ECE 315

Lab Section H31

Lab 4

Due Date: April 9, 2022

Name:

- Chanpreet Singh
- Gurbani Baweja

Table of Content

Abstract

In this lab we gain expertise in using microcontroller, running FreeRTOS real-time kernel to control the operation of a small stepper motor. We also enhance the user interface of a stepper motor control application on the Zybo Z7. This lab also enables us to gain experience with measuring experimentally the speed and acceleration limits of a small stepper motor.

The controller platform is Diligent Zybo Z7 with CPU0 in the Zynq-7000 SoC running the FreeRTOS real-kernel. We use 28BYJ-48 stepper motor with unipolar drive windings, with one 5V DC power supply, one ULN2003 type driver module, one LTV-847 opto-isolator transistor array, four 220-ohm resistors, four 10-Kohm resistors, wires, and one breadboard.
A stepper motor, also known as a step motor or stepping motor, is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can be commanded to move and hold at one of these steps without any position sensor for feedback (an open-loop controller), as long as the motor is correctly sized to the application with respect to torque and speed [1].

In this lab we complete two exercises, in the first exercise, we complete the FreeRTOS task by implementing _Task_Motor() by calling functions to move the motor. In the second task, we implement an emergency stop command for the stepper motor. We use BTN0 of Zybo Z7 board, so when it is pressed three consecutive times in a row, then the motor immediately begins decelerating to stop. A red LED on Zybo board flashes on/off with 2Hz frequency when BTN0 is pressed. The emergency stop conditions persist until Zybo Z7 board is reset by pressing the red PS-SRST reset push button.

Design:

Setup: In this lab, we connect Zybo Z7board with an optoisolation circuit (bridge signal) with ULN2003 driver module, which further connects with the stepper motor. 5V power is also connected to ULN2003 driver module so it could power the stepper motor. The Zybo board is powered by a USB connection to the computer. Our setup and connections are shown in Figure 1. After downloading the project files from EClass, we update the system.mss file BSP settings, to make kernel behaviour tick_rate from 100 to 1000. We also add a mathematics library to ensure proper compiling of the code [2].

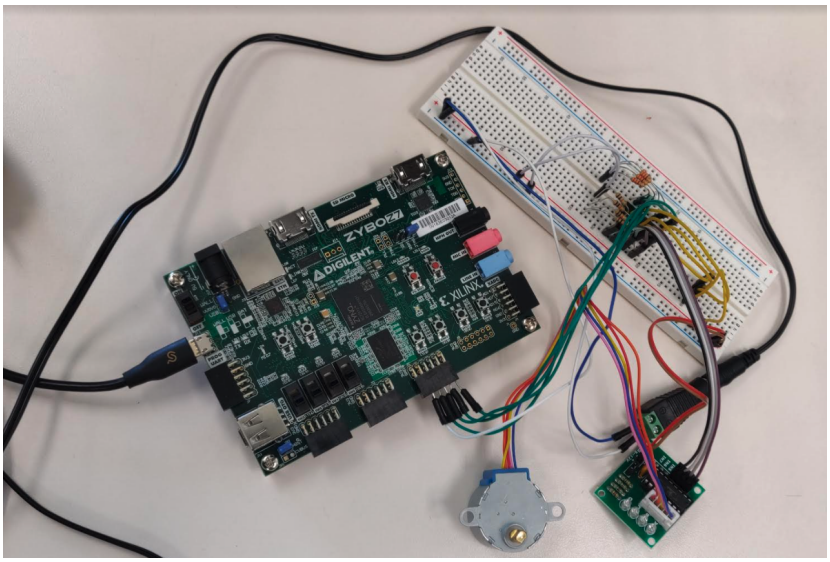

Figure 1: Lab Setup

In exercise 1, we complete the FreeRTOS task by implementing _Task_Motor() by calling functions to move the motor. We start by getting the motor parameters from the queue (FIFO1). Then we set the motor parameters by calling Stepper_setSpeedInStepsPerSecond, Stepper_setAccelerationInStepsPerSecondPerSecond, Stepper_setDecelerationInStepsPerSecondPerSecond, and Stepper_setCurrentPositionInSteps functions. Then we use destination pairs to move the motor by using Stepper_moveToPositionInSteps function with a loop, and once the desired position is reached we disable the motor and then execute the dwell time delay using the conventional vTaskDelay().

In exercise 2, we implement an emergency stop command for the stepper motor. We use an infinite loop and keep detecting if BTN0 is pressed or not on the Zybo Z7 board. Each time BTN0 is pressed we increment pressedCount variable and once pressedCount is higher than or equal to three, then we start the emergency stop procedure. We use Stepper_getCurrentPositionInSteps function to set the motion of motor so it start decelerating (targetPosition_InSteps - decelerationDistance_InSteps), then we rest of the destination position-delay pairs to zero. We flash a red LED light on Zybo on and off with a 2 Hz frequency, while the motor speed is decelerating. The emergency stop conditions persist until Zybo Z7 board is reset by pressing the red PS-SRST reset push button.

Test Suit:

Test ID	Description	Expected Result	Actual Result	Success/Failure
T1	Enter destination location as 2048 with dwell time 1000	1 rotation clockwise	1 rotation clockwise	Success
T2	Enter destination location as -2048 with dwell time 1000	1 rotation counter clockwise	1 rotation counter clockwise	Success
T3	Enter destination location as 4096 with dwell time 1000	2 rotations clockwise	2 rotations clockwise	Success
T4	Enter destination location as 0 with dwell time 1000	No movement	No movement	Success
T5	Emergency stop BTM0 is pressed	The motor stop and red-light flash	The motor stopped and red-light flash	Success

Note: In our lab, the motor was stalling thus the exact results could not be computed, but for the purpose of the report the actual results same as expected results.

Result:

In this lab, we implemented the functionality to rotate the stepper motor as per the user-entered destination posting with the dwell time. Along with that we also implemented an emergency stop button, so the user to stop the motor with a flashing red light. 2048 was equivalent to one clockwise circular rotation of the motor and -2048 was equivalent to one counterclockwise circular rotation of the motor.

Conclusion:

We were successfully able to use microcontroller running FreeRTOS real-time kernel to control the operation of a small stepper motor. The user was able to enter the destination position and delay, guiding the rotation of the motor. The Emergency button was able to decelerate the motor along with a flashing red light, the emergency button had to be pressed thrice to activate its functionality. In this lab we had one minor issue that our motor was stalling, even with an extensive root cause analysis, we were not able to find the reasoning for it. We tried replacing our hardware with other team members' hardware, and all connections were thoroughly checked.After extensive hardware checks and the code analysis, we could not find the bug that was causing the motor to stall.

References:

[1] https://en.wikipedia.org/wiki/Stepper_motor
[2] Lab handout:
https://eclass.srv.ualberta.ca/pluginfile.php/8210475/mod_resource/content/1/ECE315_Winter2022_Lab_4_handout_v0.pdf