Task 2

ESOS and the ECE 4723/6723 Target Board

"Standing in the middle of the road is very dangerous; you get knocked down by the traffic from both sides."
– Margaret Thatcher (1925-2013)

THIS TASK has you begin using ESOS and start the population and testing of your custom-developed ECE 4723/6723 lab development board. This task will implement a traffic signal controller that dynamically adjusts to traffic conditions.

2.1 Hardware

2.1.1 Schematic errors

The schematic[1] for the development board has not been verified to function properly. There may be errors[2] in the schematic. These errors may cause

- peculiar, unreliable, or sporadic operation,
- the board not to function at all, or
- physical and/or electrical damage to the board's components, its surroundings, or you!

Therefore, anything and everything on the schematic must be validated (through design inspection) before any stuffing, soldering, or power-on testing is performed. Any errors found should be noted in your task documentation.

[1]The schematic can be found on the lab website.
[2]In fact, it is almost guaranteed.

2.1.2 Component selection and justification

The schematic for the development board does not have verified component (resistor, capacitor, etc.) values. Using the schematic and the datasheets for the board's ICs, determine the correct component values. Record your results in a spreadsheet. The spreadsheet should contain columns with

- component number (R4, U3, C21, etc.)
- component description with value (10uF tantulum capacitor, 3-to-8 decoder 74HC138, etc.)
- component manufacturer
- full component part number including packaging suffix
- component's purpose
- justification with citation into the appropriate datasheet(s) for the value you deemed appropriate

2.1.3 Partial-build and test

Determine a systematic manufacturing and testing procedure for bringing up the development board subsystems. Your procedure should include the order of component stuffing and soldering with any pertinent instructions to the builder. At various steps of the build, you should have the builder power up the subsystem just completed and measure various value to ensure that stuffing is correct and solder joints have continuity. Ideally, if measured values are incorrect, your instructions should direct the builder to their mistake.

Only necessary and related components should be installed during each subsystem. For example, a resistor for LCD character module should be installed during LCD subsystem section, and should not be installed when during the MCU subsystem.

You will be provided crucial components, but minor components such as resistors and caps should be procured by you and/or your team.

For this lab task, create the build-and-test documentation for the following subsystems (in order):

- power (off-board sources to the verified PCB supply rails)
- MCU
- MCU output LEDs
- MCU serial connection to PC (FTDI $\Leftrightarrow$ MCU)
- MCU programming (ICSP)
- MCU inputs (SW1, SW2, SW3, RPG1, and POT)

DO NOT POPULATE OTHER SUBSYSTEMS AT THIS TIME!

2.2 Software

2.2.1 Software preparation

Using the schematic for the revision F14 target board, develop an appropriate header file[3] called `revF14.h` that will create user-friendly definitions and macros for manipulating *all* hardware on the target board. Your macros should be written to ensure atomic operation and prevent errors if the header file is included multiple times. Macro and definition names should be consistent with other macro names in the distro.

All code and hardware developed in this lab should follow the Python and C language coding conventions as well as the hardware conventions.

You should create macros and definitions to manage and fully utilize[4] the following components:

- LED1 (equipped with red LED)
- LED2 (equipped with yellow LED)
- LED3 (equipped with green LED)
- SW1, SW2, and SW3

2.2.2 Traffic signal controller specification #1

Your task is to build a traffic signal controller that will sequence the traffic lights for the intersection of two roads – a north-south road, and an east-west road. Each road possesses a standard red-amber-green traffic signal (represented by LEDs 1-3). Since the target board only has one set of LEDs you will select which traffic signal is illuminated on the LEDs by SW3. If SW3 is not depressed, the N-S traffic signal is displayed. If SW3 is depressed, the E-W signals are displayed.

A standard signalling pattern for R/A/G traffic signals is

duration	10 s	3 s	10 s	3 s		
N-S signal	R	R	G	A	...	repeat
E-W signal	G	A	R	R	...	repeat

Design an ESOS application `t2_rag1` to implement this traffic signal specification. Use only ESOS tasks. Be sure to properly debounce any switches.

[3]You can expect this header file to grow longer and more full-featured as the semester progresses.
[4]Initialization, use, and shutdown, as required

2.2.3 Traffic signal controller specification #2

Using the previous application as a starting point, create/extend your traffic signal controller application to incorporate a "rush hour" signalling pattern. During high traffic periods, traffic flow is improved if the traffic is allowed to move through the intersection for longer periods of time. This increases the traffic throughput.

A rush-hour signalling pattern for R/A/G traffic signals is

duration	30 s	3 s	1 s	30 s	3 s	1 s		
N-S signal	R	R	R	G	A	R	...	repeat
E-W signal	G	A	R	R	R	R	...	repeat

Using SW1 depressed to denote "rush-hour" conditions, and SW1 not depressed to indicate normal traffic conditions. Transitions from "normal" to "rush-hour" conditions should be made logically. Be sure to explain and justify your design to the TA. SW3 continues to select the desired traffic signal to display on the LEDs.

Design an ESOS application `t2_rag2` to implement this traffic signal specification. You may use ESOS tasks and ESOS-provided software timers. Again, be sure to properly debounce any switches.

2.2.4 Traffic signal controller specification #3

Using the previous application as a starting point, create/extend your traffic signal controller application to incorporate "turn arrow" operations. If an intersection has a great deal of left-turning traffic, safety dictates that left-turn arrows be provided to allow these vehicles to safely move from one road to another in the presence of oncoming traffic.

A standard signalling pattern for turn signal R/A/G traffic signals is

duration	10 s	10 s	3 s	10 s	10 s	3 s		
N-S signal	R	R	R	T	G	A	...	repeat
E-W signal	T	G	A	R	R	R	...	repeat

A rush-hour signalling pattern for turn signal R/A/G traffic signals is

duration	10 s	30 s	3 s	1 s	10 s	30 s	3 s	1 s		
N-S signal	R	R	R	R	T	G	A	R	...	repeat
E-W signal	T	G	A	R	R	R	R	R	...	repeat

The letter "T" above indicates the turn arrow is illuminated. The hardware will indicate this condition by flashing the green LED rapidly (50% duty cycle, 0.25 s period). Use SW2 depressed to denote vehicles present in the left-turn lane. SW1 and SW3 continue the functionality described above.

Design an ESOS application `t2_rag3` to implement this traffic signal specification. You may use ESOS tasks, ESOS-provided software timers, and hardware interrupts. Again, be sure to properly debounce any switches.

2.3 Check Off

Demonstrate the following things to the TA:

- Using your hardware, demonstrate the proper operation of your three traffic signal controllers

2.4 Submission

Create an archive names `*netID*_t2.zip` or `*netID*_t2.tgz`., where ***netID*** is your MSU network login ID. In this archive, include the following items:

- Schematic errata document
- Component selection and justification spreadsheet
- Stuffing, soldering, and partial-build testing procedures
- well written, well commented, MCU code[5] for all three traffic signal specifications
- any unit test or test bench files used to verify your code created for this task
- updated design review forms and software metrics

Email this archive to the TA on or before the specified due date.

[5]Your MCU firmware must adhere to the *C* language coding standards.