|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course Code** |  | **MEE428 Real Time Control** | |  |  |
| **Homework #4** |  | **DC Motor Position Control with Encoder/Potentiometer** | | | |
| **Related Learning Outcome:** | | **2** |  |  |  |
| **Group Number:** |  |  |  |  |  |
| **Group Member ID** | **Group Member Name** | | **Group Member Surname** | | **Group Member Signature** |
| 170412059 | Görkem |  | Akan |  |  |
| 160412056 | Abdullah |  | Aysun |  |  |
| 160412014 | MehmetBe | rke | Parlat |  |  |
| 180412057 | Mehmet A | kif | Demirlek |  |  |
|  |  |  |  |  |  |
|  |  |  | **Grading** |  |  |
| Q1 (40%) | | Q2 (20%) | | Q3 (40%) | |

**Introduction**

This report details the implementation and results of a DC motor position control system using MATLAB Simulink and Arduino hardware. The focus is on controlling the motor position through a feedback sensor connected to the Arduino, with the use of a potentiometer as the feedback sensor.

**Question 1: Motor Position Control Implementation**

Objective: Connect a potentiometer and a DC Motor driver to Arduino and implement a PID controller to manage the motor's position based on the potentiometer's feedback.

Methodology:

Hardware Setup: Connected the potentiometer to the appropriate pin on the Arduino to measure the motor's position. The motor driver was set up to drive the motor based on the PID controller's output.

Simulink Model: Developed a Simulink model using the MATLAB Simulink Support Package. This included reading the 10-bit analog value from the potentiometer, converting this value to a position reference, and implementing the PID controller.

Calibration: Before deploying the controller, calibrated the potentiometer to ensure accurate position measurement.

Real-Time Tuning: Utilized external mode for real-time tuning of the PID controller to achieve optimal motor control.

Results: Despite the actions we took, we could not control the position of the engine. Instead of constantly stopping at the angle we set as a reference, the motor constantly rotates. We changed the PID parameters many times, but the result did not change.

metin, ekran, görüntüleme, yazılım, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Question 2: Serial Communication Implementation**

Objective: Add a Serial Transmit block to the existing Simulink model to send the motor position values to a computer via serial communication.

Implementation:

Enhanced the Simulink model from Question 1 by adding a "Serial Transmit" block.

Deployed the model to the Arduino UNO.

Used the Arduino IDE to monitor serial communications, capturing the motor position as the potentiometer was manually adjusted.

metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Question 3: Integration with MATLAB App**

Objective: Utilize the MATLAB App created in Homework 3 to display motor position data in real-time on a GUI.

ekran görüntüsü, metin, yazılım, işletim sistemi içeren bir resim

Açıklama otomatik olarak oluşturuldu

classdef MotorControlApp < matlab.apps.AppBase

% Properties that correspond to app components

properties (Access = public)

UIFigure matlab.ui.Figure

StartButton matlab.ui.control.Button

StopButton matlab.ui.control.Button

ReadDataButton matlab.ui.control.Button

TextBox matlab.ui.control.EditField

SerialPort matlab.serial.SerialPort

end

methods (Access = private)

% Button pushed function: StartButton

function StartButtonPushed(app, ~)

try

if isempty(app.SerialPort) || strcmp(app.SerialPort.Status, 'closed')

app.SerialPort = serialport('COM3', 9600);

configureTerminator(app.SerialPort, "CR/LF");

configureCallback(app.SerialPort, "terminator", @app.readSerialData);

app.TextBox.Value = 'Connection Opened';

end

catch e

app.TextBox.Value = ['Error: ', e.message];

end

end

% Button pushed function: StopButton

function StopButtonPushed(app, ~)

try

if ~isempty(app.SerialPort) && strcmp(app.SerialPort.Status, 'open')

delete(app.SerialPort);

app.TextBox.Value = 'Connection Closed';

end

catch e

app.TextBox.Value = ['Error: ', e.message];

end

end

% Callback function to read data from serial port

function readSerialData(app, src, ~)

data = readline(src);

app.TextBox.Value = data; % Update the text box with received data

end

end

% Component initialization

methods (Access = private)

% Create UIFigure and components

function createComponents(app)

% Create UIFigure and hide until all components are created

app.UIFigure = uifigure('Visible', 'off');

app.UIFigure.Position = [100 100 400 300];

app.UIFigure.Name = 'Motor Control App';

% Create StartButton

app.StartButton = uibutton(app.UIFigure, 'push');

app.StartButton.Position = [100 200 100 22];

app.StartButton.Text = 'Start Connection';

app.StartButton.ButtonPushedFcn = createCallbackFcn(app, @StartButtonPushed, true);

% Create StopButton

app.StopButton = uibutton(app.UIFigure, 'push');

app.StopButton.Position = [210 200 100 22];

app.StopButton.Text = 'Stop Connection';

app.StopButton.ButtonPushedFcn = createCallbackFcn(app, @StopButtonPushed, true);

% Create TextBox

app.TextBox = uieditfield(app.UIFigure, 'text');

app.TextBox.Position = [100 150 210 22];

app.TextBox.Editable = 'off';

end

end

% App creation and deletion

methods (Access = public)

% Construct app

function app = MotorControlApp

% Create and configure components

createComponents(app);

% Register the app with App Designer

registerApp(app, app.UIFigure);

% Show the figure after all components are created

app.UIFigure.Visible = 'on';

end

% Code that executes before app deletion

function delete(app)

% Delete UIFigure when app is deleted

if isvalid(app.UIFigure)

delete(app.UIFigure);

end

end

end

end

Implementation:

Connected the Arduino deployment from Question 2 to the MATLAB App.

The app was designed to receive and display motor position data in real-time within a designated textbox.

Conclusion:

The project successfully demonstrated the integration of hardware and software tools to achieve precise control and monitoring of a DC motor's position. The real-time control and feedback via the MATLAB App provided practical insights into the dynamics of PID control systems.