Rodrigo Becerril Ferreyra
CECS 346 Section 03
Lab 3
30 September 2020

1 Introduction

The purpose of this lab is to extend the previous lab (Lab 2) in preparation for Project 1, where we will be tasked to create a fully-functional traffic light controller. Specifically, we were tasked with the following: add a second set of lights (for a total of six LEDs) to simulate north/south and east/west traffic lights, add a reliable delay using the SysTick timer functionality, and implement a finite state machine (FSM) controller to handle switching the lights. The timer will be responsible for controlling how long a given light is held for. The FSM will handle the current state, next state, and outputs based on the current state (a Moore FSM). The functionality of Lab 3 is as follows:

- If no switches are pressed, stay on the current state.
- If the NS (north/south) switch is pressed, make the NS light green and the EW (east/west) light red.
- If the EW switch is pressed, make EW green and NS red.
- If both switches are pressed, rotate between green on EW and NS.
- The default state on reset is NS = red, EW = green.

The delay for holding a green light is 2 s, and the delay for holding a yellow light is 1 s. Lights switch as usual (green, yellow, red, green, etc.).

2 Hardware

The hardware in this Lab is simply an extension of the last lab. We are again using pins `B[1:0]` as switch inputs (with $10\,\mathrm{k\Omega}$ pull-down resistors), and pins `E[2:0]` as outputs for the NS set of lights. We are adding pins `E[5:3]` as outputs for the EW set of lights as an extension for this Lab. All LEDs source less than $2\,\mathrm{mA}$ of current from the board and are hooked up to $690\,\Omega$ ($470\,\Omega + 220\,\Omega$) current-limiting resistors.

3 Software

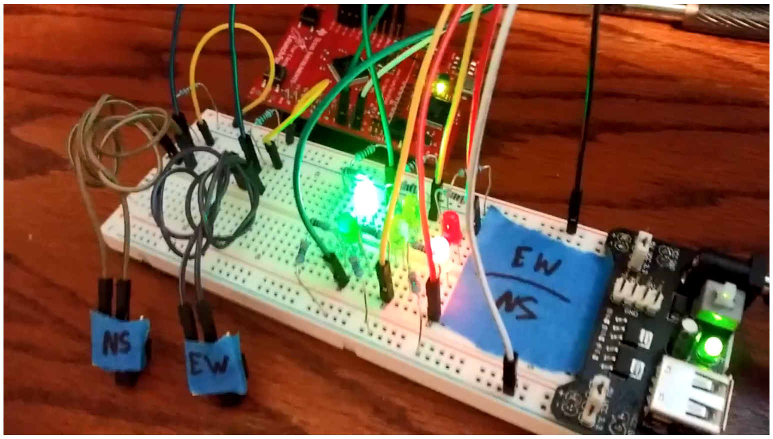
The software was easy to extend from Lab 2.

The SysTick timer is a hardware timer separate from the General-Purpose timers found on the board's microcontroller. It only has three registers to its name, and is simple to initialize. It is a 24-bit count-down timer intended for light-use applications such as delay. Before SysTick, we had to use an empty `for` loop in order to achieve an approximate delay time. The number of iterations had to be experimentally found, and that was only to a certain degree of accuracy. With the SysTick timer, we know by the datasheet exactly how fast the timer counts down: at 16 MHz, which is equivalent to one count every 62.5 ns. From this, one can calculate the amount of counts needed to achieve a 1 µs delay (which is $16 - 1$ since zero is also counted), and from there one can create a loop which runs this delay 1 000 000 times creating a precisely 1 s delay.

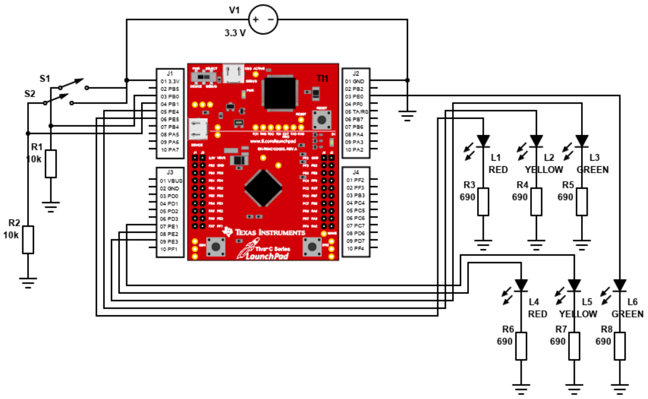
The finite state machine was implemented using a `struct`. A `struct` is similar to a class in object-oriented programming languages, as it includes variables inside of it (but cannot contain function definitions). Together with the `typedef` keyword, it can be used similarly to a class. The FSM "skeleton" has output, delay, and next state variables; we only initialized one FSM, but we could have initialized more to provide alternate decision-making functionalities if we wished. The FSM was created to the specifications above (See Section 1).

4 Media

- Link to video demonstration:
  - `https://youtu.be/fB_rXp9ZSGE`
- Picture of embedded system:



- Hardware schematic:


V1
3.3 V
S1
S2
R1
10k
R2
10k
J1
01 3.3V
02 PB5
03 PB0
04 PB1
05 PE4
06 PE5
07 PB4
08 PA5
09 PA6
10 PA7
J3
01 VBUS
02 GND
03 PD0
04 PD1
05 PD2
06 PD3
07 PE1
08 PE2
09 PE3
10 PF1
TI1
Texas Instruments
Tiva C Series
LaunchPad
J2
01 GND
02 PB2
03 PE0
04 PF0
05 TA/RS
06 PB7
07 PB6
08 PA4
09 PA3
10 PA2
J4
01 PF2
02 PF3
03 PB3
04 PC4
05 PC5
06 PC6
07 PC7
08 PD6
09 PD7
10 PF4
L1
RED
L2
YELLOW
L3
GREEN
R3
690
R4
690
R5
690
L4
RED
L5
YELLOW
L6
GREEN
R6
690
R7
690
R8
690