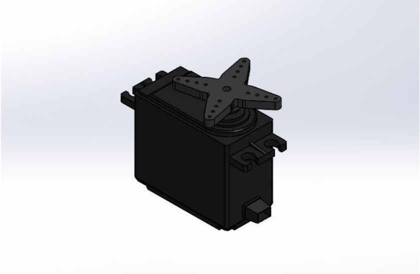
Commands:
Sketching includes Rectangle, circle and fillet commands. Then the Extrude command.

Servo Mount



Commands:
Sketching includes Rectangle, circle and fillet commands. Then the Extrude command.

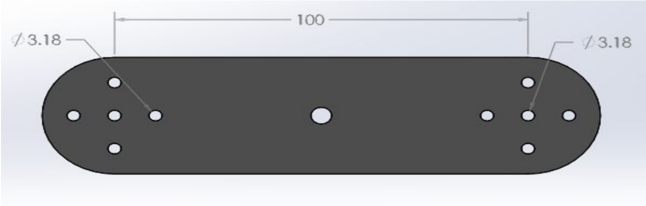
Servo Motor



Commands:
Sketching includes Rectangle, circle and fillet commands. Then the Box Extrude and cut extrude command.

Continued…

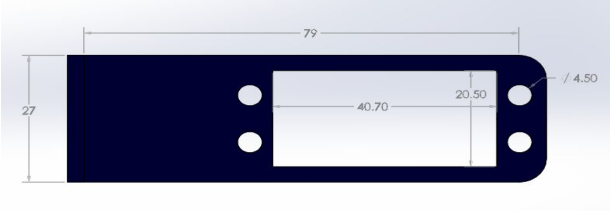
Link 1



Commands:

Sketching includes Rectangle, circle and fillet commands. Then the Extrude command.

Link 2:

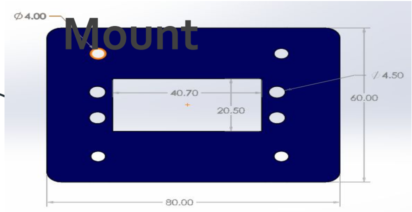


Commands:

Sketching includes Rectangle, line, circle and fillet commands. Then the Extrude command.

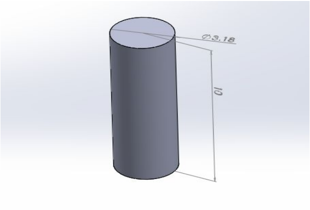
Continued...

Base Motor Mount



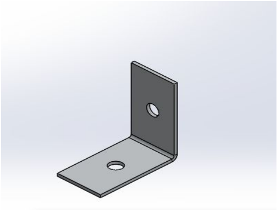
Commands:
Sketching includes Rectangle, circle and fillet commands. Then the Extrude command.

Pin



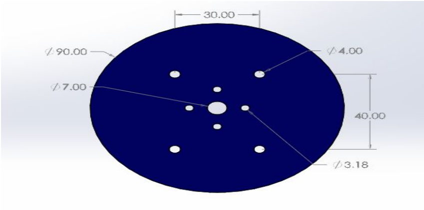
Commands:
Sketching includes circle command. Then the Extrude command.

Angle Bracket



Commands:
Sketching includes Rectangle, circle and fillet commands. Then the Extrude command.

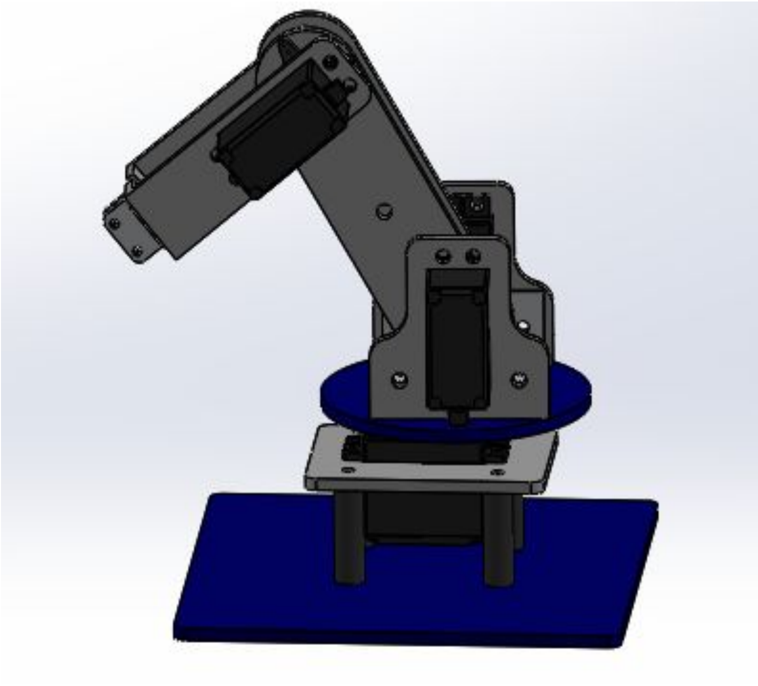
Base Circular Plate



Commands:
Sketching includes Rectangle, circle and fillet commands. Then the Extrude command.

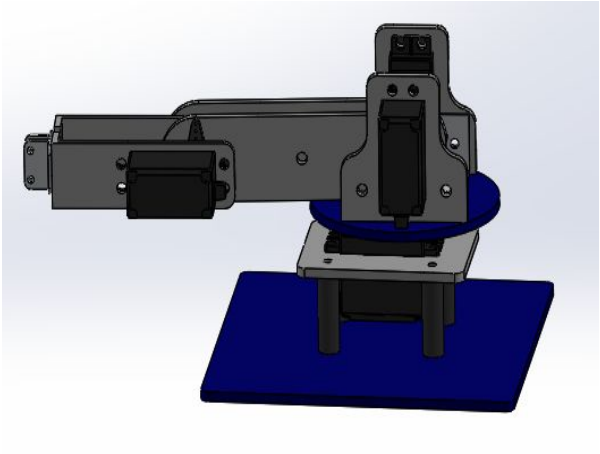
Dimensions

- All Revolute Joints & Links Breakdown
- Joint 1 (Base Rotation) – About Z-axis .
- Joint 2 (Shoulder Joint) – raises or lowers the first arm segment.
- Joint 3 (Elbow Joint) – controls bending between the gripper and lower arm.
- Joint 4 (Gripper Actuation) – servo opens and closes the gripper.



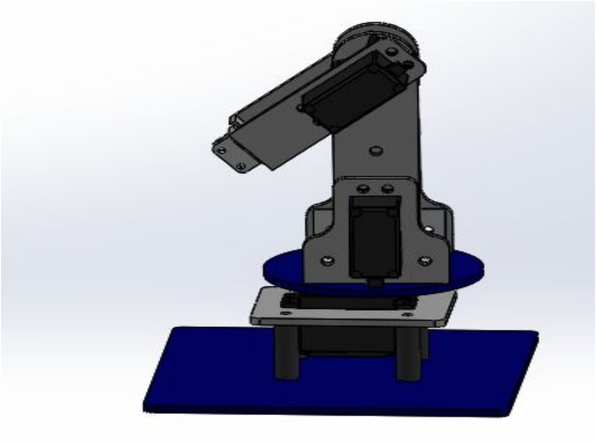
Desired Pose

	a_{i-1}(mm)	α_{i-1}	d_i	θ_i
1	0	90°	0	0
2	100	0°	0	0
3	60	0°	0	0
4	0	0°	0	θ_4



Continued...

	a_i(mm)	$\propto_i$	d_i	θ_i
1	0	90∘	0	45
2	100	0∘	0	-90
3	60	0∘	0	0
4	0	0∘	0	θ_4

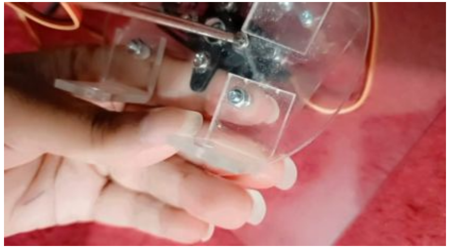


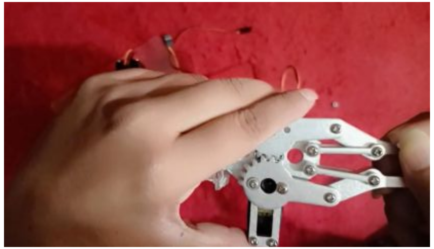
2. Component Selection ,BOM and Calibration

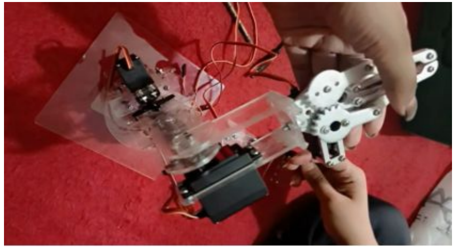
Aroosa Bibi
2021-BEE-003

Hardware Development

- Cutting of Acrylic Sheet and Assembling of robotic arm







Components used

- ESP 32
- Servo motors MG996R
- Spacers
- Gripper
- Charger
- Miscellaneous us Items
- Connecting wires
- Breadboard

Calibration

- For adjusting angles of servo motors and ensure smooth motion

Servo-1:	Angle of rotation is from 0 to 180 degrees
Servo-2: :	Angle of rotation is from 30 to 180 degrees
Servo-3: :	Angle of rotation is from 0 to 150 degrees
Servo-4: :	Angle of rotation is from 50 to 180 degrees
Servo-5: :	Angle of rotation is from 0 to 90 degrees


```
#include <ESP32Servo.h>
Servo servo1, servo2, servo3, servo4, servo5; // Declare servos sep
void setup() {
    Serial.begin(115200);
    // Attach each servo to its respective pin
    servo1.attach(18);
    Serial.println("Servo 1 initialized on pin 18");
    servo2.attach(19);
    Serial.println("Servo 2 initialized on pin 19");
    servo3.attach(21);
    Serial.println("Servo 3 initialized on pin 21");
    servo4.attach(22);
    Serial.println("Servo 4 initialized on pin 22");
    servo5.attach(23);
    Serial.println("Servo 5 initialized on pin 23");
}
void loop() {
    // Move each servo separately


    // delay(2000);
        servo2.write(120);//150 to 0 // opposite with servo3
    //delay(500);
        servo3.write(60); // 30 to 180//  opposite with servo2
}
```

BOM

BOM level	Part name	Description	Quantity	Cost
1	Acrylic sheet+ cutting	For making kit of robotic arm	-	2000+1300=3300
2	ESP 32	Microcontroller with Wi-Fi and Bluetooth	1	1300
3	Charger	For power supply charger of 5 volt	1	500
4	Servo Motors	For the movement of joints and controlling them accurately	1*5	1000*5=5000
5	Gripper	For picking up and gripping objects	1	1000
6	Jumper wires	For making electrical connections	-	100
7	Nuts and spacers	For assembling and connecting the parts of the robotic arm	-	500
			Total	11700.

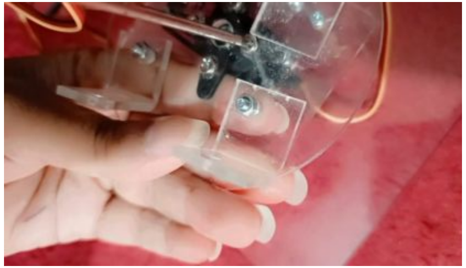
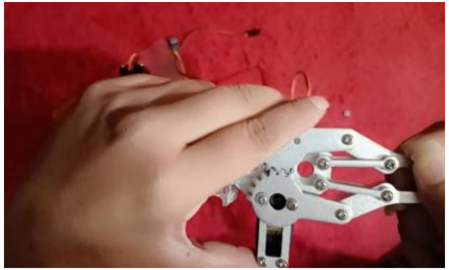
3.

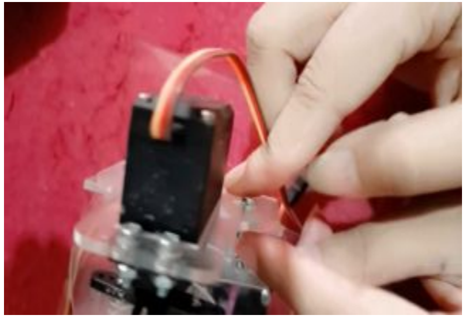
Wiring Design and Physical Assembly

Palwasha Bibi
2021-BEE-015

Hardware Development

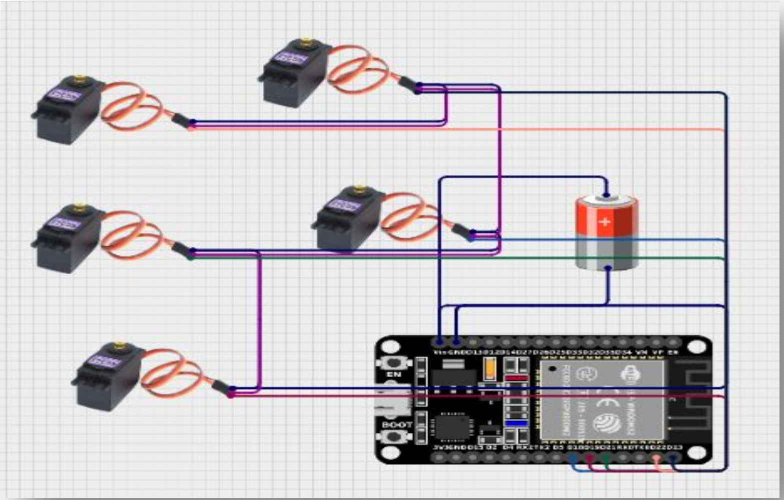
- Assembling of parts

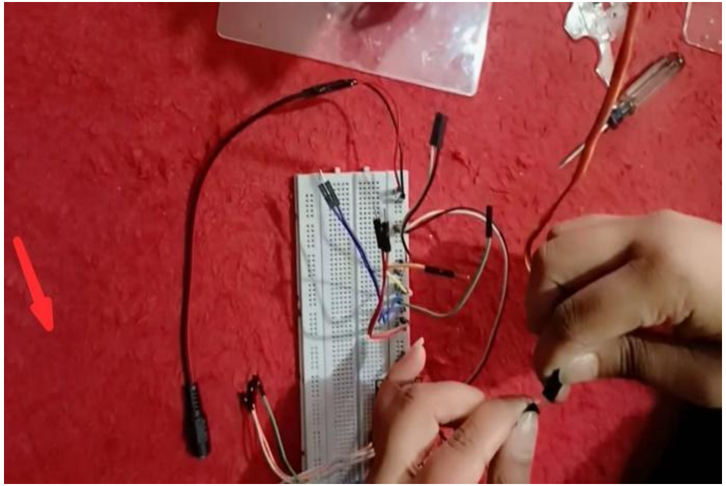






Circuit Design

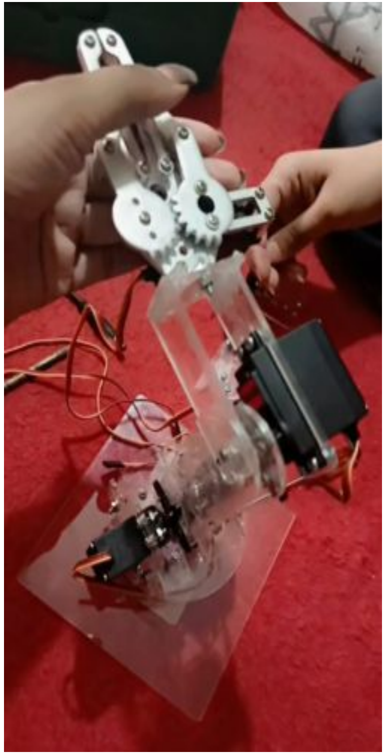
- Software: app.cirkitdesigner

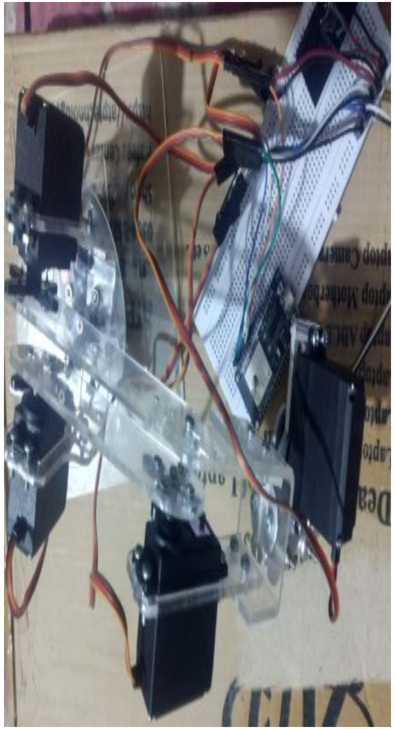




Mechanical Assembly

- Base– Servo-1
- Left side of Joint2 – Servo-2
- Right side of Joint 2 –Servo-3
- Joint 3 – Servo-4
- Gripper – Servo-5
- ESP 32 –Placed on breadboard, centralized all connections





4. Kinematics Implementation

Ayesha Shabbir Mirza
2021-BEE-004

System Overview

- Base Rotation (θ1)
- Shoulder (θ2)
- Elbow (θ3)
- Gripper is not modeled in Kinematics
- Link Lengths:
  - L1 = 100mm
  - L2 = 65mm

Coordinate System and Movements

- Base Rotation (θ1)
  - Rotates in XY plane
- Shoulder (θ2) and Elbow (θ3)
  - Moves arm in XZ plane
- Gripper/End-Effector
  - It’s position is calculated in 3D (X,Y,Z)

DH Table

Joint i	ai-1 (mm)	αi-1 (°)	di (mm)	θi	Comment
1	0	90	0	θ1	Base rotation
2	100	0	0	θ2	Shoulder (Link 1)
3	65	0	0	θ3	Elbow (Link 2)

Function 1 – DH_TRANSFORM.m

- A 4x4 matrix is created by this function
  - function A = DH_TRANSFORM(a, alpha, d, theta)
- Trigonometry is used to calculate the effect of each joint on position in space

DH_TRANSFORM.m × FKINE.m × IKINE.m × PLOT_ARM.m × test_accuracy.m ×

C:\Users\mirza\Downloads\MATLAB_KinematicsImplementation\DH_TRANSFORM.m

```
function A = DH_TRANSFORM(A_, ALPHA_DEG, D, THETA_DEG)
   ALPHA = deg2rad(ALPHA_DEG);
   THETA = deg2rad(THETA_DEG);
   A = [ cos(THETA), -sin(THETA)*cos(ALPHA), sin(THETA)*sin(ALPHA), A_*cos(THETA);
         sin(THETA),  cos(THETA)*cos(ALPHA), -cos(THETA)*sin(ALPHA), A_*sin(THETA);
         0,           sin(ALPHA),             cos(ALPHA),             D;
         0,           0,                      0,                      1 ];
end
```

Function 2 – FKINE.m (Forward Kinematics)

- Input: Joint angles θ_1, θ_2, θ_3
- Output: End-effector position (X, Y, Z)
- How it works:
  - Uses 3 DH matrices
  - Multiplies them to get the final position

DH_TRANSFORM.m × FKINE.m × IKINE.m × PLOT_ARM.m × test_accuracy.m ×

C:\Users\mirza\Downloads\MATLAB_KinematicsImplementation\FKINE.m

```
function [T_04, POS] = FKINE(THETA_1, THETA_2, THETA_3)
    L_1 = 100;
    L_2 = 65;
    A_1 = DH_TRANSFORM(0, 90, 0, THETA_1);
    A_2 = DH_TRANSFORM(L_1, 0, 0, THETA_2);
    A_3 = DH_TRANSFORM(L_2, 0, 0, THETA_3);
    T_04 = A_1 * A_2 * A_3;
    POS = T_04(1:3, 4);
end
```

Function 3 – IKINE.m (Inverse Kinematics)

- Input: Target position (X, Y, Z)
- Output: Joint angles θ_1, θ_2, θ_3
- Uses trigonometry (law of cosines)
- Calculates:
  - Base angle using atan2(Y, X)
  - Shoulder & elbow using geometry
- Checks if the point is reachable

DH_TRANSFORM.m × FKINE.m × IKINE.m × PLOT_ARM.m × test_accuracy.m ×

C:\Users\mirza\Downloads\MATLAB_KinematicsImplementation\IKINE.m

```
function [THETA_1, THETA_2, THETA_3, SUCCESS] = IKINE(X, Y, Z)
    L_1 = 100;
    L_2 = 65;
    THETA_1 = atan2(Y, X);
    R = hypot(X, Y);
    S = Z;
    D = (R^2 + S^2 - L_1^2 - L_2^2) / (2 * L_1 * L_2);
    if abs(D) > 1
        SUCCESS = false;
        THETA_1 = 0; THETA_2 = 0; THETA_3 = 0;
        return;
    end
    THETA_3 = atan2(sqrt(1 - D^2), D);
    K_1 = L_1 + L_2 * cos(THETA_3);
    K_2 = L_2 * sin(THETA_3);
    THETA_2 = atan2(S, R) - atan2(K_2, K_1);
    SUCCESS = true;
    THETA_1 = rad2deg(THETA_1);
    THETA_2 = rad2deg(THETA_2);
    THETA_3 = rad2deg(THETA_3);
end
```

Function 4 – PLOT_ARM.m

- Uses FK to find positions of joints
- Plots the robot in 3D
- Marks the end-effector
- Shows where the target is

DH_TRANSFORM.m × FKINE.m × IKINE.m × PLOT_ARM.m × test_accuracy.m × +

C:\Users\mirza\Downloads\MATLAB_KinematicsImplementation\PLOT_ARM.m

```matlab
function PLOT_ARM(THETA_1, THETA_2, THETA_3, AX)
    L_1 = 100;
    L_2 = 65;
    A_1 = DH_TRANSFORM(0, 90, 0, THETA_1);
    A_2 = DH_TRANSFORM(L_1, 0, 0, THETA_2);
    A_3 = DH_TRANSFORM(L_2, 0, 0, THETA_3);
    T_0 = eye(4);
    P_0 = T_0(1:3, 4);
    T_1 = T_0 * A_1; P_1 = T_1(1:3, 4);
    T_2 = T_1 * A_2; P_2 = T_2(1:3, 4);
    T_3 = T_2 * A_3; P_3 = T_3(1:3, 4);
    PTS = [P_0, P_1, P_2, P_3];
    plot3(AX, PTS(1,:), PTS(2,:), PTS(3,:), '-o', 'LineWidth', 2, 'MarkerSize', 6);
    hold(AX, 'on');
    plot3(AX, P_3(1), P_3(2), P_3(3), 'rs', 'MarkerSize', 8, 'MarkerFaceColor', 'r');
    hold(AX, 'off');
    xlabel(AX, 'X Axis (mm)');
    ylabel(AX, 'Y Axis (mm)');
    zlabel(AX, 'Z Axis (mm)');
    title(AX, '3-DOF Robot Arm Configuration');
    grid(AX, 'on');
    axis(AX, 'equal');
    xlim(AX, [-200, 200]);
    ylim(AX, [-200, 200]);
    zlim(AX, [-50, 200]);
end
```

Function 5 – test_accuracy.m

- Tests accuracy of FK and IK.
- Steps:
  - Define 5 test target points
  - Use IK to get joint angles
  - Use FK to get actual position
  - Compare with target
  - Calculate error
  - Plot arm and target in subplots

DH_TRANSFORM.m × FKINE.m × IKINE.m × PLOT_ARM.m × test_accuracy.m × +

C:\Users\mirza\Downloads\MATLAB_KinematicsImplementation\test_accuracy.m

```matlab
clc; clear;
TARGETS = [...
    165, 0, 0;
    100, 50, 20;
    80, -40, 30;
    50, 10, 60;
    120, 0, -10;
];
N_POSES = size(TARGETS, 1);
ERRORS = zeros(N_POSES, 1);
figure('Name', 'FK/IK Accuracy', 'Position', [100, 100, 1200, 700]);
for I = 1:N_POSES
    X = TARGETS(I, 1);
    Y = TARGETS(I, 2);
    Z = TARGETS(I, 3);
    [TH1, TH2, TH3, SUCCESS] = IKINE(X, Y, Z);
    if ~SUCCESS
        fprintf('Pose %d: Target [%.1f, %.1f, %.1f] out of reach!\n', I, X, Y, Z);
        ERRORS(I) = NaN;
        continue;
    end
    [~, POS_EST] = FKINE(TH1, TH2, TH3);
    X_HAT = POS_EST(1); Y_HAT = POS_EST(2); Z_HAT = POS_EST(3);
    ERRORS(I) = norm([X - X_HAT, Y - Y_HAT, Z - Z_HAT]);
    AX = subplot(2, 3, I);
    PLOT_ARM(TH1, TH2, TH3, AX);
    hold(AX, 'on');
    plot3(AX, X, Y, Z, 'mx', 'MarkerSize', 12, 'LineWidth', 2);
    hold(AX, 'off');
    title(AX, sprintf('Pose %d: Error = %.2f mm', I, ERRORS(I)));
    xlabel(AX, 'X (mm)\newline');
    ylabel(AX, 'Y (mm)\newline');
    zlabel(AX, 'Z (mm)');
```

```matlab
    view(AX, [45 30]);
end
fprintf('\nPose  Error (mm)\n');
for I = 1:N_POSES
    fprintf('%2d    %7.2f\n', I, ERRORS(I));
end
fprintf('Mean Error = %.2f mm\n', nanmean(ERRORS));
```

Output

- Each subplot:
  - Blue arm: actual configuration
  - Red square: end-effector
  - Magenta 'X': target point
  - Title shows error (e.g., 0.00 mm)
- Interpretation:
  - If red square touches magenta 'X', it’s accurate!

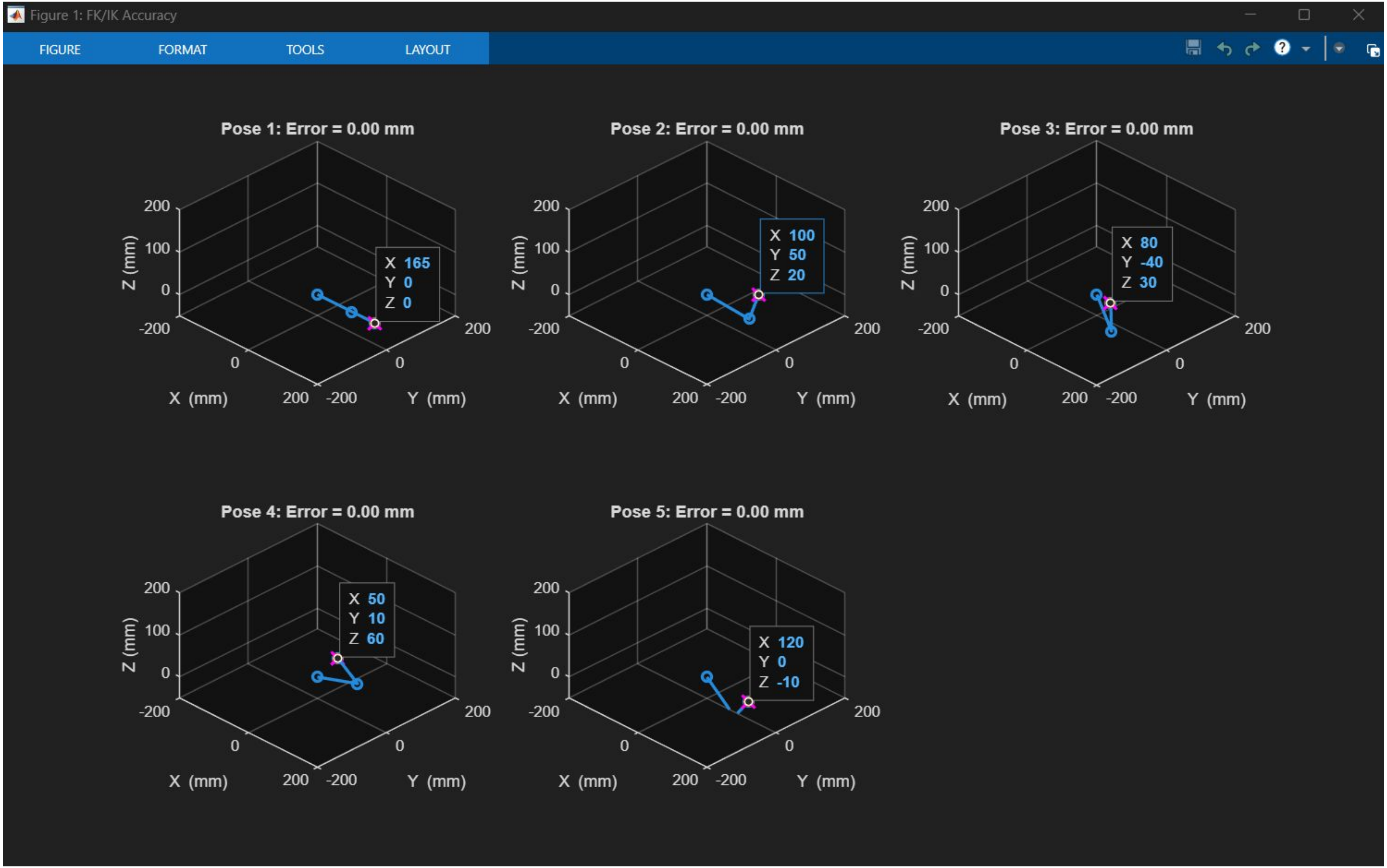


Figure 1: FK/IK Accuracy
FIGURE
FORMAT
TOOLS
LAYOUT
Pose 1: Error = 0.00 mm
X 165
Y 0
Z 0
Pose 2: Error = 0.00 mm
X 100
Y 50
Z 20
Pose 3: Error = 0.00 mm
X 80
Y -40
Z 30
Pose 4: Error = 0.00 mm
X 50
Y 10
Z 60
Pose 5: Error = 0.00 mm
X 120
Y 0
Z -10
Z (mm)
X (mm)
Y (mm)
200
100
0
-200

Summary of Results

Pose	Target (X, Y, Z)	Error (mm)
1	(165, 0, 0)	0.00
2	(100, 50, 20)	0.00
3	(80, -40, 30)	0.00
4	(50, 10, 60)	0.00
5	(120, 0, -10)	0.00

5.

MATLAB GUI Development

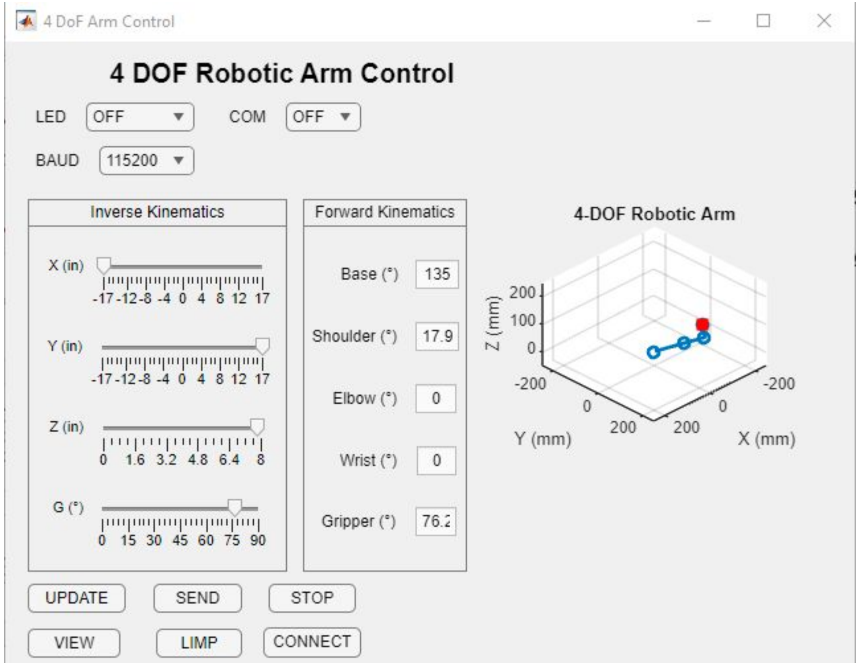
Shanza Noor
2021-BEE-016

MATLAB GUI DEVELOPMENT

- Part of the project is to design and build a MATLAB-based GUI to oversee the movements of a 4 DOF robotic arm. Real-time control, 3D imaging, and the use of user input are included in the system.

System Overview

- I have used MATLAB App Designer for design purposes.



GUI Features

- Sliders are found in the Inverse Kinematics system (X, Y, Z, Gripper).
- Fields for storing FK input that use numbers (Joint angles)
- These operations are accessed by the buttons SEND, STOP, UPDATE, and VIEW.
- There is additional support for 3D Arm Plot (ArmAxes).
- Seeing a 3D armor helps a lot.

Plots using the different 3D tools

- Changes can be made to settings as soon as they are saved.
- Describes the two countries' current areas of agreement
- Checks whether the IK & FK are working correctly

Challenges Faced

- Transferring what a user inputs to the right position of the robot arm
- Speedy response from the graphical user interface with several functions
- Concern yourself with how well your movements follow all of the expectations

6.

Integration

Afshan Bibi
2021-BEE-001

System Overview

- Integrated ESP32 with MATLAB GUI
- Controlled 5 servo motors using sliders in the GUI
- Communication via UART (serial communication)
- Built in MATLAB App Designer

ESP32 Code & Upload

- Connected ESP32 to PC via USB
- Used Arduino IDE for programming
- Code written to send serial data (e.g., LED or temperature)
- Used `Serial.begin(9600);` for UART setup
- Uploaded code successfully to ESP32

MATLAB GUI Creation

- Opened App Designer in MATLAB
- Created GUI with buttons and dropdowns
- GUI features:
    - Connect to ESP32 (COM Port)
    - Read data from ESP32
- Added serial communication functions

Errors in Execution

- Serial object not recognized in callbacks
- Dropdowns & components reported as "unrecognized"
- GUI structure had undeclared or misused elements
- Reserved keywords caused syntax errors
- GUI could not run properly due to these issues

7.

Testing

Muneeba Bibi
2021-BEE-013

Testing Lead

- Assigned role: **Testing Lead**
- Purpose: Test system functionality post-integration
- Couldn’t perform testing as integration was incomplete
- Supported the integration process actively

My Contributions

- Designed the **project poster**
- **Compiled and formatted** all team documents
- Ensured the final submission was neat and complete
- Helped ensure the team's effort was well-represented

Conclusion

- The integration of ESP32 with MATLAB GUI faced technical issues like undeclared variables and component errors.
- Due to incomplete integration, the system could not be tested as planned.
- The project helped us to learn about GUI development, serial communication, and teamwork.

Any Question?

Thank You!