School of Electrical Engineering and Computer Science

CptS466: Embedded Systems

Fall 2021

Project 3 (P3)

Code & Report Due: 10/08/2021 @ 11:59pm (Canvas)

Demo Due: 10/08/2021 in Class

1. Preparation

You will need a LaunchPad, three switches, four 10kΩ resistors, six LEDs, and six 470Ω resistors.

2. Project Description

The overall goal of this project is to build a prototype of the traffic light control system where we monitor two roads to control cars' and pedestrians' movements. Consider a 4-corner intersection of Main Street and Spring Street in Pullman. There are two streets labeled Main Street (cars travel west or east) and Spring Street (cars travel north or south). There are four inputs to your LaunchPad, two car sensors, and two pedestrian sensors. The South car sensor will be true (3.3V) if one or more cars are near the intersection on the Main Street. Similarly, the Spring Street car sensor will be true (3.3V) if one or more cars are near the intersection on the West road. The Walk sensor will be true (3.3V) if a pedestrian is present and wishes to cross in any direction. This walk sensor is different from a walk button on most real intersections. This means when you are testing the system, you must push and hold the walk sensors until the Finite State Machine (FSM) recognizes the presence of the pedestrian. Similarly, you will have to push and hold the car sensor until the FSM recognizes the presence of the car. In this simple system, if the walk sensor is +3.3V, there is a pedestrian to service, and if the walk sensor is 0V, there are no people who wish to walk. The walk sensors and walk light will service pedestrians who want to cross in any direction. This means the roads must both be red before the walk light is activated. In a similar fashion, when a car sensor is 0V, it means no cars are waiting to enter the intersection. The 'don't walk' light should be on while cars have a green or yellow light.

You will interface 6 LEDs representing the two Red-Yellow-Green traffic lights, and you will use the PF3 green LED for the "walk" light, and the PF1 red LED for the "don't walk" light. When the "walk" condition is validated, pedestrians are allowed to cross in any direction. When the "don't walk" light flashes (and the two traffic signals are red), pedestrians should hurry up and finish crossing. When the "don't walk" condition is on steady, pedestrians should not enter the intersection.

Derive an FSM (both in the form of "table" and "state diagram") that clearly describes the operation of the traffic lights. Since we only have 2 sets of LEDs for traffic lights you can also use the multicolor LED of the launchpad for pedestrian. We assign only one switch to pedestrians. This means there would be only one walk light for the entire intersection.

Just like the previous project, you will need to build circuits (using switches, LEDs, resistors, etc.) on the breadboard and connect them to the LaunchPad.
Here are some additional requirements.

- The traffic light for Main Street is initially 'Green' (i.e., the green signal/lED is ON while yellow and red signals are OFF) and walk light is 'Red'.

- Upon arrival of a pedestrian or a vehicle from Spring Street, the traffic light at Main Street turns ‘Yellow’ and after a delay it turns ‘Red’.
- Do not allow cars to crash into each other. This means there can never be a green or yellow on one road at the same time as a green or yellow on the other road. Engineers do not want people to get hurt.
- Do not allow pedestrians to walk while any cars are allowed to go. This means there can never be a green or yellow on either road at the same time as a “walk” light. Furthermore, there can never be a green or yellow on either road at the same time as the “don’t walk” light is flashing. If a green light is active on one of the roads, the “don’t walk” should be solid red. Engineers do not want people to get hurt.
- When the “walk” (green) condition is signified, pedestrians are allowed to cross in any direction. When the “don’t walk” light (red) flashes (and the two traffic signals are red), pedestrians should hurry up and finish crossing. When the “don’t walk” condition is on steady, pedestrians should not enter the intersection.
- If just the Main Street sensor is active (no walk and no Spring Street sensor), the lights should adjust so the Main Street has a green light within 10 seconds. The Main Street should stay green for as long as just the Main Street sensor is active.
- If just the Spring Street sensor is active (no walk and no Main Street sensor), the lights should adjust so the Spring Street has a green light within 5 seconds. The Spring Street light should stay green for as long as just the Spring Street sensor is active.
- If just the walk sensor is active (no Main Street and no Spring Street sensor), the lights should adjust so the “walk” light is green within 15 seconds. The “walk” light should stay green for as long as just the walk sensor is active.
- If all three sensors are active, the lights should go into a circular pattern in any order with the Spring Street light green, the Main Street light green, and the “walk” light is green. Of course, the road lights must sequence green-yellow-red each time.
- Multicolor LED on the launchpad uses only ‘Green’ and ‘Red’.

You are required to make any reasonable assumptions for your system to emulate real-life situations. The goal is not to provide you with all design decision choices and system details. Instead, you need to think (starting with high-level requirements discussed above) about details of the system, other requirements, etc. For example, you will need to choose reasonable delay while in Yellow state. Also, it is clear that if a car is passing the intersection, arrival of a second car will not change the already Green signal (pressing the switch for the second/third/… times will not change the green signal).

You need to follow guidelines in a system development process. Document your development phases (requirements, design, development, etc.) and write the software that satisfies the requirements for this lab.

Note: You will have to implement SysTick timer for this project.

2. What to submit?

Submit a .zip file with the following content:

- Your well commented C source code implementing the application described above.
- A report that documents the development process includes state machine, your observations, and a discussion of your design decisions.

Show a demo of your application in the class on the due date.

3. Report

In a separate written document (in Word or PDF), compile sections A through C as follows:

A: Requirements document. Treat this section as a requirements document and outline various requirements of the project in separate categories including (1) overview; (2) Function description; and (3) Deliverables. The overview section should contain, at least, information about objective, process, and interaction with existing systems. The function description section should contain, as a minimum, information about prototype, performance, and usability. The deliverables section should list or discuss any deliverables such as the built prototype, and report.

B: Design document. Draw a data flow graph showing schematic of the system and flow of the data from inputs to outputs. Your data flow graph should precisely shows the I/O pins, LEDs, and Switches, and the way they are used in conjunction with the microcontroller. Also, draw a flowchart showing the software design. The flowchart needs to include enough details about the hardware components used. Discuss the design of your FSM. Describe various aspects of FSM design such as inputs, outputs, states, stat transitions, etc. as discussed in the class.

C: Discussions. In this section discuss how you implemented delay in your program. Discuss any specific observations that you had, any difficulties that you ran into while designing or testing the system. Discuss

limitations of your developed system and potential ways that those limitations can be addressed in a future revision of your design.

4. Grading

Assume that the whole assignment is worth 100 points.

- 35 pts for well commented and correct C source code satisfying the project requirements.
- 35 pts for report.
- 30 pts for demo.

5. Appendix A – Example System Schematic

A potential schematic of a system for traffic light controller is shown below. This could be used as a basis for your data flow graph design. Note that the example discussed here does not necessary incorporates all requirements of the project that you will be building in this assignment. Use the information in this appendix as an example of how to begin designing and implementing a system of traffic light controller. The figure below shows a circuit diagram connecting the switch and LED components to the microcontroller.

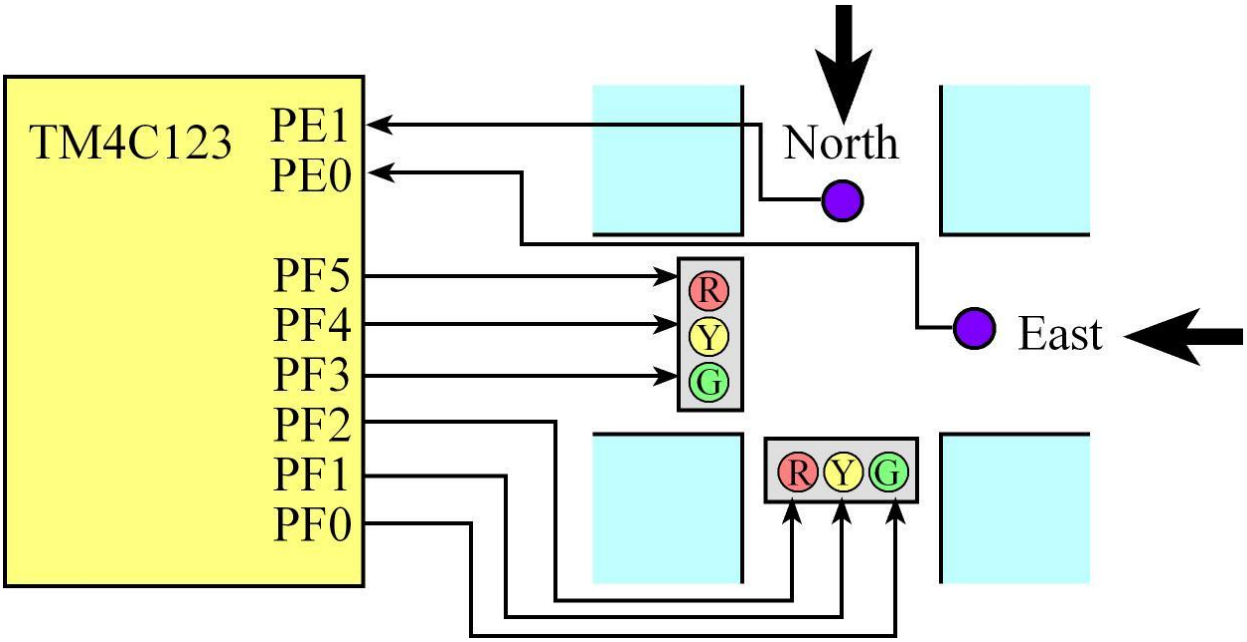
Here is an example of how to design a traffic light controller for the intersection of two equally busy one-way streets. The goal is to maximize traffic flow, minimize waiting time at a red light, and avoid accidents.

The intersection has two one-ways roads with the same amount of traffic: North and East, as shown in Figure 10.6. Controlling traffic is a good example because we all know what is supposed to happen at the intersection of two busy one-way streets. We begin the design by defining what constitutes a state. In this system, a state describes which road has the authority to cross the intersection. The basic idea, of course, is to prevent southbound cars to enter the intersection at the same time as westbound cars. In this system, the light pattern defines which road has the right of way over the other. Since an output pattern to the lights is necessary to remain in a state, we will solve this system with a Moore FSM. It will have two inputs (car sensors on North and East roads) and six outputs (one for each light in the traffic signal.) The six traffic lights are interfaced to Port B bits 5–0, and the two sensors are connected to Port E bits 1–0,

PE1=0, PE0=0 means no cars exist on either road
PE1=0, PE0=1 means there are cars on the East road
PE1=1, PE0=0 means there are cars on the North road
PE1=1, PE0=1 means there are cars on both roads

The next step in designing the FSM is to create some states. Again, the Moore implementation was chosen because the output pattern (which lights are on) defines which state we are in. Each state is given a symbolic name:

goN, PB5-0 = 100001 makes it green on North and red on East
waitN, PB5-0 = 100010 makes it yellow on North and red on East
goE, PB5-0 = 001100 makes it red on North and green on East
waitE, PB5-0 = 010100 makes it red on North and yellow on East

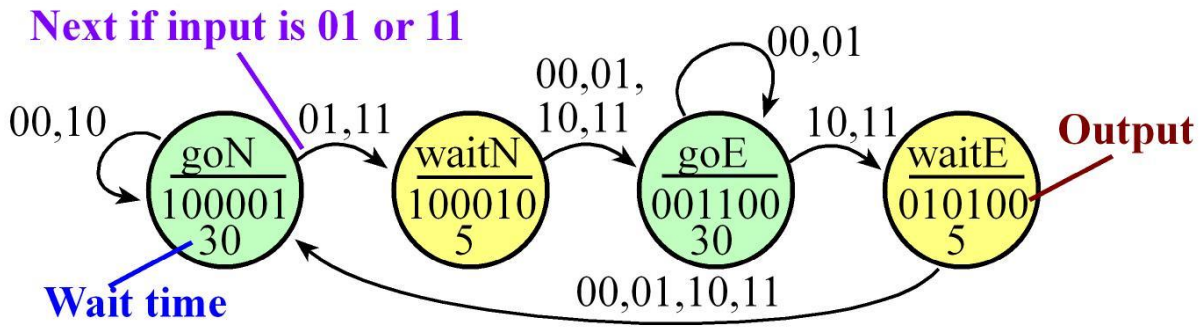


The output pattern for each state is drawn inside the state circle. The time to wait for each state is also included. How the machine operates will be dictated by the input-dependent state transitions. We create decision rules defining what to do for each possible input and each state. For this design, we can list heuristics describing how the traffic light is to operate:

- If no cars are coming, stay in a green state, but which one doesn't matter.
- To change from green to red, implement a yellow light of precisely 5 seconds.
- Green lights will last at least 30 seconds.
- If cars are only coming in one direction, move to and stay green in that direction.
- If cars are coming in both directions, cycle through all four states.
- Before we draw the state graph, we need to decide on the sequence of operations.

1. Initialize timer and direction registers
2. Specify initial state
3. Perform FSM controller
    a) Output to traffic lights, which depends on the state
    b) Delay, which depends on the state
    c) Input from sensors
    d) Change states, which depends on the state and the input

We implement the heuristics by defining the state transitions, as illustrated in figure below. Instead of using a graph to define the finite state machine, we could have used a table.



Num	Name	Lights	Time	In=0	In=1	In=2	In=3
0	goN	100001	30	goN	waitN	goN	waitN
1	waitN	100010	5	goE	goE	goE	goE
2	goE	001100	30	goE	goE	waitE	waitE
3	waitE	010100	5	goN	goN	goN	goN