Robotic Arm USB Mouse Tester Documentation

Contents

1.0 User Manual

****Note****

1. The test takes around 7 minutes per surface.
2. Method of moving robotic arm
    a. Set Polar (Input desired position based on coordination)
    b. Set Position (Input desired position based on coordination)
    c. Set Servo (Input desired position based on servo motor rotation angle)
3. Type of servo (Set Servo Method):
    a. Program mainly use this to avoid moving in trajectory path which cause more uncertainties.
    b. Servo[0]: Rotation about Y-axis
    c. Servo[1]: Stretch about X-axis
    d. Servo[2]: Change the height of the end-effector

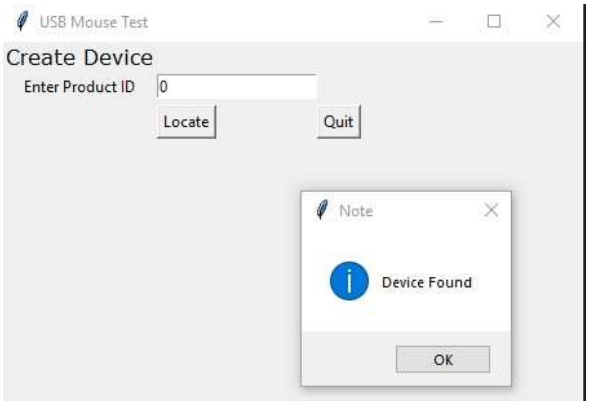
****Hardware Setup****

1. Place the surface mats (1-4) for the test before proceeding with the program.
2. Refer **Appendix[3]** for surface cutting and alignment.
3. The USB mouse holder is 90° perpendicular to the surface and tighten the screw at the end effector of the robotic arm to secure the mouse position.
4. Switch on the power supply.
5. Set the initial height of the end-effector and locate the initial angle of Servo[2]:
    a) Run program **'Run_wtGUi'** (manual reset robotic arm position)
    b) Run **'Reset()' function** (Servo[0] and Servo[1] at 90°, Servo[2] at 5°). The mouse should be positioned at the center after running the program.
    c) Slowly lower the end-effector until the USB mouse position 0mm from the surface (Do not press the mouse too much on the surface as this will hinder the movement of the mouse)
    d) Method: **Use [swift.set_servo_angle(servo_id=2,angle=θ,speed=10000)] and change angle θ**:
        i. Decrease θ to increase the height
        ii. Increase θ to lower the mouse
    e) Optimum position:
        i. When mouse is not hardly press on the test surfaces and 0mm off the surface
        ii. Test the position by using a piece of paper (sticky note) with thickness about 0.1mm and insert it under the mouse. The position is correct when user unable to insert the paper under the mouse.
    f) Once achieved the optimum position, observe angle θ and **update initial angle** in the main program 'Run_GUi' (refer 'Code_note.txt' -> 1)

Software Setup

1. **Copy 'usbmouse_project' to local disk(C:)**
2. **Initialize USB mouse:**
    a. Go to [**Local Disk(C:)/usbmouse_project/LibMouse_TCL/tcl**] and find "**run**"
    b. Input: **Dolphin3 -> o -> w 00 01** (depend on the type of mouse)
    c. In case of new device (uninstalled device/driver), go to 'Filter Wizard' and install the required driver.
3. In case of fail to initialize the USB mouse such that program could no longer track and read from the USB mouse due to deactivation:
    a. Check the stats from 'Bus Hound' and try to scroll using the mouse.
    b. Move the mouse around and check the changes in the stats of the mouse.
4. **Initialize Robotic Arm:**
    a. Open Spyder and search program from File Explorer: [**usbmouse_project -> Code**]
        i. Run_GUi (Main Program)
        ii. Run_wtGUi (without GUI, for manual reset purpose)
5. Make sure the following folders and programs are available in your PC:
    a. Result_data (For data storage)
    b. Modules, libraries and the main program are in the correct location.
    c. In case of transferring program to a new PC, check the location of the directory and make sure that it is synchronized with the directory address in the main program.
6. There are issues when program fail to run:
    a. Exit the GUI application by click 'Quit' and rerun the program
    b. Reason is due to the failure in reading the class 'Functions', rerun the main program solve the issue. (Refer 'Code_note.txt' -> 2)

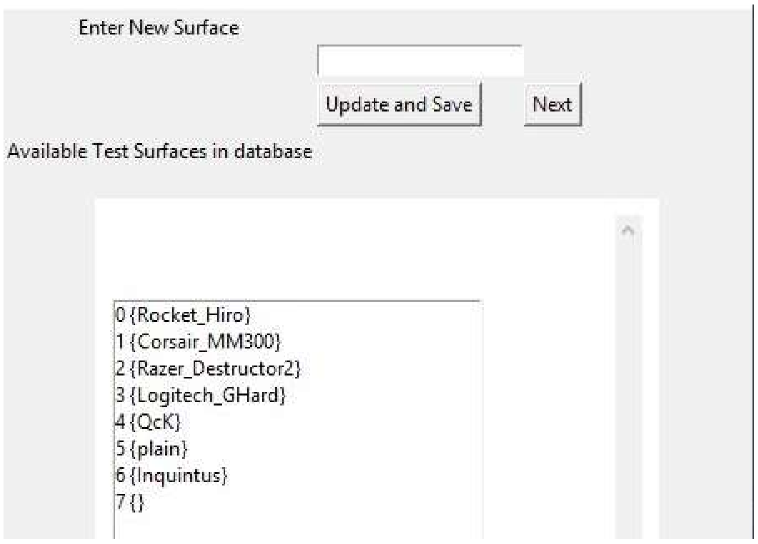
<u>**Step 1: Program activation and initialization**</u>



- ➢ Enter Product ID and click 'Locate'
- ➢ Output: 'Device Found' message box or 'Device not found' message box

Figure 1: Program display for Step 1

Step 2: Program update

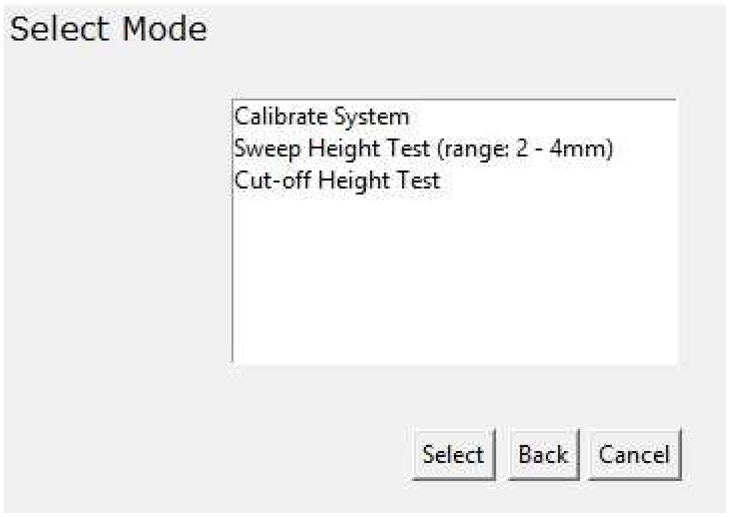


```
0 {Rocket_Hiro}
1 {Corsair_MM300}
2 {Razer_Destructor2}
3 {Logitech_GHard}
4 {QcK}
5 {plain}
6 {Inquintus}
7 {}
```

- Enter new surface and click 'Update and Save' (GUI)
- Click 'Next' to proceed

Figure 2: Program display for Step 2 (Exit update – Any key)

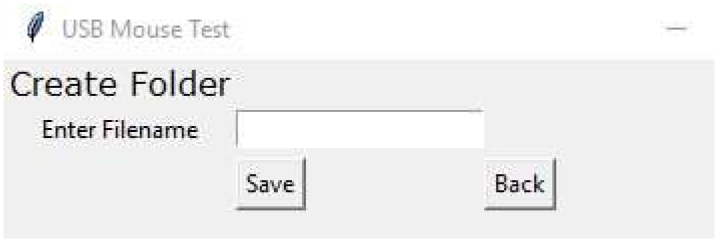
Step 3: select mode



- Select mode from the list and click 'Select'

Figure 3: Program display for Step 3 (mode selection)

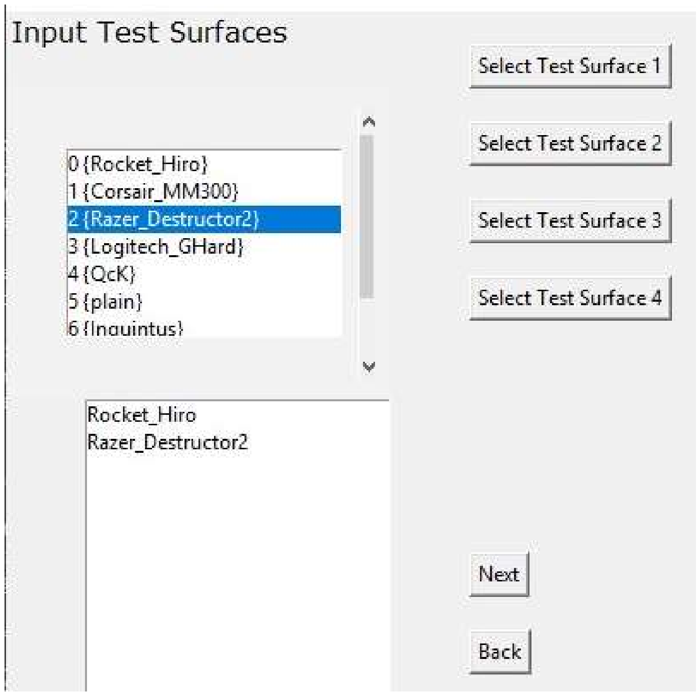
Step 4: Create new directory (Store result data)



- Enter Filename and click 'Save' (GUI)
- Only for mode 2 and 3

Figure 4: Program display for Step 4

Step 5: Input Test Surfaces



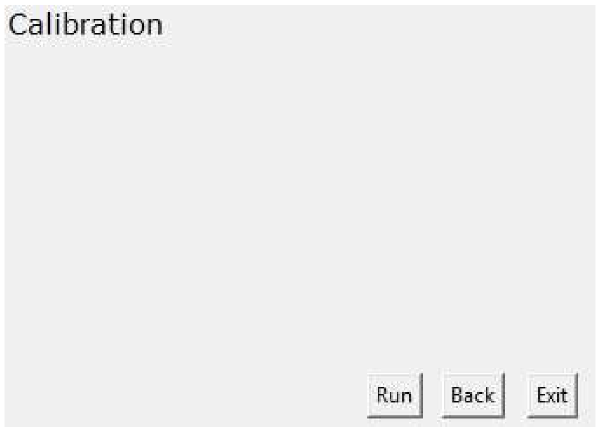
0 {Rocket_Hiro}
1 {Corsair_MM300}
2 {Razer_Destructor2}
3 {Logitech_GHard}
4 {QcK}
5 {plain}
6 {Inquintus}

Rocket_Hiro
Razer_Destructor2

- ➢ Select the test surface and click the respective 'Test Surface [i]'
- ➢ Refer the bottom frame on your selections
- ➢ Change the selection by select the desired surface from the list and click on the respective button.
- ➢ Click 'Next' to confirm
- ➢ Only for mode 2 and 3

Figure 5: Program display for Step 5 (Selected surfaces for test)

Step 6: Run selected mode



- ➢ Click 'Run'

Figure 6: Program display for Mode 1

Step 7: Program Output

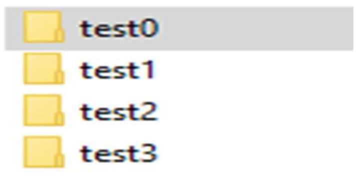


Figure 7: Program creates 4 subfolders to store data for the respective tests

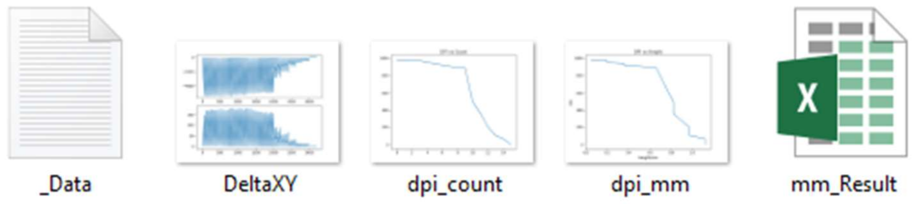
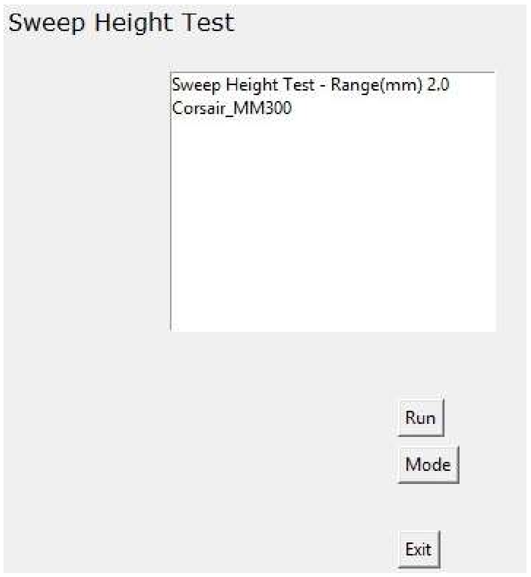
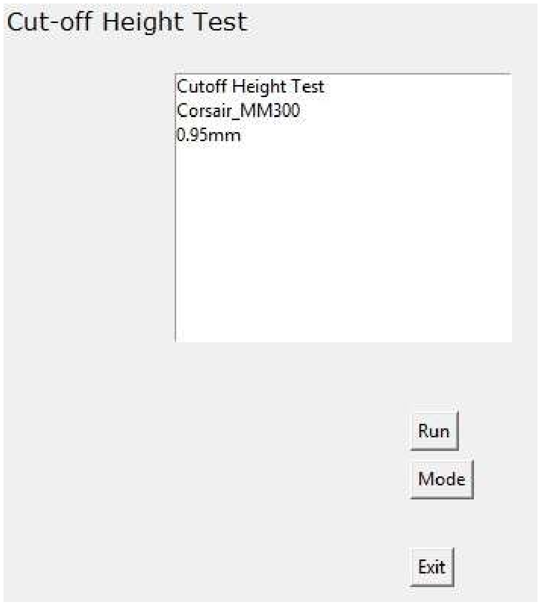


Figure 8: Program generate txt, plot and excel files in collecting the data and save them in the created subfolders.



```
Sweep Height Test - Range(mm) 2.0
Corsair_MM300
```

Figure 9: Program display output for Sweep Height Range Test



```
Cutoff Height Test
Corsair_MM300
0.95mm
```

Figure 10: Program display output for Cut-off Height Test

2.0 Flowchart

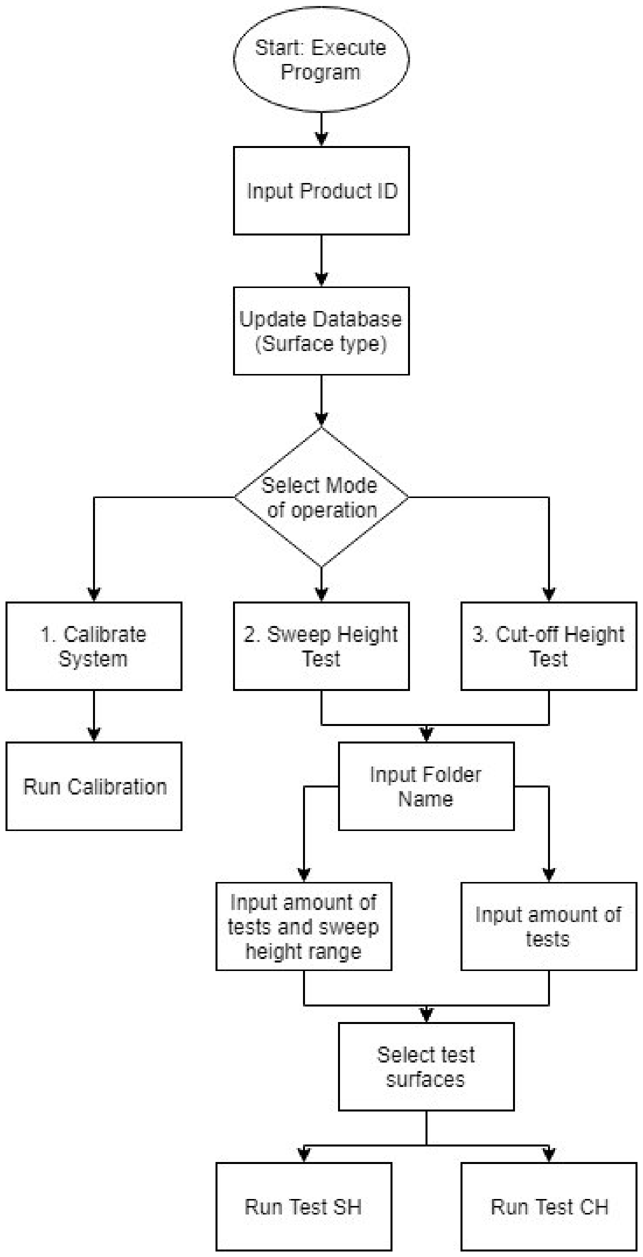


Figure 13: Overall flowchart on the system operation.

Note:

- ✓ Users select either go back to <Mode> selection or exit the program after completed the <Run> program or the tests.
- ✓ There are various limitations set such as the amount of tests, the sweep height range and so on. Do take note on the instructions.

3.0 Appendix

Default Running Conditions:

a. Motion Speed (Python speed):
   - Fixed parameter
   - Rotation angle transition = 30000 (~76mm/s)
   - Swing = 10000 (~25.4 mm/s)

b. Number of repetitions: 4 (swings)

c. Movement Distance: ~4 inches (per swing)

d. Optimum condition: Robot arm at 90° (servo[1])

Manipulated parameters:

e. Mode:
   - Calibrate:
     - To calibrate the robotic arm and make sure it moves to the desired position
   - Sweep Height Test:
     - Analyze the USB mouse output via sweep height
     - (2 – 4mm of surface)
     - Output: Plot
   - Cut-off Height Test:
     - Determine the USB mouse cut-off height
     - Output: Plot, Text, Excel

f. Number of tests: 1 – 4 (min=1, max=4)

g. Type of test surface:
   - Input the type of surface from the displayed database

Test Surface:

h. Cut-out size:

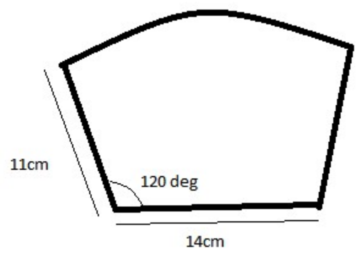


Figure 14: Surface Cutting

i. Thickness:
   - Optimum= 2mm
   - Add layers (paper, cardboard etc) to increase the thickness in the case of unequal surface thickness

System Accuracy

The robotic arm accuracy test was conducted by drawing various shapes of different dimensions such as circles, triangles and squares. The variations varied when the robotic arm conducted the tests at different height.

j. Circular path

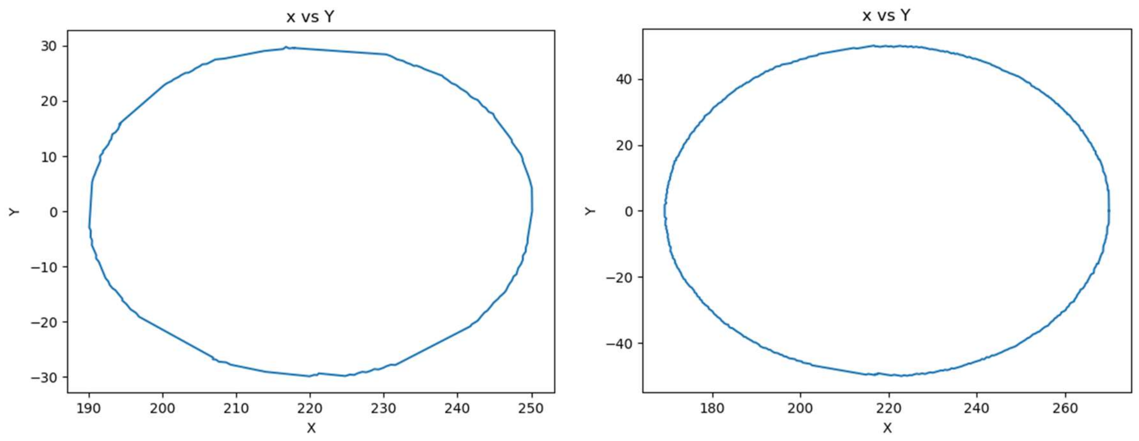


Figure 15: Circular path at radius 30 units and 50 units respectively.

The circle at radius 50 units is better defined than that of the circle at radius 30 units. This is due to the increase in the displacement per step (100 steps in total) of the circle with a greater radius which leads to the increase in the range of motion.

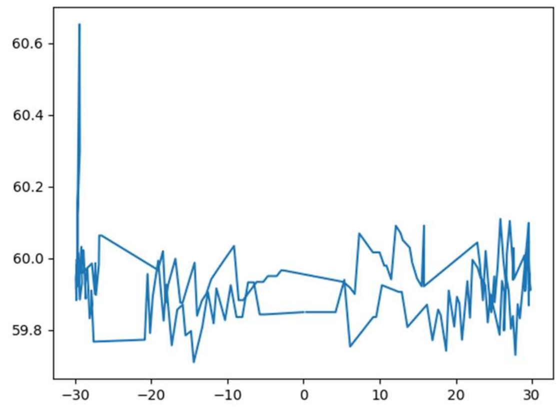
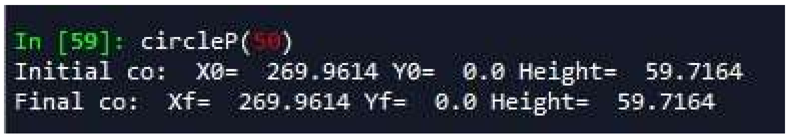


Figure 20: The z-axis (height) vs y-axis plotting.

The changes in height shows the arm travelled in a trajectory in order to reach the desired position from an initial position. The difference in height is averagely in a range of 0.4 units.



```
In [59]: circleP(50)
Initial co:  X0=  269.9614 Y0=  0.0 Height=  59.7164
Final co:  Xf=  269.9614 Yf=  0.0 Height=  59.7164
```

Figure 16: The initial position and the respective final position of the end-effector.

The above result shows that the end-effector returned to its initial coordination including the height which proved the accuracy of its positioning ability as the robotic arm met its initial position.

k. Square path

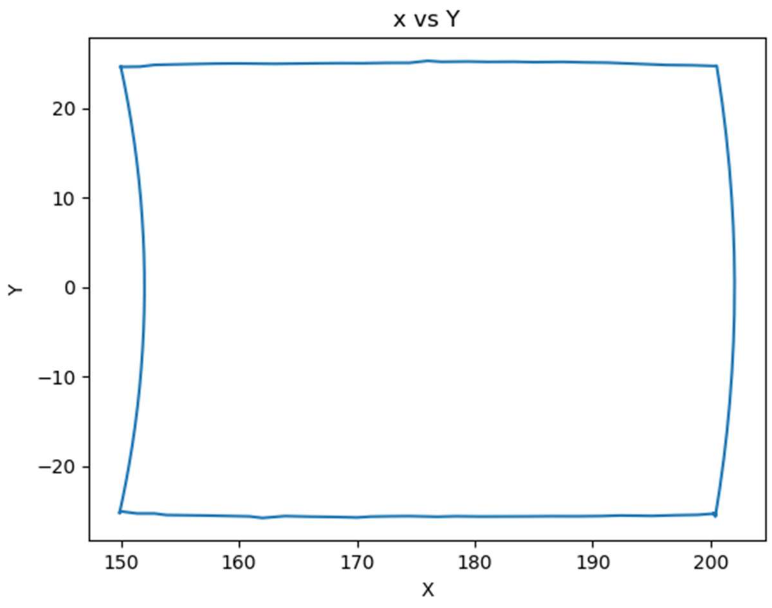


Figure 17: The square path constructed based on x-y coordination (design perimeter of 200 units).

The left line was slightly curved due to the trajectory path travelled by the robotic arm. To construct a straight line, divide the path into smaller steps.

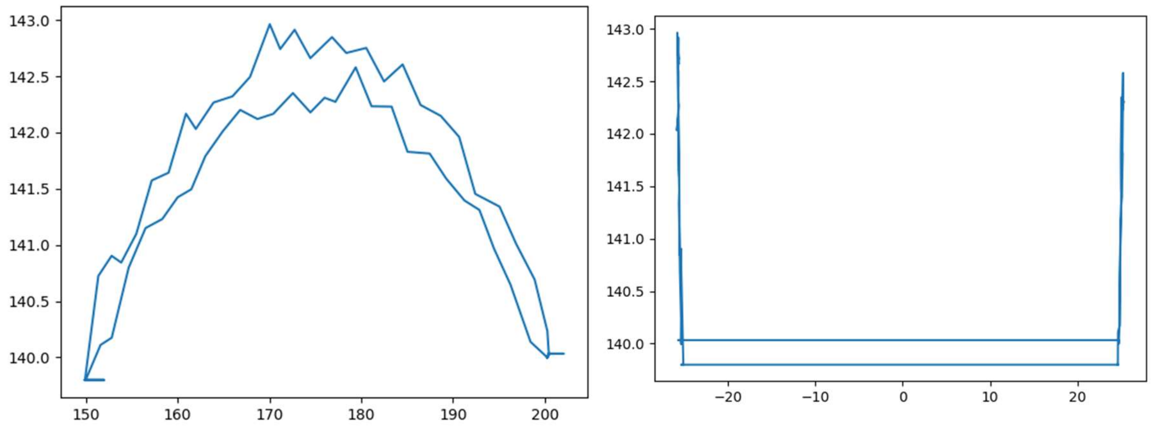
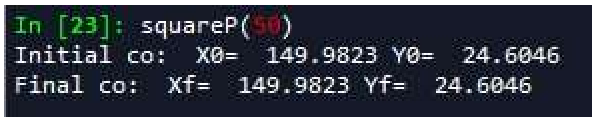


Figure 18: The changes in units of z-axis (height) in relative to x-axis (a) and y-axis(b).

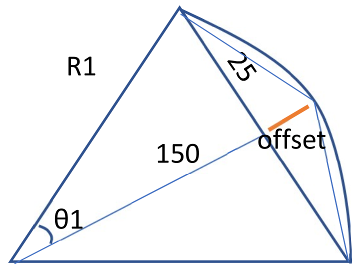
The changes in height shows the arm travelled in a trajectory path in order to reach the desired position from an initial position. The difference in height is averagely in a range of 3 units.



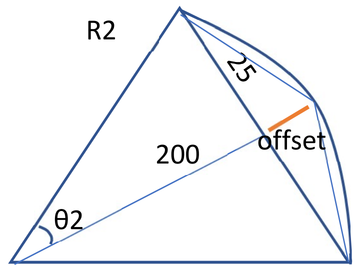
```
In [23]: squareP(50)
Initial co:  X0=  149.9823 Y0=  24.6046
Final co:  Xf=  149.9823 Yf=  24.6046
```

Figure 19: The initial and final position of the end-effector.

The robotic arm return to its initial position after completed the square without presetting the final coordination. The robotic arm is precise with the completion of the square however offset in construction of the perfect square existed due to the trajectory path travelled. The offset and error(perimeter) are calculated:



(a)



(b)

Figure 20 (a)&(b): Offset and error calculation

The square is supposed to be 50 units in term of length and width. 200 units in perimeter.

$$R1 = \sqrt{150^2 + 25^2} = 152.07\ units$$

$$\theta 1 = \ tan^{-1}\, {}^{25}\!/_{150} = 9.46°$$

$$arc1 = 2 \times {}^{9.46}\!/_{360} \times 2\pi \times 152.07 = 50.22 units$$

$$R2 = \sqrt{200^2 + 25^2} = 201.56\ units$$

$$\theta 2 = \ tan^{-1}\, {}^{25}\!/_{200} = 7.13°$$

$$arc2 = 2 \times {}^{7.13}\!/_{360} \times 2\pi \times 201.56 = 50.17 units$$

$$offset\ (\text{peak}) = 2.07 + 1.56 = 3.63 units$$

$$error(perimeter) = {}^{(200.39 - 200)}\!/_{200} \times 100\%$$

$$= 0.195\%$$

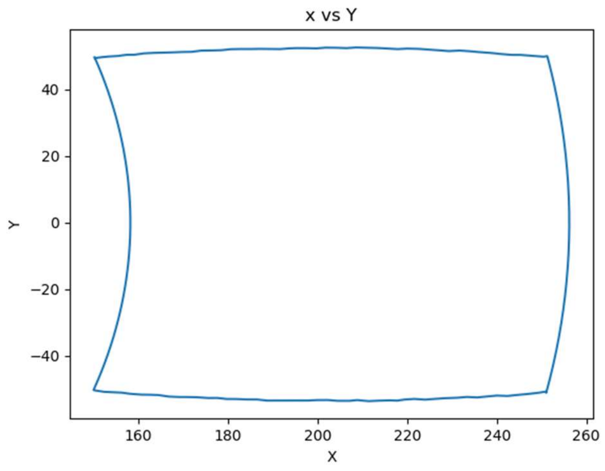
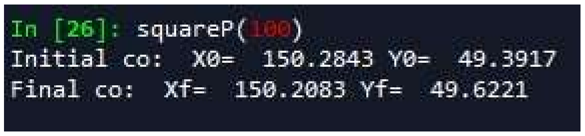


Figure 21: Square with design perimeter of 400 units.

By designing a square with greater perimeter, we observed there is an increase in the offset and error in perimeter. This is due to the increase in the trajectory path travelled. However, the robotic arm still able to return to its initial position.



```
In [26]: squareP(100)
Initial co:  X0=  150.2843 Y0=  49.3917
Final co:  Xf=  150.2083 Yf=  49.6221
```

Figure 22: The initial and final position of the end-effector.

I. Triangular path

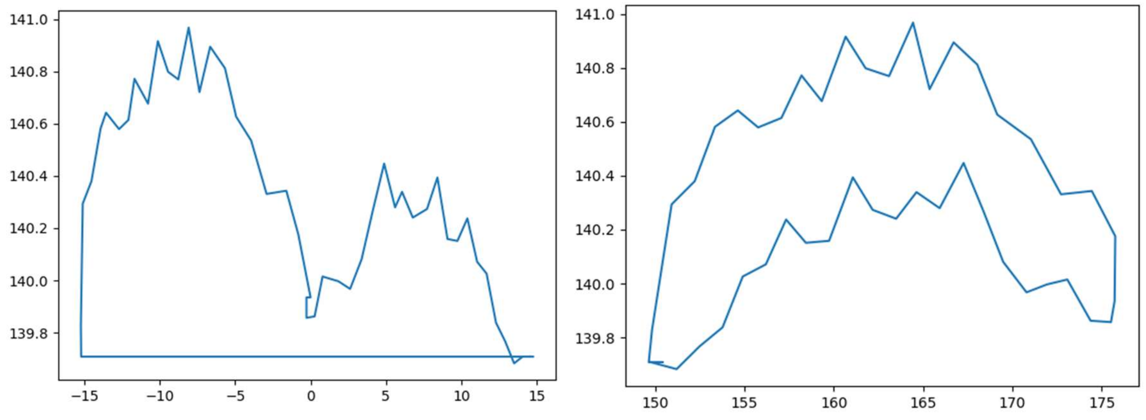


Figure 23: The changes in units of z-axis (height) in relative to x-axis (a) and y-axis(b).

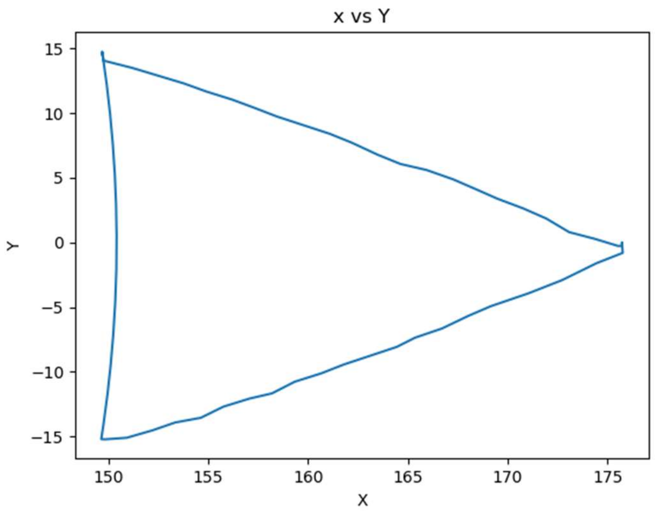
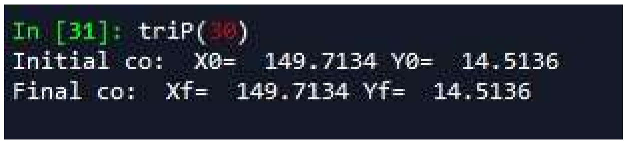


Figure 24: The triangular path constructed based on x-y coordination (design perimeter of 90 units).

The line was slightly curved due to the trajectory path travelled by the robotic arm due to the changes in height. To construct a straight line, divide the path into smaller steps.



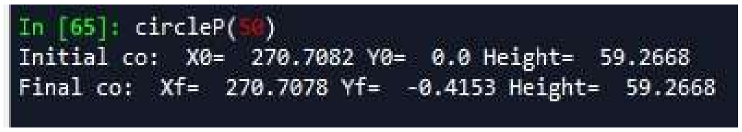
```
In [31]: triP(30)
Initial co:  X0=  149.7134 Y0=  14.5136
Final co:  Xf=  149.7134 Yf=  14.5136
```

Figure 25: The initial and final position of the end-effector.

System Consistency

The robotic arm consistency test was conducted by drawing various shapes of different dimensions such as circles, triangles and squares in 5 iterations.

m. Circular path



```
In [65]: circleP(50)
Initial co:  X0=  270.7082 Y0=  0.0 Height=  59.2668
Final co:  Xf=  270.7078 Yf=  -0.4153 Height=  59.2668
```

Figure 26: The initial and final position of the end-effector.

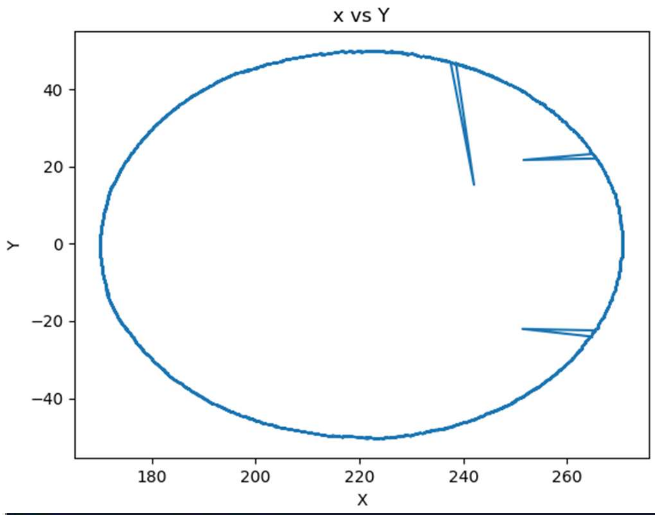
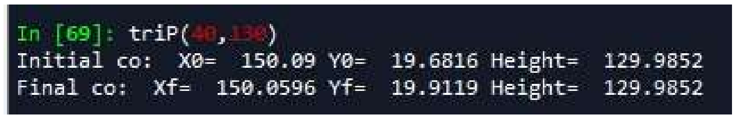


Figure 32: The circular path constructed based on x-y coordination (design radius 50 units).

One out of five of the circles has slight offset along the track.



```
In [69]: triP(40,130)
Initial co:  X0=  150.09 Y0=  19.6816 Height=  129.9852
Final co:  Xf=  150.0596 Yf=  19.9119 Height=  129.9852
```

Figure 27: The initial and final position of the end-effector.

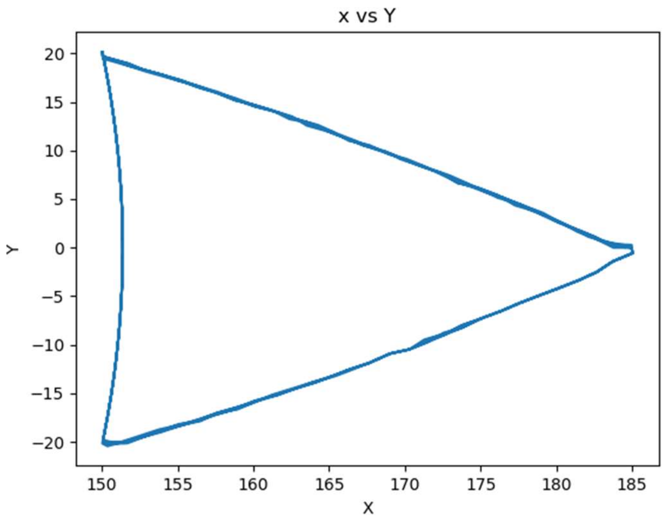


Figure 28: The triangular path constructed based on x-y coordination (design perimeter 120 units).

One out of five of the triangles has slight offset along the track.

4.0 Hardware

- **Uarm Swift Pro**
    a. Limitation:
        i. 0.2 units per step (unable to achieve 0.1mm accuracy)
        ii. 0.0879° per step for servo motor rotation
    b. 1 unit ~= 0.8*1mm
    c. Overview (max angle of rotation)

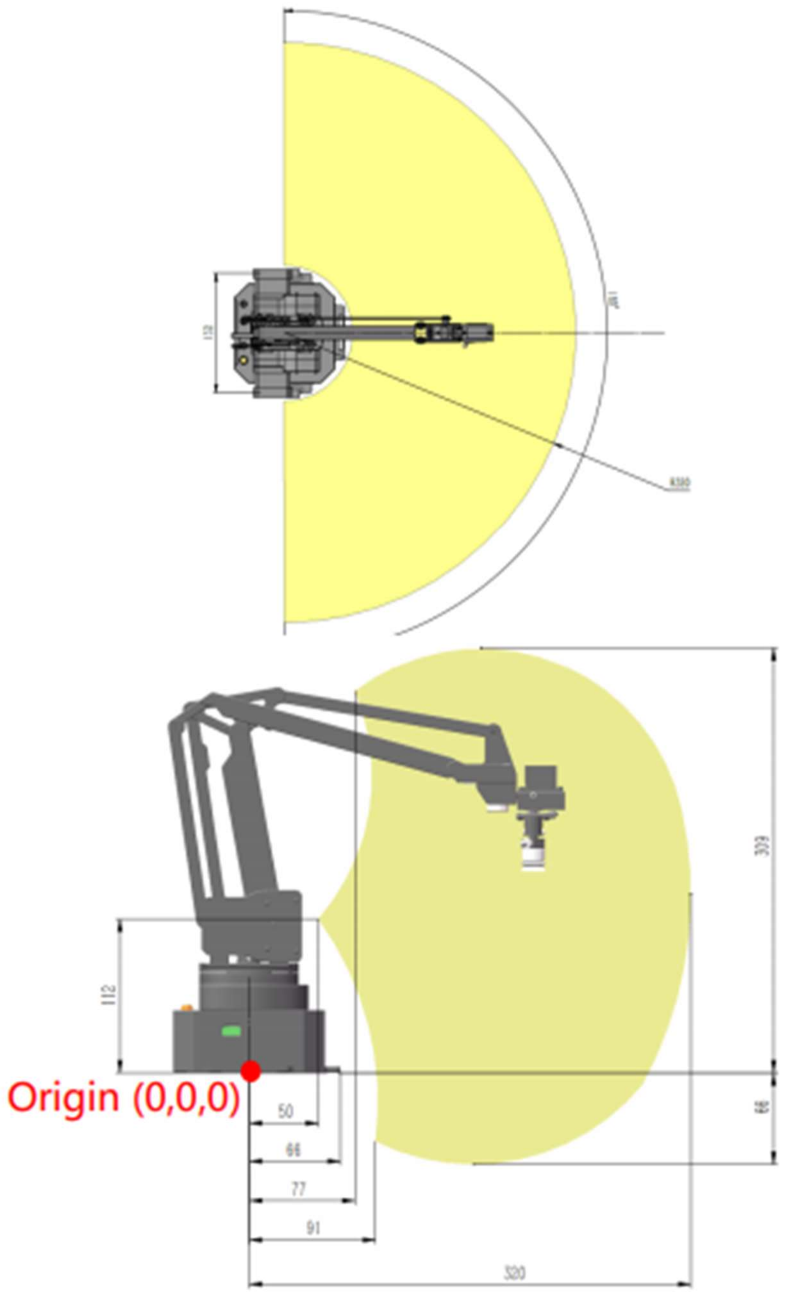


Figure 29: Uarm range of rotation (adapted from Uarm Swift Pro manual)

<u>The approximate unit for each step</u>

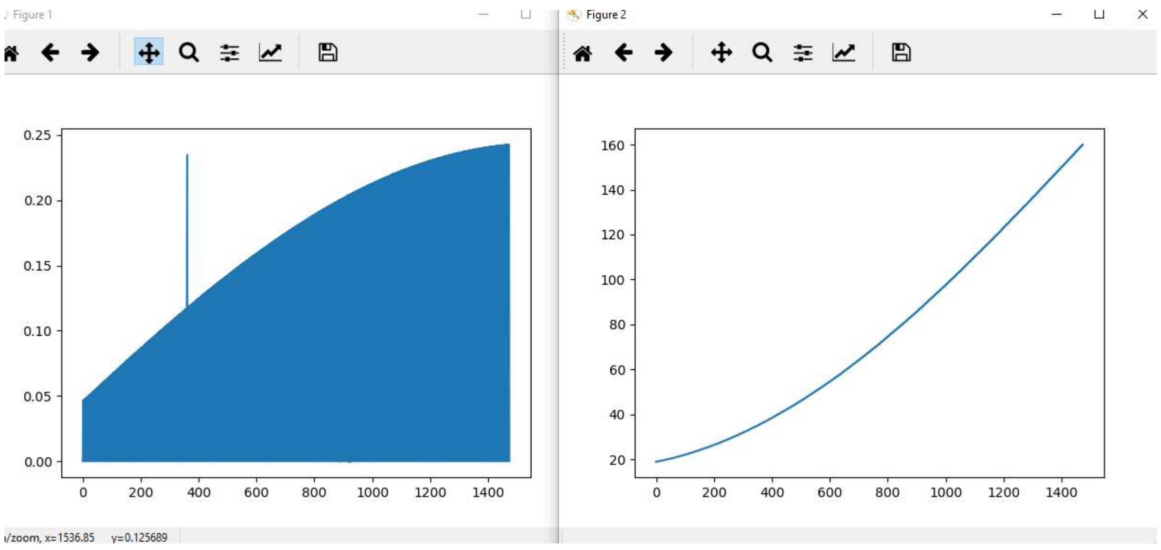


Figure 30: Increment of height of end effector by increasing the units at z-axis.

From the above figure, we observed that the robotic arm able to perform linear increment from 20 unit to 160 unit at z-axis with a slight blending (200 – 600 iterations). The robotic arm was programmed to increase height by 0.1 unit at z-axis. The height increased by 0.05 units initially and eventually increased by 0.24 units at z-axis. This is due to the changes in the position angle of Servo[1] as the robotic arm moved in a trajectory path.

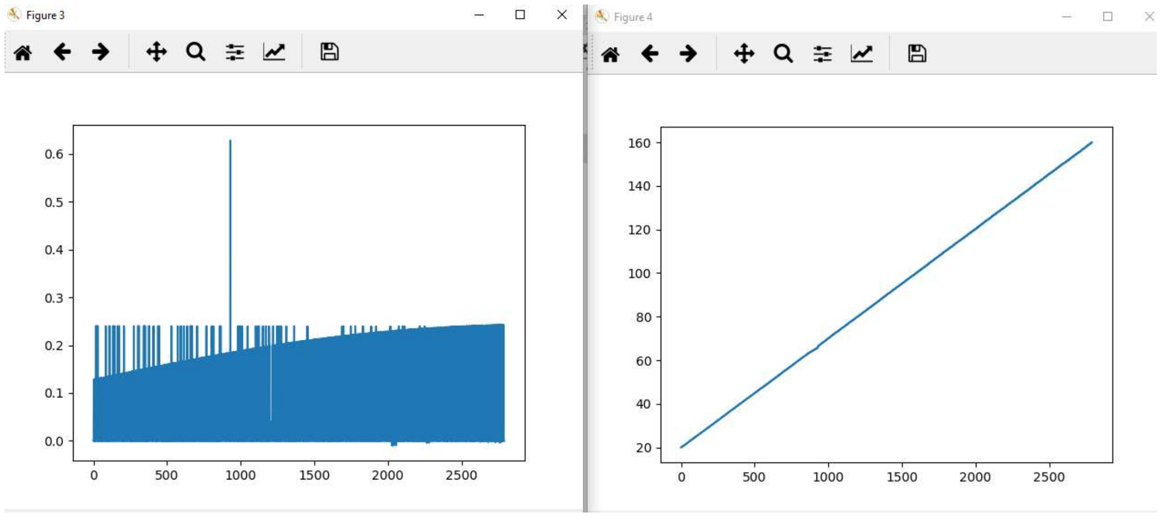


Figure 31: Increment of height of end effector by decreasing the position angle of Servo[2].

From the above figure, we observed that the robotic arm able to perform linear increment from 20 unit to 160 unit at z-axis. The robotic arm was programmed to increase height by decreasing the angle position of Servo[2] to 0.1 degree per iteration however due to its limitation, it could only increase by 0.24 unit at z-axis. The other servo motors were not involved in the operation. Hence, we concluded that the method of increasing the height by decreasing the position angle of Servo[2] is much more stable as compared to the first case.

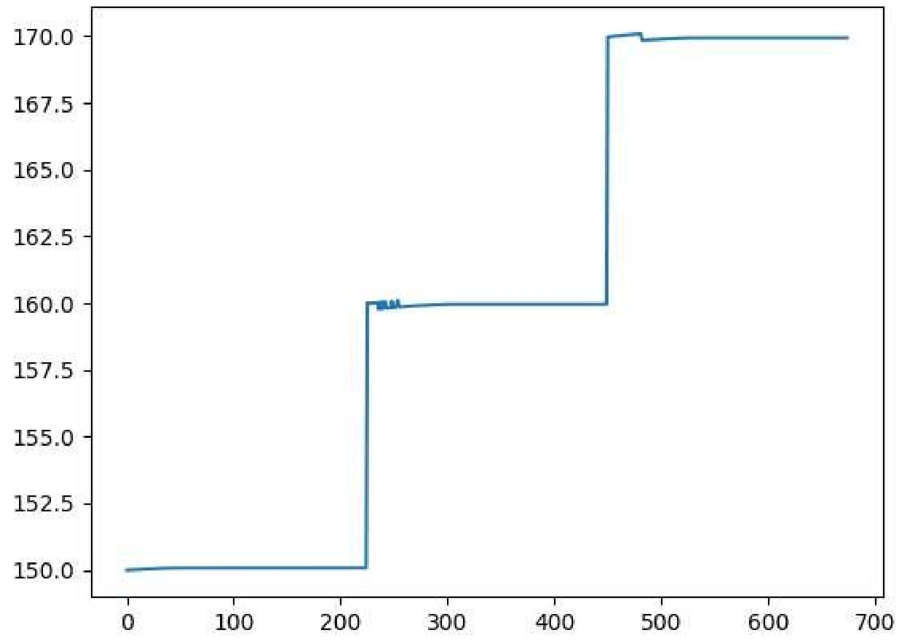


Figure 32: The Swing motion performed by robotic arm at 3 different height (150,160,170).

Figure 32 shows the changes in z-axis unit at the end effector when the robotic arm performed swing motion from left to right **with only Servo[0] working**. Servo[2] function to increase the height of the end effector. The plotting shows that the robotic arm could **perform a stable rotation from left to right (Swing motion)**. Minor instability in the initial iteration of the swing motion at height 160 and 170 units due to the minor shake when the arm started to move

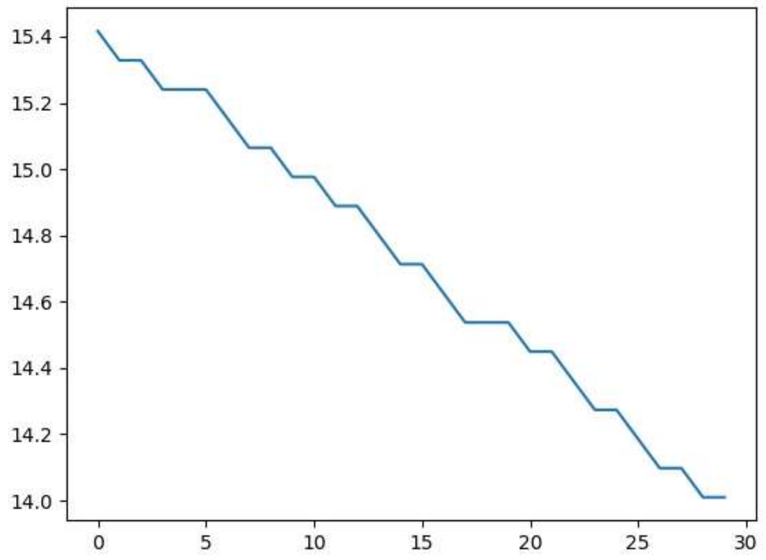


Figure 33: The decrement in angle rotation of Servo[2].

Result shows the decrement in angle rotation of Servo[2] exceed 0.1° .

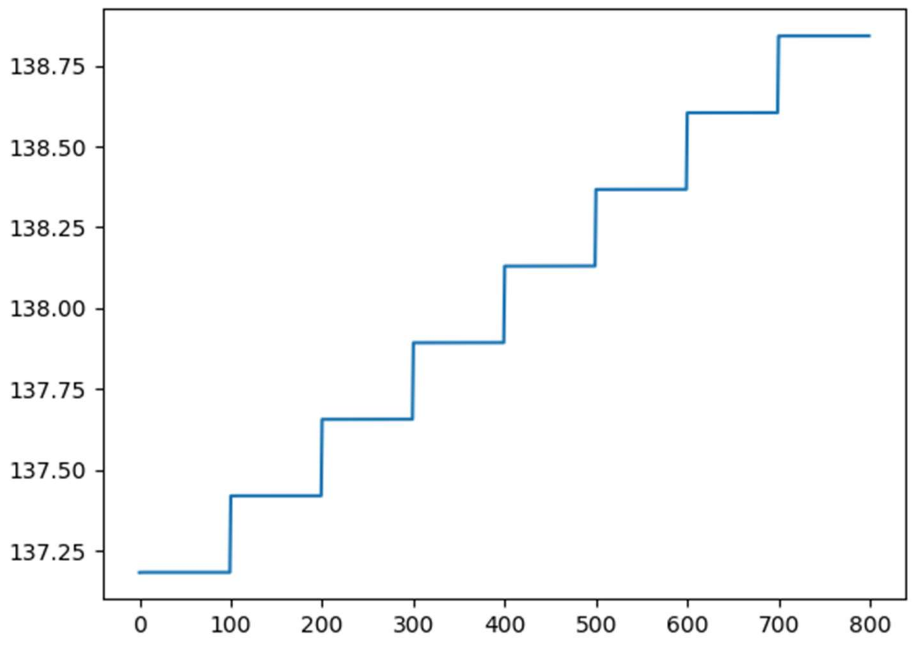


Figure 34: The Swing motion performed by robotic arm with initial height at 137.15 and increase by 0.23 units (appx. 0.19mm) per step.

Figure 34 shows the swing motion performed when the mouse is not pressed toward the surface, instead slightly placed on top of the surface with 0mm distance off the ground. Hence users should take note about this when setting up the USB mouse hardware.

5.0 Project Setup

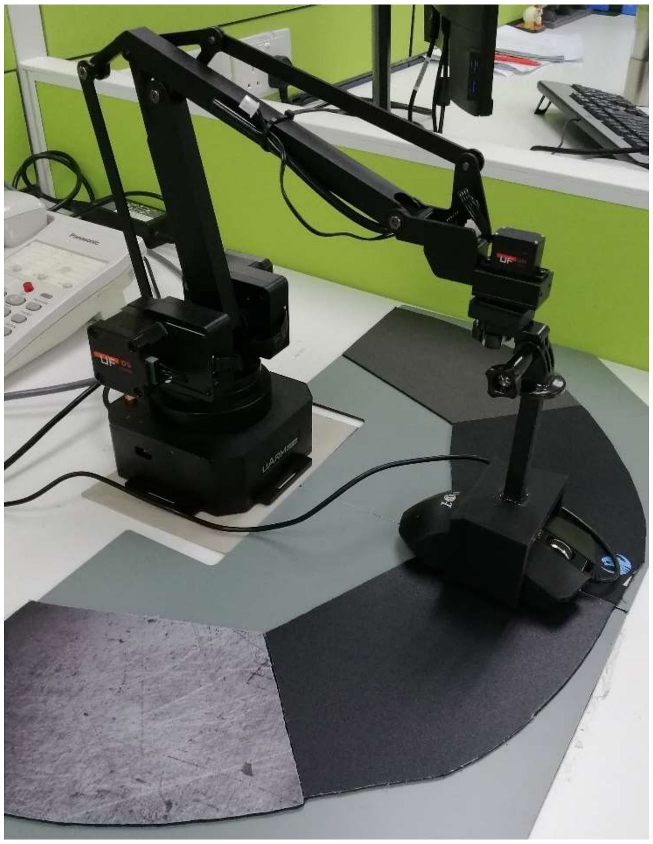


Figure 35: Project setup

- ✓ The test surfaces are to be arranged side by side and form a hemisphere.
- ✓ The USB mouse is to be at the center of the test surfaces as shown when the robotic arm is at 0 y-axis (Servo[0]=90°)

6.0 Test Result

****Dolphin3****

Sweep Height Range Test

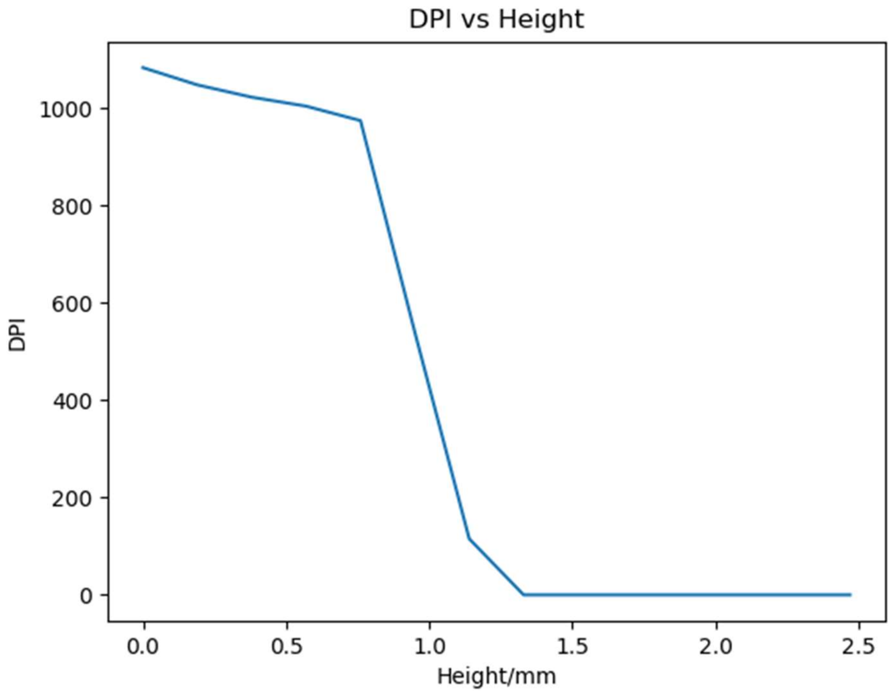


Figure 36: DPI vs Height (mm) plotting at sweep height range of 2.5mm

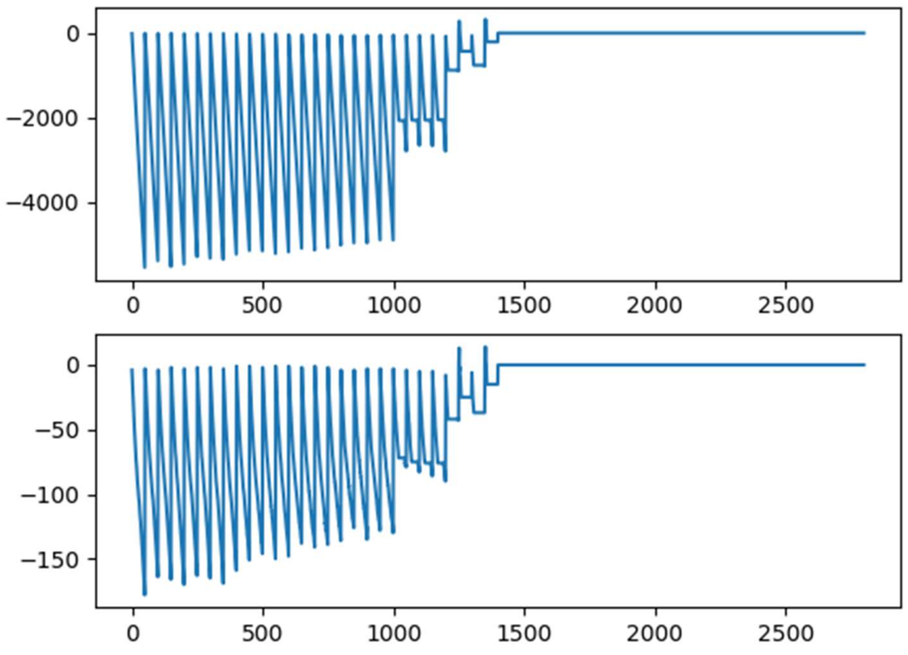
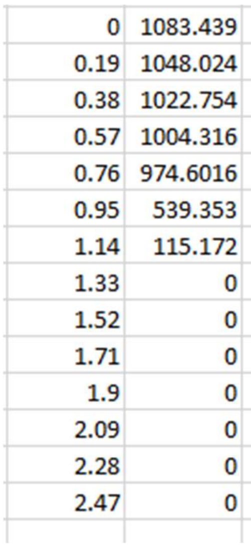


Figure 37: Delta X and Delta Y plotting

Plotting shows that the USB Mouse travelled for 5 inches per swing.



0	1083.439
0.19	1048.024
0.38	1022.754
0.57	1004.316
0.76	974.6016
0.95	539.353
1.14	115.172
1.33	0
1.52	0
1.71	0
1.9	0
2.09	0
2.28	0
2.47	0

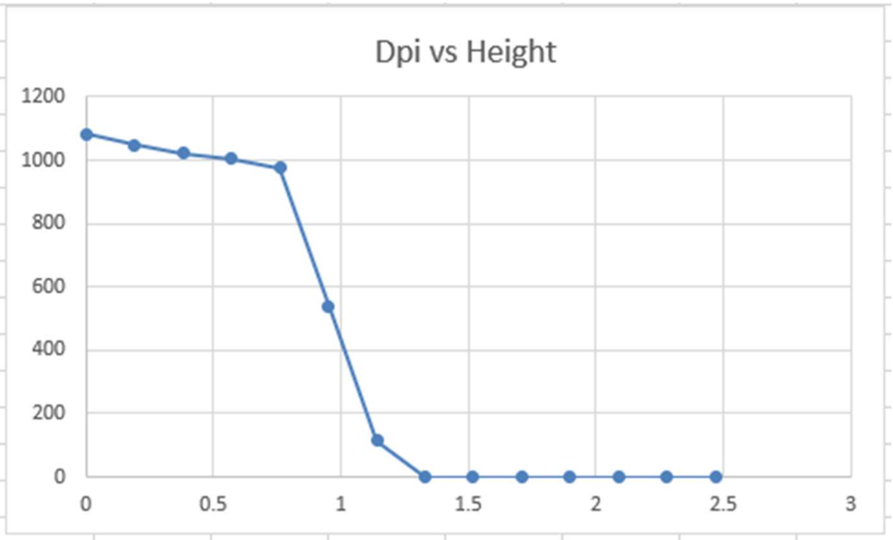


Figure 38: Data collected and generated in excel.

Cut-off Height Test

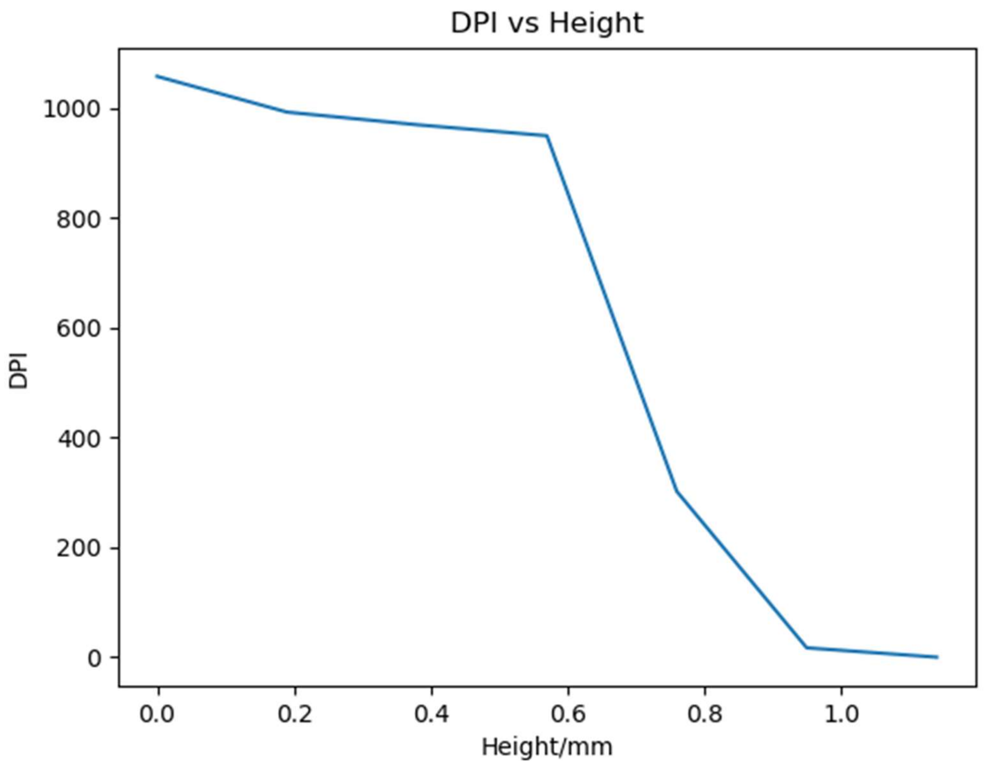


Figure 39: DPI vs Height (mm) plotting for lift-off height test at 1.16mm off ground

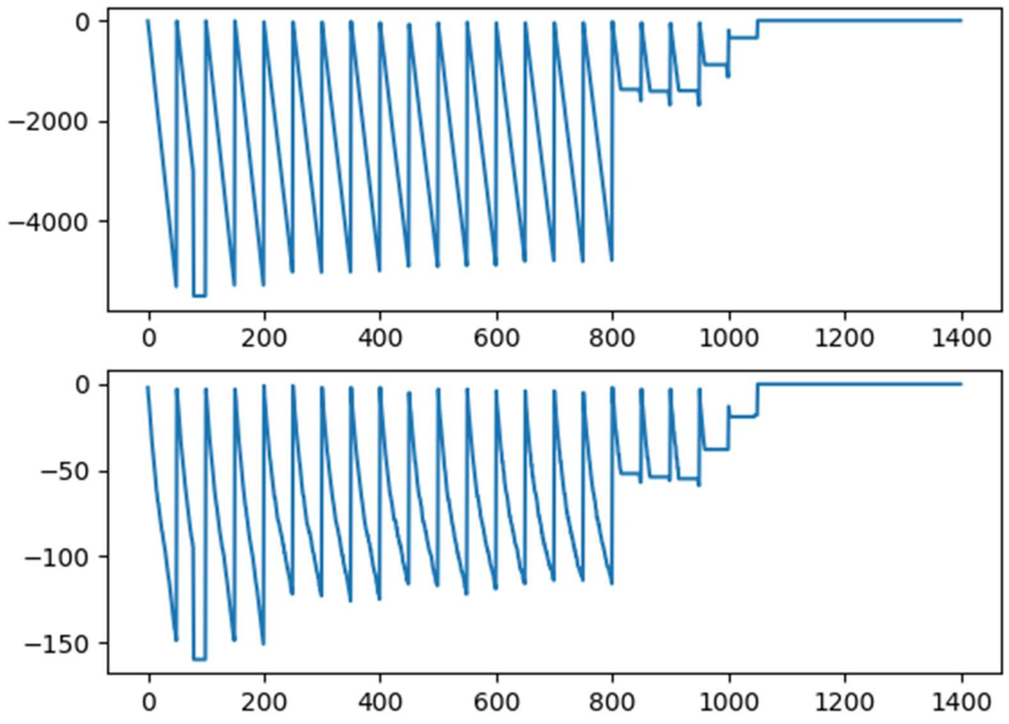
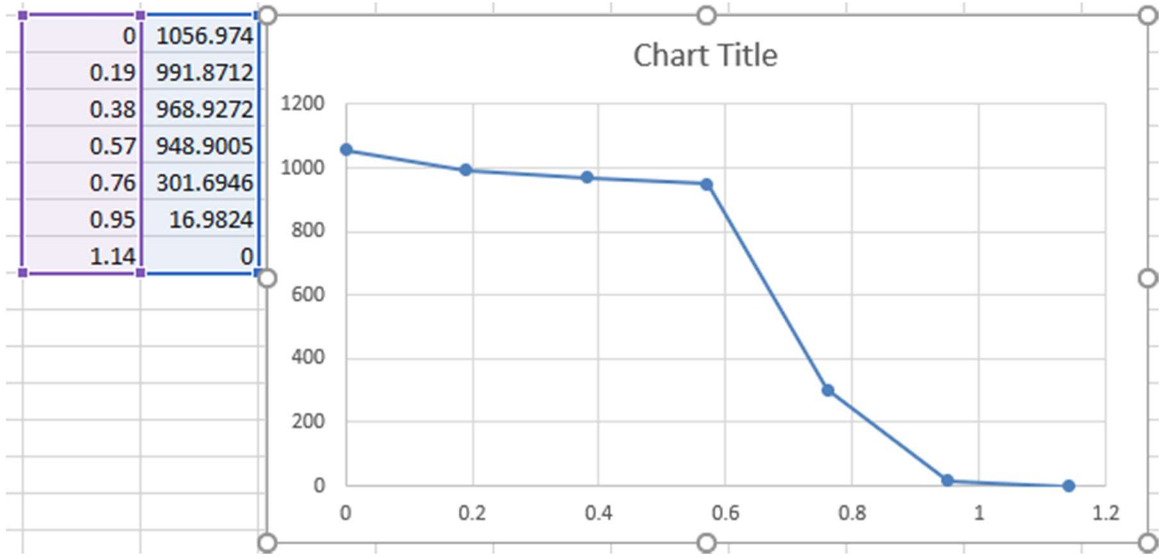


Figure 40: Delta X and Delta Y plotting



0	1056.974
0.19	991.8712
0.38	968.9272
0.57	948.9005
0.76	301.6946
0.95	16.9824
1.14	0

Figure 41: Data collected and generated in excel.