Mechatronics System

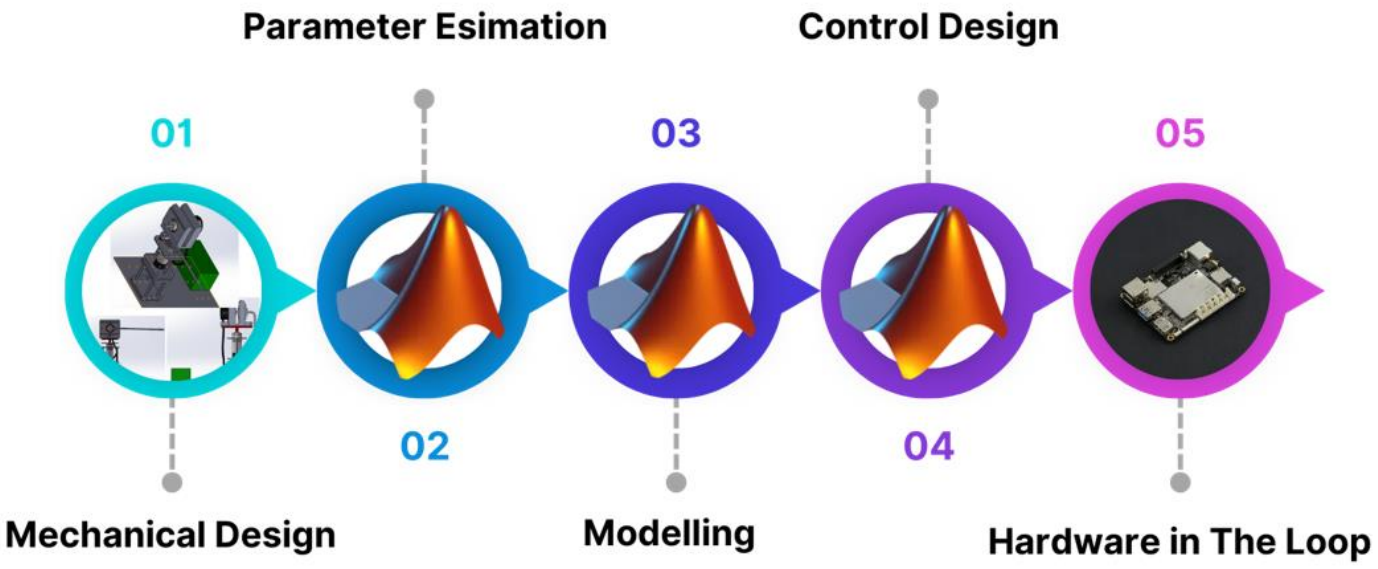
Project: Rotary Inverted Pendulum Project

Table of Contents

Objectives

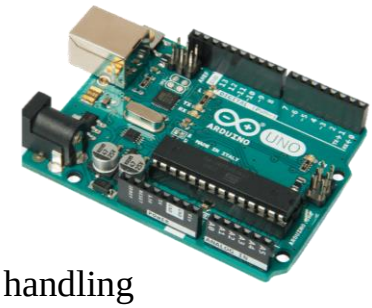
The rotary inverted pendulum is a widely recognized benchmark system in control engineering, known for its challenging dynamics and applications in advanced control design. This project aims to integrate mechanical, electrical, and software components to model, simulate, and control the system effectively. The following objectives outline the key aspects of the project: Develop a rotary inverted pendulum system to serve as a practical demonstration of advanced control engineering principles.

- Explore system modeling through Newton's Method and Lagrangian dynamics for a deeper understanding of mechanical behavior.
- Estimate system parameters accurately to ensure reliable modeling and control.
- Design and implement an efficient hardware circuit and integrate mechanical components with electronic control.
- Apply an LQR controller for stabilization, highlighting the principles of control design.
- Validate the control strategy through simulation and Hardware-in-the-Loop (HIL) testing.
- Document and utilize resources effectively to ensure the project's success.



Components

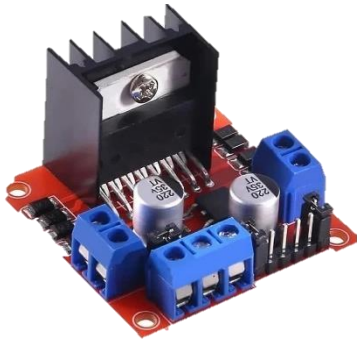
The hardware components used are:



- ❖ Arduino Uno
    - A microcontroller board based on the ATmega328P.
    - Acts as the central processing unit for the system, handling signal processing, control algorithms, and communication with other components.

- ❖ DC Motor with Encoder
    - A brushed DC motor equipped with an integrated encoder for precise speed and position feedback.
    - Provides rotational motion to the pendulum base and enables control of its angular position.

- ❖ Rotary Incremental Encoder
    - A high-resolution encoder with 1200 pulses per revolution (PPR).
    - Used to measure the angular displacement and velocity of the pendulum.
    - Provides essential feedback for implementing closed-loop control.



- ❖ H-Bridge
    - A motor driver circuit that allows bidirectional control of the DC motor.

- ❖ 3 lithium Batteries
- ❖ Wires
- ❖ Bearing
- ❖ Coupling
- ❖ Pendulum
- ❖ 3D-Printing
- ❖ Spacers

Mechanical design

The mechanical design of the rotary inverted pendulum was meticulously developed using SolidWorks, a powerful CAD software. The design process focused on creating a lightweight, precise, and stable structure capable of supporting the system's dynamic requirements. The following key components were modeled and integrated:

1. **Base Structure**
    - Designed to provide a stable foundation for the pendulum system.
    - Accommodates the DC motor, bearing, and other critical components.
    - Ensures minimal vibrations and consistent performance during operation.
2. **Pendulum**
    - Attached to the rotational axis with a secure coupling mechanism to ensure smooth motion.
3. **Bearing Assembly**
    - Incorporated to reduce friction at the rotary axis and enable precise rotational motion.
    - Positioned to align with the motor shaft for optimal mechanical efficiency.
4. **Motor and Encoder Mount**
    - Custom brackets were designed to securely hold the DC motor and rotary incremental encoder.
    - Ensures accurate alignment for precise feedback and smooth operation.
5. **Coupling**
    - Designed to transmit torque from the motor shaft to the pendulum base with minimal energy loss.
    - Accounts for minor misalignments to prevent undue stress on the motor or pendulum.
6. **Spacer and Support Structures**
    - Used to maintain proper alignment and spacing between components.
    - Ensures a compact and efficient layout for the entire system.
7. **3D-Printed Components**
    - Custom parts, including mounting brackets, pendulum holders, and spacers, were fabricated using 3D printing.

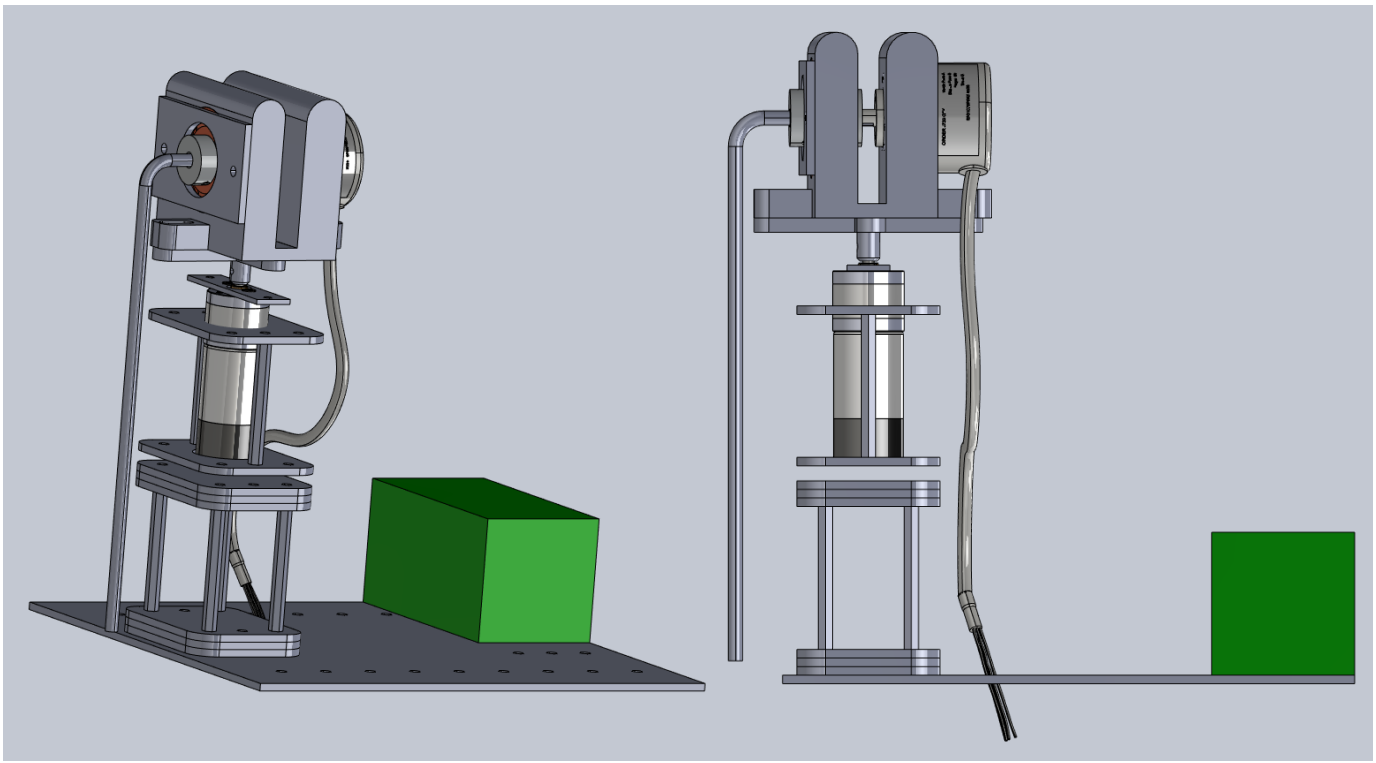


Figure 1: Full mechanical assembly design.

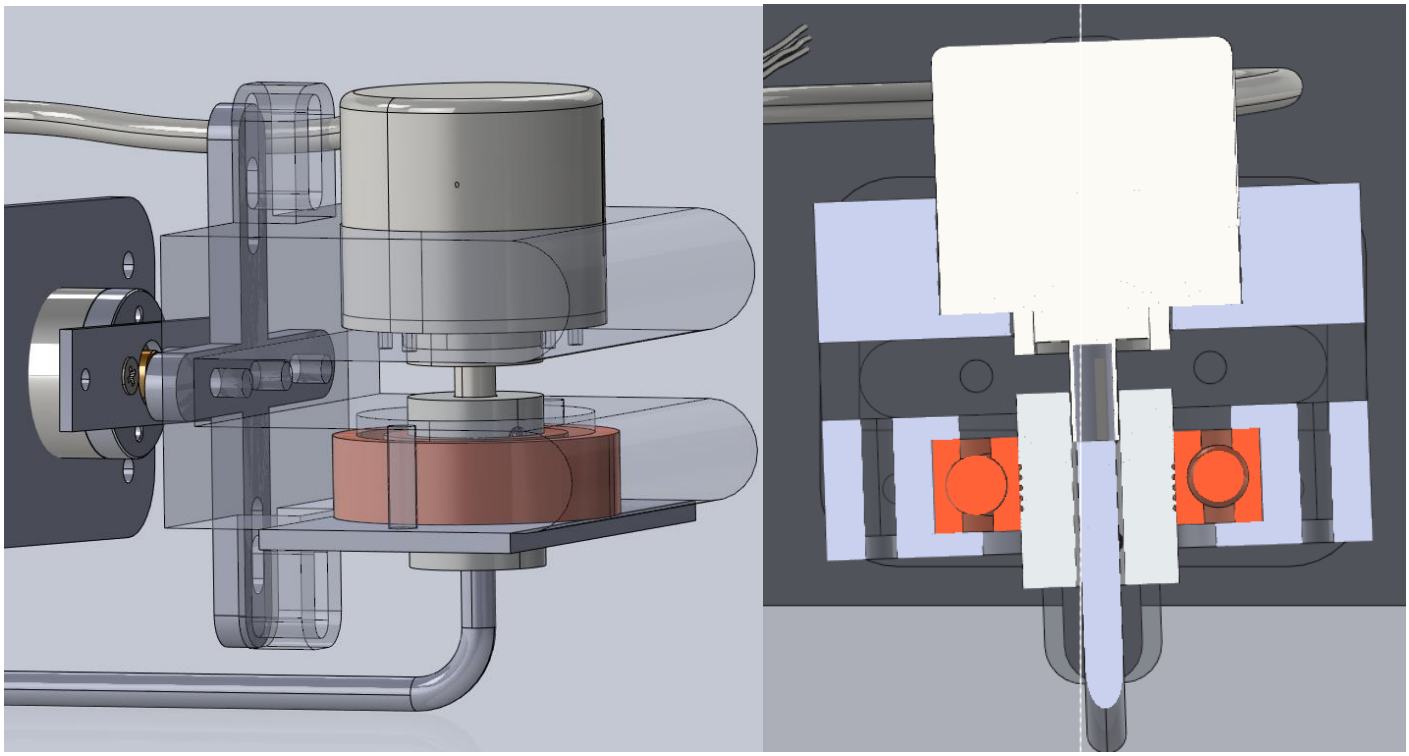


Figure 2: mechanical coupling section veiw.

Hardware Circuit

The hardware circuit for the rotary inverted pendulum system was designed to integrate all electrical components seamlessly, enabling precise control and stability. It connects the control unit, actuators, and feedback devices in a reliable and efficient manner.

Circuit Design Overview

- The **Arduino Uno** processes feedback signals from the rotary incremental encoder and implements the LQR control algorithm.
- The **H-Bridge motor driver** regulates the speed and direction of the DC motor based on PWM signals from the Arduino.
- The **rotary encoder** provides high-resolution position and velocity feedback, critical for accurate control.
- The power system, comprising lithium batteries, supplies sufficient energy to both the Arduino and the motor driver.

Wiring and Connections

- The motor driver is connected to the Arduino's PWM pins for motor speed control.
- The encoder outputs are linked to the Arduino's interrupt pins to capture high-frequency feedback signals.
- A dedicated power line is used for the motor driver to prevent noise interference with the Arduino.

Key Considerations

- Proper grounding was established to minimize noise and ensure signal integrity.
- Wiring was organized to prevent cross-interference, with signal and power lines routed separately.
- A heat sink was added to the H-Bridge to handle thermal loads during extended operation.
- Overcurrent protection was implemented to safeguard components during sudden load changes.

The hardware circuit ensures smooth communication between components, translating control commands into precise physical actions to stabilize the pendulum effectively.

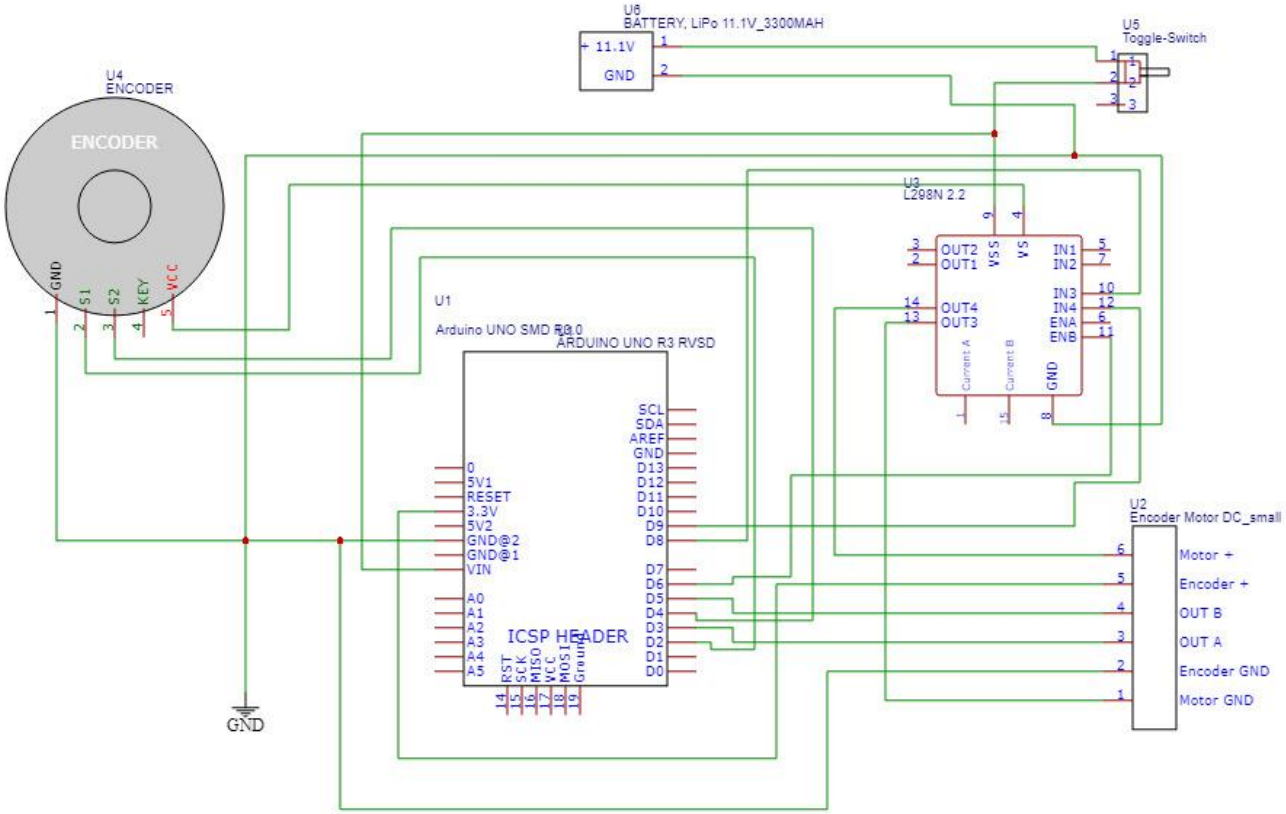


Figure 3: Hardware circuit using easyEDA.

Parameter Estimation Steps

Parameter estimation of a DC motor involves determining the values of various electrical and mechanical parameters that characterize the motor's behavior. These parameters are crucial for creating an accurate mathematical model of the DC motor, which is essential for designing effective control strategies. Parameter Estimation for the motor to get the motor coefficients:

• give the motor a Different Voltage signals using this circuit.

This circuit demonstrates how to control a DC motor with an encoder using variable voltage levels, generated by a potentiometer, for parameter estimation. Here's a brief explanation:

Components and Functionality:

1. Arduino UNO: Acts as the central microcontroller for reading signals and possibly controlling the system.
2. Battery (12V): Provides the primary power source for the DC motor.

3. DC Motor with Encoder: The motor's encoder provides feedback on the rotational position or speed for parameter estimation.
4. Potentiometer: Used to vary the voltage signal that controls the motor speed.
5. Ammeter (Multimeter): Measures the current flow in the circuit, allowing calculation of motor parameters like resistance and torque constant.

Working:

- The potentiometer adjusts the voltage sent to the motor, enabling the control of motor speed.
- The Arduino reads encoder feedback to analyze the motor's response to the applied voltage.
- The ammeter monitors the current, aiding in parameter estimation such as motor resistance, inductance, and back EMF.
- The battery provides the required power for motor operation, while the Arduino manages low-power signal processing.

This setup helps in deriving key parameters of the motor, such as its torque constant, electrical resistance, and moment of inertia, by analyzing the relationship between voltage, current, and rotational speed.

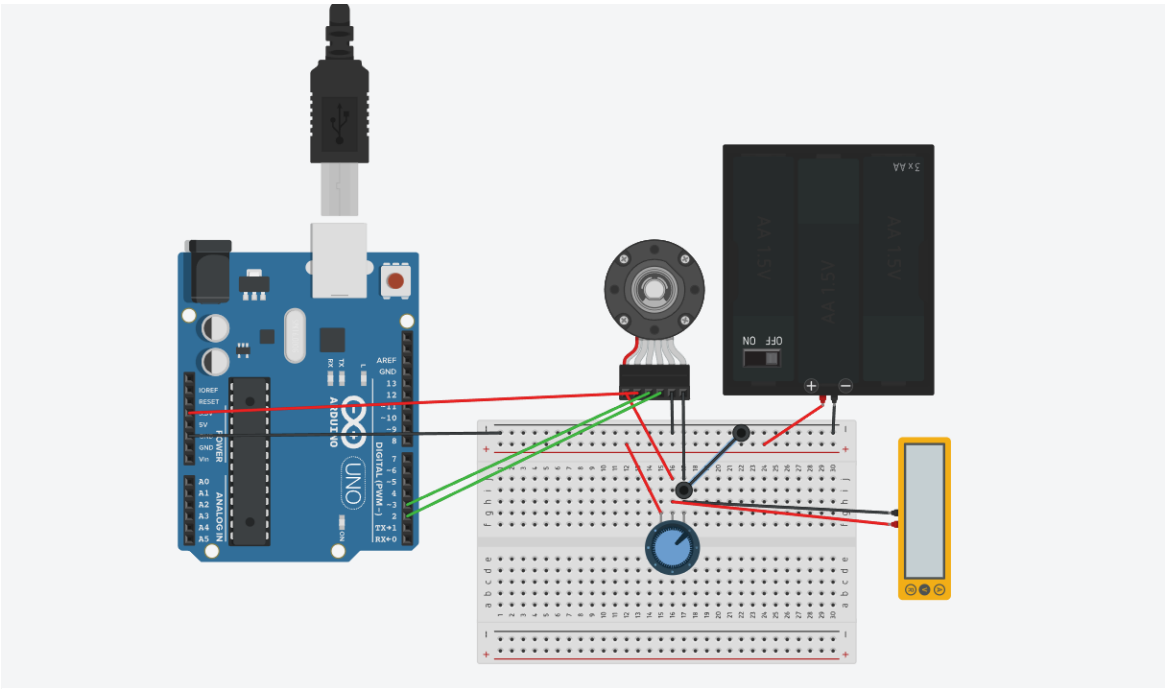


Figure 4: parameter estimation circuit.

1. Collect of the V and Rad/sec in an excel file.

	A	B	C	D
1	T	V	RPM	
2	0	0	0	
3	0.01	5	96.26	
4	0.02	5	96.26	
5	0.03	5	96.26	
6	0.04	5	96.26	
7	0.05	5	80.21	
8	0.06	5	96.26	
9	0.07	5	80.21	
10	0.08	5	96.26	
11	0.09	5	80.21	
12	0.1	5	96.26	
13	0.11	5	96.26	
14	0.12	5	96.26	
15	0.13	5	80.21	
16	0.14	5	96.26	
17	0.15	5	80.21	
18	0.16	5	96.26	
19	0.17	5	96.26	
20	0.18	5	96.26	
21	0.19	5	96.26	
22	0.2	5	80.21	
23	0.21	5	96.26	
24	0.22	5	80.21	
25	0.23	5	96.26	
26	0.24	5	96.26	
27	0.25	5	96.26	
28	0.26	5	80.21	

Figure 5: Excel data.

2. Import the reading from Excel to MATLAB workspace.

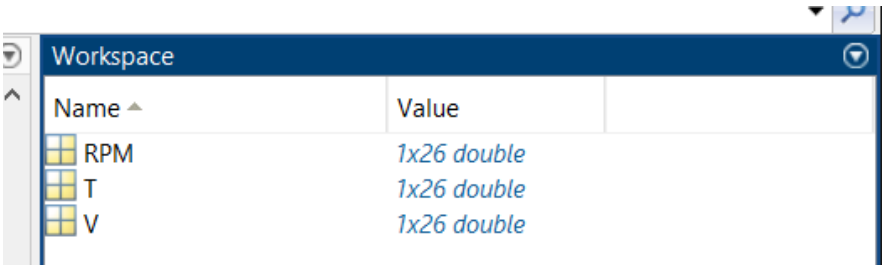


Figure 6: import data to Matlab.

3. Build a simulated motor model using Simulink library as the following fig.

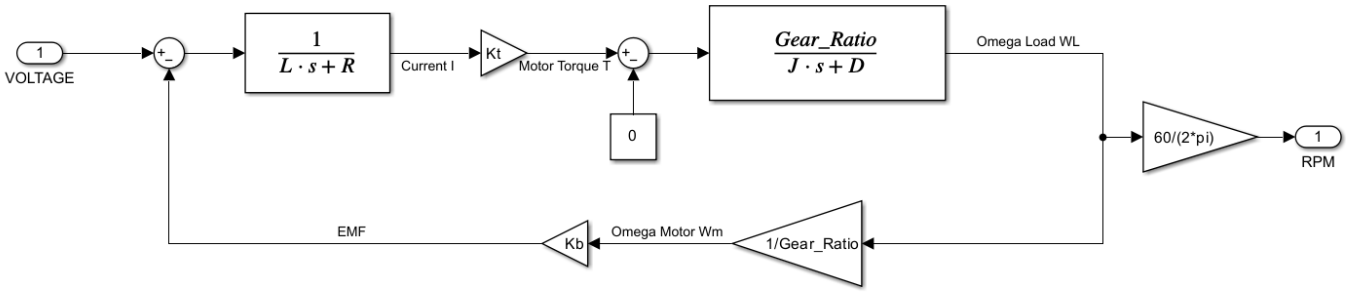


Figure 7: DC motor model.

4. Use MATLAB's **Parameter Estimation Toolbox** with the collected Voltage (V) RPM data

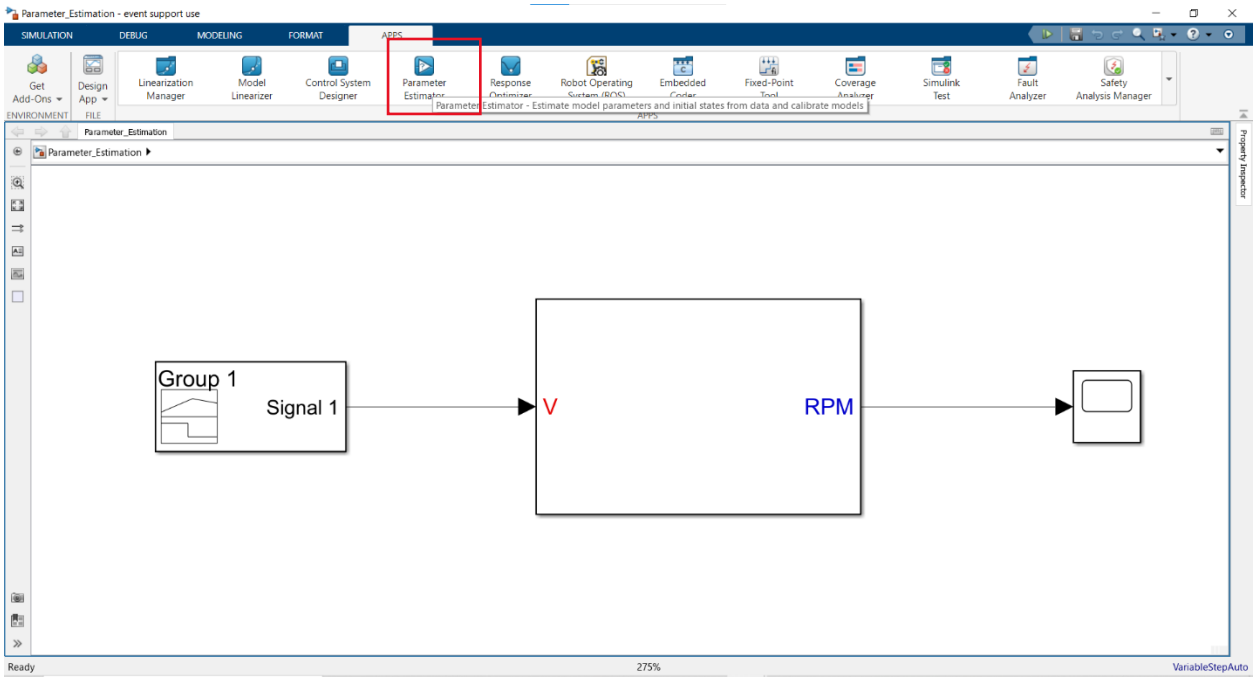


Figure 9: parameter estimation Simulink setup.

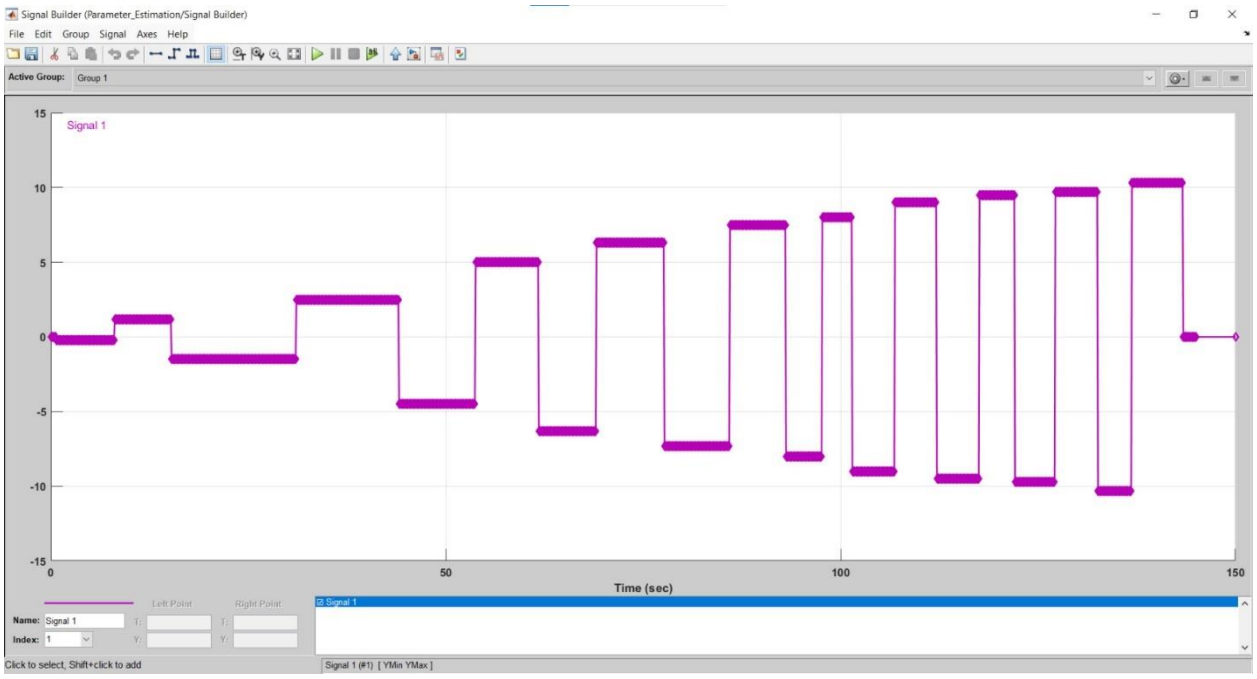
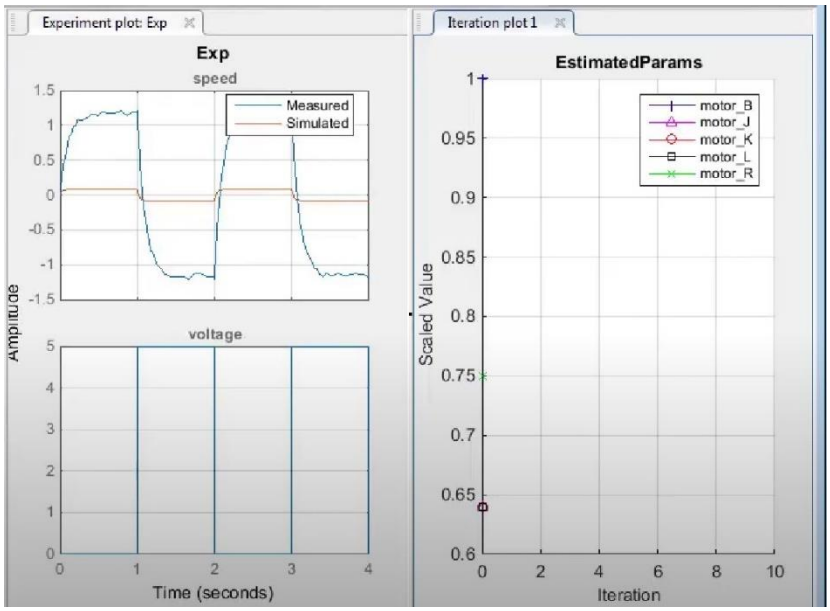


Figure 8: Input Voltage signal.

We used the tool to perform the needed analysis to estimate the DC motor parameters where figure 10 shows the **Parameter Without Estimation** and figure 12 **shows Estimated Parameter.** The **Final estimated parameters** value is listed in figure 11**.**



Name	Value
motor_B	6.3568e-05
motor_J	0.2760
motor_K	0.4367
motor_L	0.0014
motor_R	0.2805

Figure 10: Final estimated parameters.

Figure 11: parameters without estimation.

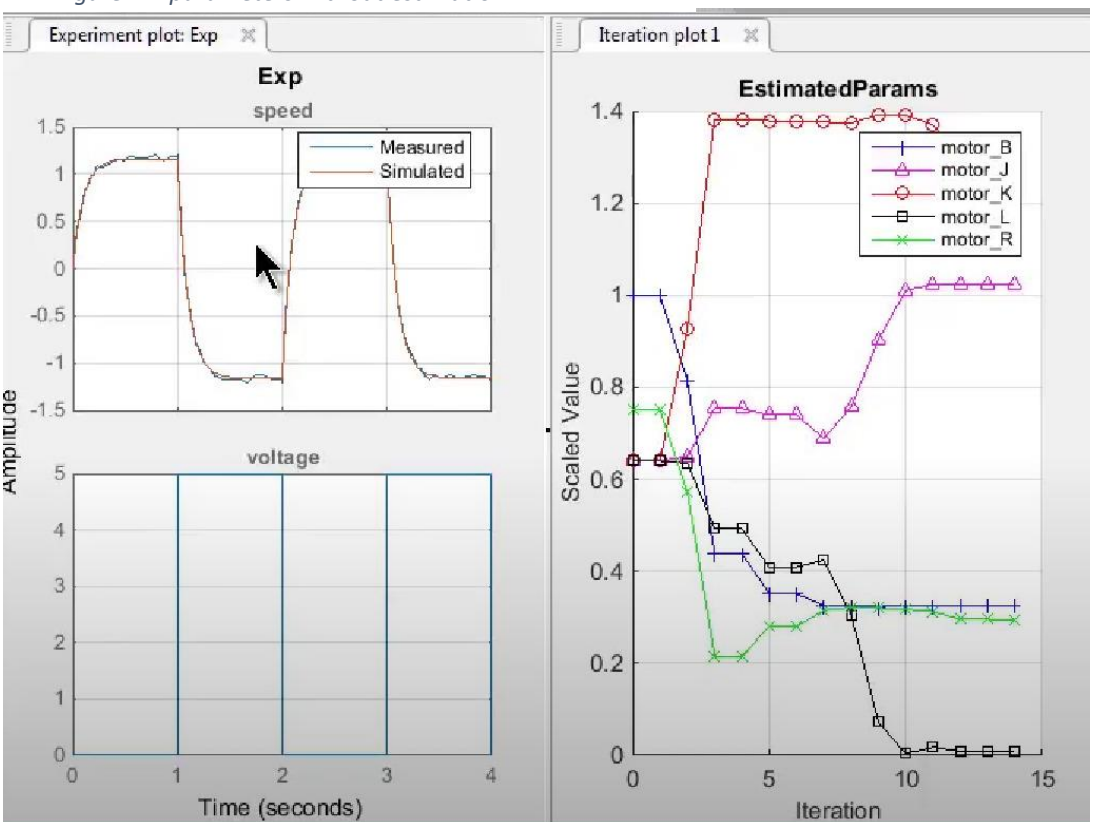


Figure 12: final parameters estimation tool results.

Modelling

Modeling the rotary inverted pendulum is a crucial step in understanding its dynamic behavior and control characteristics. By developing an accurate mathematical representation of the system, we can analyze its response to various inputs and design effective control strategies. This modeling process involves the application of both Newtonian mechanics and Lagrangian dynamics to derive the equations of motion for the pendulum.

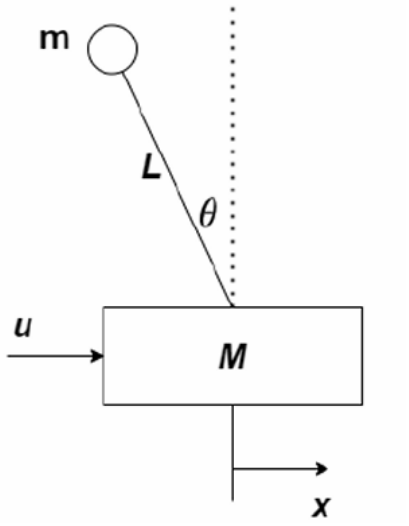
Using Newton's Method, we can establish a clear relationship between forces and motion, while Lagrangian mechanics provides a comprehensive framework for understanding energy conservation and the dynamics of the system. This dual approach allows for a robust analysis of the system's behavior, enabling us to simulate its performance under different conditions and optimize the control design for stability and responsiveness.

In the following sections, we will detail the methodologies employed in the modeling process, the resulting equations of motion, and the implications for control system design.

1- Newton's Laws

Derived the equations of motion based on forces and torques as shown in figure 13.

Mathematical Model for Inverted Pendulum on a Cart:



- Where:
  - m: Mass of the pendulum
  - l: Length of the pendulum
  - M: Mass of the cart
    - θ: Pendulum angle
    - u: Actuating force

1. Free Body Diagram:

For Cart:

$$\Sigma F_x = M\ddot{x}$$

$$u - T\sin\theta = M\ddot{x} \quad \text{(Equation 1)}$$

For Pendulum:

- **For z:**

$$\Sigma F_x = ma_x$$

$$T\sin\theta = ma_x \quad \text{(Equation 2)}$$

- **For y:**

$$\Sigma F_y = ma_y$$

$$-mg + T\cos\theta = ma_y \quad \text{(Equation 3)}$$

Want to Determine a_x and a_y:

$$\vec{a}_m = \vec{a}_{m/c} + \vec{a}_t + \vec{a}_c$$

- $\vec{a}_{m/c} = \ddot{x}\hat{i}$
- $\vec{a}_t = l\ddot{\theta}(\sin\theta\hat{i} - \cos\theta\hat{j})$
- $\vec{a}_c = l\dot{\theta}^2(-\cos\theta\hat{i} - \sin\theta\hat{j})$

For the pendulum:

$$\vec{a}_m = \vec{a}_{m/c} + \vec{a}_t + \vec{a}_c$$

$$\vec{a}_{m/c} = \ddot{x}\hat{i}$$

$$\vec{a}_t = l\ddot{\theta}(\sin\theta\hat{i} - \cos\theta\hat{j})$$

$$\vec{a}_c = l\dot{\theta}^2(-\cos\theta\hat{i} - \sin\theta\hat{j})$$

The system consists of a cart with a mass (M), moving horizontally by input (u), with a massless rod with a length (L) holding a small mass (m), we want to keep the rod vertical (θ = Zero).

You can use Newtonian mechanics, or Euler-Lagrange equation to derive the equations of motion, but for the sake of brevity, you are given the equations in their final form.

$$(M+m)\ddot{x} - mL\ddot{\theta}\cos\theta + mL\dot{\theta}^2\sin\theta = u$$

$$L\ddot{\theta} - \ddot{x}\cos\theta - g\sin\theta = 0$$

We have two 2nd order differential equations, thus a total of 4 state variables is needed, selecting the states to be θ, θ', x, x'

Then we need to linearize these non-linear equations and use them to build our Linear State Space Model. (Full proof is discussed in the extra resources section at the end) Important note, our linearized model is valid only around small angles.

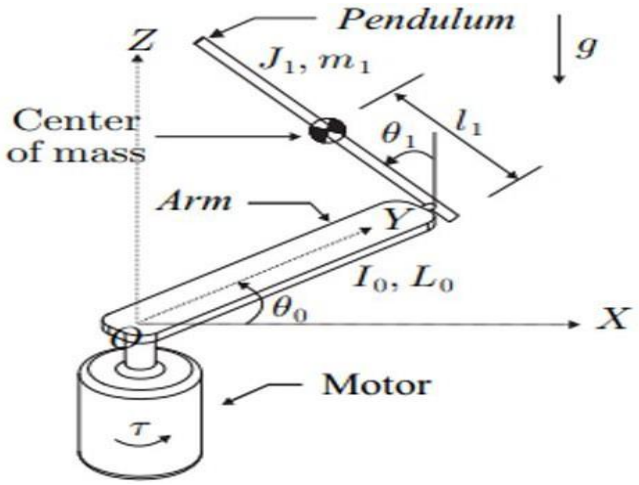
if delta= 1 the model is linearized about the Upwards position and if delta= -1 the model is linearized about the downwards position.

Final State Space

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{\Delta(M+m)g}{ML} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{mg}{M} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\Delta}{ML} \\ 0 \\ \frac{1}{M} \end{bmatrix} u$$

2- Euler-Lagrange Method

The Euler-Lagrange method is a powerful technique used to derive the equations of motion for dynamic systems, including the rotary inverted pendulum. This method is based on the principle of least action, which states that the path taken by a system between two states is the one for which the action integral is minimized. In this section, we apply the Euler-Lagrange formulation to our system, enabling us to model the dynamics effectively.



According to this paper**" Modeling and control of a rotary inverted pendulum using various methods, comparative assessment and result analysis**'' We can conclude the state space shown below:

$$\begin{bmatrix} \dot{\theta} \\ \dot{\alpha} \\ \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{bd}{E} & \frac{-cG}{E} & 0 \\ 0 & \frac{qd}{E} & \frac{-bG}{E} & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \alpha \\ \dot{\theta} \\ \dot{\alpha} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ c\frac{\eta_m \eta_g K_t K_g}{R_m E} \\ b\frac{\eta_m \eta_g K_t K_g}{R_m E} \end{bmatrix} V_m$$

Here, $a = J_{eq} + mr^2$, $b = mLr$, $c = 4/3mL^2$, $d = mgL$,

$$E = ac - b^2, \quad G = \frac{\eta_m \eta_g K_t K_m K_g^2 - B_{eq} R_m)}{R_m}.$$

Control Design

Before designing an LQR controller, **controllability** of the system must be verified. This ensures that the states of the system can be fully controlled using the input u(t)u(t)u(t). The system is controllable if the controllability matrix has full rank.

Controllability Matrix

For a state-space system:

$$\dot{x} = Ax + Bu$$

The **controllability matrix** $\mathcal{C}$ is defined as:

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix}$$

where:

- A: State matrix of size $n \times n$,
- B: Input matrix of size $n \times m$,
- n: Number of states.

The system is controllable if:

$$\text{rank}(\mathcal{C}) = n$$

If $\text{rank}(\mathcal{C}) < n$, it means the system is not fully controllable, and the LQR cannot control all states.

Why Controllability is Critical for LQR

- LQR relies on the ability to control all states of the system.
- Without controllability, some states cannot be regulated, leading to poor or unstable performance.

If the system is not controllable, consider redesigning the system, such as modifying A and B matrices, or adding actuators.

LQR Controller (Linear Quadratic Regulator) is an optimal control strategy used to design controllers for linear systems. It minimizes a cost function that balances state performance and control effort. LQR is widely applied in robotics, aerospace, and mechanical systems, including the inverted pendulum, robotic arms, and vehicle control.

Objective of LQR

To find an optimal control input u(t)u(t)u(t) for a linear system:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

that minimizes the quadratic cost function:

$$J = \int_0^\infty \left(x(t)^T Q x(t) + u(t)^T R u(t) \right) dt$$

where:

- x(t): State vector.
- u(t): Control input.
- A, B: System matrices.
- Q: State weighting matrix (penalizes deviations from the desired state).
- R: Control weighting matrix (penalizes excessive control effort).

Steps to Design an LQR Controller

1. Model the System

Obtain and linearize the state-space representation:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

where:

- $x(t)$: State vector (e.g., position, velocity, angle).
- $u(t)$: Input vector (e.g., force or torque).
- A: State matrix (describes system dynamics).
- B: Input matrix (describes how control input affects the system).

2. Choose the Weighting Matrices

Select the matrices Q and R:

- Q: Penalizes the deviation of states from their desired values. Larger values on the diagonal elements of Q increase the importance of minimizing specific states.
- R: Penalizes control effort. Larger values reduce control input magnitude, avoiding aggressive or excessive actuation.

Tip: Start with trial-and-error tuning of Q and R, then refine based on system response.

3. Solve the Algebraic Riccati Equation

Solve the Riccati equation to find the optimal state-feedback gain matrix K:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

where P is a symmetric positive definite matrix. MATLAB or similar software tools can solve this equation numerically.

4. Compute the Gain Matrix KKK

The optimal feedback gain matrix K is:

$$K = R^{-1}B^T P$$

This matrix defines how the control input u(t) is computed based on the current state x(t):

$$u(t) = -Kx(t)$$

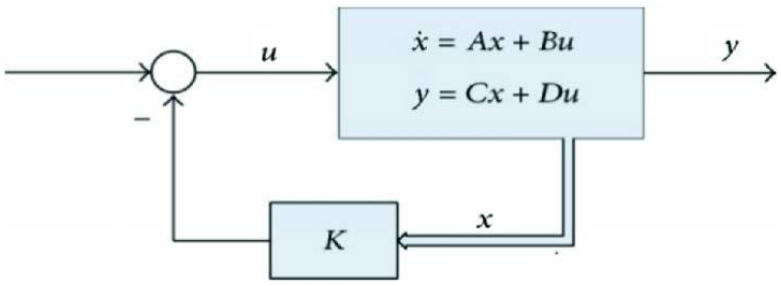
5. Implement the Controller

The control law becomes:

$$u(t) = -Kx(t)$$

Substitute $u(t)$ into the system to form a closed-loop system:

$$\dot{x}(t) = (A - BK)x(t)$$



Simulation

By Coding this up in MATLAB and see how it behaves by using Newton Law to modelling

Test.m

```
clear all
close all
clc
M=5; %Cart Mass [Kg]
m=1; %Pendulum Mass [Kg]
L=2; %Pendulum Length [m]
g=9.81; %Gravitational acceleration [m/s^2]
%Delta represents choice of linearization point
%Delta =1 upwards position
%Delta =-1 downwards position
Delta = 1;
A = [0 , 1 , 0 , 0;
    (Delta*g*(m+M))/(M*L) , 0 , 0 , 0;
    0 , 0 , 0 , 1;
    (m*g/M) , 0 , 0 , 0];
B= [0;
    Delta/(M*L);
    0;
    1/M;
    ];
C= eye(4);
D=0;
eig(A)
```

The Poles of the system before control

```
Poles =

         0
         0
    2.4261
   -2.4261
```

We can detect the pole in the right-hand side and that's why the system is **unstable**.

Then we Implement LQR Controller to achieve the stability

```
Q = [3 0 0 0;
    0 1 0 0;
    0 0 2 0;
    0 0 0 1];
R = 0.0001;

K = lqr(A,B,Q,R)
Poles after control=eig(A-B*K)
```

This is The K matrix which help to achieve stability

```
K =

   1.0e+03 *

    1.6763    0.7550   -0.1414   -0.2380
```

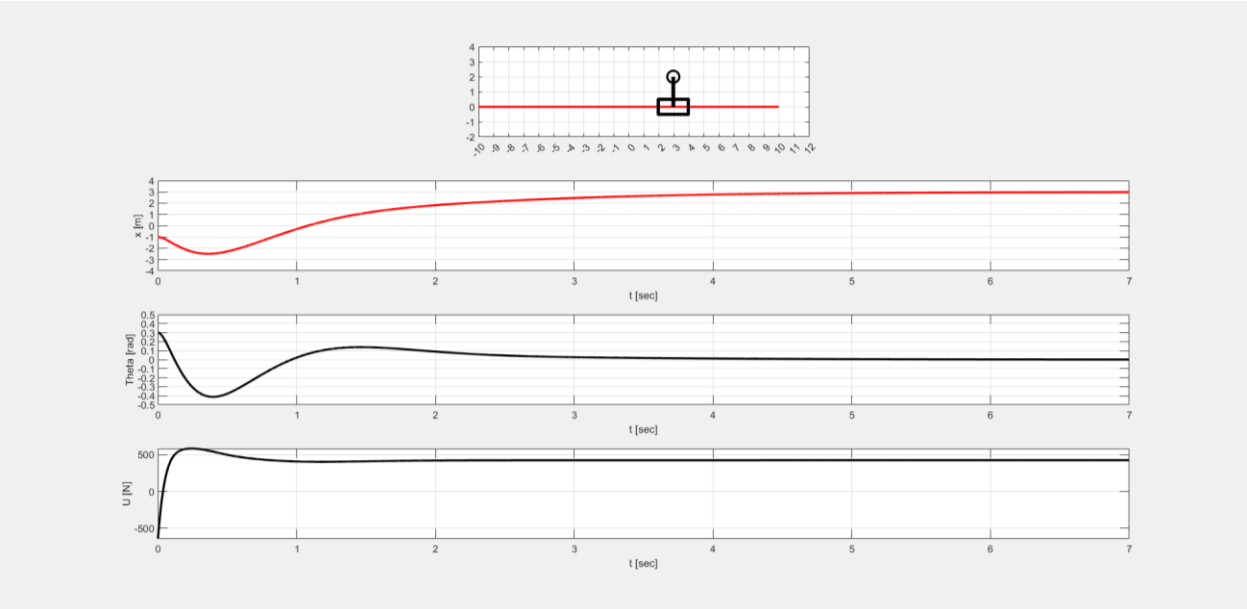
Poles after control will be:

```
Poles_after_control =

 -22.4000 + 0.0000i
  -2.0192 + 0.3823i
  -2.0192 - 0.3823i
  -1.4665 + 0.0000i
```

We can show the all poles in the left-hand side of S-plan and the LQR achieve stability of the system

Simulation Results



Hardware in the Loop

Hardware-in-the-Loop (HIL) Using Arduino Support Package in Simulink Hardware-in-the-Loop (HIL) simulation is a powerful technique for testing control algorithms by interfacing them with real-world hardware in real-time. Using Simulink's Arduino Support Package

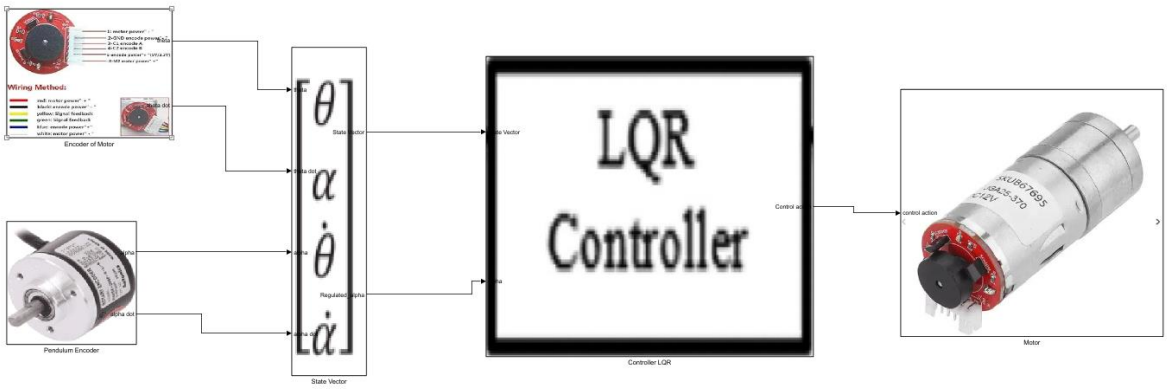


Figure 14: Simulink model for Hardware in the loop testing.

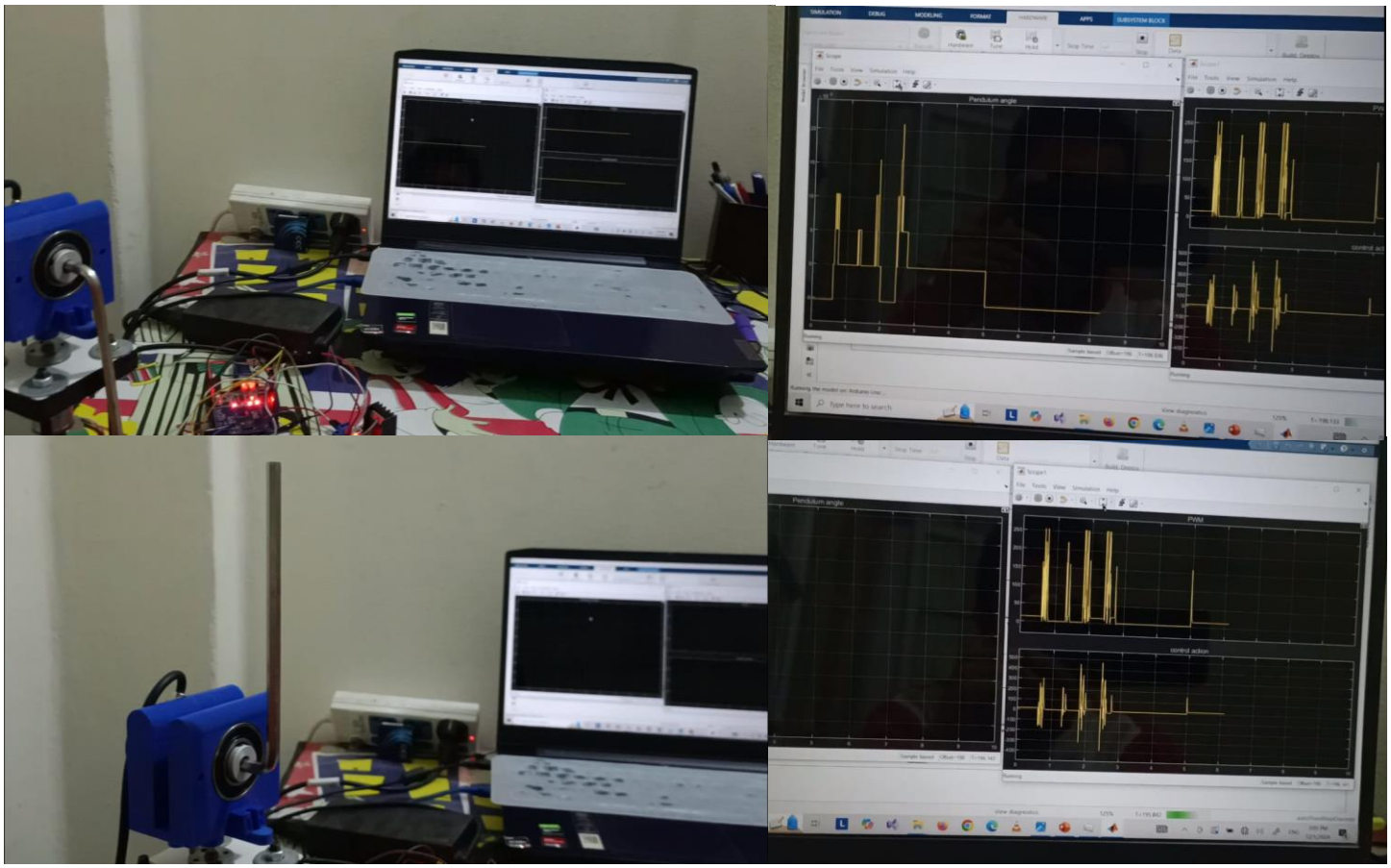


Figure 13: real life figures from the HIL testing.

Resources

- Paper "**Modeling and control of a rotary inverted pendulum using various methods, comparative assessment and result analysis**"
  - https://ieeexplore.ieee.org/document/5589450
- **LQR**
  - https://www.mathworks.com/help/control/ref/lti.lqr.html
  - https://www.youtube.com/watch?v=E_RDCFOlJx4
- **Classic Inverted Pendulum - Equations of Motion**
  - https://www.youtube.com/watch?v=5qJY-ZaKSic