YILDIZ TECHNICAL UNIVERSITY

DC Motor Controller

DC Motor Control via MATLAB/Simulink

Mustafa Tursun 19016909 – Deniz Gül Demir Ç190B1053

5/4/2024

KOM4620 REAL TIME CONTROL SYSTEM

1 Introduction

Problem approach and Objectives

The main purpose of this project is to develop a system for controlling and analyzing the behavior of a DC motor. An Arduino board will be used to interface with the motor and an encoder. The encoder will provide data on the motor's position and speed, while the Arduino board will control motor speed through Pulse Width Modulation (PWM) and collect data for further analysis. Simulink software will be used to visualize and analyze the collected data as well as to design and implement control algorithms for regulating the motor's speed. The project will dive through motor characterisation and the development of both open-loop and closed-loop control methods. The efficiency of these control methods will be evaluated through experimentation and data analysis.

1.1 Equipment Needed

● Arduino Uno
● L298N DC-Step motor driver board
● 12V DC power supply

1.2 Structure of Work

- **System Setup and Characterization**
  In this section, The necessary libraries and toolboxes have been installed to ensure communication between MATLAB and Arduino. Following the setup, PWM commands and corresponding motor speed information were obtained using Arduino commands on MATLAB.

- **Construction of 1D Look-Up Table**
  By using the obtained monotonic PWM Command-Motor Speed graph from the previous step, a 1D Look-Up table is constructed.The Look-Up Table was used to convert these motor speed values into appropriate PWM commands, which were then sent to the motor via the Arduino board to provide commands for motor control.

- **Open-Loop Control Design**
  The reference speeds for the motor were given and simulated in an open-loop configuration to evaluate the system's performance. This step aims to observe the motor's response to speed inputs without any feedback control.

- **Closed-Loop Control with P-Controller Design**
  A proportional controller is designed to enable the motor to follow the reference speed values in a closed loop system. (This involves tuning the controller parameters to achieve the desired performance). This controller uses feedback from the motor's actual speed to adjust the PWM commands it sends. By comparing the reference speed with the actual speed, the P-Controller applies a proportional correction to the PWM signal, allowing the motor to follow the target speed.

- **System Identification**
  Using the data obtained, the theoretical transfer function which represents the mathematical model that reflects the relationship between the control input (PWM) and the motor's output (speed) ,was estimated with the help of the Simulink Parameter Estimator application. A

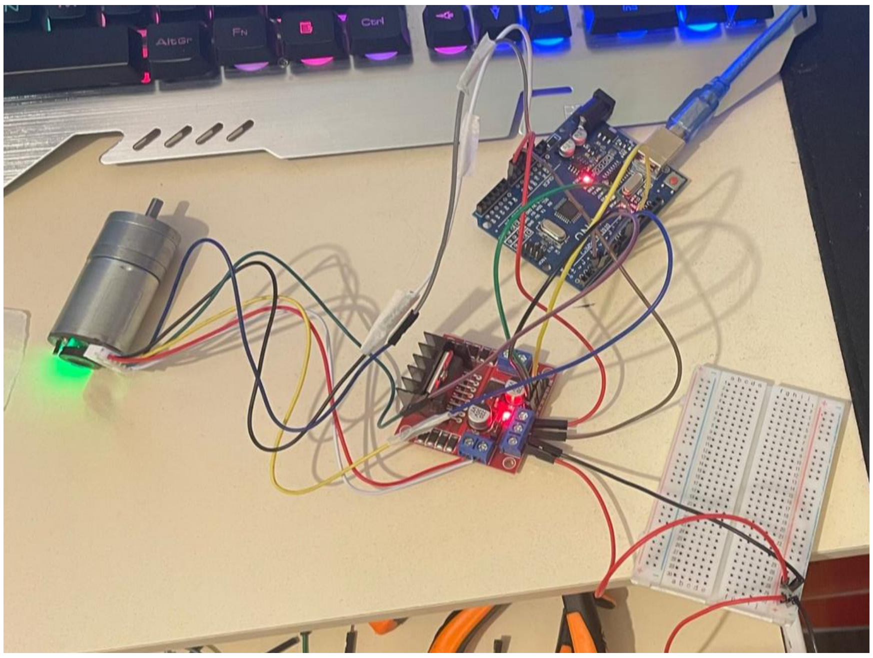
comparison between the actual system behavior and the theoretical transfer function was made to assess its accuracy. Thus, the overall performance of the system is evaluated.

2.Implementation

In this section the process from communicating with Arduino to motor control on MATLAB/Simulink environment will be described.

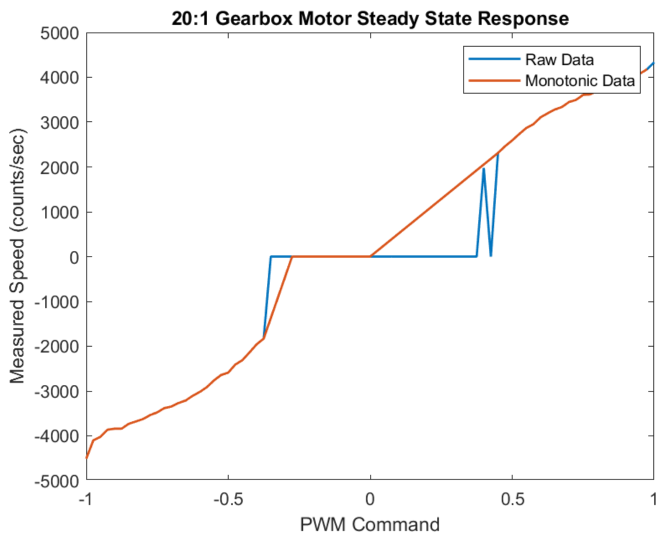
2.1 Real Time System Design of the DC Motor Control with Arduino Uno

Before controlling the motor, a connection was established between MATLAB and the Arduino Uno board to control the motor using the Arduino Uno. This was accomplished by installing the libraries for the Simulink Arduino IO package, the Simulink support package for Arduino hardware, and the MATLAB support package. 1 Arduino Uno R3, 1 L298N DC-Step motor controller board, 12V DC power supply and 6V DC motor with sensor were used for the experiment. The following figure illustrates the real-time experimental setup:



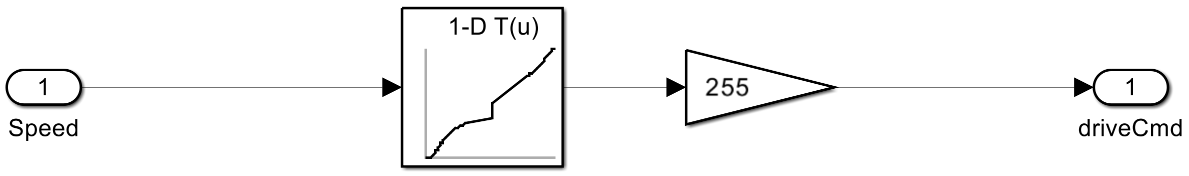
After the required libraries were installed and the connections established, the Arduino hardware installation was performed using the "arduinosetup" command. Then, the script in "KOM4620_Real_Time_Control_Systems_19016909_Ç190B1053_HW.m" was compiled to obtain 20:1 gearbox motor steady-state response. The graph shows that sending different PWM commands through the Arduino to the motor driver gives the corresponding motor speed using the Arduino commands implemented in MATLAB and measuring the resulting motor speed. The dead zone was defined almost between -0.2 and 0 PWM signal. Next, monotonically increasing data were obtained using a script to create a 1D lookup table. So, the lookup table was created successfully.

The static component of Coulomb friction can be potentially identified from the PWM command and measured speed graph obtained. **Coulomb friction** is a type of friction that acts as an opposing constant force until a certain threshold is passed. This threshold creates the dead zone in motor control. The flat region in the graph at low PWM values where the PWM command values increase but the measured speed remains zero represents the dead zone caused by static friction.



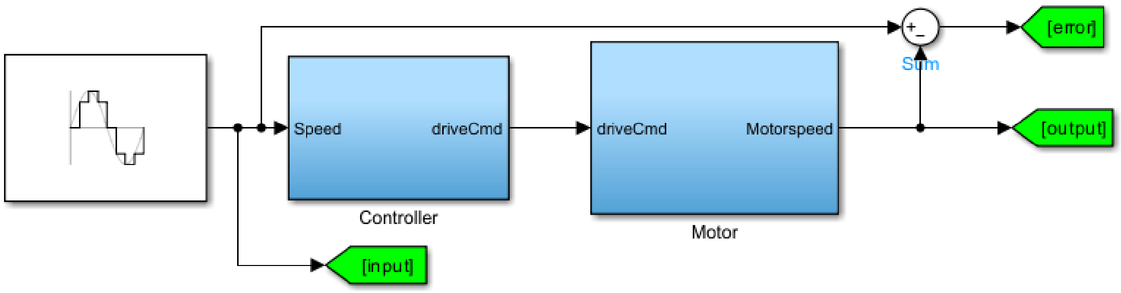
The collected data will be used to create a 1D Look-Up Table in Simulink. This table will transform desired motor speeds into corresponding PWM commands. The Look-Up table is useful for this case because it eliminates the need for complex calculations within the control loop and simplifies future control efforts for the simulation environment.

The generated look-up table was multiplied by 255 to obtain normalized PWM values.

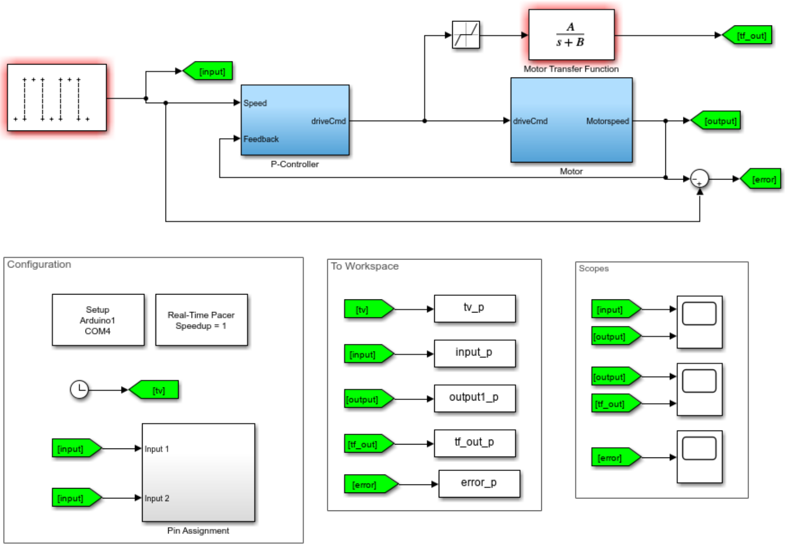


2.2 DC Motor system dynamics and control

In the beginning, the motor was tested in **open-loop configuration** with a sine wave input. In the controller subsystem, the system that produces the normalized PWM value, was used.

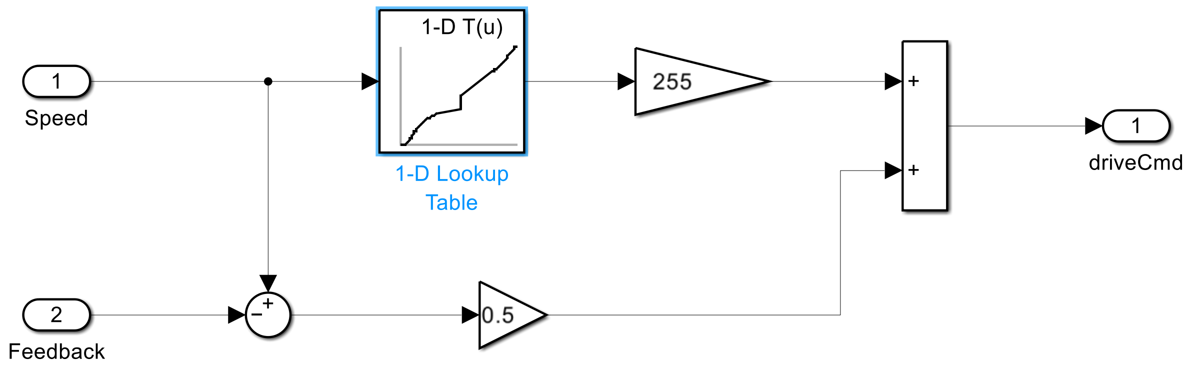


After than, the motor was controlled using **closed-loop** P-controller with pulse wave as a input.

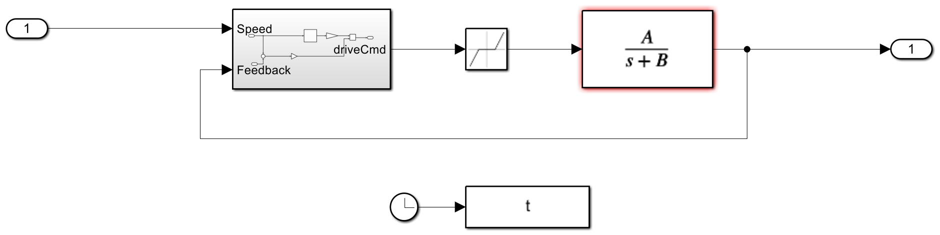


The P-Controller will receive reference speed which represents the desired speed the motor should achieve and the actual speed which is the feedback signal obtained from the simulated motor model.

The P-Controller will compare the reference speed with the actual speed and calculate the error between them. Proportional to this error, the P-Controller will adjust the PWM command signal sent to the motor. The error is attempted to be minimized for the motor to track the reference speed values.



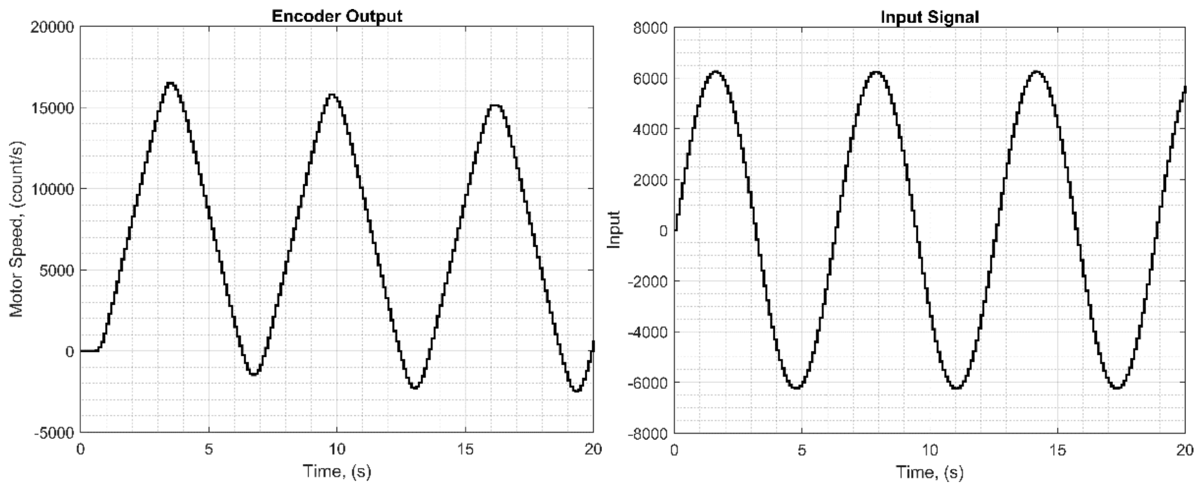
The speed of the motor managed with P-controller, the pulse input in the closed-loop dynamics have been recorded, and the Parameter Estimator was used to get an approximate transfer function of the motor by using experimental datasets.

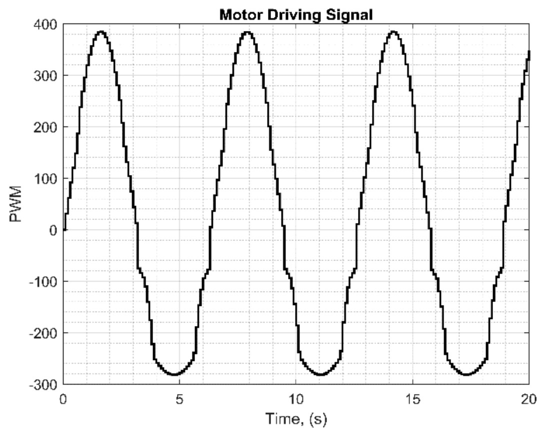


2.3 System results

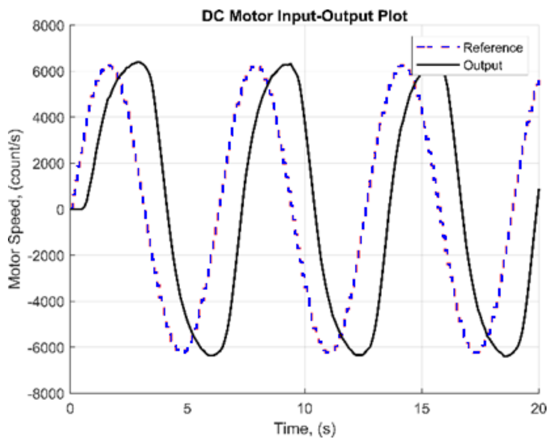
2.3.1 Generate PWM signals for a DC motor and Read Encoder Measurement Speed

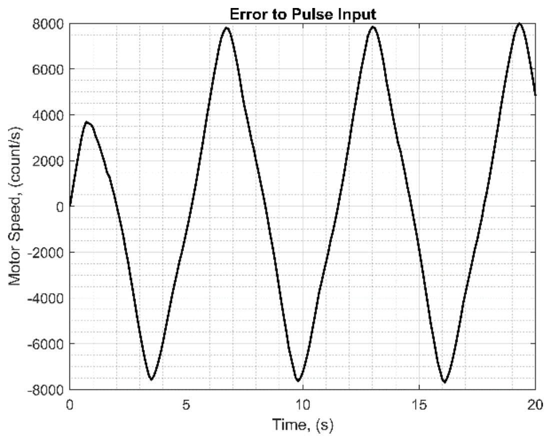
The input command given by the 1D Look-Up Table is transmitted to the motor as a normalized PWM signal.





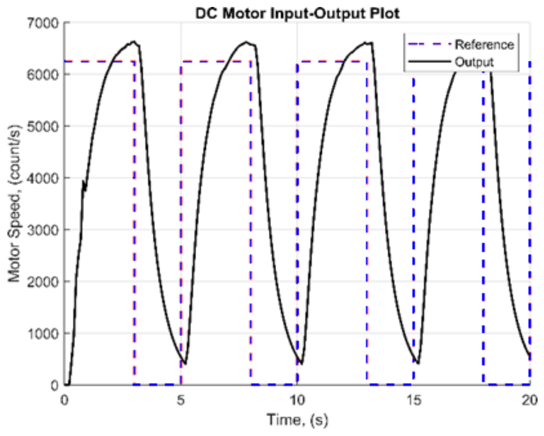
2.3.2 Open Loop Configuration of DC Motor

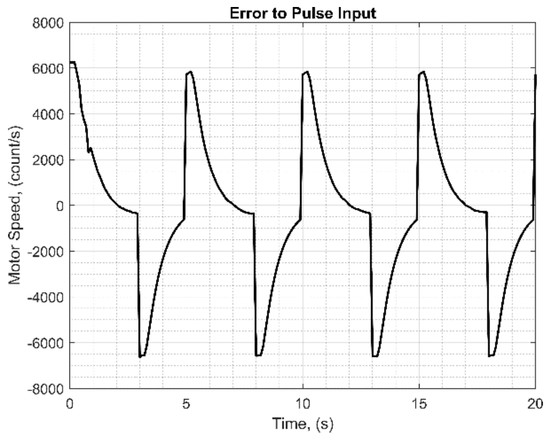




In an experimental setup of a DC motor with an open-loop configuration, a sine wave was provided as a reference signal to make the system follow it. However, it can be seen that there is a data transfer delay in the system. The main reason for this delay is reading data from the Arduino Uno board. As a result, a **phase shift** is observed.

2.3.3 Close Loop Control of DC Motor



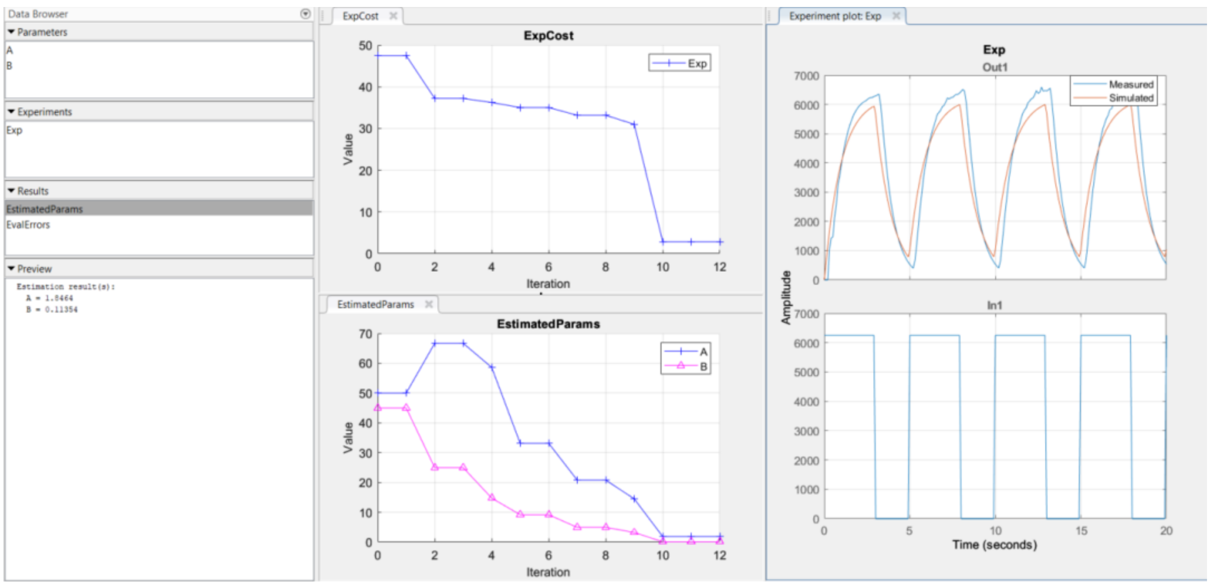


In a closed-loop configuration, a P-controller was used to monitor the pulse reference signal with a stable error. An integrator can be added to the controller to reduce this steady state error. The value of P gain used in the controller is 0.5.

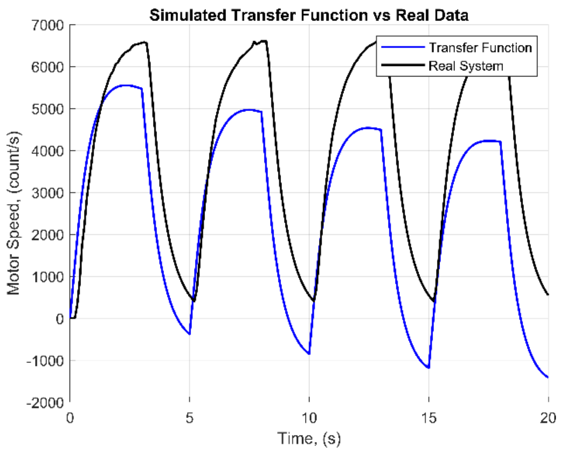
2.3.4. Parameter Estimation

The experimental data collected during the motor control in closed-loop configuration, will be utilized for system identification. The Simulink Parameter Estimator application will be employed to analyze this data as shown in the figure above and the transfer function of the DC motor was determined independently of the experimental setup.

This procedure is used to identify the mathematical model, in the form of a transfer function based on experimental data, that best represents the relationship between the control input (PWM signal) and the motor's output (speed). The identified transfer function serves as a theoretical model of the actual motor system. In addition, different control strategies can be developed by identifying the mathematical model of the real system using experimental data.



```
Estimation result(s):
  A = 1.8464
  B = 0.11354
```



In the graph above, the simulated transfer function is compared with real data. It cannot be expected to be exactly the same, but it should be close. According to this graph, it can be said that a transfer function that can be considered close is obtained.

2.4 Video Links of the Real Time Application

https://drive.mathworks.com/sharing/d18e7e22-dde2-4e05-95af-edac45eaaa96

3. Conclusion

This study resulted in a detailed study and implementation of DC motor control using MATLAB / Simulink with Arduino Uno. First, the necessary libraries and experimental setup were created to achieve communication between Arduino Uno and MATLAB/Simulink. Engine speed was controlled using a closed-loop control configuration. By designing the P controller, the stability error has been reduced, but it cannot be eliminated. In addition, the controller allows the system to control the desired reference. In addition, the Simulink Parameter Estimator tool was used together with the experimental data to understand the theoretical part of this study. This study provided a theoretical and practical overview of DC motor control using MATLAB/Simulink with Arduino Uno. This research enables us to combine the theoretical part of control theory with the practical part. In conclusion, this study presents a comprehensive overview of DC motor control using MATLAB/Simulink with Arduino Uno.

4. Personal Comment

In this study, we learned about communication between Arduino Uno and MATLAB, transfer of PWM signals to a DC motor, control of motor speed in two different configurations, open loop and closed loop. In addition, we learn how to obtain the transfer function of a DC motor using its experimental data. By doing this, we gained a better understanding of the mathematical model of the engine seen earlier. However, some of the results did not meet our expectations and we believe this may be due to delays between the motor, Arduino board and controller components.

5. Reference List

1. [https://drive.google.com/drive/folders/1-2wXda73C7dAg9rIp1Dt-CcMCC48sYdZ?usp=drive_link]

2. DC-Motor-Dynamics-Modeling-and-Position-Control [https://www.mathworks.com/matlabcentral/fileexchange/164976-dc-motor-dynamics-modeling-and-position-control]

3. Control Tutorials for MATLAB and Simulink : DC Motor Speed: System Modeling [https://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling]

4. DC Motor Position: System Modeling

   [https://ctms.engin.umich.edu/CTMS/index.php?example=MotorPosition&section=SystemModeling]

5. YAŞAR, C. F. (2024) Real Time Control Systems Lecture Note 5th Downloaded from Avesis

6. Control Tutorials for MATLAB and Simulink : Activity 6 Part (a): Time-Response Analysis of a DC Motor [https://ctms.engin.umich.edu/CTMS/index.php?aux=Activities_DcmotorA]