
# AIN SHAMS UNIVERSITY

## FACULTY OF ENGINEERING

### International Credit Hour Programs (ICHEP)

# MCT233 – Dynamic Modelling & Simulation

# Major Task Report

**Team 10 Group Members:**

**ID: NAME:**

**21P0410: Omar ElShawaf**

**21P0017: Mahmoud Eldwakhly**

**21P0088: Yousef Ahmed**

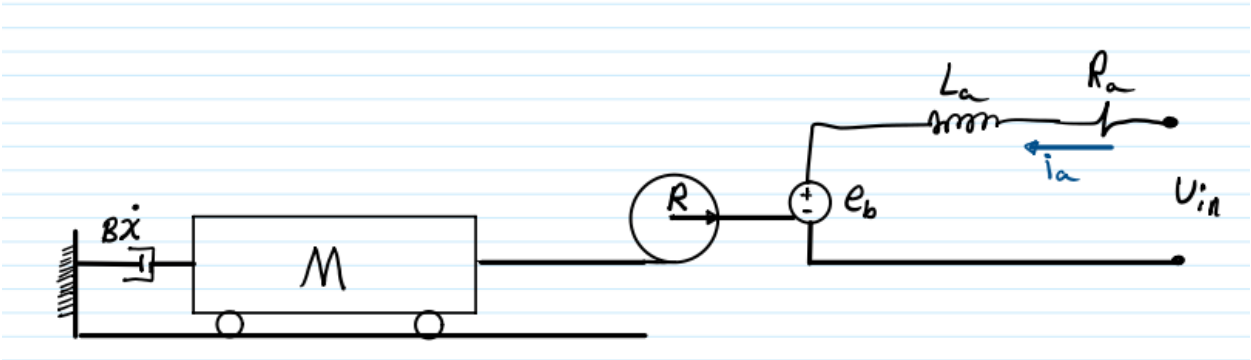
**21P0137: Mohamed Mohamed Taha**

# Contents

## Introduction:

The project simply is a linear cart that moves linearly on a V-slot aluminium rod. The cart is attached to the motor that rotates by a belt, and the motor is equipped with an encoder to track its movement. To control the motor's speed, we've used tools like MATLAB, Arduino, and to control the motor's speed, an H-bridge is used.After fixing the physical system completely. Using MATLAB, A simulation and mathematical model is made, The Task is to make the simulation behave like the model in daily life. This will be done by estimating parameters like different types of friction affecting the real-life speed of the cart. So, it should be put it in simulation and so on. This process involves applying input signals which will be PWM in our project, measuring the cart's output velocity, and using Simulink/Simscape to model and simulate the system. Through analytical tools and the Parameter Estimation feature in MATLAB Simulink, we aim to pinpoint the values of system parameters, ensuring our virtual model closely mirrors the real-world behavior. This endeavor not only calls for technical skills but also prompts reflection on the acquired simulation expertise, evaluating and comparing responses of both the physical and simulated systems under different input signals.

## Schematic Graphical Representation of System:



# Mathematical Model of System:

**Given:**

$R = 1.1\ cm = 0.011\ m$

$L_a = 0.01\ H$

$V_{in} = 12\ V$

$R_a = \frac{V_{in}}{I_{stall}} = \frac{12}{1.5} = 8\ \Omega$

$M = 107\ gm = 0.107\ Kg$

$J_r = 0.0002\ Kg.m^2$

$K_t = 0.494\ N.m/A$

$b = 5 \times 10^{-6}\ \frac{N.m}{r/s}$

$Friction = 7.92\ N$

---

**For electrical system:**

$$R_a I_a(s) + L_a s I_a(s) + e_b(s) = V_{in}(s)$$

$$e_b(s) = K_b s \theta_m(s)$$

$$T_m = K_t I_a(s)$$

$$\frac{(R_a + L_a s) T_m(s)}{K_t} + K_b s \theta_m(s) = V_{in}(s)$$

$$\frac{(R_a + L_a)}{K_t} \times T_m(t) + K_b \omega_m(t) = e_b(t)$$

$$\frac{(8 + 0.01)}{0.494} \times T_m + 0.494 \dot{\theta}_2 = V_{in}(t) \Rightarrow (1)$$

For mechanical system:

$$J_{eq}\ddot{\theta}_2 = T_{meq} - b_{eq}\dot{\theta}_2 - FR$$

$$0.0002\,\ddot{\theta}_2 = T_m - 5\times10^{-6}\dot{\theta}_2 - 0.011\times 7.92$$

$$\boxed{T_m = 2\times10^{-4}\ddot{\theta}_2 + 5\times10^{-6}\dot{\theta}_2 + 0.08712} \rightarrow (2)$$

$$V_{in}(t) = 16.2145\times(2\times10^{-4}\ddot{\theta}_2 + 5\times10^{-6}\dot{\theta}_2 + 0.08712) + 0.494\dot{\theta}_2$$

$$V_{in}(t) = 3.2\times10^{-3}\ddot{\theta}_2 + 8.1\times10^{-6}\dot{\theta}_2 + 1.4126 + 0.494\dot{\theta}_2$$

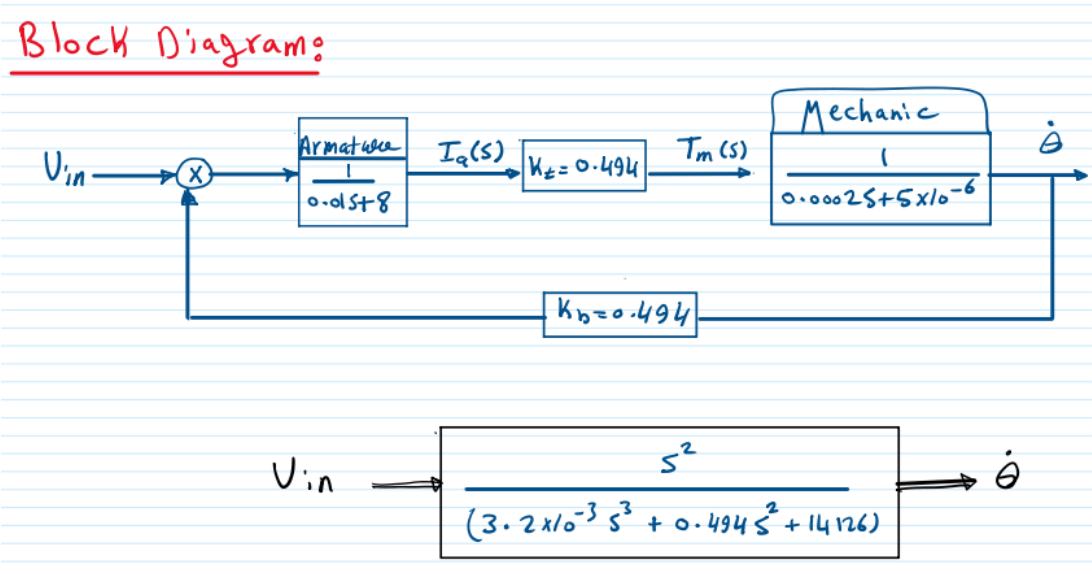
$$V_{in}(t) = 3.2\times10^{-3}\ddot{\theta}_2 + 0.494\dot{\theta}_2 + 1.4126$$

$$V_{in}(s) = \left(3.2\times10^{-3}s + 0.494 + \frac{1.4126}{s^2}\right)\dot{\theta}(s)$$

$$\boxed{\frac{\dot{\theta}(s)}{V_{in}(s)} = \frac{s^2}{(3.2\times10^{-3}s^3 + 0.494s^2 + 1.4126)}} \rightarrow \text{Mathematical model}$$

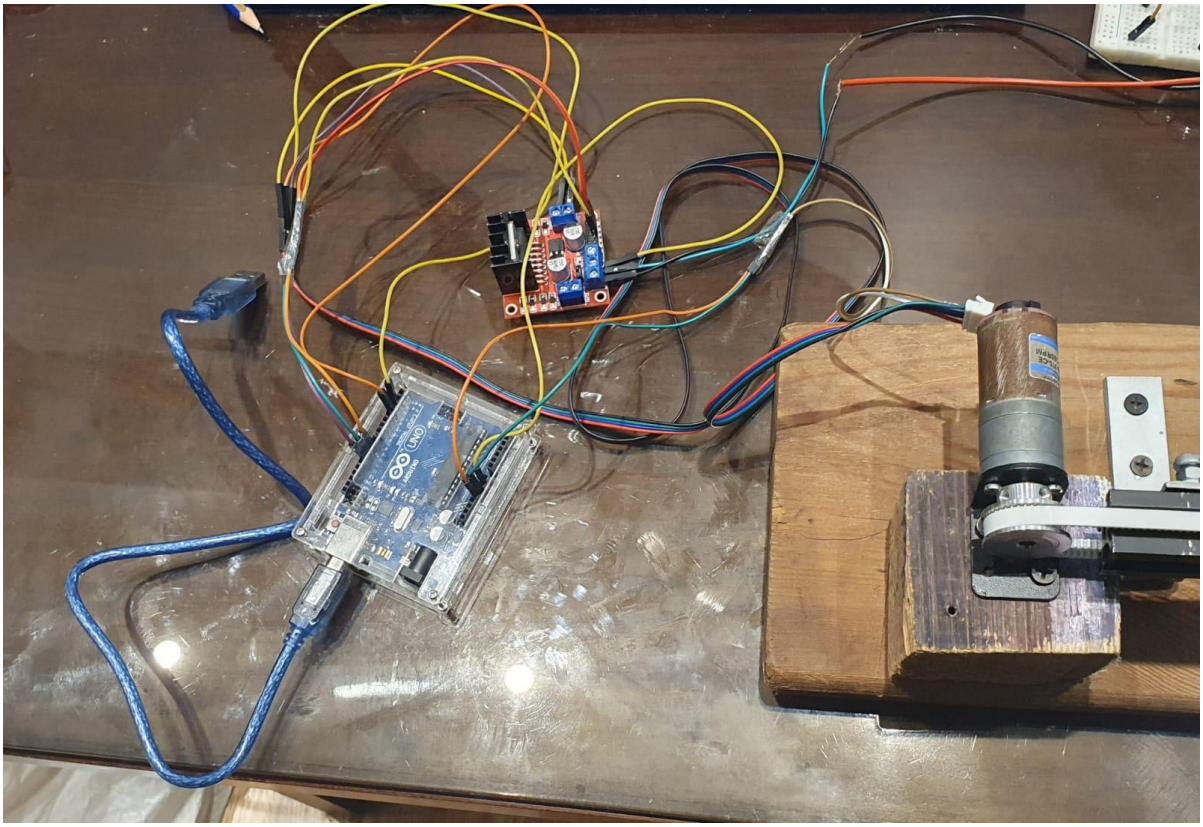
## Block Diagram:

Block Diagram:



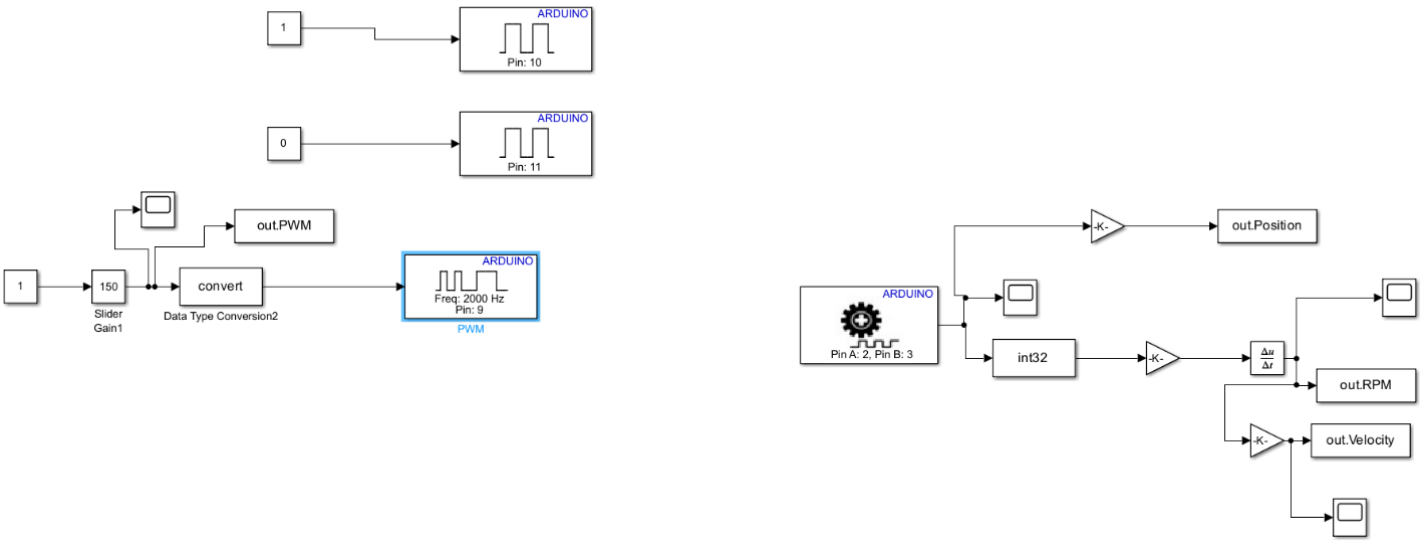
# General Steps to perform project:

## 1. Build & connect our system hardware components



## 2. Build a data acquisition model in Simulink

- The aim of this step is to get readings (pulses) from the encoder and then convert it into whatever I want (RPM, Linear Velocity, Linear Position, etc…). Below is our data acquisition model:




Data Acquisition Model

- With the aid of the Simulink Support Package for Arduino, the above data acquisition model was created. To start with, a PWM block was used, and this is the block which will send the PWM signal from the Arduino to the H-bridge Enable pin. A slider gain is connected to this block (or instead a constant block could've been used) to change the PWM.
- 2 digital output blocks have been used and are connected to 2 constant blocks of values 1 & 0. These digital blocks are just for the direction of rotation of motor. By using values of 1 & 0, the motor is made to rotate in a certain direction. These pins are 11 & 12 in our circuit in Arduino that'll be connected to IN1 & IN2 pins in Arduino.
- An encoder block is used to get readings from our physical encoder. This block the pins at which the phases of encoder are connected on Arduino are defined.

- ❖ To get RPM out of the encoder, we used the derivative block & a gain block of 60/(370*4). Derivative makes the output pulses/s. To convert pulses/s into RPM, multiply by 60 and divide by the encoder's pulses per revolution (ppr) multiplied by 4 since our encoder is a quadrature encoder (It uses 2 output channels to indicate the position of the motor)
- ❖ There are 2 To Workspace blocks (PWM & RPM). The data will be sent to Matlab Workspace as 2D arrays. These will be used to run the simulated model.

### *3. Create a simulation model similar to the real-life dynamic system (Include dead zones)*

- The simulation model must behave exactly like the real-life model. For this to occur, the dead zones in our system must be identified. Dead zone is the range of PWM at which the motor won't rotate when given these values.

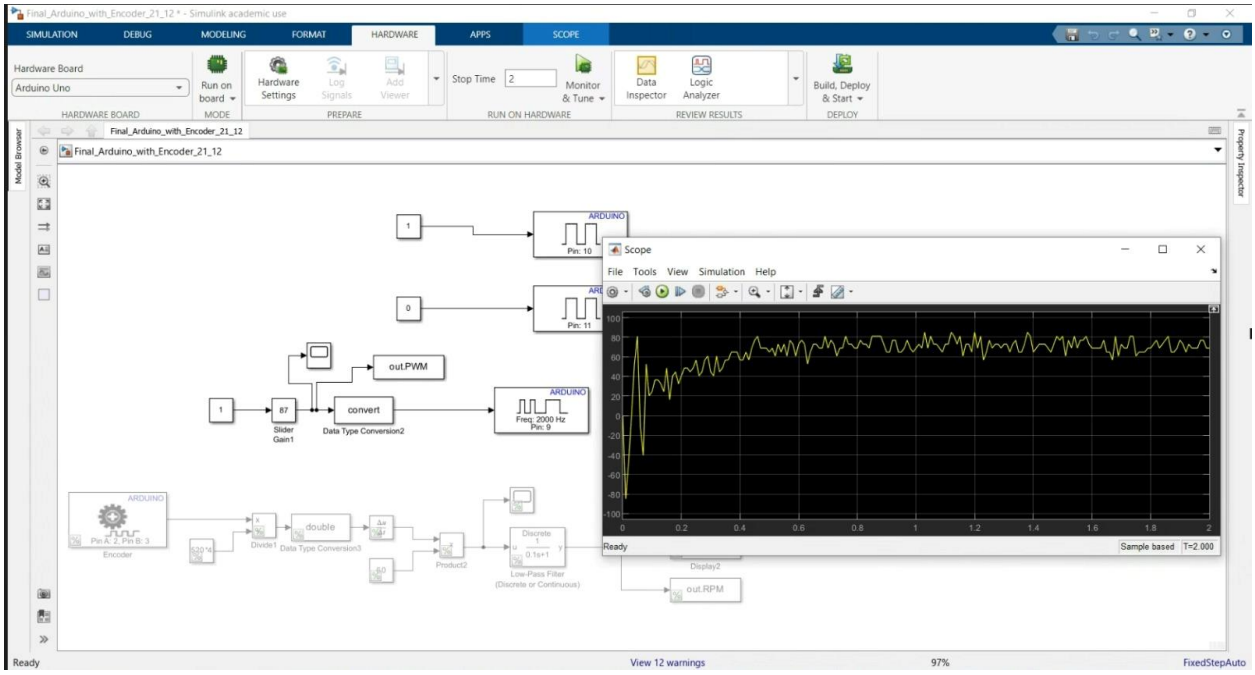
## DEAD Zones:

- There are two dead zones in our model. The greater value of dead zone will be inserted to our model, but we'll calculate both since their difference will be used to bring an initial value for Coulomb & Breakaway Friction forces which are inside the Translational Friction Block.

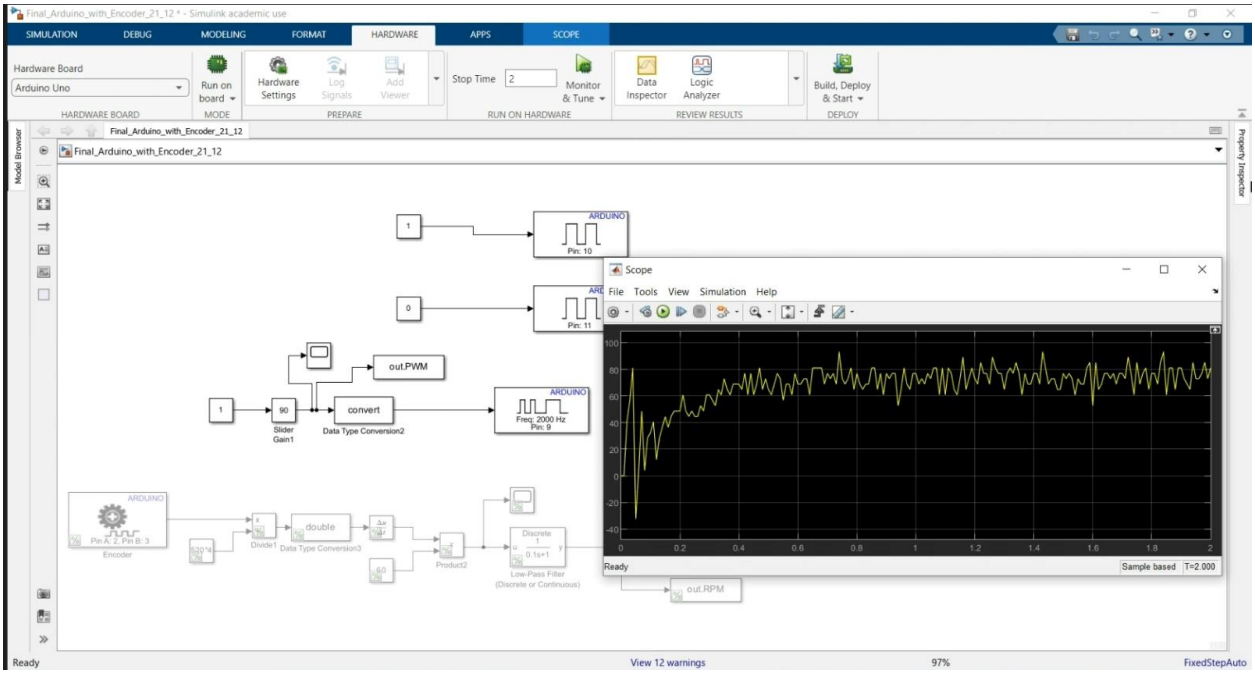
1. One will be calculated when the motor has no load nor friction. It is required to know the voltage needed to make the motor start rotation with no-load (Only connecting the H-bridge & Motor). We calculated it by applying different PWM on motor via H-bridge. Consequently, the value of PWM at which the motor will start rotation can be determined. The motor starts rotation at **81** PWM.

### ➢ Our sequence of testing:

- 90 PWM without load**:**




- 87 PWM without load:

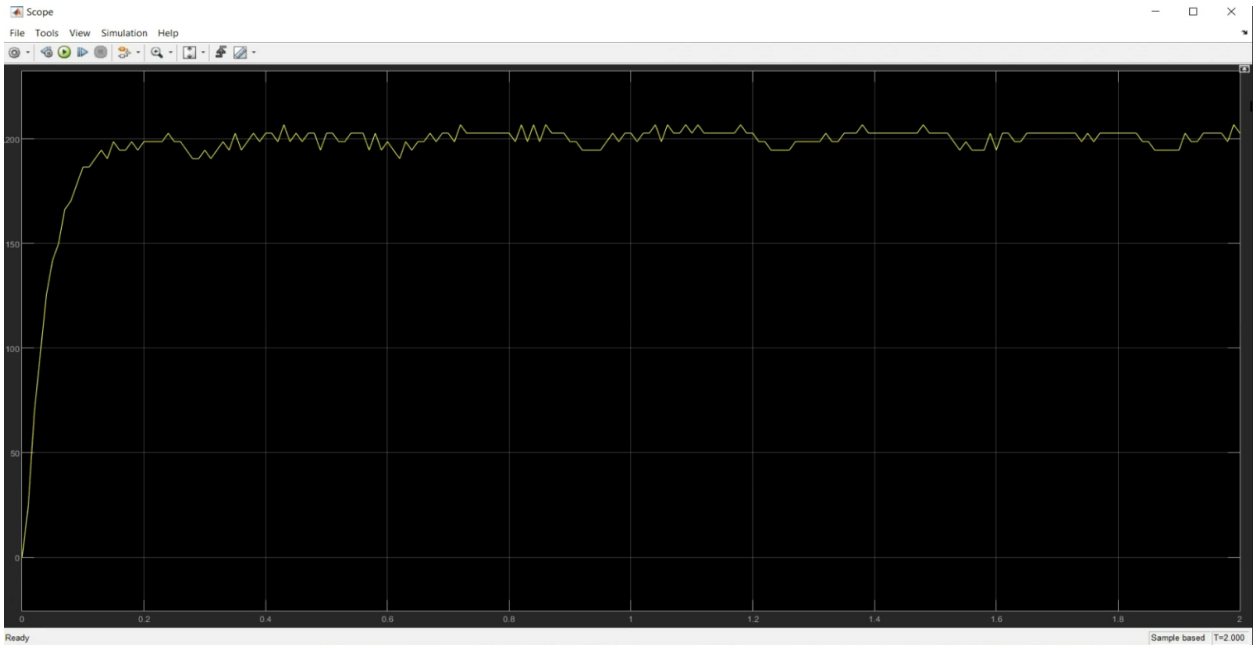



- At 86 PWM the motor rotated but, at 85 PWM it did not rotate and the same at 84 PWM. We thought that the right value at 85 PWM., but we

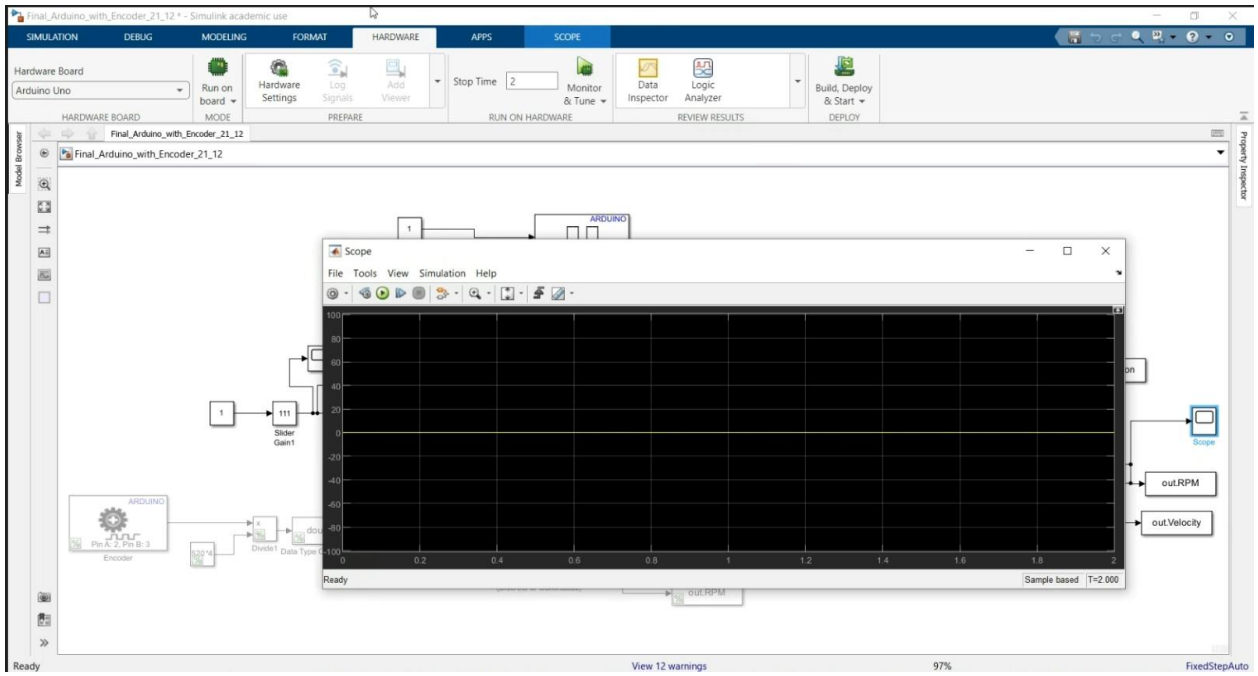
found out that the motor rotated at PWM 83 and PWM 82 but did not rotate at PWM 81. Eventually, we decided that 81 PWM is our first dead zone value

2. The other dead zone is to calculate friction, and we calculated it after putting the load on motor which is belt with the cart. We tested like the other dead zone by applying different input voltage. We found that it is at PWM **111**.

- At 255 PWM:




- At 111 PWM:




RPM=0, so this is the value of the second dead zone

- By subtracting the 2 dead zones, this is the change in PWM the friction caused. We will use that to calculate coulomb and breakaway friction forces and put them in transitional friction force block.

## Calculations:

- We need to bring initial values for $R_a$ (armature resistance) & $K_b$ (Back-emf Coefficient). A K.V.L is done (Inductance is very small so it's negligible). The following equation is reached:

$$E = IR + Kb * w$$

- This will be calculated @ 2 conditions: stall & no-load to bring $R_a$ & $K_b$

$$E = IR + 0$$

$$12 = 1.5 * R$$

$Ra = 8$ @ Stall Conditions (Stall Current=1.5 A & w=0)

$$E = IR + kb * w$$

$$12 = 0.1 * 8 + kb * 26.2$$

$Kb = 0.493$ @ Stall Conditions (No-load Current=0.1 A & No-load Speed = 250 Rpm = 26.2 rad/s

- Now, the difference in PWM in the dead zones is 30. This will be converted into voltage and will be substituted in the following equation to bring the friction force:

$$E = \frac{Ra}{Kb} * F * r$$

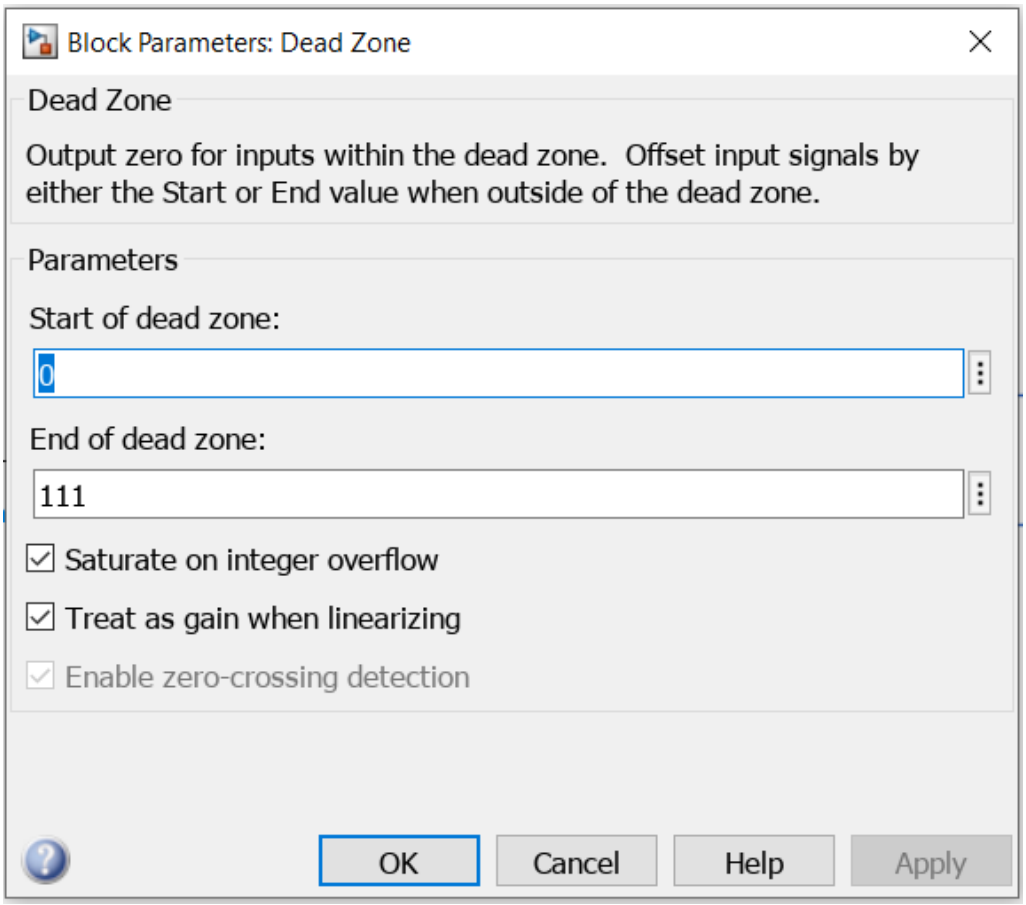
@ r=radius of pulley (0.011 m) & E=difference in PWM but in voltage

$$\frac{30}{255} * 12 = \frac{8}{0.493} * \mathrm{F} * 0.011$$

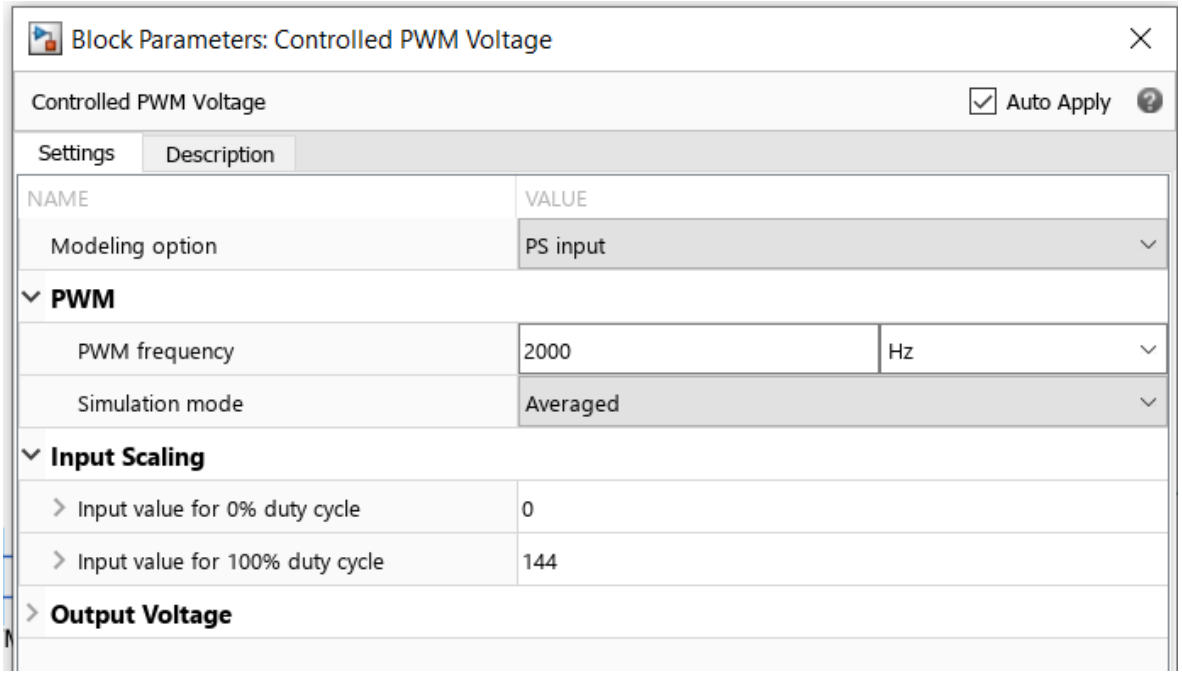
$F = 7.92$

## Notes about model:

- The input signal to this simulation is a constant PWM signal
- The dead zone block has an end of 111 PWM

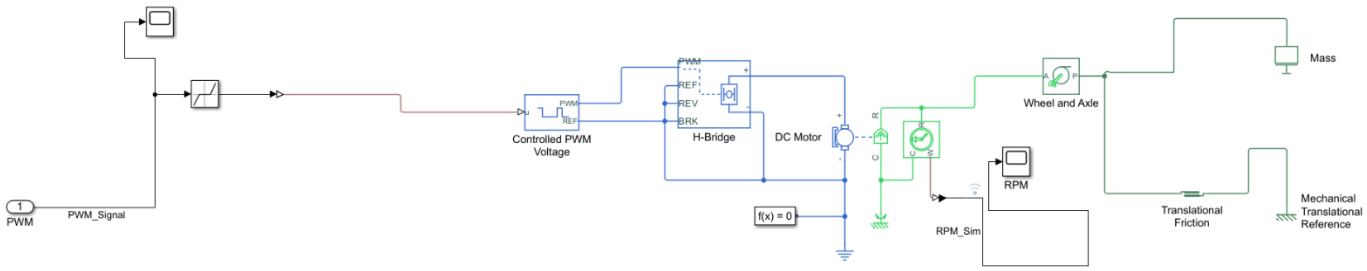



- The controlled PWM Voltage Block will have an input of 144 PWM for 100 % duty cycle (255-111)



- Wheel and axle block is to simulate the pulley and we determined the pulley's radius (1.1 cm)
- Mass block is the mass of the cart (107 g)
- We defined variables for the circuit's parameters and gave them initial values

## Final Simulation Model




### *4. Use parameter estimation to estimate the parameters of the system at one PWM signal*

1. Firstly, we monitor & tune the Data Acquisition Model. PWM & RPM signals will be sent to the simulation model.
2. Next, convert the PWM & RPM 2D arrays into timeseries and store them in a new variable. Moreover, create a time array with 0.01 step (Our model step-size) & a final value of the stop time beginning with 0 (This is to be able to convert the 2-D array into timeseries)
3. Thirdly, in simulation model, put and determine your input signal that'll operate the model (PWM signal). Also, initialize your variables.
4. Log the output signal to RPM signal generated from data acquisition model (So, simulation knows what the graph of RPM should look like)
5. Run the simulation (take care, that the stop time of simulation and data acquisition model must be identical)
6. Cut the 2D arrays of RPM & PWM to 0.4 seconds (as this is what we only want to parameter estimate on). Furthermore, create a new time array with the same step size of model but a final value of 0.4 (So later, the 2-D array can coincide with the time)

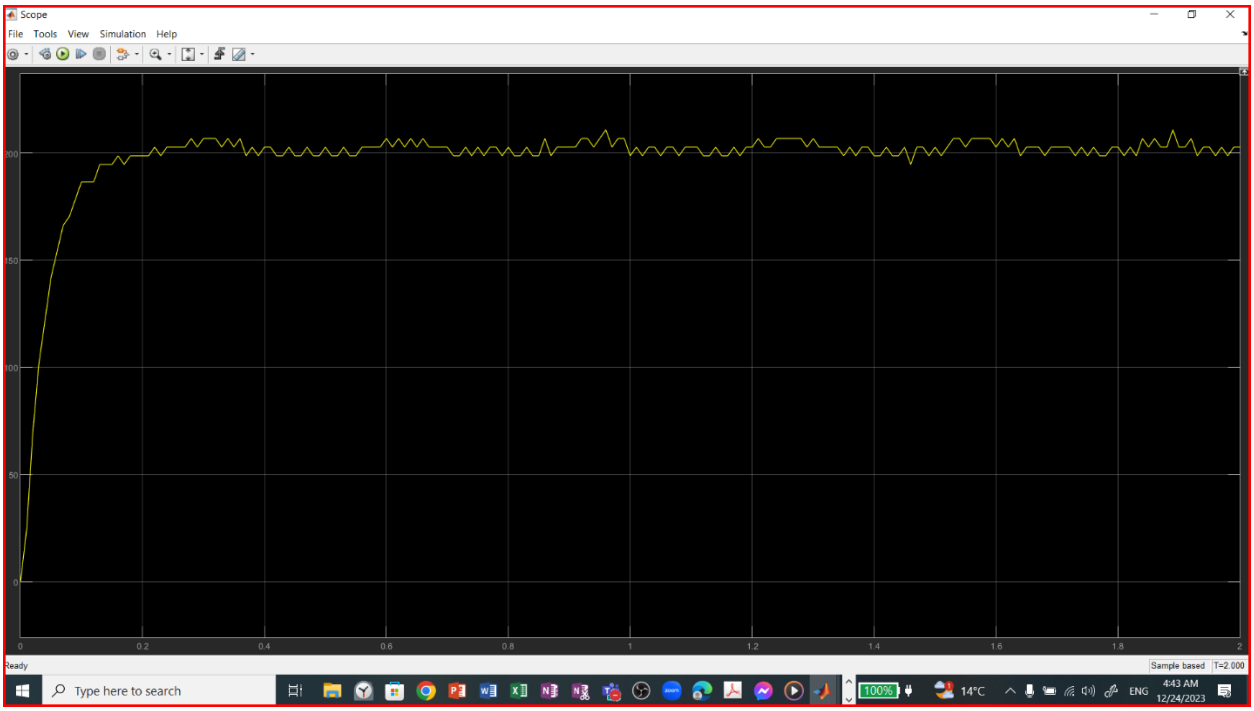
7. Open parameter estimator app and identify your input and output signals (data & time arrays)
8. Click plot & simulate to plot the simulated & measured graphs
9. Now, select the parameters you want to estimate, then click estimate
10. Eventually, the measured graph will coincide on the simulated graph

***We performed our parameter estimation on 255 PWM, and here are our results:***

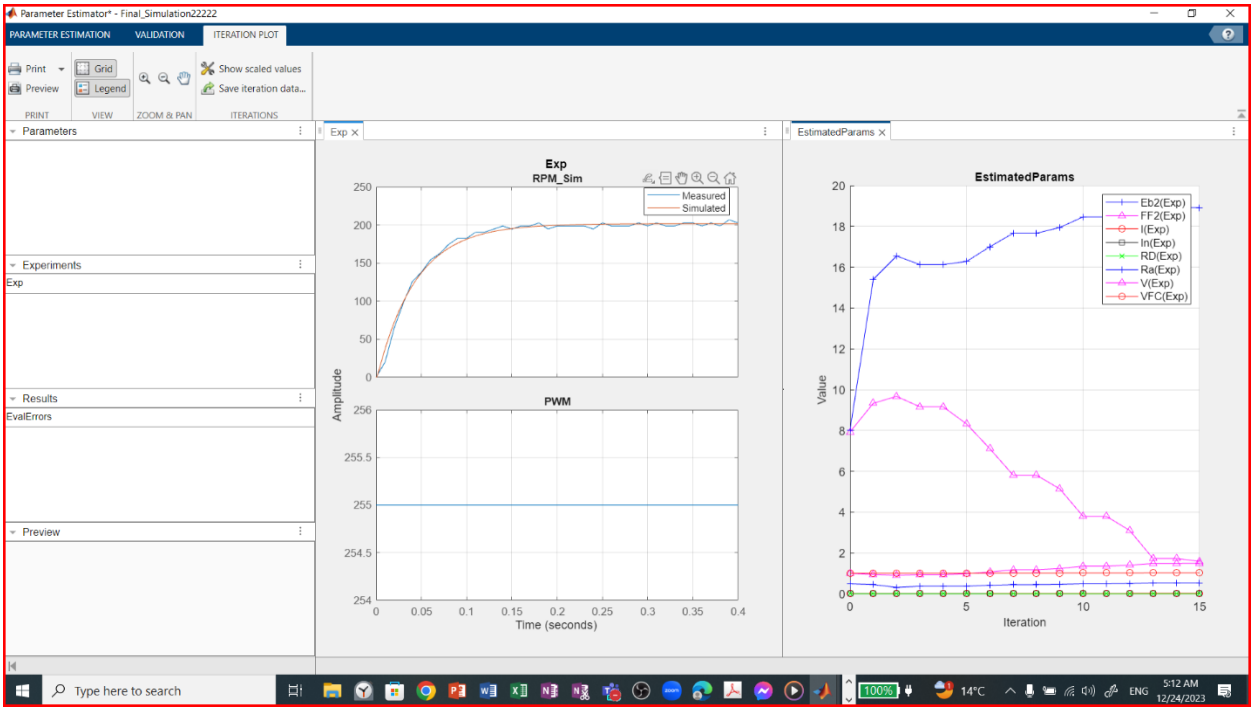
255 PWM

- Simulated graph


- Measured Graph

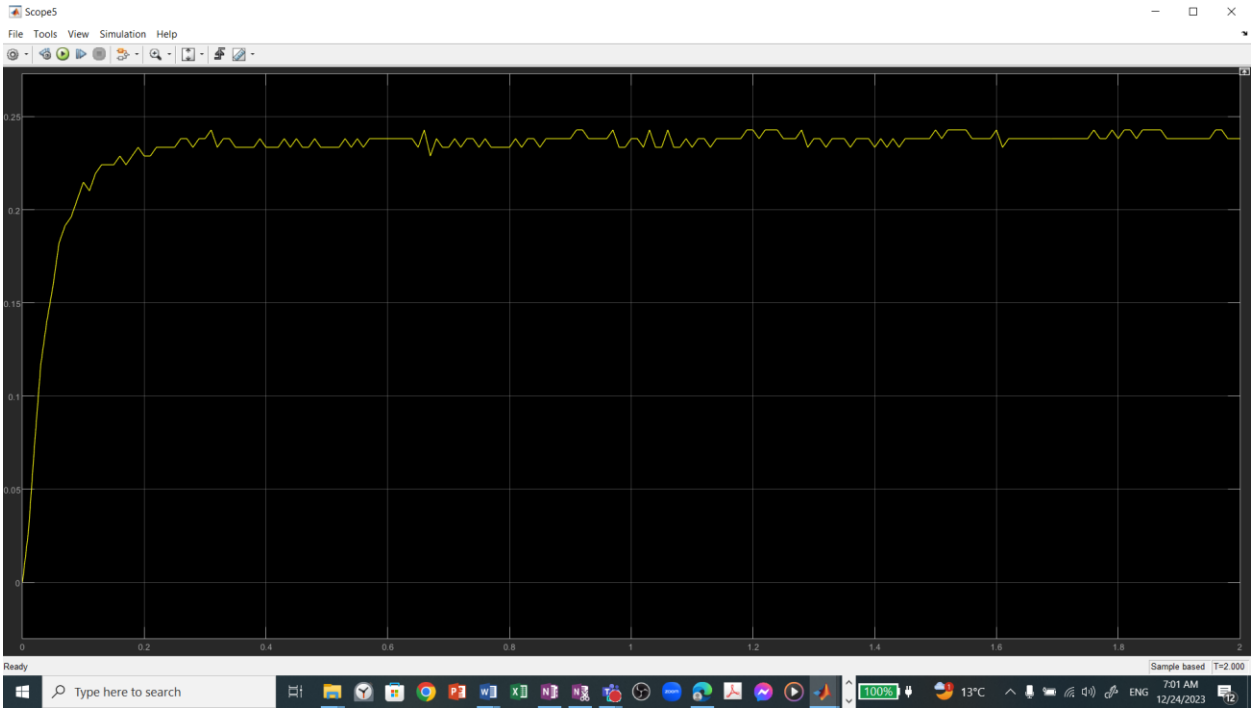



- Result after parameter estimation




- Since this is a 1 degree of freedom, by knowing the RPM or by making the simulation an exact copy of the measured, so will the linear velocity of the cart as well. Below is the result of the measured graph of the cart's linear velocity:

- Velocity




Velocity graph looks the same like the RPM graph (Multiply RPM by constant to bring linear velocity)

- Parameters estimated:

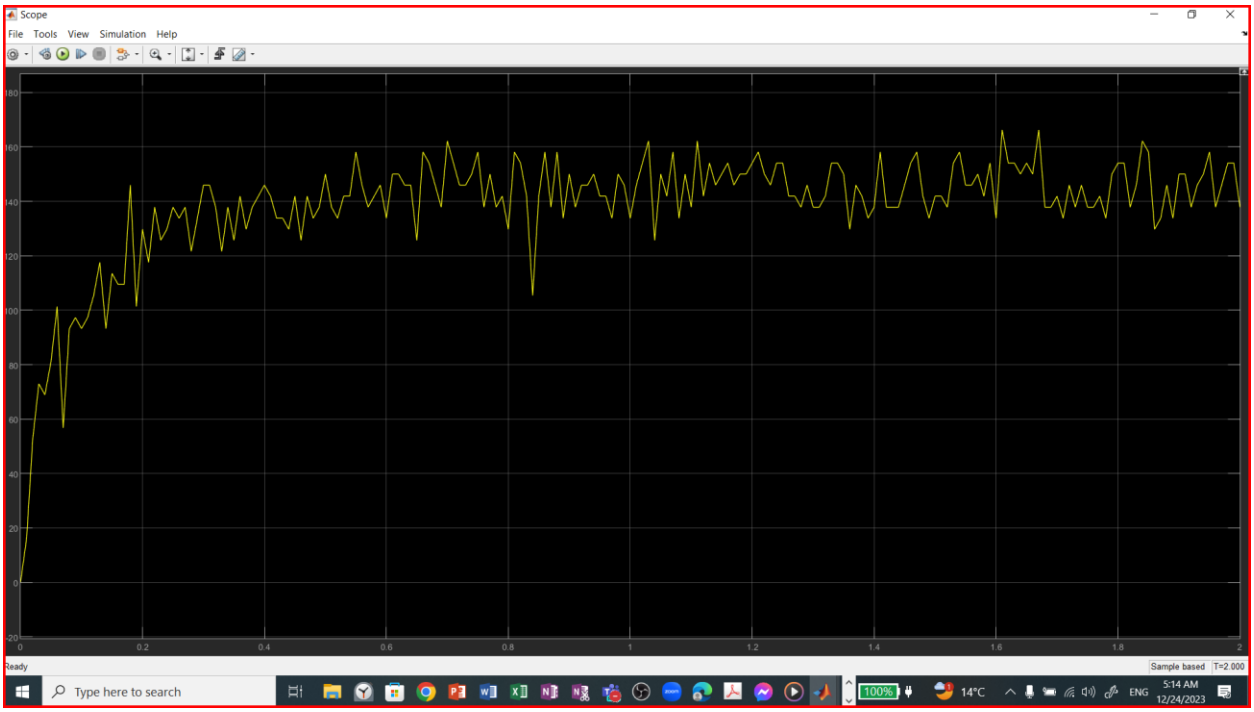
| Parameter | Initial Value |
|---|---|
| Eb2 | 0.536753678380987 |
| FF2 | 1.5907845922581 |
| I | 0.0235727851988216 |
| In | 0.000681256412882049 |
| RD | 5.01200021054277e-06 |
| Ra | 18.9159727588287 |
| V | 1.48833776582249 |
| VFC | 1.02494560752655 |

## 5. Try at different PWMs, the measured graph and simulated graph should coincide on each other

- Now, the parameters are determined, so when trying out different PWMs, the measured & simulated graphs should be the same like each other
- Here are some of our results:

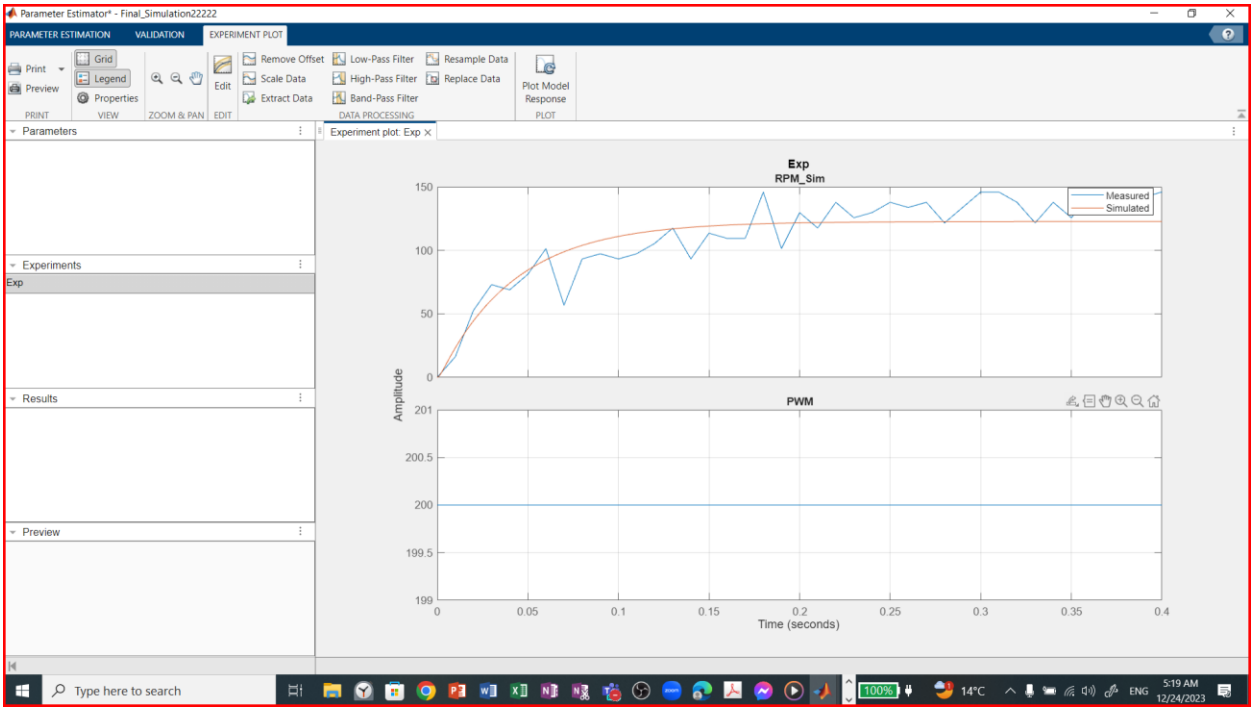
### ❖ 200 PWM

- Measured Graph

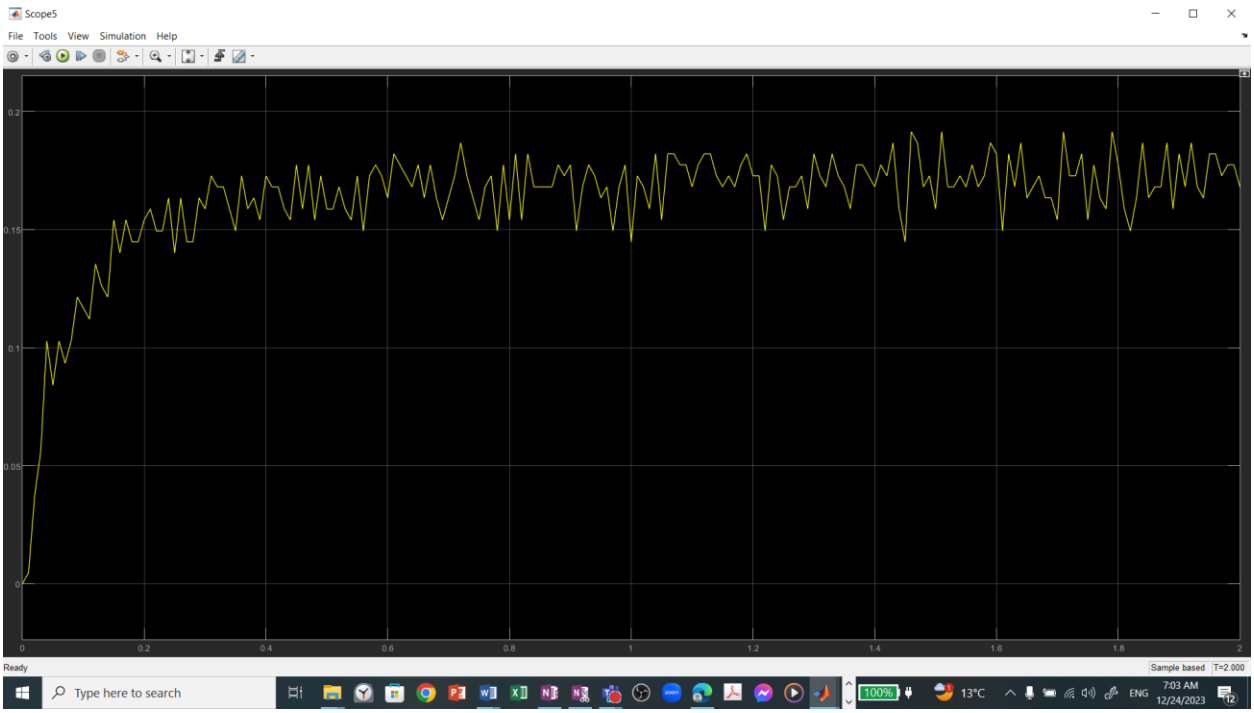



- Simulated Graph


- Measured & Simulated Graphs (200 PWM)

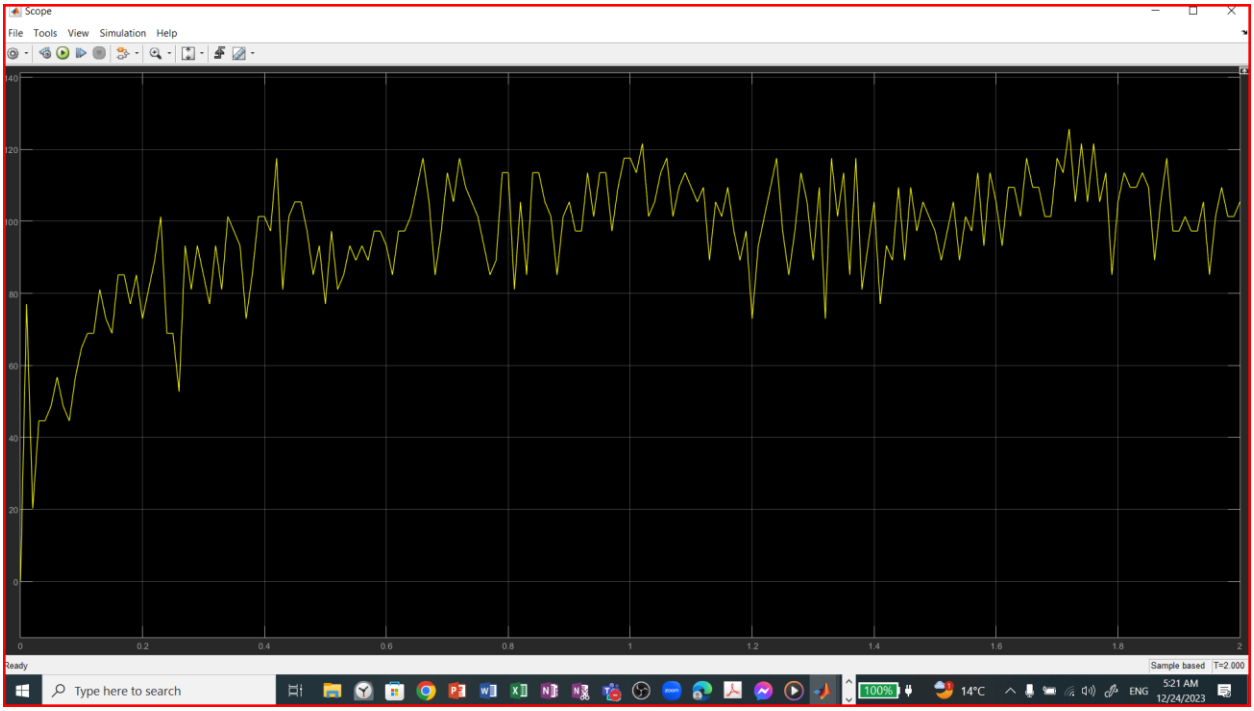



- Velocity



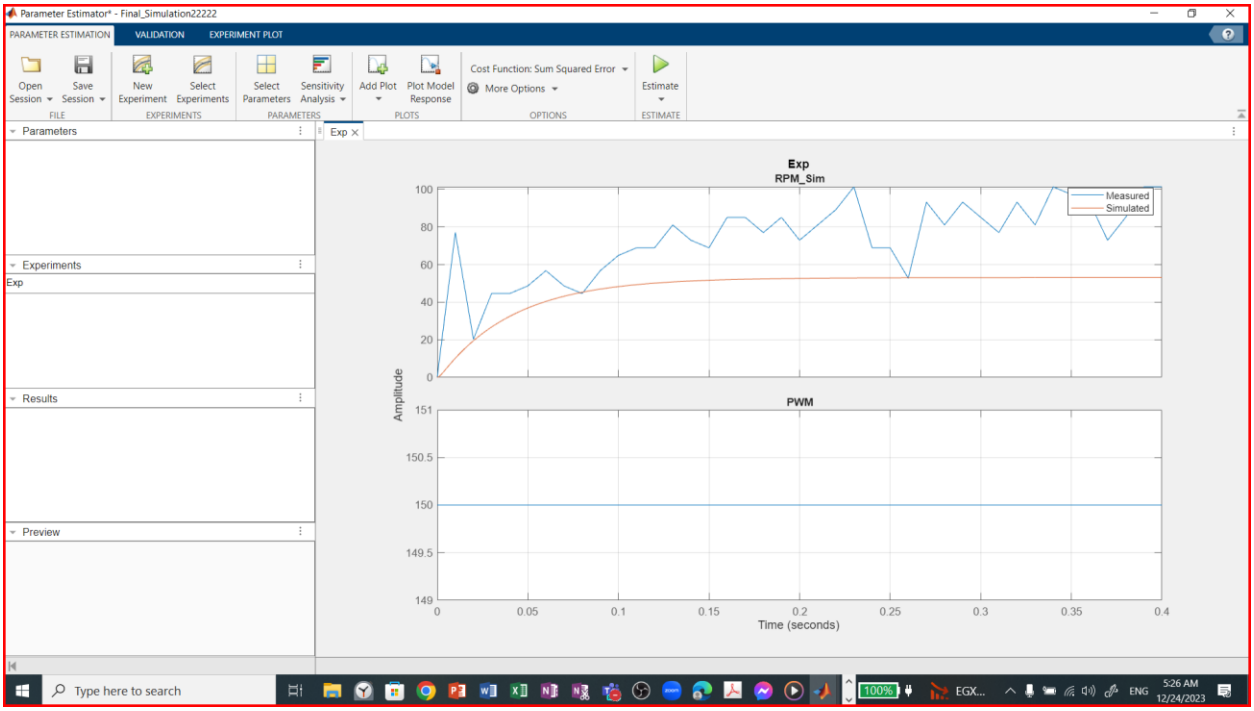

## ❖ 150 PWM

- Measured Graph



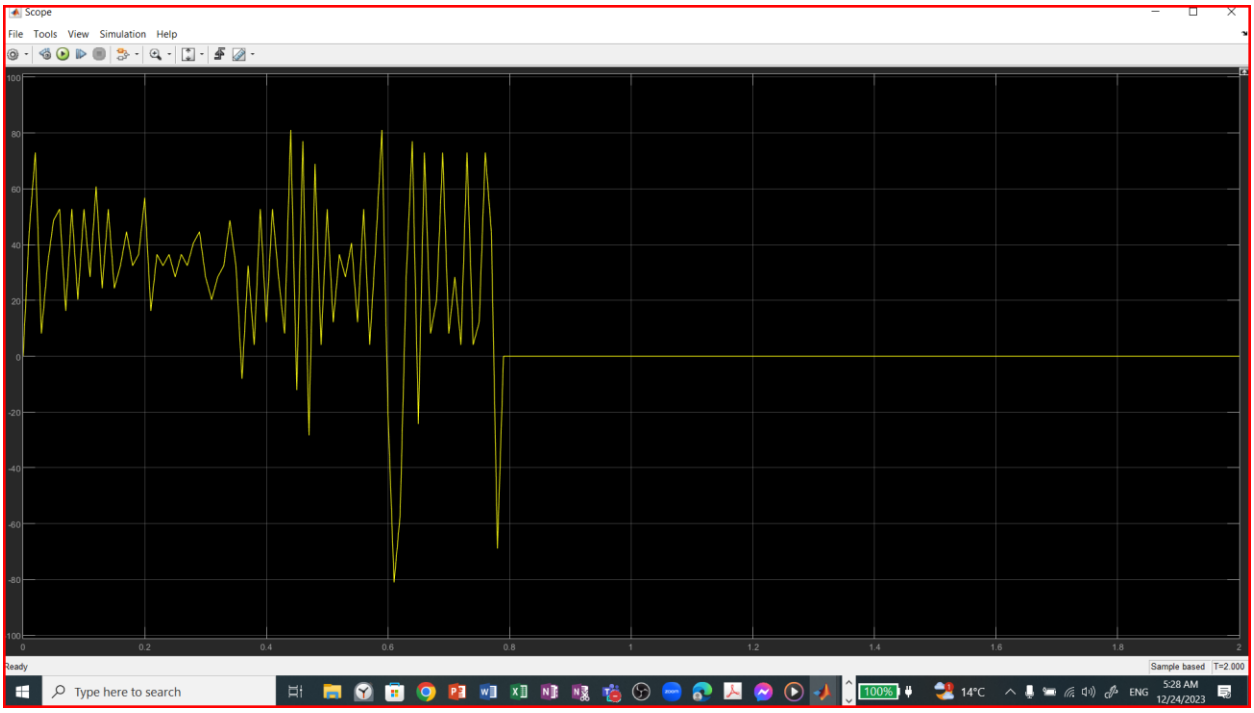

- Simulated Graph


- Graphs without parameter estimation



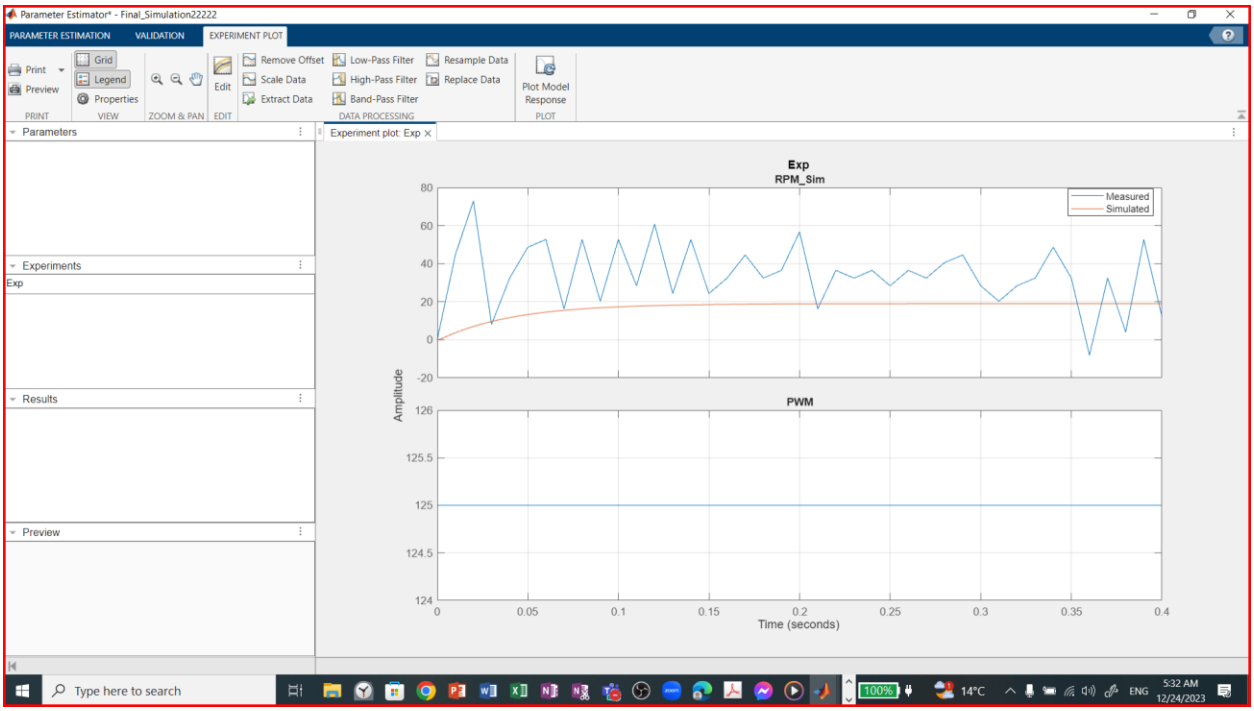
## ❖ 125 PWM

- Measured Graph



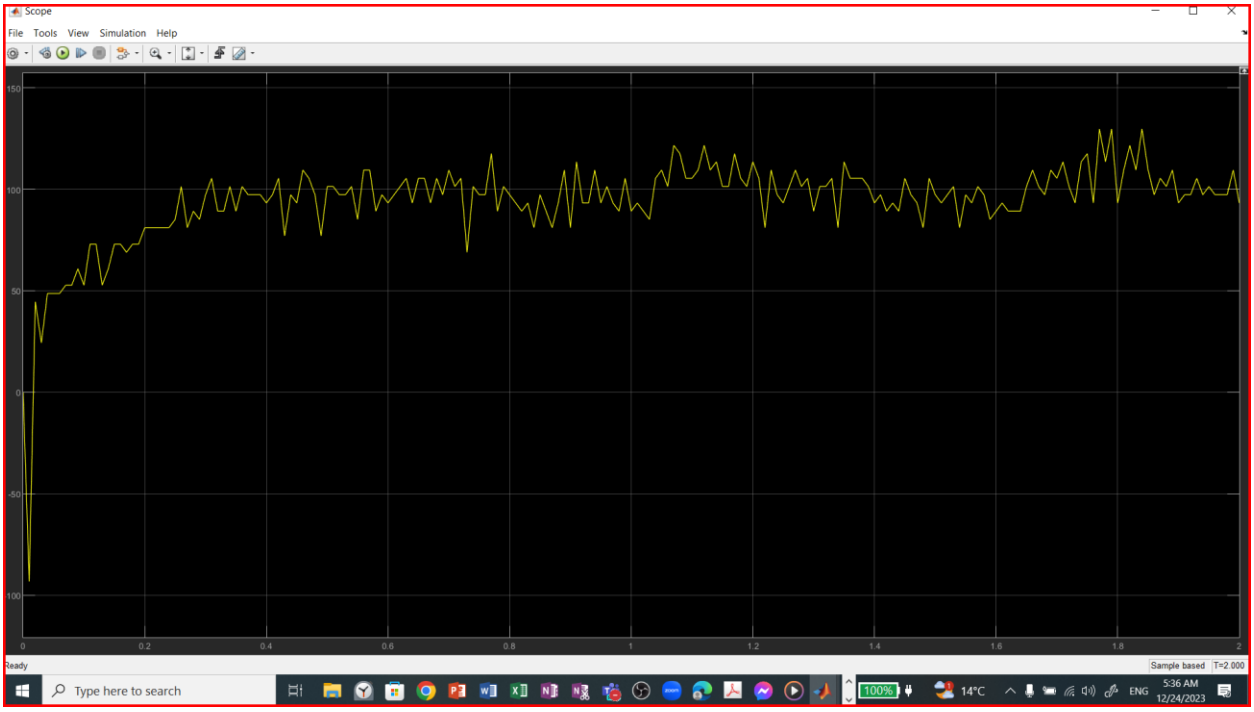

- Simulated Graph


- Graphs without parameter estimation:

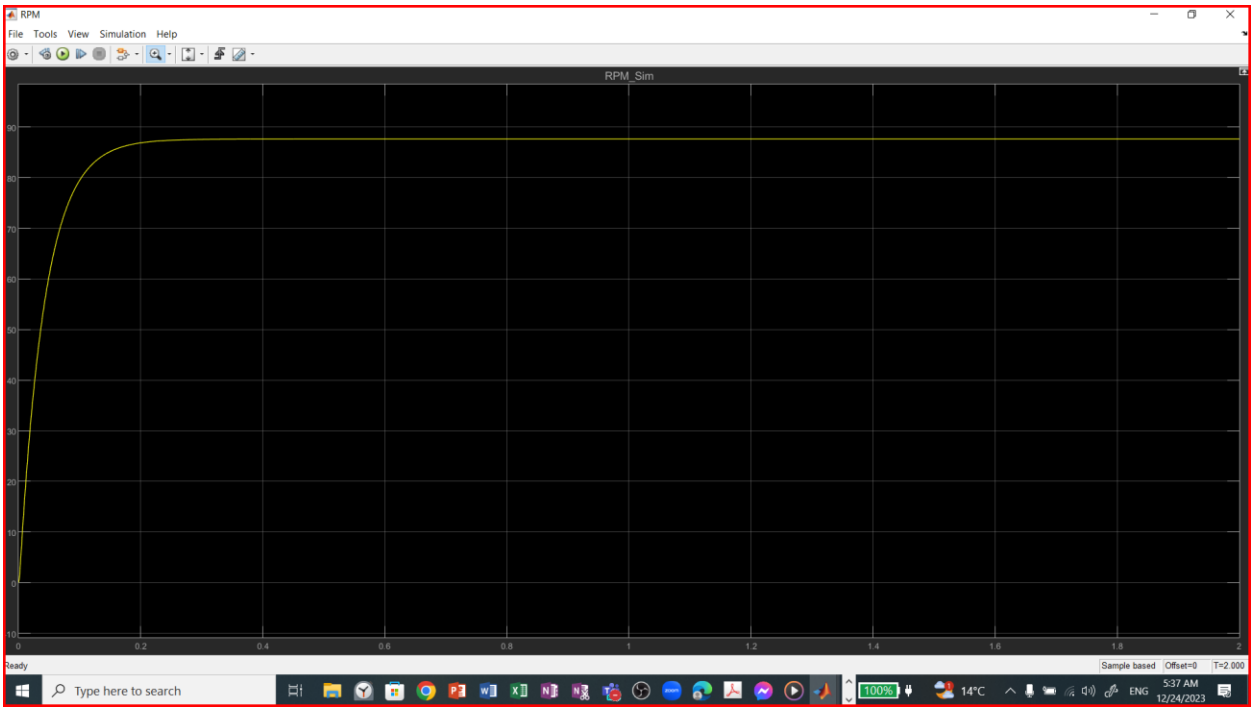



## ❖ 175 PWM

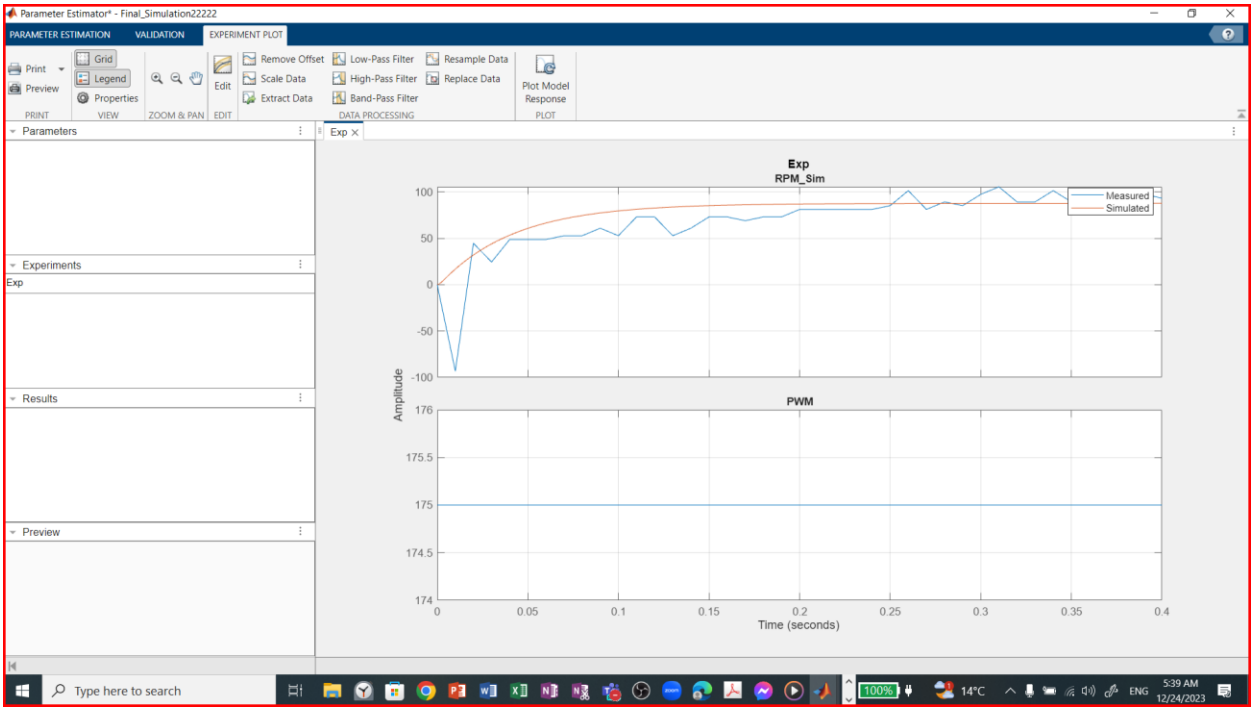
- Measured Graph




- Simulated Graph

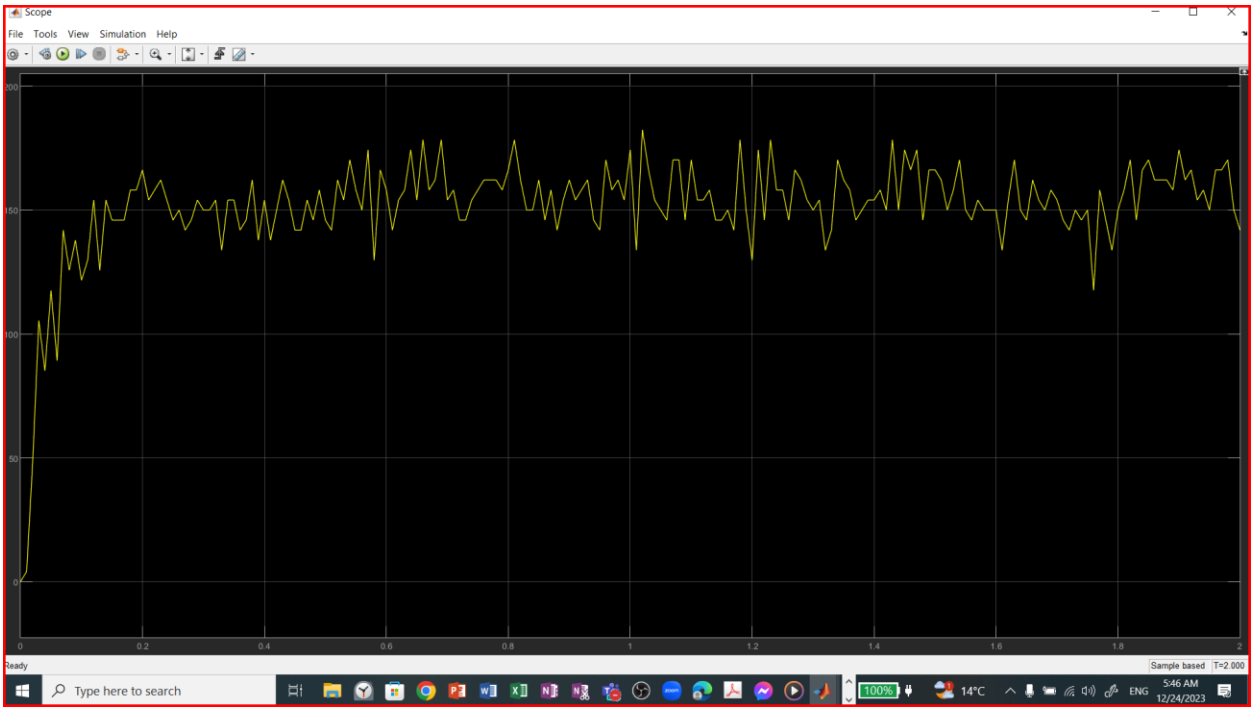



- Graphs without estimation:

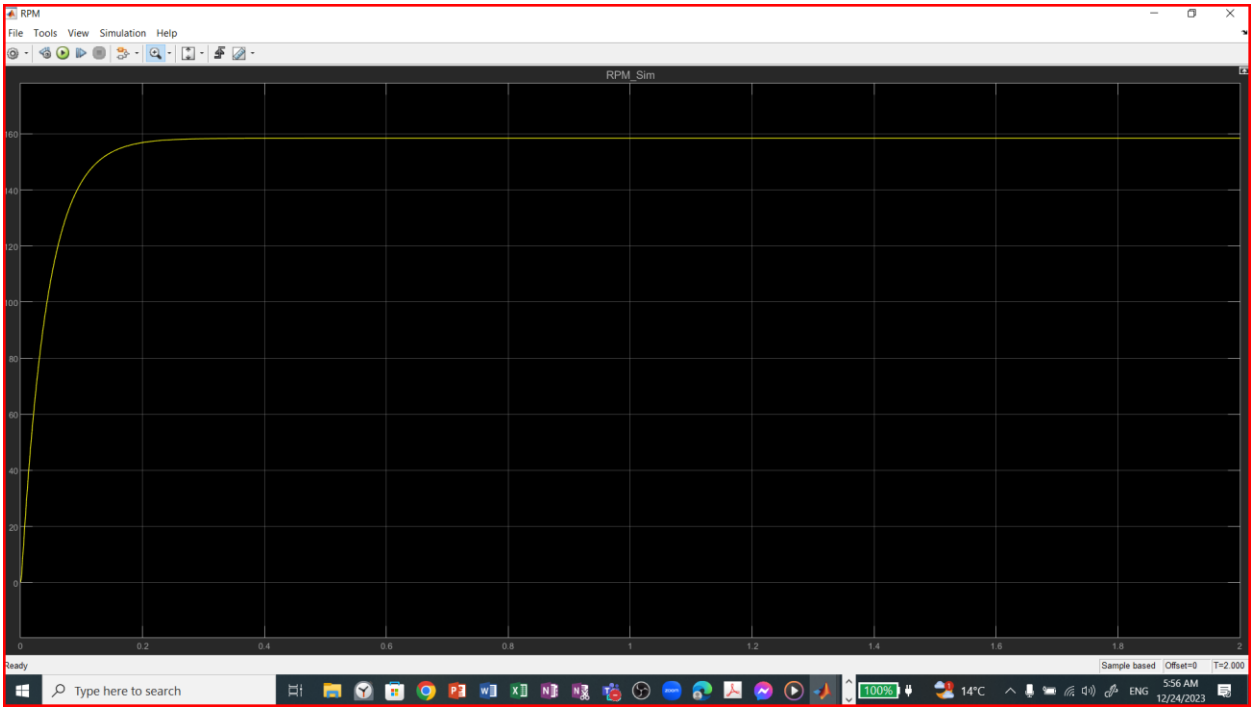



## ❖ 225 PWM

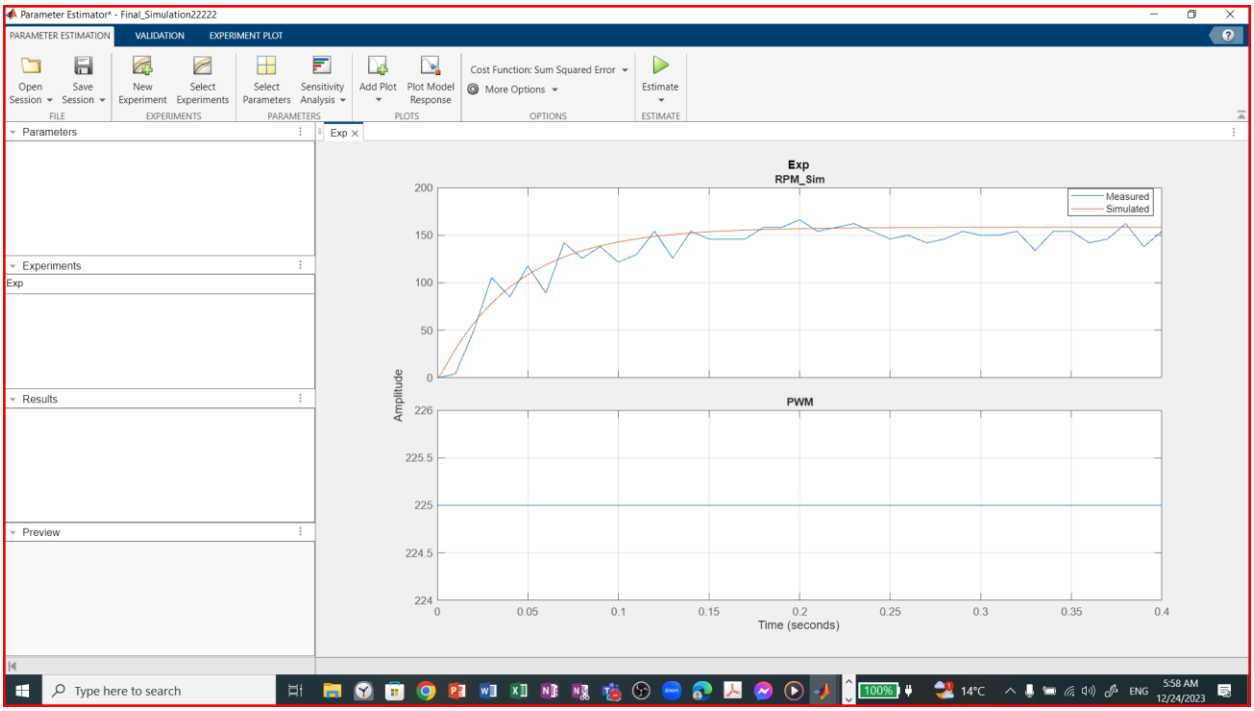
- Measured Graph




- Simulated Graph:




- Graphs without estimation:




## Comparing results of physical model & simulation

Overall, at high PWMs, both graphs look identical to each other with the same system parameters. But at low PWMs, they don't exactly coincide to each other, but the graphs are very close. This maybe be to at low PWMs, noise has greater effect in the readings. Eventually, the parameter estimation succeeded in making the simulation model resemble our physical model.

## Conclusion

After putting in a lot of effort, we faced many challenges in our project. We had to deal with problems like choosing the right settings like step size and solver settings for our calculations to get accurate results. One big issue was that the encoder and motor we used. There accuracy was not good. They had a lot of errors, making the readings jump around a lot. The fluctuation was very high as well as quantization was very high, we worked hard to fix this and make things smoother in our main model. Even though the encoder wasn't very accurate, we eventually made our simulation response similar to the real dynamic system response. We used MATLAB well, adding features like a dead zone and figuring out parameters. We also tested different inputs

to see how they affected the output. In the end, despite the challenges, we succeeded in making our simulation behave like our physical system.

Drive Link for All source files:
https://drive.google.com/drive/folders/1hCYYkQOMATr2RnHjz5u8DDOJz7kCzqXK?usp=sharing