CECS 346 Project 1 -- Traffic Light Controller

By Dr. Min He

Preparation

You will need a LaunchPad, 2 sets of red/green/yellow LEDs, 3 switches (Preferably slide switches or dip-switches), two LEDs for pedestrian lights: red and green, and resistors if needed.

Book Reading	Sections 6.4 to 6.9
Reference project	SimpleTrafficLight, SysTick
Starter project	Lab 4.

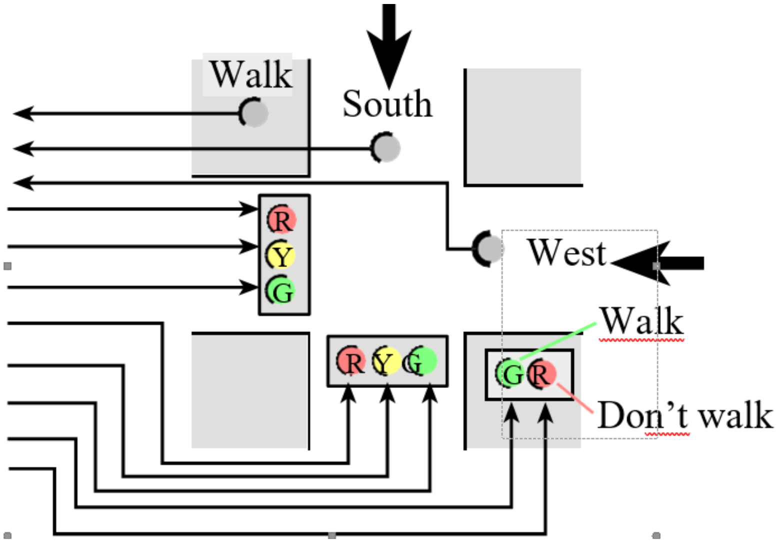
Purpose

The objectives of this project are: 1) the understanding and implementing of indexed data structures; 2) learning how to create a segmented software system; 3) the study of real-time synchronization by designing a finite state machine controller; 4) learn how to use a hardware timer. Software skills you will learn include defining data structure for FSM, building Moore FSM engine, creating fixed-time delays using the SysTick timer, and debugging real-time systems. Please read the entire project before starting.

System Requirements

Design a traffic light controller for the intersection of two equally busy one-way streets. The goal is to maximize traffic flow, minimize waiting time at a red light, and avoid accidents.

Consider a 4-corner intersection as shown in Figure 1. There are two one-way streets which are labeled South (cars travel toward South) and West (cars travel toward West). There are three inputs to your LaunchPad, two are car sensors, and one is a pedestrian button. The *South* car sensor will be true (3.3V) if one or more cars are near theintersection on the South Road. Similarly, the *West* car sensor will be true (3.3V) if one or more cars are near the intersection on the West Road. The *Walk* button will be true (3.3V) if at least one pedestrian is present, and he or she wishes to cross in any direction. If more than one pedestrian pressed the Walk button before traffic lights change, it should be treated as one press. Since the car sensors and pedestrian button need to be on until the FSM recognizes them, it is preferrable to use slide switches or dip-switches to simulate them. If you use push buttons, you will have to push and hold the push buttons until the FSM recognizes them. When a sensor is 0V, it means no cars/pedestrain are waiting to enter the intersection. You will interface 6 LEDs that represent the two Red-Yellow-Green traffic lights, and you will use the PF3 green LED for the "walk" lightand the PF1 red LED for the "hurry up" and "don't walk" light. The walk sequence should be showing three separate conditions: 1) "walk": green LED on, 2) "hurry up": redLED flash, and 3) "don't walk": red LED on. When the "walk" condition is signified, pedestrians are allowed to cross. When the "don't walk" light flashes (and the two trafficsignals are red), pedestrians should hurry up and finish crossing. When the "don't walk" light is on steady, pedestrians should not enter the intersection.



We would like to call the two streets plus the pedestrian three participants for the convenience of description. When none of the three participants need green light, stay in current state or finish transition to green. If one participant needs green, turn on the green for that participant and stay on as long as no other participant need green. If there are more than one participant need green: cycle through the requests servicing them in a round robin fashion, i.e., take turns to go through green-yellow-red for traffic light, walk- hurry-don't walk for pedestrian. In order to provide fair chances for each participant involved in the cycle, observe the following rules:

1. The only valid transition for traffic lights is green-yellow-red. No other transition is allowed.
2. The only valid transition for pedestrian lights is walk-hurry-don't walk. No other sequence is allowed.
3. If one participant starts a green-yellow-red transition, it needs to finish the whole transition and give green to one of the other two participants.
4. When two participants compete for the green light, make sure to give each one a chance. For example: when the current state is GoS, there can be two situations: 1. All three participants need green light. 2. West Street and pedestrian need green light. For both cases, the next green cannot be south street as south street just had green light. For both cases west street and pedestrian compete for green. So a fair solution is to assign green to one participant in one case and assign green to another participant for another case: case 1: GoW, case 2:GoP or vise verse.

The time durations for traffic lights are green/walk 2 seconds, yellow/hurry 1 seconds, red/don't walk 3 seconds. "Hurry up" uses a flashing LED that flashes on for 0.25 second and off for 0.25 second then repeat for a total of 1 second.

The system will start with green on south.

Implementation Requirements:

Implement a Moore machine: each state in the graph has a name, an output, a time to wait, and 8 next states (one for each input value). Do not embed functionality (e.g., flash 2 times) into the software that is not explicitly defined in the state graph. There can be no conditional branches in the FSM engine. This will simplify debugging and make the FSM engine trivial. You are required to use **SysTick timer** to generate time delay.

GPIO pin assignments for the inputs and outputs are given below:

The "don't walk" and "walk" lights must be PF1 and PF3 respectively, GPIO pin assignments for other inputs and outputs are given below.

Red west	PB5
Yellow west	PB4
Green west	PB3
Red south	PB2
Yellow south	PB1
Green south	PB0

South sensor	PE2
West sensor	PE1
Pedestrian Walk button	PE0

You are required to use the bit addresses to access traffic lights, pedestrian lights, sensors and pedestrian push buttons. Complete the following #define statements in your code:

```
#define P_LIGHTS        (*(volatile unsigned long *)__________)
#define T_LIGHTS        (*(volatile unsigned long *)__________)
#define SENSORS         (*(volatile unsigned long *)__________)
```

Implementation Steps:

1) Design a finite state machine that implements the required traffic light system. Draw a graphical picture of your finite state machine showing the various states, inputs, outputs, wait times and transitions.

2) Complete the following state table. It is based on the input order: South, West, Pedestrian.

Current	Time	Outputs		Inputs (South, West, Pedestrian) PE2, PE1, PE0							
State	(0.25s)	PB5-0	PF3,1	000	001	010	011	100	101	110	111
GoS	8	100001	01	GoS	WaitS	WaitS	WaitS	GoS	WaitS	WaitS	WaitS
WaitS				GoW	GoP	GoW	GoW	GoP	GoP	GoW	GoP
GoW											
WaitW											
GoP				GoP	GoP	WaitPOn1					
WaitPOn1				WaitPOff1							
WaitPOff1				WaitPOn2							
WaitPOn2				WaitPOff2							
WaitPOff2											

3) Write and debug the C code that implements the traffic light control system using Keil simulator, use the logic analyzer to visualize the input/output behavior of your system.

4) After testing your software on Keil simulator, you will build your traffic light control system circuit step by step. *Do not place or remove wires on the protoboard while the power is on*.

   a. The first step is to interface two switches for the two traffic sensors and one push button for pedestrian. You should implement positive logic switches. Write a simply main to test your switches. Example: initialize the GPIO ports used for the switches, each one of the three switches can be used to turn on/off one of the onboard LEDs.

   b. The next step is to build the six LED output circuits. You shouldimplement positive logic for the 6 LEDs. Align the LEDs in a shape that matches a traffic intersection. You will use the PF3&1 LED interfaces forthe pedestrian walk light (green for walk and red for don't walk). Write a simple main program to test the LED interfaces. Example: initialize the GPIO ports used for the LEDs, turn on all the LEDs for a period, then turn all of them off.

5) Debug the complete traffic light control system onboard.

Deliverable:

1. Demonstrate your system on LaunchPad: you need demonstration all cases, including just west, just south, just walk, two of the three, and all three.
2. Submit a video or a link to your video that records all the required behavior of your embedded system.
3. Submit a project report: follow the project report template for report format.
   1) Include the following design items in the "**Hardware Design**" section:
      A schematic and a picture of your embedded system.
   2) Include a video link in the "**Operation**" section.
   3) Include the following information in your report "**Software Design"** section:
      a. State table for your Moore FSM
      b. State diagram for your Moore FSM
      c. Software source code
   4) For "**Conclusion"** section: provide a brief description on your implementation experience, such as challenges, how did you solve it, what did you learn most in this project, etc.