ME477, Lab 4

Ali Jones

2/11/2022

Description:

This program communicates with a DC motor by sending a PWM current generated by a finite state machine. The overall hierarchical structure and various states of the program are shown below. The *main* function loops through pointers to the various states, activating them if conditions are met.

```
main
|_ initializeSM()          - state machine initialization
|   |_ Dio_WriteBit()      - sets channel high or low
|   |_ printf_lcd()        - prints to LCD display
|_ double_in()             - user keypad input
|_ wait()                  - waits for 5ms
|
stateLow                   - current on
|_ Dio_WriteBit()          - sets channel high or low
|
stateHigh                  - current off
|_ Dio_WriteBit()          - sets channel high or low
|_ Dio_ReadBit()           - detects channel state
|
stateSpeed                 - current on, printing speed
|_ vel()                   - calculates motor speed in BDI/BTI
|   |_ Encoder_Counter()   - reads the position of the encoder
|_ printf_lcd()            - prints to LCD display
|
stateStop                  - motor stopped
|_ printf_lcd()            - prints to LCD display
|_Dio_WriteBit()           - sets channel high or low
```

Testing:

1. Run the program
2. The LCD screen will display the prompt “Enter N”
    a. Enter the desired number of 0.05ms ‘wait’ intervals in each Basic Time Unit (5 suggested)
    b. Hit enter
3. The LCD screen will display the prompt “Enter M”
    a. Enter the desired number of “wait” intervals that the motor signal will be on (must be less than N total intervals. 3 suggested)
    b. Hit enter.
4. The motor will start running and the oscilloscope will display the PWM signal
5. Press the “Ch7” button
    a. The LCD screen will print the calculated RPM of the motor
    b. The oscilloscope will deviate from its initial signal during the ‘print’ command

6. Press the “Ch6” button
    a. The LCD screen will print “Stopping”
    b. The motor will stop

Results:

Considering all of the operation codes in the wait() function, the number of cycles in its loop, and the frequency of the microcomputer, the “wait” delay should be approximately 5ms. The conversion from BDI/BTI to RPM is then BDI = 1/2000 rev, and BTI = (0.005 sec)(1min / 60sec)(N). Using a delay routine for time measurement in this way is simple, but if the time delay is too large it can lead to a lagging controller and insurmountable steady-state error.

The oscilloscope showing the PWM waveform of the motor can be seen in figure 1, below. All oscilloscope pictures are using N=5 and M=3.

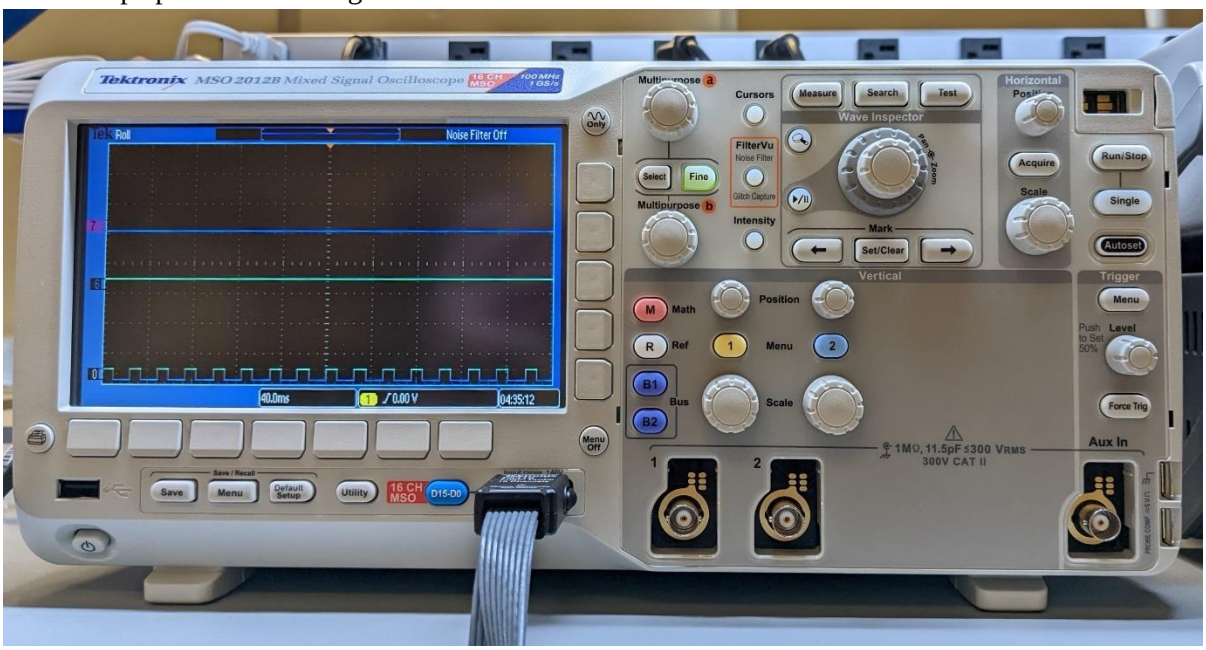

Figure 1. Oscilloscope with PWM waveform at the base of the screen

Each dotted segment on the screen represents 40.0ms and there are 5 small tickmarks per segment, so each small tickmark represents 8ms. Each BTI covers slightly more than 3 tickmarks, so **1 BTI is just over 24ms**. With a “wait” interval of approximately 5ms and N=5 “wait” intervals per BTI, **each BTI should be 25 ms.**

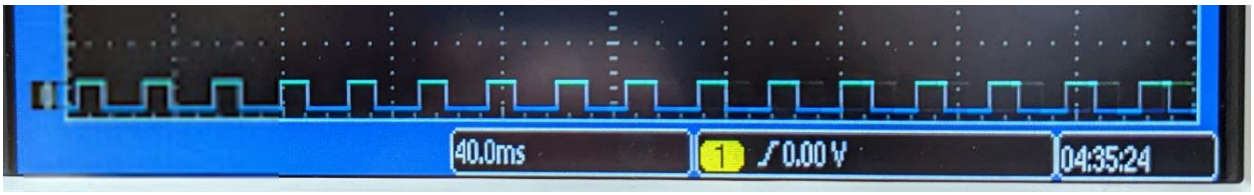

Figure 2. Segment of oscilloscope showing waveform and time slices

The *speed()* function prints out the RPM of the motor, but since printing takes a significant amount of time it introduces delays in the system and distorts the PWM signal. A distorted BTI takes almost the entire 40ms.

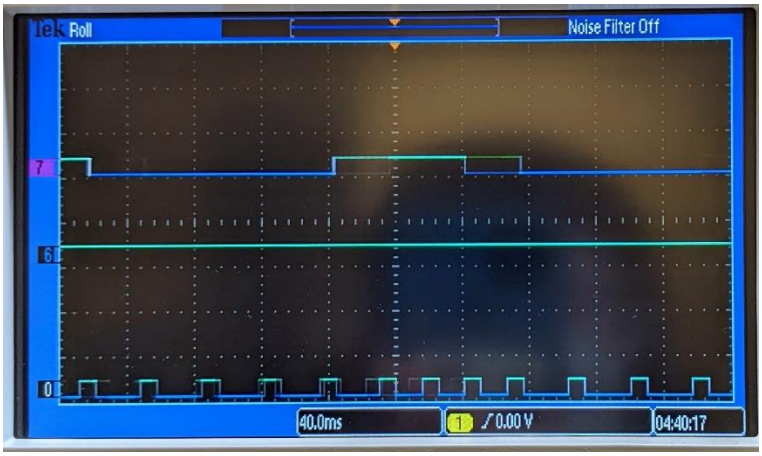

Figure 3. Oscilloscope showing distorted PWM signal when printing speed

To find the true velocity of the motor, the print statement must be suppressed before exporting the data. Assuming that each data point was added to the exported buffer at the "wait" delay of 5ms, the open-loop motor behavior can be plotted. The following figures show the graph of the motor as it ramps up to steady state and a measurement of the motor made by a digital tachometer. From this data, we can determine that the time constant τ is 1.12 sec and the steady state velocity is approximately 650 rpm.

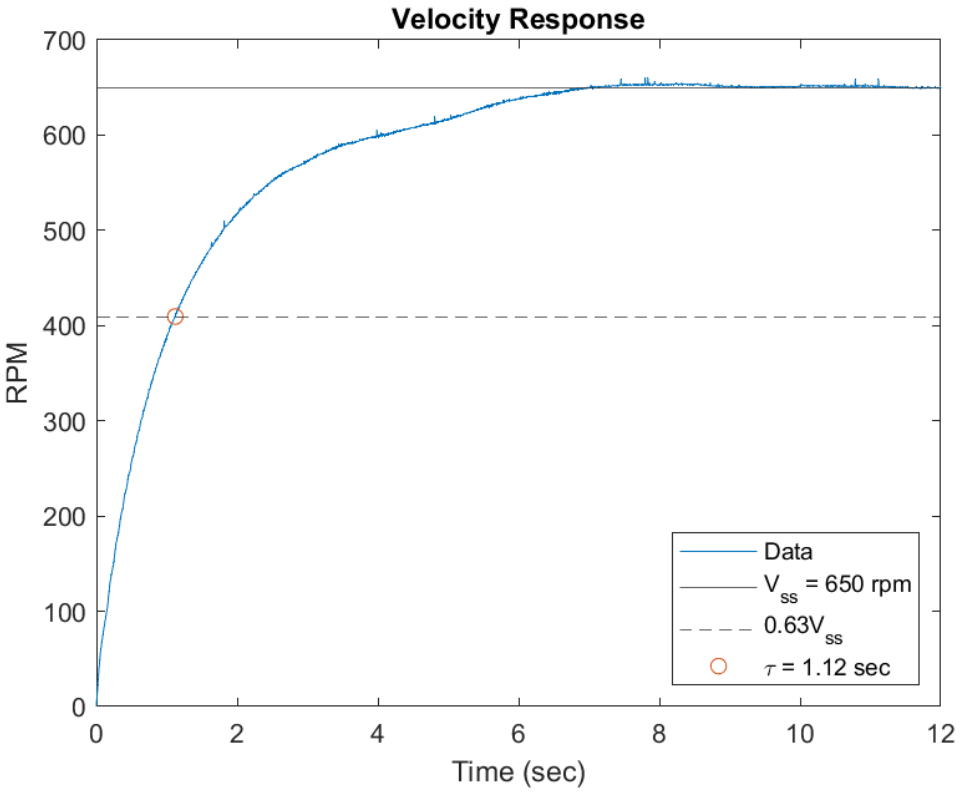

Figure 4. Graph of velocity plot with time constant **τ = 1.12 seconds** and corresponding steady state measurement

Potential improvements:

As discussed above, the printed velocity displayed on the LCD screen differs from the true velocity because the time it takes to print chagnes the length of 1 BTI. A more robust program that can display the accurate current speed would be necessary in a practical setting, as it would allow the user to monitor the motor more accurately.

MATLAB Code:

```
clear all; close all; clc

load('Lab4.mat')

v_ss = 650; % found by inspection
t = linspace(0,length(vel)*0.005,length(vel));  % assuming each data point took 0.005 sec to add
to buffer

vel_1_tau = (650*0.63);     % velocity at one time constant, 63% of steady state

% find the index of the time constant
limit = 0.5;
for i = 1:length(vel)
    if abs(vel(i)-vel_1_tau) < limit
        index = i;
        break
    end
end

tau=t(index);

plot(t, vel)
    hold on
    yline(v_ss)
    yline(vel_1_tau, '--')
    plot(tau, vel_1_tau, 'o')
    legend(["Data", "V_{ss} = 650 rpm", "0.63V_{ss}", "\tau = 1.12 sec"], location = "southeast")
    title("Velocity Response")
    xlabel("Time (sec)")
    ylabel("RPM")
```