Lab 3 Prelab: Software Based Finite State Machines

Owen Blair

# Goal and Background Information

The goal for Lab 3 is to move a stepper motor at a constant 15 RPM. The mode and direction that the stepper motor moves in is determined by the inputs from button 1 and button 2.

New subsystems and concepts used are Enums, software finite state machines, and open loop control. Enums is a defined set of values that a specific Enum variable can be. For example, if you have written enum apartment\_numbers{bottom\_floor = 1, middle\_floor = 2, top\_floor = 3} then you could use them to ensure that only certain values were used when using the Enum when executing instructions. Software finite state machines are like FSM that were defined in digital logic (ECE-240) but are defined using software and switch statements instead of logic gates. Lastly, open loop control or real-time open loop control implies that this type of scheduling is time dependent and tasks must happen at specific times. For lab 3, the code will read, decode, implement the software FSM, and write to the stepper motor as fast as possible but then will delay using the sw\_delay function. This kind of timing is real-time loop control.

For most of the lab I intend to copy and paste the functions needed because I have already written most of them. This includes functions like system\_init(), sw\_delay(), and read\_buttons(). The function decode\_buttons() will be mostly identical with a little modification to output the needed delay to maintain the 15 RPM depending on the step mode. The functions stepper\_state\_machine() and output\_to\_stepper\_motor() will be needed to written from scratch.

The stepper\_state\_machine() will take two inputs, the current state of the stepper motor and the decoded inputs from the buttons and will output a next state. The logic of the function will made of a switch statement that is controlled by the current state and then if statements that output the next state depending on the button inputs.

Output\_to\_stepper\_motor() will utilize an Enum that defines the different instructions that can be written to the stepper motor. This way anything written to the stepper motor using the Enum can’t write an invalid instruction to the stepper motor control board.

**Data Flow Diagram**

**Diagram

Description automatically generated**

**Control Flow Diagram**

**Diagram

Description automatically generated**