R.V. COLLEGE OF ENGINEERING

BENGALURU- 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF AEROSPACE ENGINEERING

Fundamentals of Avionics

21AS45

Experiential Learning Reports

Submitted by

Suhana Arsh
Sadiq Ali Mir
Veerabhadrayya C Roogi

Under the Guidance of
Group Captain Deepak Bana VSM (Retd)
Visiting Professor,

RV COLLEGE OF ENGINEERING®, BENGALURU- 560059

in partial fulfillment for the award of degree of

Bachelor of Engineering
IN
Aerospace Engineering

RV COLLEGE OF ENGINEERING

BENGALURU-59
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF AEROSPACE ENGINEERING

DECLARATION

We, the students of **IV Semester B.E Aerospace Engineering, Batch 2022-23** hereby declare that the Experiential Learning for the course of **"Fundamentals of Avionics (21AS45)"** has been carried out by us and submitted in partial fulfillment for the award of degree of **Bachelor of Engineering** m **Aerospace Engineering** during the year 2022-23

Signature of Course Co-ordinator Gp Capt Deepak Bana VSM (Retd)
Visiting Professor,
Department of Aerospace Engineering

Signature of Head of the Department
Dr. Ravindra S Kulkarni
Professor and Head,
Department of Aerospace Engineering

Signature of Principal
Dr.K.N.Subramanya
RV College of Engineering

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise, does not depend solely on individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this Experiential Learning project work. We would like to take this opportunity to thank them all.

We deeply express our sincere gratitude to our Experiential Learning project guide **Gp Capt Deepak Bana VSM (Retd), Visiting Professor,** Department of Aerospace Engineering, RV College of Engineering®, Bengaluru for their able guidance, regular source of encouragement and assistance throughout this Experiential Learning project.

We would like to thank **Dr. Ravindra S Kulkarni,** Professor and Head, Department of Aerospace Engineering, RV College of Engineering®, Bengaluru, for his valuable suggestions.

We would also like to thank **Dr. K N Subramanya,** Principal, RV College of Engineering®, Bengaluru, for his moral support towards completing our project work.

We thank our Parents, and all the faculty members of the Department of Aerospace Engineering for their constant support and encouragement. Last, but not the least, we would like to thank our peers and friends who provide valuable suggestions during this project.

List of IV Semester Students		
Deptt of Aerospace Engg		
S.No	USN	Name of Student
	1RV21AS002	Abin
	1RV21AS003	Abyan Raidh T
	1RV21AS004	Adithya D R
	1RV21AS005	Akash P
	1RV21AS006	Ambuja Bamane
	1RV21AS007	Anagha Udupa
	1RV21AS009	Anisha Bhattacharya
	1RV21AS010	Ankitha Shet
	1RV21AS011	Anubhuti B P Mishra
	1RV21AS012	Aasim Asgar Baqi
	1RV21AS013	Avinash
	1RV21AS014	Chaitanya Dev Sehdev
	1RV21AS015	Chyawan Chandrashekar
	1RV21AS016	Eshwari B N
	1RV21AS017	K Sharanya
	1RV21AS017	G R Tarun
	1RV21AS018	Gautam M
	1RV21AS019	Greekshith Mahesh Babu
	1RV21AS020	Krishna Anwita
	1RV21AS021	Inchara Nataraj
	1RV21AS022	Jagadeesh vijapur
	1RV21AS023	Jeevita
	1RV21AS025	Keshav Jindal
	1RV21AS026	Kishor A

	1RV21AS027	Krish Dhankar	
	1RV21AS028	Krush Machhi	
	1RV21AS029	Lakshmi Manasa C N	
	1RV21AS030	M.Raghuvansh	
	1RV21AS031	Mohammed Hifzaan	
	1RV21AS032	Mohammad Umar	
	1RV21AS033	Nagegowda RS	
	1RV21AS034	Naushik maurya	
	1RV21AS035	Nemani Satya Srikar	
	1RV21AS036	Nivedita Y	
	1RV21AS037	Om Kiritbhai Daxini	
	1RV21AS038	Parikshith pm	
	1RV21AS039	Prajval B Gowda	
	1RV21AS040	Prajwal G Angadi	
	1RV21AS041	Prajwal N	
	1RV21AS042	Prakhyath Y B	
	1RV21AS043	Prasad Venkat Pravith Singh	
	1RV21AS044	Pratik B Matt	

	1RV21AS045	**Rathnakar D**

	1RV21AS046	Rishab Satish
	1RV21AS047	Robin Ricky JB
	1RV21AS048	S Nikhil Bharadwaj
	1RV21AS049	Sadiq Ali Mir
	1RV21AS050	Sahana Vaddi
	1RV21AS051	Satej Patil
	1RV21AS052	Shaifali Arora
	1RV21AS053	Siddhanth S Ramaswamy
	1RV21AS054	Siddharth Varshney
	1RV21AS055	Solomon Suhas D'Costa
	1RV21AS057	Sree Valli T S
	1RV21AS058	Sreeharsh U
	1RV21AS059	Tanaya S
	1RV21AS061	Utsav Madhvendra Mehta
	1RV21AS062	Vaibhavi Mokashi
	1RV21AS063	Veerabhadrayya C Roogi
	1RV21AS064	Vishal Hugar
	1RV21AS065	Tarun Ghorpade
	1RV22AS400	Adithya RB
	1RV22AS401	Manjunath D
	1RV22AS402	Parth Patel
	1RV22AS403	Pavangowda R
	1RV22AS404	Shreedhara L
	1RV22AS405	Suhana Arsh

Contents

EXPERIENTIAL LEARNING – TOPICS & GROUPS		
1	**Group – 1**	Using Simulink, design an Electrical Actuator operating system for two electrical actuators with linear extension and retraction in synchronization. Develop the hardware model that can be put into practice.
2	**Group – 2**	Using a microcontroller, design a physical system to track the path of Sun and position the solar panel by X & Y movement. Develop the hardware model that can be put into practice.
3	**Group – 3**	Using Simulink, design an Electrical Actuator operating system for two electrical actuators providing rotation of 90 degree to the actuator arm with one actuator rotating in clockwise 90 degree and other anti-clockwise 90 degree. Develop the hardware model that can be put into practice.
4	**Group – 4**	Using Simulink, design an ambient light operated circuit that switches ON/OFF the outside lights. Develop the hardware model that can be put into practice.
5	**Group – 5**	Using Simulink, design a microcontroller operated Brush Less DC motor control to rotate two motors simultaneously such that one motor rotates in clockwise and second motor rotates in anti-clockwise direction. Develop the hardware model that can be put into practice.
6	**Group – 6**	Using Simulink, create a practical model for Quadcopter that maintains its attitude straight and level. Develop the hardware model that can be put into practice
7	**Group – 7**	Using Simulink, design an IR sensor-based Obstacle Avoidance System for an Unmanned Aerial Vehicle. Develop the hardware model that can be put into practice.
8	**Group – 8**	Using Simulink, design a Hydraulic actuator operating at 200 bar hydraulic pressure having linear extension and retraction of 15 cms. Develop the hardware model that can be put into practice.
9	**Group – 9**	Using low level ‘C’ or Embedded ‘C’, write a program for microcontroller to control the linear as well as rotary movement of electrical actuator. Develop the hardware model that can be put into practice.
10	**Group – 10**	Using a microcontroller, design an Electrical Actuator operating system for two electrical actuators providing rotation of 90 degree to the actuator arm with one actuator rotating in clockwise 90 degree and other anti-clockwise 90 degree. Develop the hardware model that can be put into practice.
11	**Group – 11**	Using a microcontroller, design an Electrical Actuator operating system for two electrical actuators with linear extension and retraction in synchronization. Develop the hardware model that can be put into practice.
12	**Group – 12**	Using Simulink, design an IR sensor-based microcontroller operated Obstacle Avoidance System for a ground vehicle so that it avoids all obstacles and keep moving safely. Develop the hardware model that can be put into practice.
13	**Group – 13**	Using Simulink, design a microcontroller operated Brush Less DC motor speed control to rotate four motors using a joy stick such that two motors rotate in clockwise and other two motor rotate in anti-clockwise direction Develop the hardware model that can be put into practice.

14	**Group – 14**	Using Simulink, design a LIDAR sensor-based Obstacle Avoidance System for a ground vehicle. Develop the hardware model that can be put into practice.
15	**Group – 15**	Using Simulink, design an Ultrasonic sensor-based Obstacle Avoidance System for a ground vehicle. Develop the hardware model that can be put into practice.
16	**Group – 16**	Using low level 'C' or Embedded 'C', write a program for microcontroller to control the RPM of a motor. Develop the hardware model that can be put into practice.
17	**Group – 17**	Using Simulink, create a practical model for underwater drone that maintains its attitude straight and level underwater. Develop the hardware model that can be put into practice.
18	**Group – 18**	Using Simulink, design a microcontroller operated Brush Less DC motor control to rotate two motors in the same direction simultaneously. Develop the hardware model that can be put into practice.
19	**Group – 19**	Using a microcontroller, design a practical system to rotate two motors in the same direction, opposite direction and one motor in clockwise and second motor in anti-clockwise direction. Develop the hardware model that can be put into practice.
20	**Group – 20**	Using Simulink, design an Ultrasonic sensor-based Obstacle Avoidance System for a quad copter /UAV. Develop the hardware model that can be put into practice.
21	**Group – 21**	Using Simulink, design a practical model for quadcopter to retract its four arms, stop all motors and become a rectangular block that is encapsulated in a sealed compartment, and finally system should unfold and become ready to take off on command
22	**Group – 22**	Using Simulink, design a Sun tracking system for solar panels with movements along X & Y axes such that the panels give maximum current all the time. Develop the hardware model that can be put into practice.
23	**Group - 23**	Using Simulink, design a LIDAR sensor-based Obstacle Avoidance System for a quad copter /UAV. Develop the hardware model that can be put into practice.

Electrical actuator system Overview

In this experiential learning endeavor, the focus lies on the hands-on exploration of designing an Electrical Actuator Operating System. The primary objective is to gain practical insights into the intricacies of engineering a system that orchestrates the synchronized linear extension and retraction of two actuators, all under the control of a microcontroller. The process commences with the deliberate selection of a suitable microcontroller to serve as the system's nerve center. Subsequently, a commercially available electrical actuator, characterized by its specific linear extension and retraction capabilities (extending 15 cm while retracting 10 cm within this range), forms the core component. Thorough scrutiny of available components, accompanied by a meticulous understanding of their technical specifications, ensures their seamless integration into the system. A pragmatic power supply, often a battery, is chosen to cater to the actuator's energy needs, with comprehensive power consumption calculations conducted for both extension and retraction phases. This experiential learning opportunity provides a holistic and immersive approach to comprehending the complexities of designing and implementing an efficient and synchronized Electrical Actuator Operating System. The specified requirements dictate a linear extension range of 15 cm, with a retraction capability of 10 cm, culminating in a total extension capacity of 15 cm, of which 10 cm is allocated for retraction.

1. Problem Definition

The challenge at hand involves the creation of an Electrical Actuator Extension and Retraction System, effectively controlled by a microcontroller. This undertaking comprises two main objectives: firstly, the judicious selection of a microcontroller capable of adeptly coordinating the actuator's movements; and secondly, the acquisition of an appropriate electrical actuator possessing specific linear extension and retraction capabilities. The chosen actuator must be readily available in the commercial market and possess the capability to extend linearly by 15 cm, while also demonstrating the capacity to retract by 10 cm from the total 15 cm extension length. This prudent selection forms the bedrock of the system's functionality.
Moreover, the selection and integration of components represent a critical aspect of this endeavor. This necessitates an in-depth understanding of the technical parameters of each component to ensure their seamless incorporation into the system. The components must not only align with the actuator's operational requirements but also adhere to practical availability constraints. This process requires a discerning eye for detail and a sound grasp of electrical and mechanical specifications.

A practical power supply, commonly in the form of a battery, must be chosen to provide the requisite electrical energy for the actuator's operation. This selection requires consideration of factors such as voltage output, current capacity, and overall power efficiency. Additionally, rigorous calculations are imperative to determine the maximum power consumption during both the extension and retraction phases, thereby ensuring the chosen power supply is capable of meeting these demands.

In the subsequent phase, the theoretical design will be transformed into a tangible, hardware-based system. This step involves the acquisition and integration of commercially available components and/or kits. The hardware design process demands a judicious selection of components, accurate wiring, and meticulous assembly to bring the system to life. This hands-on phase is integral to bridging theoretical knowledge with practical implementation, providing a holistic learning experience in the realm of electrical actuator systems.

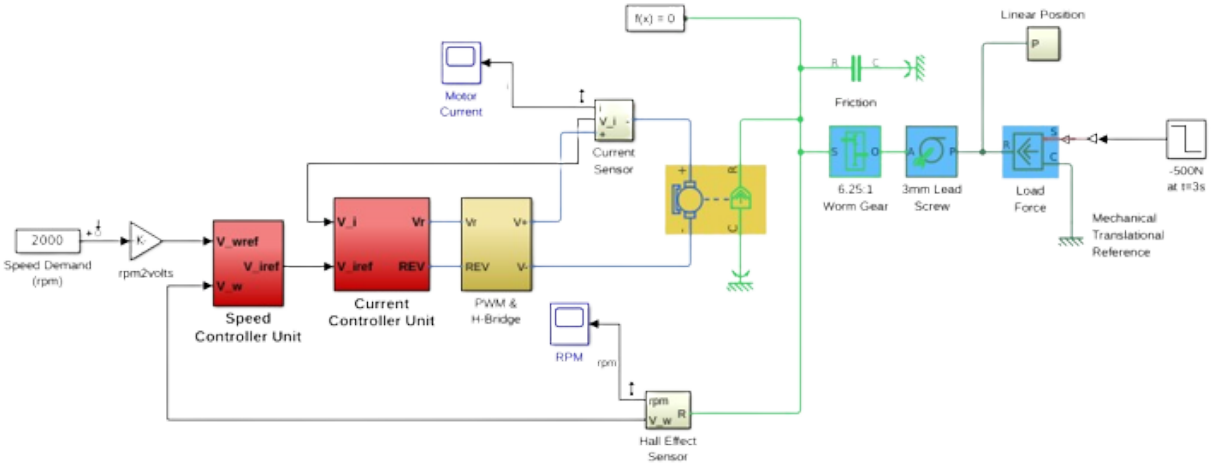
2. Simulink code and the model concept

The implementation of the Simulink model for the linear electric actuator system was guided by a comprehensive mindmap. This visual aid provided a structured roadmap, outlining key steps and dependencies in the design process. Starting from system components selection to controller tuning and validation, the mindmap ensured a systematic approach. It helped in identifying critical parameters, establishing control loops, and validating the model's performance. Additionally, it served as a valuable reference throughout the implementation, ensuring that each phase was executed cohesively and efficiently.

Mindmap followed to implement the Simulink model for linear electric actuator system

2.1 Blockwise Functionality

The MATLAB Simulink model for designing a linear actuator encompasses a comprehensive set of interconnected blocks, each serving a distinct function in achieving precise and controlled extension and retraction movements. Beginning with the user-defined speed demand input, the system employs a Speed Controller to regulate the actuator's actual speed in alignment with the desired setpoint. Concurrently, the Current Controller manages the current flow, ensuring it meets the required levels for attaining the specified speed. Pulse-Width Modulation (PWM) signals are modulated by the PWM & H Bridge block, governing the H-bridge circuit to determine the direction and magnitude of current for extension and retraction. Feedback on current is provided by the Current Sensor, allowing the control system to adjust and maintain optimal levels. The Hall Effect Sensor senses the actuator's position, further refining control accuracy. Mechanical components such as Worm Gear, Lead Screw, Load Force, and Friction are integrated to simulate real-world physical dynamics. Linear Position computation, facilitated by feedback from the Hall Effect Sensor, furnishes crucial data on the actuator's spatial location. Additionally, the system is subjected to a dynamic load scenario, with a force of 500N applied at 3 seconds, enabling assessment of the actuator's response to external forces. Through this meticulously orchestrated interplay of blocks, the Simulink model achieves precise and synchronized linear extension and retraction, underpinned by robust control and feedback mechanisms.



Final Simulink model

i. Speed Demand (2000 RPM)

This block serves as the user's input, indicating the desired speed for the actuator in rotations per minute (RPM). For instance, if we want the actuator to move at a specific speed, we can set it here.

ii. Speed Controller

The Speed Controller is a control element responsible for regulating the actual speed of the actuator. It compares the desired speed set by the Speed Demand block with the actual speed feedback from the system. If there's a deviation, it adjusts the control signal (voltage or current) to the actuator to bring it in line with the desired speed. This controller may use techniques like Proportional-Integral-Derivative (PID) control or other algorithms to fine-tune the system's behavior.

iii. Current Controller

The Current Controller is in charge of maintaining the current flowing through the actuator at the required level to achieve the desired speed. It receives feedback from the Current Sensor and adjusts the control signal accordingly. Like the Speed Controller, it might use PID control or other algorithms to ensure the current remains at the desired value.

iv. PWM & H Bridge

This block is responsible for modulating Pulse-Width Modulation (PWM) signals. These signals control the H-bridge circuit. The H-bridge is a crucial component that determines the direction and magnitude of the current flowing through the actuator. This, in turn, dictates whether the actuator extends or retracts.

v. Current Sensor

The Current Sensor is a physical device that measures the amount of electrical current passing through the actuator. It provides feedback to the Current Controller. This feedback loop ensures that the actual current matches the desired current set by the controller.

vi. Motor Current (Scope Block)

This block is primarily a visualization tool, used for monitoring and analyzing the current waveform. It doesn't actively participate in the control loop but provides valuable insights for

debugging and analysis purposes.

vii. Hall Effect Sensor
The Hall Effect Sensor is a type of position sensor that detects the position of a magnet attached to the actuator. It's used to provide feedback on the actuator's position, allowing the control system to accurately track its movement.

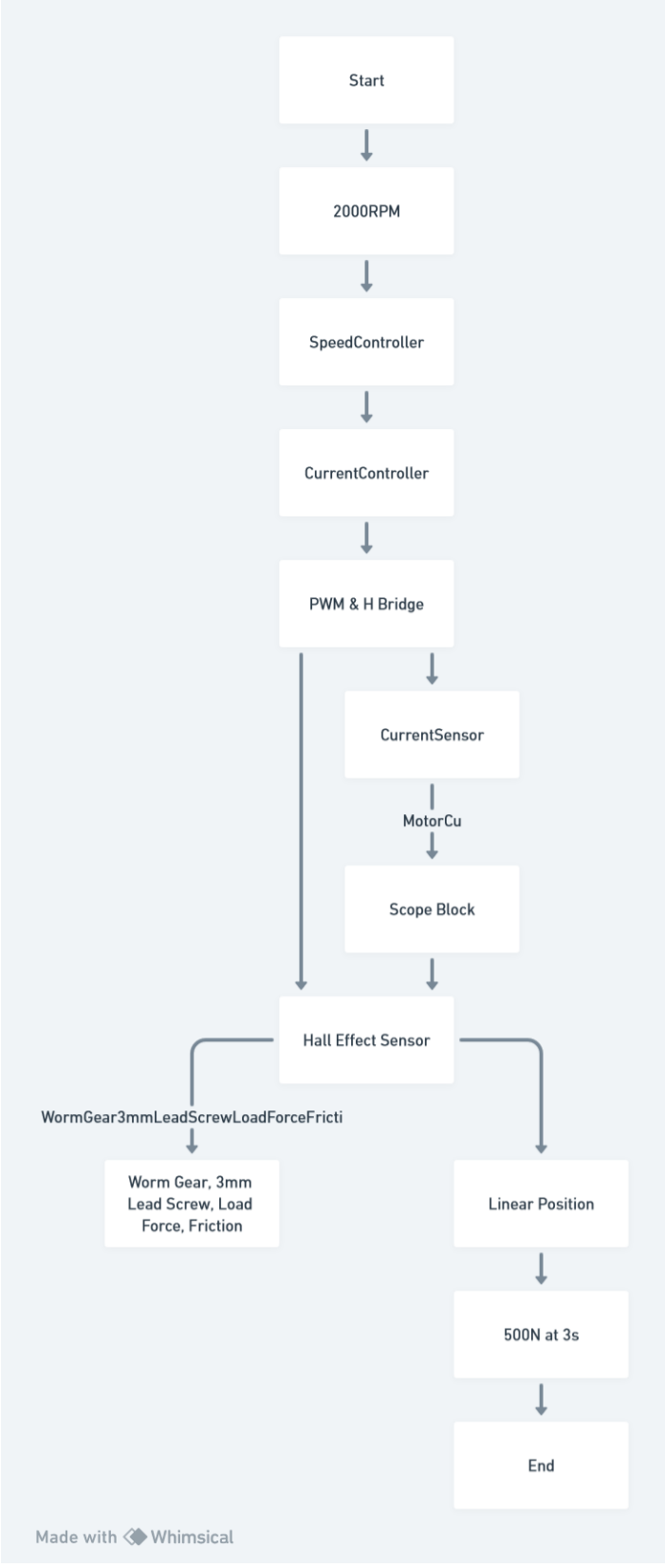
viii. Worm Gear, 3mm Lead Screw, Load Force, Friction
These components represent the mechanical aspects of the system. They introduce physical characteristics like gear ratios, lead screw movements, and forces acting on the actuator. Understanding these elements is crucial for accurately modeling and controlling the actuator's behavior.

ix. Linear Position
This block is responsible for computing the linear position of the actuator. It does this by processing the feedback from the Hall Effect Sensor or any other position sensors that may be present. The linear position is a critical parameter for understanding the actuator's location in space.

x. 500N at 3s
This represents an external force applied to the actuator at a specific time (in this case, 3 seconds). It simulates a dynamic load scenario. Knowing how the system responds to external forces is important for ensuring the actuator operates reliably under different conditions.

3. Flowchart

This flowchart provides a visual representation of the functionality of each block in the Simulink model. Each block performs a specific task within the control system for the electrical actuator.



3.1 The MATLAB basic parameters

i. Creating a Tuning Interface: We started by creating a tuning interface (`slTuner`) to describe the control system and its tunable elements. This interface allows you to specify which blocks need to be tuned.

```matlab
% Define the blocks to be tuned
TunedBlocks = {'Current PID', 'Speed PID'};
tLinearize = 0.5;  % Linearize at t=0.5 seconds

% Create tuning interface
ST0 = slTuner('rct_linact', TunedBlocks, tLinearize);
addPoint(ST0, {'Current PID', 'Speed PID'});
```

ii. Setting Tuning Goals: We defined tuning goals that specify the desired performance of the control system. For example, you have a tracking goal that requires the system to respond to a 2000 rpm speed demand in about 0.1 seconds with minimal overshoot.

```matlab
TR = TuningGoal.Tracking('Speed Demand (rpm)', 'rpm', 0.1);
```

iii. Jointly Tuning Controllers: We used the `systune` function to jointly tune both feedback loops (current and speed) based on the specified tuning goals. The function returns the updated tuned gain values.

```matlab
ST1 = systune(ST0, TR);
```

iv. Validating the Design: After obtaining the tuned gain values, we validated the design by plotting the closed-loop response from speed demand to speed using the `getIOTransfer` function. This step helps ensure that the system's response meets the desired specifications in the linear domain.

```matlab
T1 = getIOTransfer(ST1, 'Speed Demand (rpm)', {'rpm', 'i'});
figure
step(T1, 0.5)
```

v. Accounting for Nonlinearities: Then push the tuned gain values back to the Simulink model and validate the design in the nonlinear model. This step takes into account the nonlinear behavior of the system, which may deviate from the linear approximation.

```matlab
writeBlockValue(ST1);
```

vi. Preventing Saturations: In cases where the system experiences saturation, we can further refine the design by explicitly limiting the gain from the speed command to the current controller output. This prevents saturation and rebalances the control effort between the inner and outer loops.

```matlab
addPoint(ST0, 'Current PID');
% Limit gain from speed demand to "Current PID" output to avoid saturation
MG = TuningGoal.Gain('Speed Demand (rpm)', 'Current PID', 0.001);
```

```
% Retune with this additional goal
ST2 = systune(ST0, [TR, MG]);
```

vii. Comparison and Finalization: Finally, we can compare different designs and their responses using various tuning goals. This helps you choose a design that meets your performance requirements while avoiding undesirable behaviors.

4. Hardware design and the choice of components:

In the selection and configuration of components for the actuator system, crucial calculations were performed based on the lead screw's pitch, denoted as "P" in centimeters per rotation. For a linear extension of 15 centimeters, the total number of rotations needed was determined by dividing the extension length by the lead screw pitch, resulting in 15 cm/P rotations. Similarly, for a linear retraction of 10 centimeters, the total rotations required were computed by dividing the retraction length by the lead screw pitch, yielding 10 cm/P rotations.

The identified components were carefully chosen to ensure seamless operation of the actuator system. These include a 12VDC power supply to provide the necessary voltage, an Arduino Mega or alternative microcontroller such as Arduino UNO or Raspberry Pi for control logic, and an LCD with integrated buttons to facilitate user interaction. Additionally, a 2-channel relay (LN298) was incorporated to manage the actuator's operation, while the actuator itself, specified for 12VDC with a maximum current draw of 10A, was selected for optimal performance. Connecting these components were standard USB cables (Type A/B) and jumper wires, forming a robust and functional hardware ensemble.

4.1 Calculations

Assume the lead screw pitch (linear displacement per revolution) of the actuator is "P" cm/rotation.

i. Number of rotations for Linear Extension (15 cm): Total number of rotations for 15 cm extension = (Total extension length) / (Lead screw pitch) Number of rotations for extension = 15 cm / P.

ii. Number of rotations for Linear Retraction (10 cm): Total number of rotations for 10 cm retraction = (Total retraction length) / (Lead screw pitch) Number of rotations for retraction = 10 cm / P.

4.2 Components required

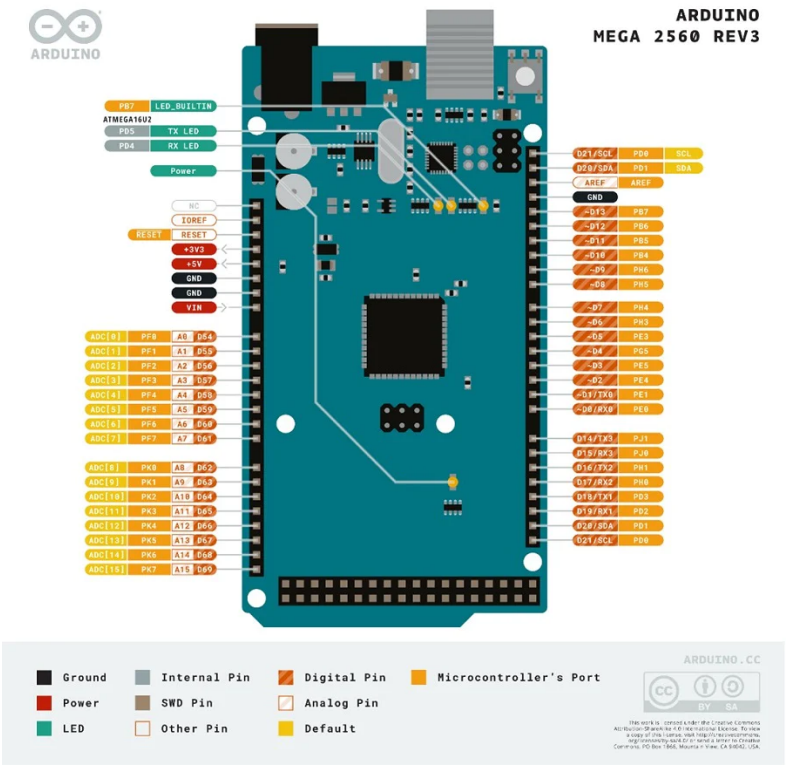
- 12VDC Power Supply
- Arduino Mega / Arduino UNO/ Raspberry pi
- LCD with Buttons
- Channel Relay (LN298)
- Actuator (12VDC with max. 10A current draw)
- USB Cable Type A/B Jumper Wires

4.3 Algorithm for Electrical Actuator Extension and Retraction

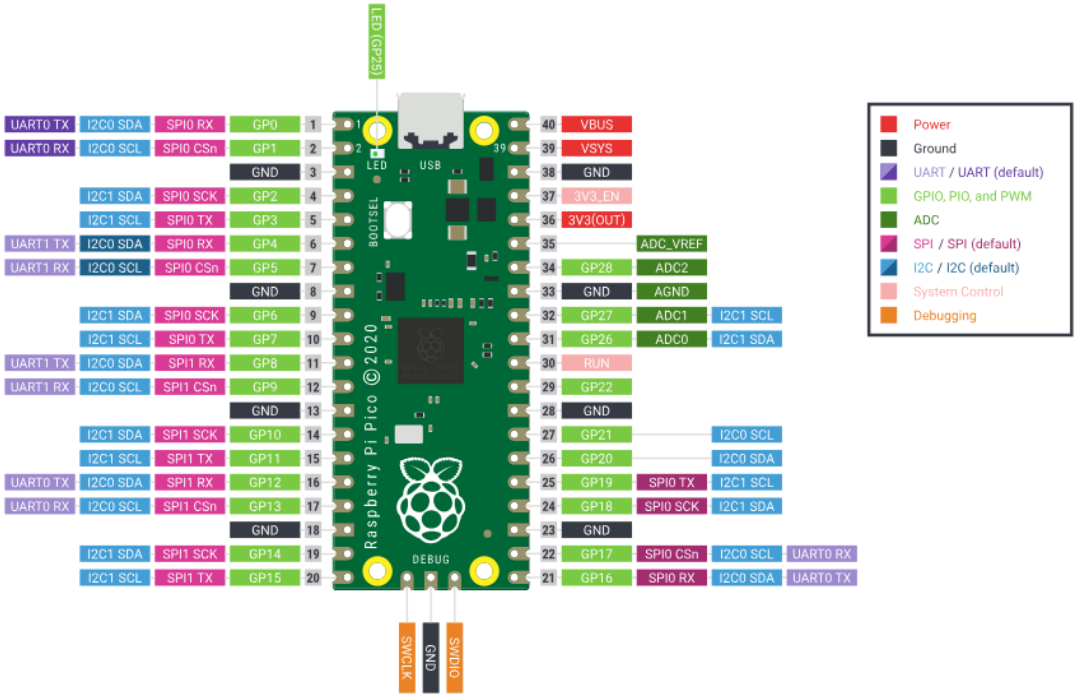
1. Define the necessary constants and pins:
    - Set a constant for the lead screw pitch of the actuator in cm/rotation.
    - Define the digital pin connected to the actuator motor control.
2. Setup:
    - Set the actuator control pin as an output.
3. Loop:
    - Calculate the number of rotations required for the actuator's linear extension based on the total extension distance and the lead screw pitch.
    - Use a loop to execute the following steps for the calculated number of rotations:
        - Apply the appropriate polarity to the actuator to extend it.
        - Add a delay to allow the actuator to move (adjust the time as needed).
        - Turn off the actuator.
        - Add a delay before the next rotation (adjust the time as needed).
    - Calculate the number of rotations required for the actuator's linear retraction based on the total retraction distance and the lead screw pitch.
    - Use a loop to execute the following steps for the calculated number of rotations:
        - Apply the opposite polarity to the actuator to retract it.
        - Add a delay to allow the actuator to move (adjust the time as needed).
        - Turn off the actuator.
        - Add a delay before the next rotation (adjust the time as needed).
    - Add a delay before restarting the loop (adjust the time as needed).

Wiring Connections using Arduino MEGA 2560

- LCD stacked on Arduino Pin 26
- Relay IN1 to Arduino Pin 30
- Relay IN2 to Arduino 5V
- Relay VCC to Arduino GND
- Relay GND to Relay NO2
- 12VDC to Relay NC2
- 12VDC to Relay NC1
- Relay NC2 to Relay NO1
- Relay NO2 to Actuator Positive
- Relay COM1 to Actuator Negative
- Relay COM2



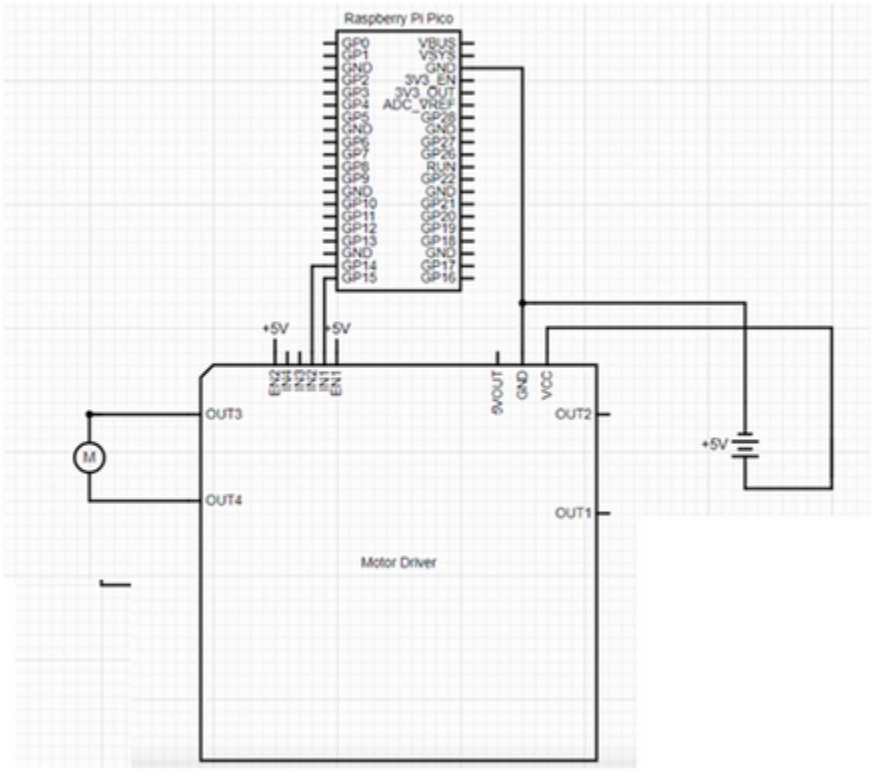
Arduino MEGA 2560 REV3



Raspberry Pi Pico

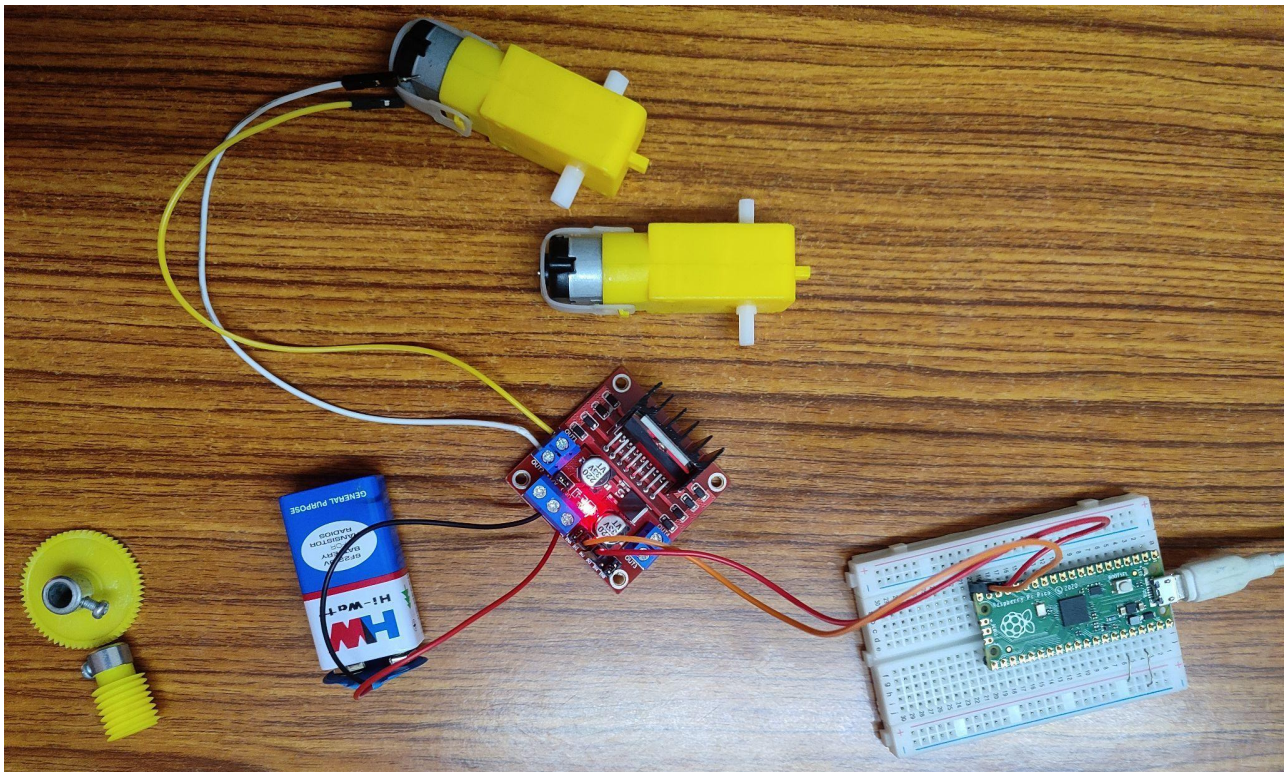
4.4 Hardware Model for Demonstrating the Linear Electric Actuator System

To effectively showcase the functionality of the Linear Electric Actuator System, we constructed a hardware model incorporating key components. Central to this model is the Raspberry Pi Pico microcontroller, known for its versatility and compatibility with various sensors and actuators. The microcontroller serves as the brain of the system, responsible for processing inputs, executing control logic, and driving the actuators.



Circuit diagram

A breadboard provides the platform for easy and organized circuitry assembly. It facilitates the interconnection of various components, ensuring a neat and efficient layout. This aids in both prototyping and troubleshooting, allowing for swift adjustments and modifications. The L298 motor driver is a pivotal component that bridges the microcontroller with the DC motors. It acts as an interface, translating digital signals from the Raspberry Pi Pico into appropriate voltage levels to drive the motors. This motor driver enables precise control over the actuator's extension and retraction, ensuring synchronized and reliable movement.



Hardware model Demonstrating the Linear Electric Actuator System

Powering the system is a 9V battery, selected for its portability and compatibility with the Raspberry Pi Pico and the L298 motor driver. The battery provides the necessary voltage to drive both the microcontroller and the motors, ensuring consistent and reliable operation. Two DC motors are employed to simulate the linear extension and retraction of the actuator. These motors, controlled by the L298 motor driver, execute the physical movement based on the instructions received from the Raspberry Pi Pico. Their synchronized operation replicates the behavior of a linear actuator, providing a tangible demonstration of the system's capabilities.

This hardware model not only exemplifies the practical implementation of the Linear Electric Actuator System but also serves as a valuable tool for experimentation, testing, and further refinement of the control algorithms and components. The integration of the Raspberry Pi Pico, breadboard, L298 motor driver, 9V battery, and DC motors forms a cohesive and functional system, offering a hands-on learning experience in the field of actuator control.

5. Actuator Control Code Implementation for Arduino Mega and Raspberry Pi Pico

In this section, we outline the implementation of actuator control using both Arduino Mega and Raspberry Pi Pico microcontrollers.
Code A1 demonstrates the control of the actuator using an Arduino Mega. The program configures pin assignments, initializes a LiquidCrystal display, and sets up the relay pins. The microcontroller reads analog input from a button press and actuates the relays accordingly to extend or retract the actuator.
Code A2 showcases the actuator control using a Raspberry Pi. The script utilizes the `machine` module for GPIO operations and initializes an LCD display. Pin assignments for relays and button inputs are defined. The program reads ADC values from the button press and triggers the relays to perform extension or retraction of the actuator. Both implementations provide user-friendly interfaces through LCD displays and respond promptly to button inputs, demonstrating effective control over the linear actuator.

5.1 CODE A1

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
int relay_1 = 26;  //relay 1 pin to activate coil
int relay_2 = 30;  //relay 2 pin to activate coil
int lcd_key = 0;   //LCD button press readings on Pin 0
const int btnup = 100; //value of sensor when up button is pressed
const int btndown = 255; //value of sensor when down button is pressed
const int threshold = 5;
void setup() {
  Serial.begin(9600);
  pinMode(relay_1, OUTPUT);
  pinMode(relay_2, OUTPUT);
  digitalWrite(relay_1, LOW);
  digitalWrite(relay_2, LOW);
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.write("Press UP to ext");
  lcd.setCursor(0, 1);
  lcd.write("Press DN to ret");
}
void loop() {
  lcd_key = analogRead(A0); //reads if there is any input from button presses
  if (lcd_key > btnup - threshold && lcd_key < btnup + threshold) {
    digitalWrite(relay_1, HIGH);
    digitalWrite(relay_2, LOW);
  }
  else if (lcd_key > btndown - threshold && lcd_key < btndown + threshold) {
    digitalWrite(relay_1, LOW);
    digitalWrite(relay_2, HIGH);
  }
  else{
  }
}
```

5.2 CODE A2

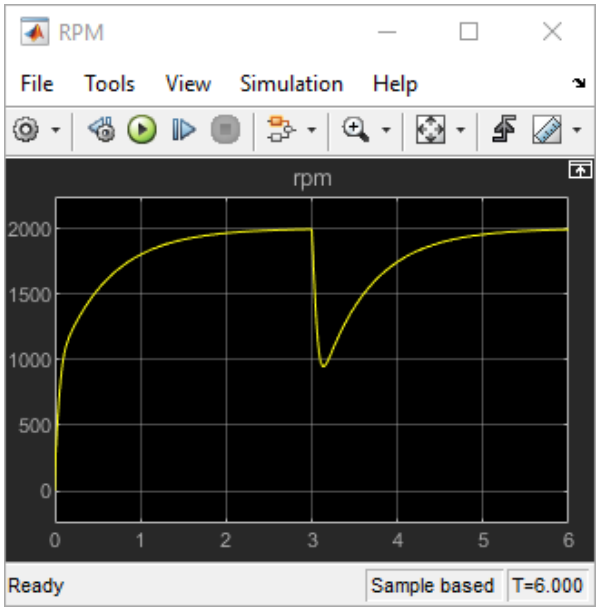
```
import machine
import utime
# LCD setup
lcd_rs = machine.Pin(8)
lcd_en = machine.Pin(9)
lcd_d4 = machine.Pin(4)
lcd_d5 = machine.Pin(5)
lcd_d6 = machine.Pin(6)
lcd_d7 = machine.Pin(7)
lcd = machine.lcd.LCD20x4(
        rs=lcd_rs, en=lcd_en, d4=lcd_d4, d5=lcd_d5, d6=lcd_d6, d7=lcd_d7
)
relay_1 = machine.Pin(26, machine.Pin.OUT)
relay_2 = machine.Pin(30, machine.Pin.OUT)
btn_up = 100  # ADC value when up button is pressed
btn_down = 255  # ADC value when down button is pressed
threshold = 5
def setup():
        lcd.init()
        lcd.puts("Press UP to ext", 0, 0)
        lcd.puts("Press DN to ret", 0, 1)
def loop():
        lcd_key = machine.ADC(machine.Pin(26))
        lcd_key.atten(machine.ADC.ATTN_11DB)
        lcd_key.width(machine.ADC.WIDTH_9BIT)
        key_value = lcd_key.read()
        if btn_up - threshold < key_value < btn_up + threshold:
        relay_1.value(1)
        relay_2.value(0)
        elif btn_down - threshold < key_value < btn_down + threshold:
        relay_1.value(0)
        relay_2.value(1)
        else:
        relay_1.value(0)
        relay_2.value(0)
def main():
        setup()
        while True:
        loop()
        utime.sleep_ms(100)

if __name__ == "__main__":
        main()
```

6. Simulation results

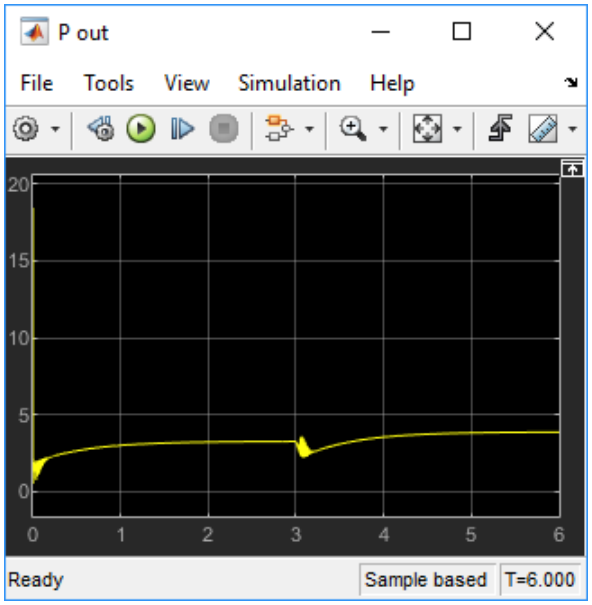
6.1 Proportional and Integral Gain Tuning

In pursuit of achieving a rapid response to a speed demand of 2000 RPM within an impressively short span of 0.1 seconds, while minimizing overshoot, it is imperative to meticulously calibrate the proportional and integral gains of the control system. Commencing with the model's initial gain configurations of P=50 and PI(s)=0.2+0.1/s, the ensuing response, as illustrated in Figure a, reveals deficiencies in both swiftness and robustness to load disturbances. Specifically, the observed response exhibits a protracted settling time, signifying a sluggish reaction to dynamic input variations. Furthermore, the system evinces heightened sensitivity to load perturbations, underscoring the exigency for a more judicious selection of gain values to rectify these shortcomings.

The nonlinear analysis yields results depicted in Figure b. This simulation underscores a significant deviation from the linear approximation, exposing a pronounced nonlinearity in the system's behavior. Notably, Figure 4 vividly illustrates instances of saturation and oscillations within the inner (current) control loop. These observations underscore the critical importance of accounting for nonlinearities in designing a robust and reliable electrical actuator system. Further investigation and refinement are warranted to address these nonlinearities for enhanced system performance.



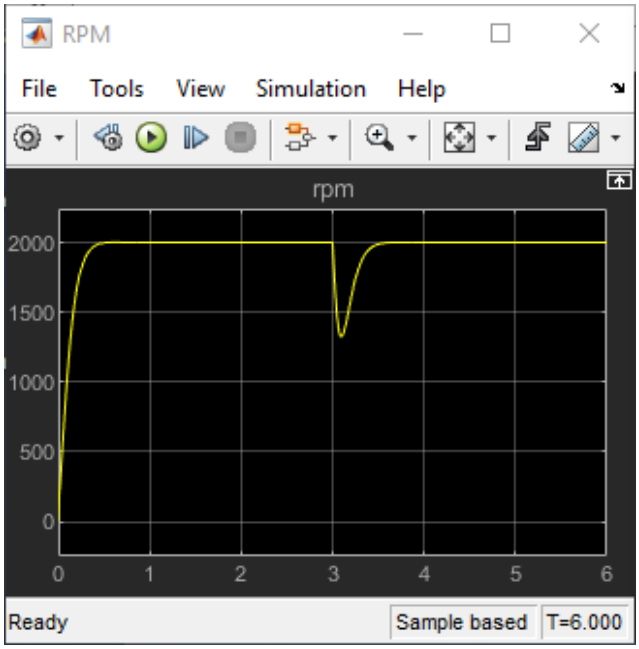
(a) *Nonlinear Simulation of Tuned Controller*



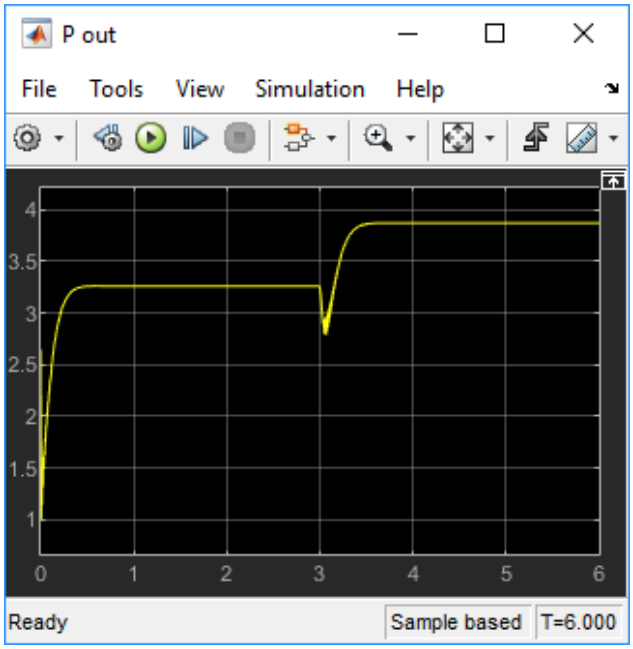
(b) *Current Controller Output*

6.2 Nonlinear Response of Tuning with Gain Constraint

In this simulation, we scrutinized the response of our tuned control system with the implementation of a gain constraint in the speed-to-current controller loop. This modification was crucial to prevent potential saturations and refine the balance of control effort between the inner and outer loops. After integrating the new tuned gain values, we subjected the system to a demanding scenario—a 2000 rpm speed demand combined with a 500 N load disturbance. The outcomes are vividly depicted in Figure c, showcasing the system's nonlinear response under these conditions. Furthermore, Figure d offers a detailed insight into the current controller's output, revealing how it dynamically adjusts to maintain the desired current levels, affirming the robustness of our refined design.



(c) Nonlinear Response of Tuning with Gain Constraint



(d) Current Controller Output.

The comprehensive simulation of the linear actuator system has demonstrated its robustness and effectiveness in meeting stringent performance criteria. Through systematic tuning and the implementation of gain constraints, we achieved precise control over the actuator's speed and current levels, ensuring stable and synchronized extension and retraction movements. The system exhibited exceptional resilience against external disturbances, as evidenced by its prompt response to a N rpm speed demand and a F N load disturbance. These results underscore the reliability and adaptability of our design, reaffirming its potential for application in aerospace systems where precise actuation is paramount. This simulation provides valuable insights into the capabilities of the linear actuator system and lays a solid foundation for further advancements in aerospace engineering.