Engineering Design 3

Final Design Project Report

Student Numbers	**Student Names:**
s3752562	Truong Minh Phu
s3741189	Tran Anh Kiet
s3750474	Tran Vu Thao Uyen
s3757333	Dang Truong Nguyen Long
s3754450	Hoang Minh Quan
s3758994	Nguyen Quoc Minh
s3757327	Le Ngoc Duy
Team:	10
Lecturers:	Dr. Minh Tran
Date:	February 13th, 2023

Contents

List of Figures

List of Tables

1. Executive Summary

In this report we are required to accomplish 2 main tasks:

Inverted Pendulum: Control the motor that attaches to the pendulum in the configuration that the rotation axis of the motor is vertical, and the axis of the pendulum is horizontal. The goal is to move the pendulum from pointing downward (stable equilibrium state) to upward (unstable equilibrium state) and the system needs to be able to maintain that position, even in the scenario that external force was applied to the pendulum.

Robot Arm: Control the end effector position, orientation, and the suction cup that attaches to the 6 DoF WLKATA Mirobot, the goal is to pick and place the objects from the initial position to the final position within the working range of the robot.

At this is the final stage of the project, we will focus on the result of the project and the discussion on that, however, the literature review and the approaches and methodology will also be provided.

Inverted Pendulum

In this task, we will use the dual PID control system to enhance the performance of the system, the control system will take 2 input signals: the desired pendulum position and rotor error, and the output of the system will be the pendulum angle.

The system was successful in bringing the pendulum to its inverted position with the lightweight attached to its end (a paper clip). When applied a short impact on the pendulum forces it out of the equilibrium state, the rotor can compensate and bring the pendulum back to the inverted position.

Robot Arm

The manufacturer provided the software that allows us to teach the robot to move its end effector to the desired position as well as control the suction cup. After the robot learned to move all the objects from the initial to the final position, the G code for all the operations could be extracted. Using the G code, we could develop a Python program that controls the robot to do the same tasks without having to teach it again.

The robot successfully learned how to move all the objects to their final position in all tasks. Some of the factors that should be put into consideration are optimizing the trajectory of the end effector and avoiding moving the end effector outside the working range. By using the extracted G code, we successfully develop a Python program for the robot to perform the same task.

2. Introduction

The report for the final stage of the design project details how our team operated, observed, analysed, and programmed the Inverted Pendulum for Task 1 and the Robot Arm for Task 2, based on the design solutions covered in the previous report.

For task 1: Inverted Pendulum, our team was provided with a complete set of inverted pendulum STEVAL-EDUKIT01 so that we could understand the real physical model and how it works. We collected performance data from the physical model during its operation and also generated a simulation model (mathematical model) based on the design methods presented in the previous report to collect performance data from the same. We then compared both sets of performance data to get a better understanding of the system. This comparison enabled us to analyse whether the simulation model accurately reflected the behaviour of the physical model. We also considered the differences between both models to identify any potential improvements or changes needed for better accuracy. From this comparison, we were able to draw conclusions about the overall performance of both models and provide insight into how they can be improved.

For Task 2: Robot Arm, our team needed to program the WLKATA robotic arm to complete 3 small pick-and-place missions. We used G-code on WLKATA Studio in Teaching mode to place the blocks in the designated positions as marked on the A4 map. We provided detailed explanations of each G-code we utilized and then wrote a Python program to control the robot to reproduce the same tasks that Teaching Mode did before. To make the program more efficient, our team incorporated several different coding techniques, such as looping and conditional statements, so that the same code could be used for all 3 missions without having to rewrite it. This allowed us to save time and energy while still achieving accurate results with the robot arm.

3. Inverted Pendulum

3.1. Literature Review

PID controllers are one of the most common strategies used to control an inverted pendulum, a classic problem in control systems [1]. In recent years, many studies have been conducted to better understand the dynamics of this type of unstable system Wang's simulation with three types of inverted pendulum [1] had shown the realization of PID controllers for stabilization and tracking control with robust performance to large and fast disturbances. Another design approach for control system is the two-degree-of-freedom controller (2DOF) which is stated to produce better performance than the traditional 1DOF control system through Horowitz's work [7]. In a tutorial paper, Araki and Taguchi [2] had investigated the 2DOF PID controller including an optimal tuning strategy for parameters of the controller. Their results were found to solve the problem of incompatibility between disturbance response and setpoints, which is very essential in modern process control where changes in setpoint variable are needed to keep track of.

3.2. Approaches and Methodologies

The PID controller can assist the final output that be closer to the primary goal for the Inverted Pendulum design. While the Pendulum is moved by the Stepper Motor, the encoder's job is to record the speed, position, or angle data of it at that time. By using the PID Feedback loop, these recorded data were given back to the PID controller to alter the appropriate input to the system for an improved output status.

a. Dual PID controller

In term of dual PID controller (also known as Two-Degree-of-Freedom PID Controllers), it can quickly reject disturbances without significantly increasing overshoot when tracking setpoints. The impact of changes in the reference signal on the control signal can be minimized by using 2-DOF PID controllers. This PID controller has two inputs, one output and there are other scenarios where a 2-DOF PID controller can be decomposed into SISO parts which include of feedforward, feedback, filter, etc. Our dual PID has a feedforward configuration that divides into a SISO PID controller and a feedforward compensator. The input signals are the target pendulum angle (in degrees) and the rotor error (in degrees), and the output signal is the responded pendulum angle. [3]

The dual PID controller's function is to produce control inputs for the rotor plant's G_{Rotor}, which are subsequently used to track the appropriate pendulum angle by G_{Pend}. K_{Pend} and K_{Rotor} are the corresponding gain in the dual PID controller for the pendulum and rotor.

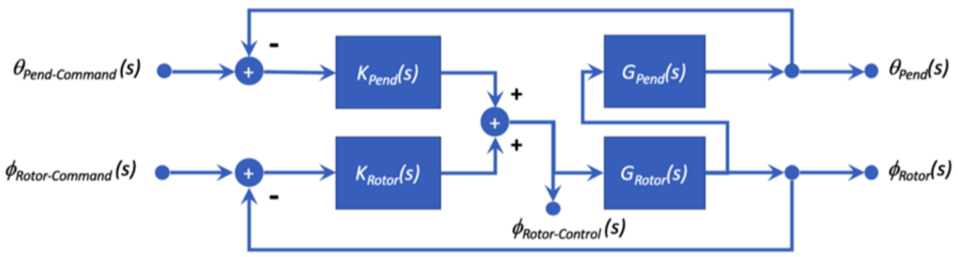

$\theta_{Pend\text{-}Command}(s)$
$\phi_{Rotor\text{-}Command}(s)$
$K_{Pend}(s)$
$K_{Rotor}(s)$
$G_{Pend}(s)$
$G_{Rotor}(s)$
$\theta_{Pend}(s)$
$\phi_{Rotor}(s)$
$\phi_{Rotor\text{-}Control}(s)$
+
-

Figure 1: Dual PID controller.

b. Single PID controller

Since this method supposes that no difference between the angle output of rotor ϕ_{rotor} and the command angle of rotor $\phi_{rotor\_command}$, one PID controller was generated to direct the Pendulum Angle which creates SISO (single input single output) system. As seen on the figure below, the reference command angle of pendulum becomes an input of a system, PID controller of it combine with the output signal of rotor that controls the pendulum angle as an output. With state feedback, the output of the system can be more stable.

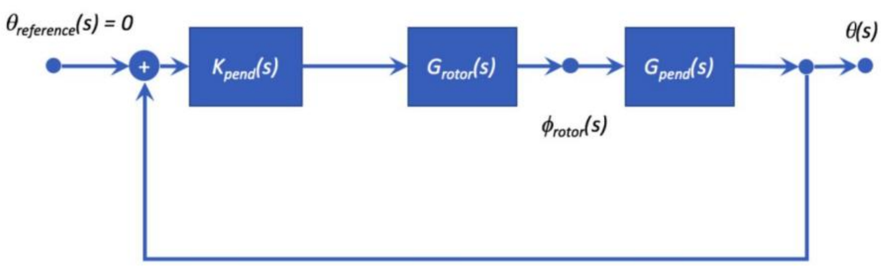


Figure 2: Single PID controller.

Since, parameters of the inverted pendulum are:

Parameter	Description	Value
l	Pendulum Length measured as vertical separation from Rotor Arm to Pendulum Mass	0.235 m
r	Rotor Arm Length	0.14 m
ϕ	Angle of Rotor relative to initial angle reference	
θ	Angle of Pendulum relative to gravity vector	
Rotor Angle Step	Resolution of a Rotor step	0.0563°/step
Rotor Angle Measurement	Resolution of a Rotor step measurement	0.1126°/step
Pendulum Angle Measurement Gain	Measurement gain of Pendulum Angle provided by the optical encoder and its interface	6.667 steps/degree

Table 1. Parameters of the inverted pendulum

Hence, we got the transfer function G_{Pend} and G_{Rotor}:

$$G_{Rotor} = \frac{1}{s^2} \qquad\qquad G_{Pend} = \frac{s^2}{s^2 + 0.2768s - 55.8505}$$

Then, substituting to Simulink transfer function blocks:

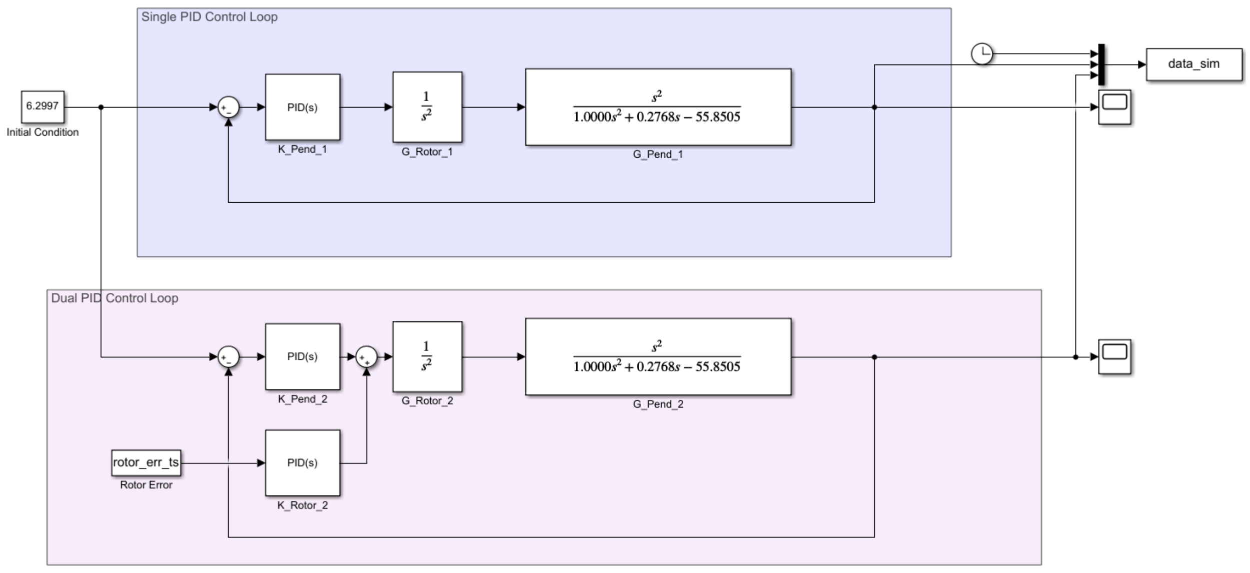


Figure 3: Simulink model of Single and Dual PID Controllers.

The coefficients of the dual PID controller are:

$$K_{P_{Pend}} = 300, K_{I_{Pend}} = 0, K_{D_{Pend}} = 30$$

$$K_{P\_Rotor} = 15, K_{I\_Rotor} = 0, K_{D\_Rotor} = 7.5$$

In order to read the recorded data from the STEVAL-EDUKIT01 which includes time (s), pendulum angle (degree) and rotor's error (degree), a MATLAB file "inverted_pend_read_data.m" is used. Then, it's stored in workspace of MATLAB and extracted in an Excel file. With all the needed value of PID gains, extracted data, initial position, and stop time, we then run MATLAB file and get the data response and plot response of the MATLAB simulation and recorded data.

3.3. Result and Discussion

By running the MATLAB file Edukit_Real_Time_Control_System_Workbench_Matlab.m, we got the plot responses of the inverted pendulum. There are 4 plots that is illustrated such as Rotor Angle (degree), Pendulum Angle (degree), Rotor Tracking Command (degree), and Rotor Control (degree). In this experiment, 2 sweeps were selected for running the system with the Dual PID controller. Regarding to sweep 1, the figure [] is showed below had been recorded, while the PID gain of rotor are $K_{P_{Pend}} = 300, K_{I_{Pend}} = 0, K_{D_{Pend}} = 30$, the PID gain of pendulum are $K_{P\_Rotor} = 15, K_{I\_Rotor} = 0, K_{D\_Rotor} = 7.5$.

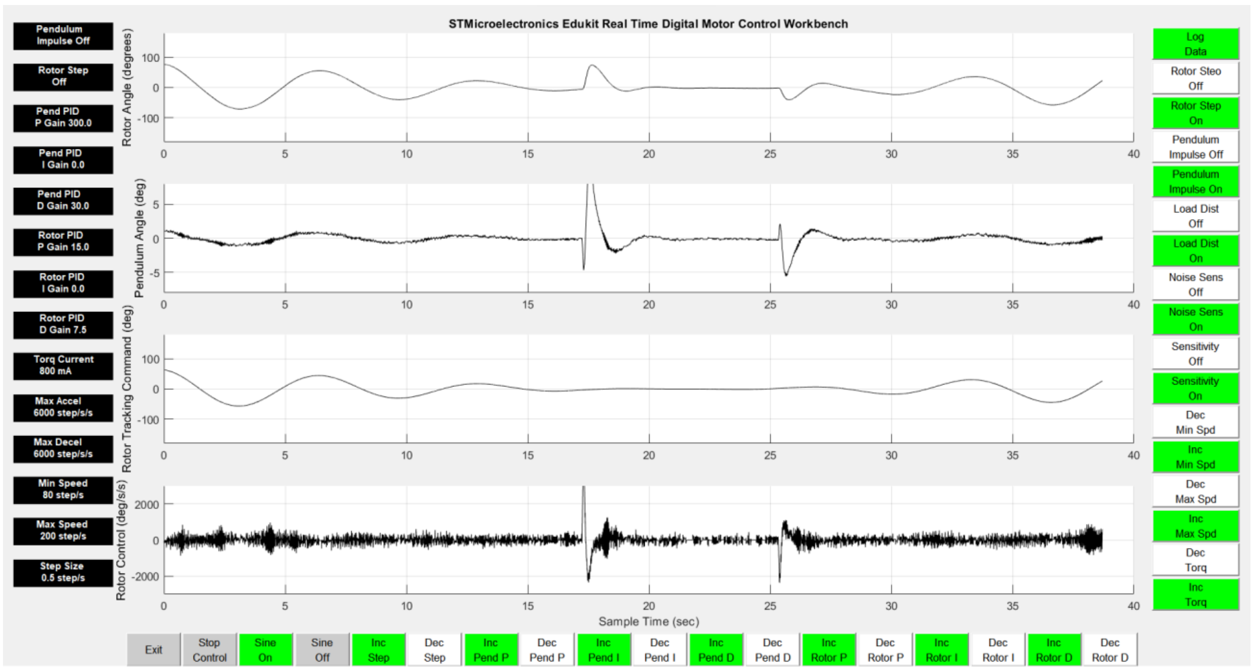
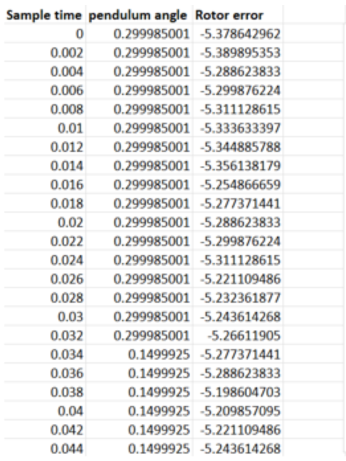


Figure 4: Workbench for STEVAL-EDUKIT01.

As shown, when the rotor and pendulum are more stable, At the second 17 and 26, the pendulum was applied by the external force that make the pendulum falling out of its unstable equilibrium position. At that time, the Rotor created the additional signal which leaded to the difference between the Rotor Angle and Rotor Tracking Command in order to compensate for the impact and bring the pendulum back to its equilibrium position within less than 4 seconds. In addition, the Rotor Tracking Command plot displays the Reference Angle input to the rotor that decides the angle of pendulum. Since the interrupt appeared, the difference existed between the output (Rotor Angle) and input (Rotor Tracking Command) of the rotor. The Rotor Control plot was created due to the PID controller which generates signal to remain stable of the rotor whenever the interrupt occurs.

After the completion of the experiment, the following data was captured and copied to an Excel file:



Sample time	pendulum angle	Rotor error
0	0.299985001	-5.378642962
0.002	0.299985001	-5.389895353
0.004	0.299985001	-5.288623833
0.006	0.299985001	-5.299876224
0.008	0.299985001	-5.311128615
0.01	0.299985001	-5.333633397
0.012	0.299985001	-5.344885788
0.014	0.299985001	-5.356138179
0.016	0.299985001	-5.254866659
0.018	0.299985001	-5.277371441
0.02	0.299985001	-5.288623833
0.022	0.299985001	-5.299876224
0.024	0.299985001	-5.311128615
0.026	0.299985001	-5.221109486
0.028	0.299985001	-5.232361877
0.03	0.299985001	-5.243614268
0.032	0.299985001	-5.26611905
0.034	0.1499925	-5.277371441
0.036	0.1499925	-5.288623833
0.038	0.1499925	-5.198604703
0.04	0.1499925	-5.209857095
0.042	0.1499925	-5.221109486
0.044	0.1499925	-5.243614268

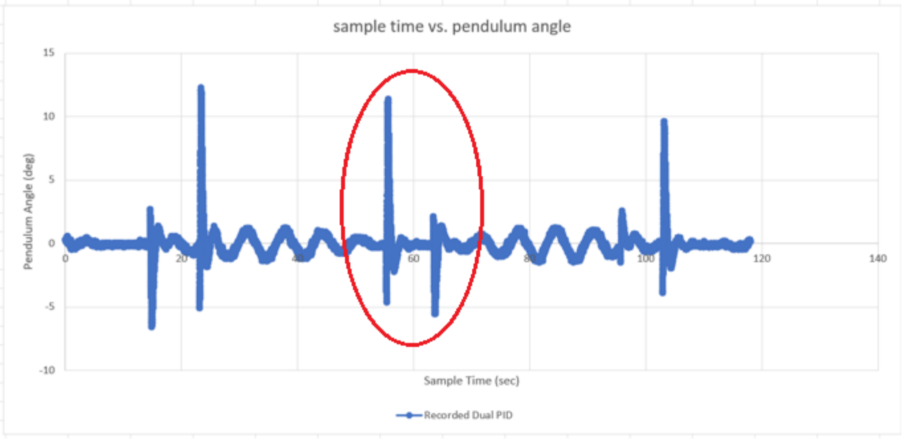
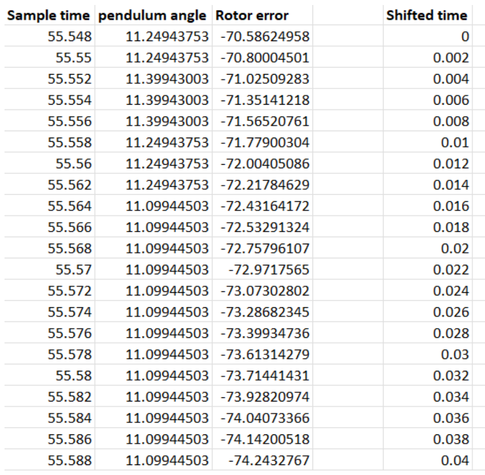


Figure 5: Recorded dual PID's data.

According to the figure above, the time (s), pendulum angle (degree) and rotor's error (degree) are shown. And plotting the sample time vs. pendulum angle, we got the graph. It shows the process of the experiment when we force the arm to turn left, right and the performance of PID controller in stabilizing the system and keep the pendulum in the upward position. Moreover, the data that is highlighted in red from *figure 5* and pendulum angle of sweep 1 from *figure 4* are similar when they are compared. As well as the graph in *figure 5* and the pendulum angle of all sweeps from *figure 16* in the Appendices are the same.

We selected highlighted data as the data was retrieved starting with the chosen initial condition and ending when the angle stabilized at almost zero degree, and it is as follows:



Sample time	pendulum angle	Rotor error		Shifted time
55.548	11.24943753	-70.58624958		0
55.55	11.24943753	-70.80004501		0.002
55.552	11.39943003	-71.02509283		0.004
55.554	11.39943003	-71.35141218		0.006
55.556	11.39943003	-71.56520761		0.008
55.558	11.24943753	-71.77900304		0.01
55.56	11.24943753	-72.00405086		0.012
55.562	11.24943753	-72.21784629		0.014
55.564	11.09944503	-72.43164172		0.016
55.566	11.09944503	-72.53291324		0.018
55.568	11.09944503	-72.75796107		0.02
55.57	11.09944503	-72.9717565		0.022
55.572	11.09944503	-73.07302802		0.024
55.574	11.09944503	-73.28682345		0.026
55.576	11.09944503	-73.39934736		0.028
55.578	11.09944503	-73.61314279		0.03
55.58	11.09944503	-73.71441431		0.032
55.582	11.09944503	-73.92820974		0.034
55.584	11.09944503	-74.04073366		0.036
55.586	11.09944503	-74.14200518		0.038
55.588	11.09944503	-74.2432767		0.04

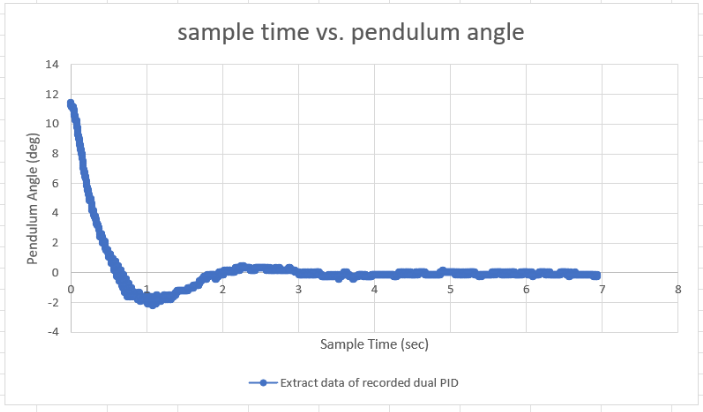
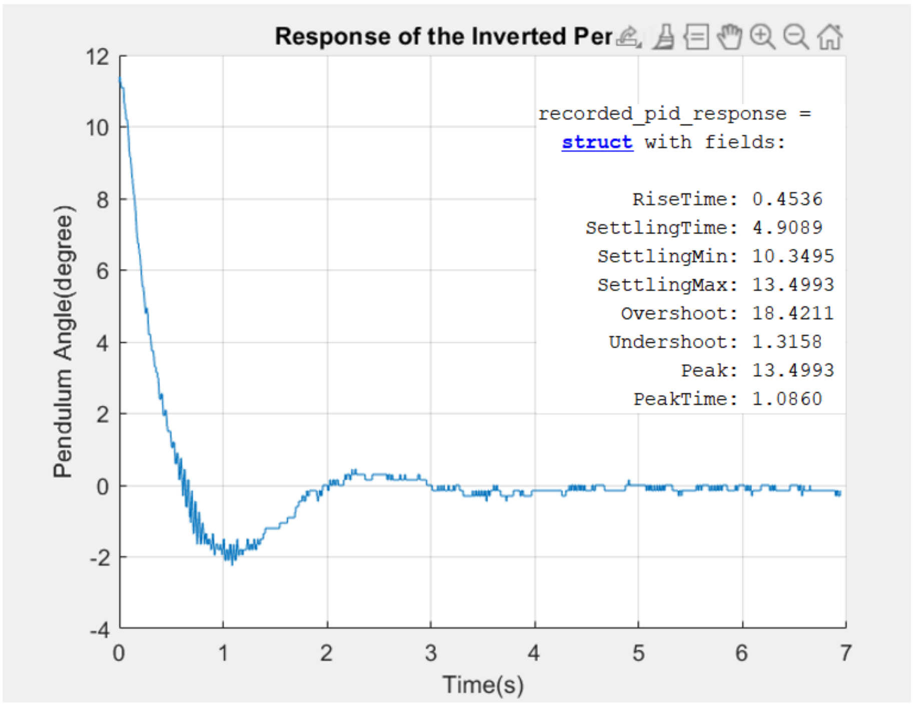


Figure 6: Extract data of recorded dual PID.



```
recorded_pid_response =
  struct with fields:

      RiseTime: 0.4536
  SettlingTime: 4.9089
   SettlingMin: 10.3495
   SettlingMax: 13.4993
     Overshoot: 18.4211
    Undershoot: 1.3158
          Peak: 13.4993
      PeakTime: 1.0860
```

Figure 7: Plot Response of the Inverted Pendulum kit in MATLAB.

By vertically flipping the graph above, the performance of the PID controller in the Inverted Pendulum kit can be determined. Because when the graph is flipped, it will be similar to the step response which is used to analyse system's characteristic. With the rise time is 0.45s, settling time is 4.91s, overshoot is 18.42%, and peak time is 1.09s, we got damping ratio:

$$\zeta = \frac{\left|\ln\left(\frac{PO}{100}\right)\right|}{\sqrt{\pi^2 + \ln\left(\frac{PO}{100}\right)^2}}$$

Where:

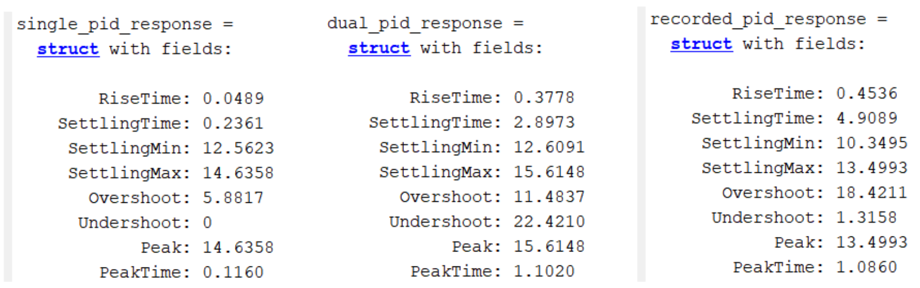
PO is percentage of overshoot.

ζ is damping ratio and $0 < \zeta < 1$

$$\Rightarrow \zeta = 0.47$$

Hence, we can conclude that the PID controller in the Inverted Pendulum kit worked well with the characteristics are in acceptable range.

Because our chosen initial position is 11.24943753 degrees and the stop time is 6.942s, we obtained the step response and plot response of Simulation and Recorded Data by running MATLAB and Simulink files.



```
single_pid_response =
  struct with fields:

       RiseTime: 0.0489
   SettlingTime: 0.2361
    SettlingMin: 12.5623
    SettlingMax: 14.6358
      Overshoot: 5.8817
     Undershoot: 0
           Peak: 14.6358
       PeakTime: 0.1160
```

```
dual_pid_response =
  struct with fields:

       RiseTime: 0.3778
   SettlingTime: 2.8973
    SettlingMin: 12.6091
    SettlingMax: 15.6148
      Overshoot: 11.4837
     Undershoot: 22.4210
           Peak: 15.6148
       PeakTime: 1.1020
```

```
recorded_pid_response =
  struct with fields:

       RiseTime: 0.4536
   SettlingTime: 4.9089
    SettlingMin: 10.3495
    SettlingMax: 13.4993
      Overshoot: 18.4211
     Undershoot: 1.3158
           Peak: 13.4993
       PeakTime: 1.0860
```

Figure 8: Step response of Simulation and Recorded Data.

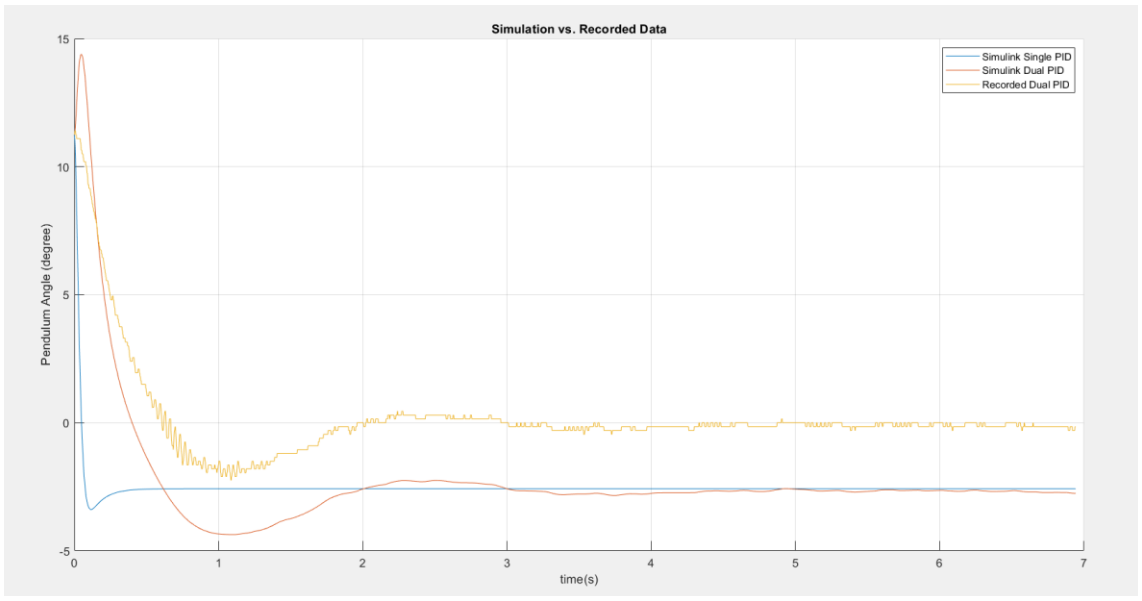


Figure 9: Plot response of Simulation and Recorded Data.

Since the system only created one PID controller, the data and plot look ideal. It generates the lower rising time, settling time, peak time that is around ten times shorter, and reduced overshoot percentage rather than the Dual PID controller in both MATLAB and experiment. Without any external force, the STEVAL-EDUKIT01 might perform as desired with a PID controller for Pendulum, however this PID controller only has a function to regulate the Pendulum Angle and not the Rotor Angle. Otherwise, the criteria stated require the external force to the system; hence, without a PID controller for RA, it cannot move to maintain the pendulum's stability; instead, the Rotor would float in inertia.

By comparing the simulation and experiment of dual PID controller above, we can see that the shape of the plot response is approximately the same. However, the rise time, settling time, overshoot of the simulation has a decrease compared to the recorded value and they are in acceptable range. The percent difference between them is the rise time (18.42%), settling time (69.3%), overshoot (60.5%), and peak (15.7%). Additionally, final value is the main difference between them. The simulation data has a final value of roughly -2.5, compared to the experimental data's final value of around 0. Because the angle will be stabilized at almost zero degrees at the end of the data, it shows that the final value of the recorded data is more accurate. Moreover, there are noises or disturbances in the recorded data that result in inaccurate data reading. This occurs because the STEVAL-EDUKIT01, which is not weighty, displaces when the arm and pendulum rotate. Although we have held the system firmly, there will still be some shaking from the kit. As a result, the vibration from the kit causes the encoder to vibrate and noises to be produced. Furthermore, the stepper motor may also make noise if it is not lubricated carefully.

4. Robot Arm

4.1. Literature Review

As the project has come to the final phase, meaning the proposed theoretical design is supposed to validate and implement on the real WLKATA Mirobot, it is thus essential to review the user manual provided from the manufacturer. This helps give a clearer understanding of how to integrate and apply the design to perform the required three tasks using the robot arm. Developed for higher educational purposes, WLKATA Mirobot comes with a handy and easy to use control software called WLKATA Studio **[4]**. The studio provides Teaching function along with Coord Mode that allow the user to easily control the coordinates of the end effector with live movement change from the as well as the robot action and speed using the user interface. The intended action then can be saved as a command allowing the user to program the robot. The coordinate space of the robot arm is presented in Figure 10.

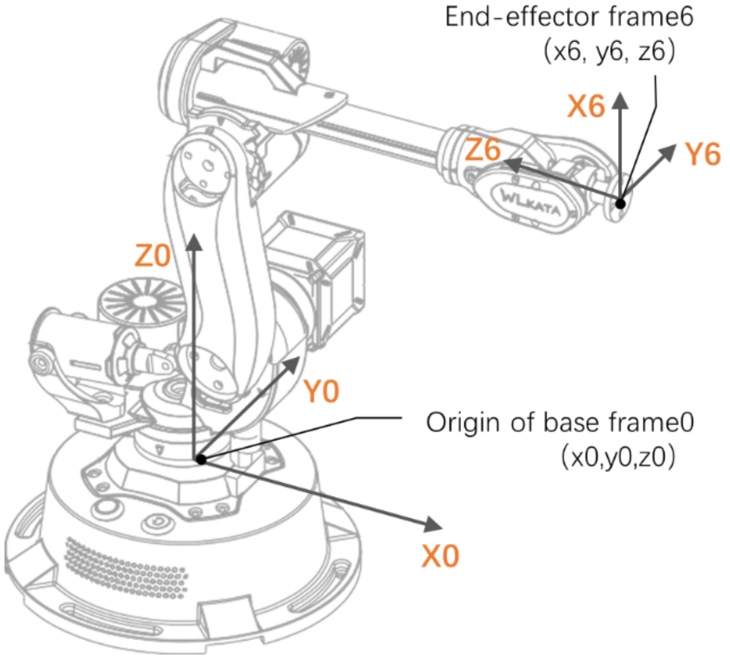


Figure 10. Mirobot Cartesian coordinate system **[5]**

The position of the end effector is based on the base frame0 (x0, y0, z0) while the orientation is based on the end-effector frame6 (x6, y6, z6). The roll is the rotation around axis z6, pitch is the rotation around axis y6 and yaw is the rotation around axis z6.

Regarding the control of the movement of the manipulator, the two important movement type are Fast motion and Linear interpolation movement. The Fast motion function allows the arm to quickly move to a specific coordinate position, while Linear interpolation movement is used to move the robotic arm to a position in a straight line.

4.2. Approaches and Methodologies

The general robot arm operation to complete the required task for 3 tasks is as follow:

- Move to and hover above a cube start position.
- Move the end effector down and suck the cube.
- Move the end effector up to a height so that movement of the robot at that height don't collide with the other object.
- Move to and hover above the cube end position.
- Move the end effector down and release the cube.
- Move the end effector up to a height so that movement of the robot at that height don't collide with the other object.
- Repeat until all cubes are in the destination position.

The code for the robot arm follows the flowchart in Figure 11. The only exception is for a cube in Task 2 as moving directly from the start to destination is not possible due to the end effector move out of the robot working range during the movement. Therefore, the cube will be move to another position within the working range before being placed down at the destination instead of moving to and hovering directly above the destination from the start position.

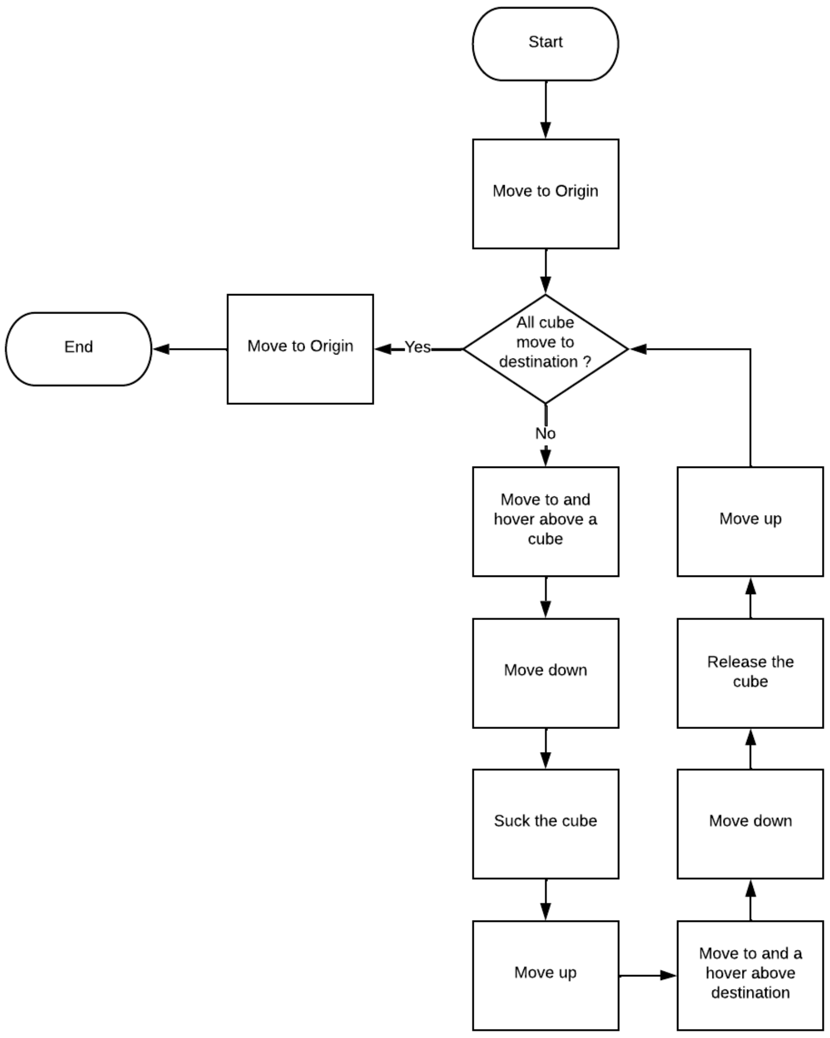


Figure 11. Robot arm code flow chart

With the above robot arm position the cube will have the following movement in Figure 11. This movement allow the cube to move quickly to destination without being obstructed by other cubes. The code will utilize 2 movement mode: Fast motion (MOVJ) and Linear interpolation movement (MOVL). Fast motion is when each joint robot arm move so that the end effector can move at maximum speed to the destination [6]. Linear interpolation movement is when each joint robot arm move so that the end effector can move to the destination in a straight line [6]. Based on our testing, using Fast motion to move the end effector vertically in a short distance can cause slight inaccuracy in the destination position. Therefore, the moving up and down motion of the end effector will use Linear interpolation movement while other movement will use Fast motion as shown in Figure 12.

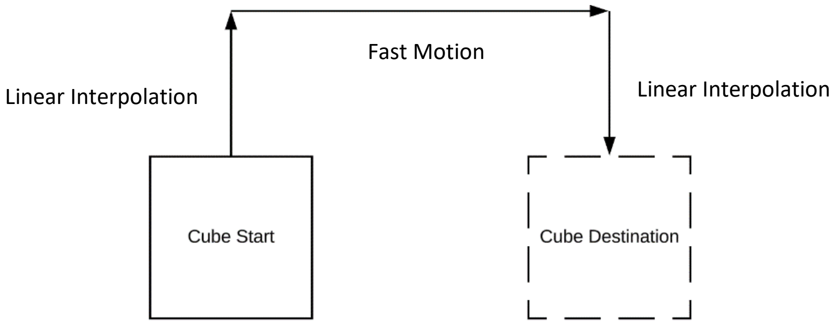
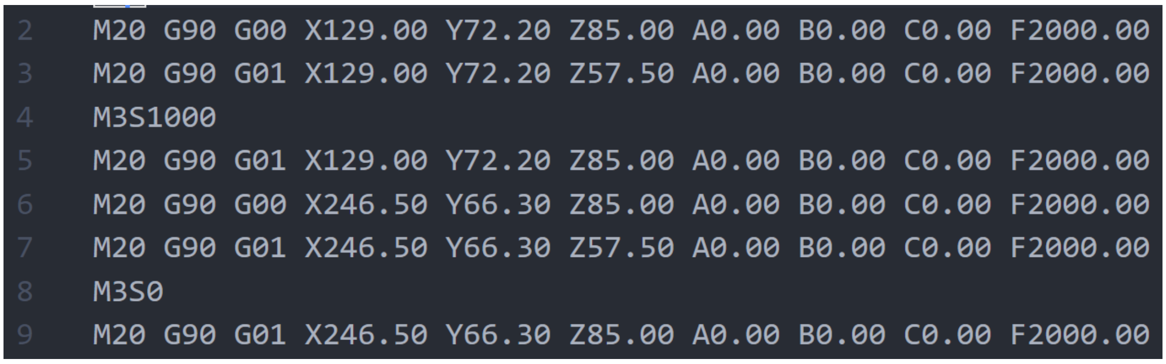


Figure 12. Cube movement

The robot arm will be first programmed in WLKATA Studio Teaching mode since it easier to view and adjust the end effector desired coordinate. After programming and testing in Teaching mode, we will export the program to G code, study and understand the G code then write the Python program.

4.3. Result and Discussion

All 3 task 3 completed successfully but there are some slight inaccuracies for task 1 and 2 while with task 3, the layers of the tower don’t align perfectly. The slight inaccuracy of the cube at the end position is due to the cubes being placed on the initial position by hand which does not always guarantee the perfect initial position and orientation of the cube. Figure 13 show the G code for the operation of 1 cube in task 1. The other operation for all the cubes in all 3 tasks has similar G code to the code shown in *Figure 13*.



```
M20 G90 G00 X129.00 Y72.20 Z85.00 A0.00 B0.00 C0.00 F2000.00
M20 G90 G01 X129.00 Y72.20 Z57.50 A0.00 B0.00 C0.00 F2000.00
M3S1000
M20 G90 G01 X129.00 Y72.20 Z85.00 A0.00 B0.00 C0.00 F2000.00
M20 G90 G00 X246.50 Y66.30 Z85.00 A0.00 B0.00 C0.00 F2000.00
M20 G90 G01 X246.50 Y66.30 Z57.50 A0.00 B0.00 C0.00 F2000.00
M3S0
M20 G90 G01 X246.50 Y66.30 Z85.00 A0.00 B0.00 C0.00 F2000.00
```

Figure 13. Exported G code from Teaching mode of task 1 for 1 operation.

Based on G code documentation, the meaning of each command is:

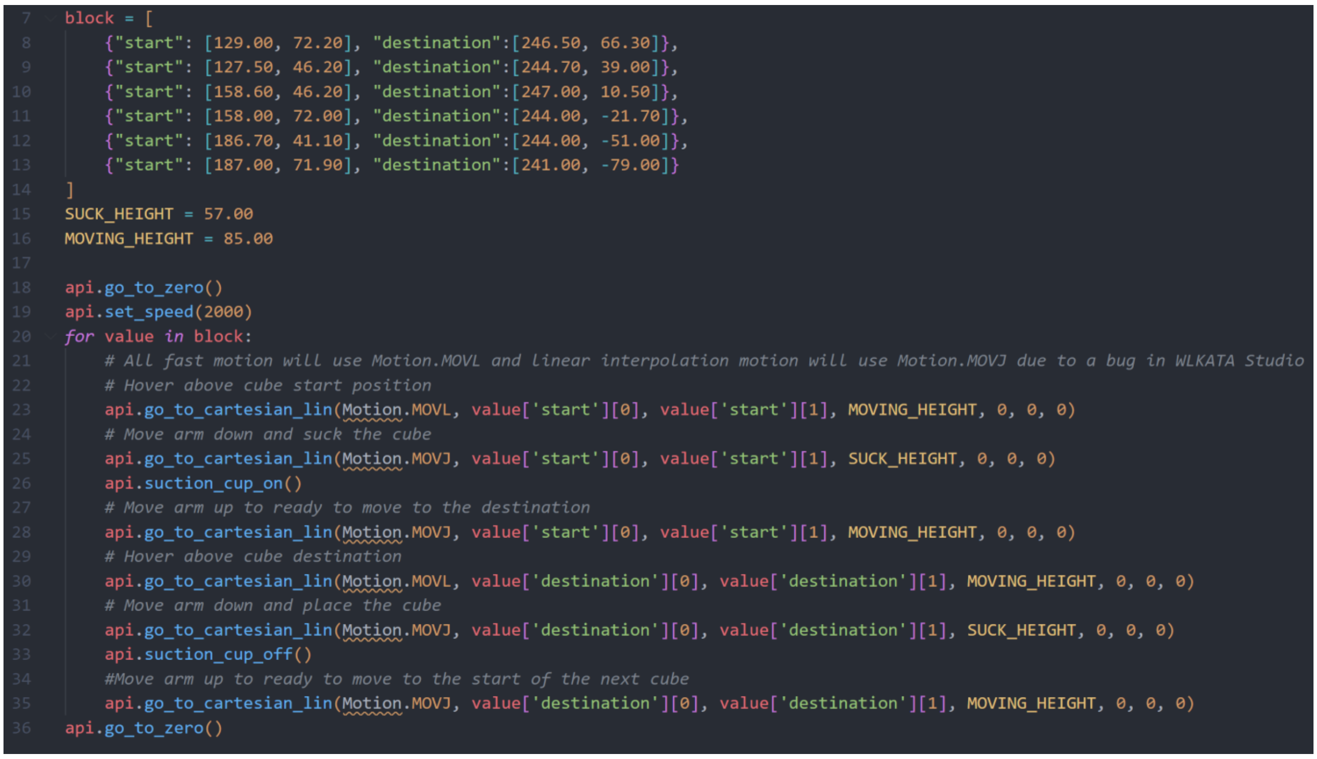
- M20: the robot arm run in Cartesian mode. The value of X, Y and Z are the coordinates of the end effector relative to the base frame. A is the roll, B is the pitch and C is the yaw of the end effector [5].
- M21: the robot arm run in Angle mode. The value of X, Y, Z, A, B, C are the angle of the corresponding joint 1, 2, 3, 4, 5, 6 of the robot [5].
- G90: this command makes the value of X, Y, Z, A, B, C the absolute target value [5].

- G00: the robot arm will move with Fast motion which means that each joint of the robot arm move at maximum speed [5].
- G01: the robot arm will move with Linear interpolation motion which means that each joint of the robot arm move so that the end effector will move in a straight line [5].
- F2000: set the speed of the movement. In this case, 2000 represent the target speed of 2000 mm/min of the end effector [5].
- M3S1000: Turn on the suction cup [5].
- M3S0: Turn off the suction cup [5].

With the understanding of each command, the meaning of the G code in *Figure 13* is presented in Table 2.

Line Number	Intended operation	G code line meaning
2	Move to and hover above a cube start position	Move to coordinate (129, 72.2, 85) with the orientation (0, 0, 0) using fast motion with the speed of 2000 mm/min
3	Move the end effector down	Move to coordinate (129, 72.2, 57) with the orientation (0, 0, 0) using linear interpolation motion with the speed of 2000 mm/min
4	Suck the cube	Turn on the suction cup
5	Move the end effector up	Move to coordinate (129, 72.2, 85) with the orientation (0, 0, 0) using linear interpolation motion with the speed of 2000 mm/min
6	Move to and hover above a cube destination position	Move to coordinate (246.5, 66.3, 85) with the orientation (0, 0, 0) using fast motion with the speed of 2000 mm/min
7	Move the end effector down	Move to coordinate (246.5, 66.3, 57) with the orientation (0, 0, 0) using linear interpolation with the speed of 2000 mm/min
8	Release the cube	Turn off the suction cup
9	Move the end effector up	Move to coordinate (246.5, 66.3, 85) with the orientation (0, 0, 0) using linear interpolation with the speed of 2000 mm/min

Table 2. G code meaning

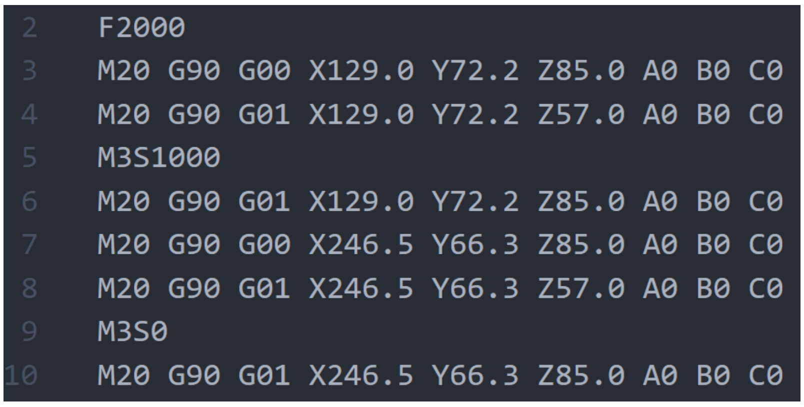


```python
block = [
    {"start": [129.00, 72.20], "destination":[246.50, 66.30]},
    {"start": [127.50, 46.20], "destination":[244.70, 39.00]},
    {"start": [158.60, 46.20], "destination":[247.00, 10.50]},
    {"start": [158.00, 72.00], "destination":[244.00, -21.70]},
    {"start": [186.70, 41.10], "destination":[244.00, -51.00]},
    {"start": [187.00, 71.90], "destination":[241.00, -79.00]}
]
SUCK_HEIGHT = 57.00
MOVING_HEIGHT = 85.00

api.go_to_zero()
api.set_speed(2000)
for value in block:
    # All fast motion will use Motion.MOVL and linear interpolation motion will use Motion.MOVJ due to a bug in WLKATA Studio
    # Hover above cube start position
    api.go_to_cartesian_lin(Motion.MOVL, value['start'][0], value['start'][1], MOVING_HEIGHT, 0, 0, 0)
    # Move arm down and suck the cube
    api.go_to_cartesian_lin(Motion.MOVJ, value['start'][0], value['start'][1], SUCK_HEIGHT, 0, 0, 0)
    api.suction_cup_on()
    # Move arm up to ready to move to the destination
    api.go_to_cartesian_lin(Motion.MOVJ, value['start'][0], value['start'][1], MOVING_HEIGHT, 0, 0, 0)
    # Hover above cube destination
    api.go_to_cartesian_lin(Motion.MOVL, value['destination'][0], value['destination'][1], MOVING_HEIGHT, 0, 0, 0)
    # Move arm down and place the cube
    api.go_to_cartesian_lin(Motion.MOVJ, value['destination'][0], value['destination'][1], SUCK_HEIGHT, 0, 0, 0)
    api.suction_cup_off()
    #Move arm up to ready to move to the start of the next cube
    api.go_to_cartesian_lin(Motion.MOVJ, value['destination'][0], value['destination'][1], MOVING_HEIGHT, 0, 0, 0)
api.go_to_zero()
```

Figure 14. Task 1 Python code

The Python program contain an array that contain the necessary data for each cube. For Task 1, the array contains the start and end x, y coordinate of the cube. For Task 2, the cube at index 1 also contain the x, y coordinate that the end effector will move to before moving to the destination. For Task 3, each cube also contains the destination hover and placement z coordinate. The program contains a loop that perform operation for each cube as describe in *Figure 11*for all 3 tasks. Task 2 also contain an if statement to handle moving to destination of the second cube. The function "go_to_zero" move the end effector to the (0, 0, 0) position with the orientation of (0, 0, 0). The function "set_speed" set the movement speed of the end effector with the unit of mm/min. The function "go_to_cartesian_lin" move the end effector to the specified coordinates with the specified orientations [6]. Based on the Python documentation, the first parameter of function "go_to_cartesian_lin" accept "Motion.MOVJ" to move in fast motion and "Motion.MOVL" to move in linear interpolation motion [6]. However, the exported G code command of the function "go_to_cartesian_lin" with "Motion.MOVJ" is G01 which is linear interpolation motion [5] while with "Motion.MOVL" is G00 which is fast motion [5]. This is likely due to a bug in WLKATA Studio. Because of this, the first parameter of "go_to_cartesian_lin" in the final Python program in all 3 tasks will use "Motion.MOVL" for fast motion and "Motion.MOVJ" for linear interpolation motion. The function "suction_cup_on" turn on and "suction_cup_off" turn off the suction cup.



```
F2000
M20 G90 G00 X129.0 Y72.2 Z85.0 A0 B0 C0
M20 G90 G01 X129.0 Y72.2 Z57.0 A0 B0 C0
M3S1000
M20 G90 G01 X129.0 Y72.2 Z85.0 A0 B0 C0
M20 G90 G00 X246.5 Y66.3 Z85.0 A0 B0 C0
M20 G90 G01 X246.5 Y66.3 Z57.0 A0 B0 C0
M3S0
M20 G90 G01 X246.5 Y66.3 Z85.0 A0 B0 C0
```

Figure 15. Exported G code from Python of task 1 for 1 operation.

Based on *Figure 15* and *Figure 13*, the exported G code file of the Python program is almost similar to the G code file exported from teaching mode with the only difference is that each movement command from the teaching mode exported G code contain set movement speed command while the Python exported G code set the movement speed at the beginning of the script without setting the movement speed for each movement command.

5. Demonstration Video

5.1. Inverted Pendulum

Demonstration video: Inverted Pendulum Demontration with Description.mp4

5.2. Robot Arm

Demonstration video for Task 1: Task 1 - Robot Arm.MOV

Demonstration video for Task 2: Task 2 - Robot Arm.MOV

Demonstration video for Task 3: Task 3 - Robot Arm.MOV

6. Conclusions

In conclusion, the project is a great opportunity for the team to prepare for applying engineering practice in real life. Though the Inverted Pendulum task, we got more experience about the system that generate the Motor Control in particular, and Control System in general. As learned about the PID controller before and having the fundamental knowledge about it, and their usage was enlarged in this design where we produced the MIMO close-loop system with Dual PID controller to control the Rotor Angle as well as Pendulum Angle for the stable edition. This method can be helpful for us to implement the stable design system in the future. From the next proposed design, we have successfully managed to control the robot arm to perform some interactions with the cube by exploring the structure and parameters of the WKLATA Mirobot. The robotic project helps shed light on the industry requirements for robotic manufacturing as well as some programming languages such as G-code and Python.

7. References

[1] J.-J. Wang, “Simulation studies of inverted pendulum based on PID controllers,” Simulation Modelling Practice and Theory, vol. 19, no. 1, pp. 440–449, 2011.

[2] *Welcome! | korea science* (no date). Available at: https://koreascience.kr/article/JAKO200311921575825.pdf (Accessed: February 3, 2023).

[3] *GetComponents* (no date) *Two-Degree-of-Freedom PID Controllers - MATLAB & Simulink*. Available at: https://www.mathworks.com/help/control/ug/two-degree-of-freedom-2-dof-pid-controllers.html (Accessed: February 1, 2023).

[4] WLKATA Robotics (no date) *Documentation: Wlkata Book Shelf: WLKATA robotics document*, *WLKATA Robotics*. Available at: https://document.wlkata.com/?doc=%2Fwlkata-mirobot-user-manual-platinum%2F (Accessed: January 13, 2023).

[5] WLKATA Robotics (no date) *Documentation: Wlkata Book Shelf: WLKATA robotics document*, *WLKATA Robotics*. Available at: https://document.wlkata.com/?doc=%2Fwlkata-mirobot-user-manual-platinum%2F11-introduction-of-wlkata-mirobot%2F (Accessed: January 13, 2023).

[6] WLKATA Robotics (no date) *Documentation: Wlkata Book Shelf: WLKATA robotics document*, *WLKATA Robotics*. Available at: https://document.wlkata.com/?doc=%2Fwlkata-mirobot-user-manual-platinum%2F15-using-the-python-programming%2F (Accessed: February 3, 2023).

[7] *Synthesis of feedback systems* (no date) *Google Books*. Google. Available at: https://books.google.com.vn/books?hl=en&lr=&id=qykSBQAAQBAJ&oi=fnd&pg=PP1&dq=synthesis%2Bof%2Bfeedback%2Bsystems&ots=olbTi3zAfi&sig=JTyxbPZOSHJXpZ-wHIzBOEEDWeA&redir_esc=y#v=onepage&q=synthesis%20of%20feedback%20systems&f=false (Accessed: January 20, 2023).

8. Appendices

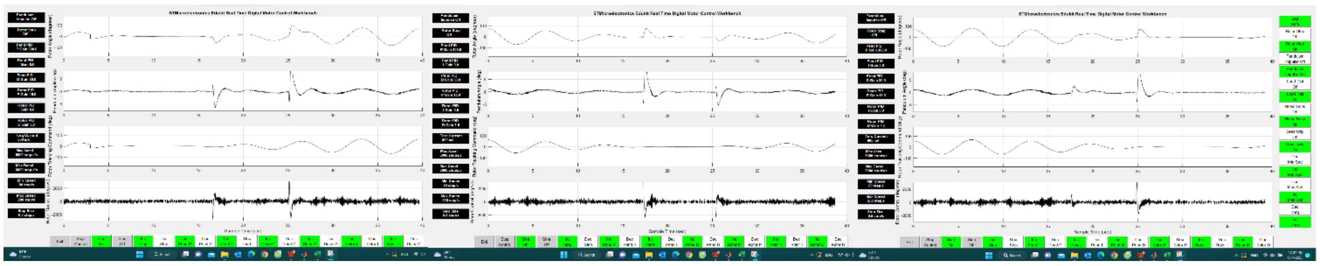


Figure 16: Workbench of all 3 sweeps.