**Laboratory 4: Finite State Machines….. or the tricky art of crossing the road at IIT main gate**

This assignment asks you to setup a traffic light control system for a junction of two busy roads. Most modern traffic light control systems, when not using a policeman with a whistle, are implemented using a finite state machine (FSM) design. We will use the Arduino to implement the FSM and use LED’s driven from the digital I/O pins to demonstrate the FSM’s operation.

**Procedure:**

The assignment is in two parts – complete part A first. The second part asks you to add a feature to your design. *Note that your grade for this assignment will depend on:*

1. A state transition diagram and transition table for the FSM (write these on the space provided on page 3 with this sheet).
2. *Neatly* written programs: you should write your code with functions corresponding to states.
3. The physical layout of the circuit on the breadboard (some push buttons and LED indications) should correspond to the diagrams indicated (*make it neat and pretty!)*

**Part A: Two-way junction traffic light**

* Think about how a real traffic light system works: both the highway and the crossroad have a set of lights – red, yellow, green on both sides of the road for two-way traffic. Use your understanding to draw the state diagram of an FSM for controlling the traffic lights.
* You can assume fixed delays between change of states – i.e. each light stays on for fixed times. Further, let us simplify the experimental setup by assuming that no traffic turns right or left – it just goes straight sticking to the lanes indicated in Fig 1.
* In the real world, the switching times are usually set based on time of day since that correlates to the expected amount of traffic. A smarter system has sensors mounted at the junction which monitor traffic volume and dynamically adjust the switching delays.
* Write a program to implement the FSM for the traffic intersection in Fig. 1 on your Arduino board and use LED’s connected to the digital I/O pins to demonstrate its operation.

**Note:** Plan your usage of digital I/O + analog I/O lines carefully considering the additional signals required in Part B. There are 14 digital I/O pins and 6 analog inputs. Pin 13’s use as a digital input is tricky because of the blink LED permanently connected to it on the Arduino board. The analog lines can be used for the standard analogRead(..) function, and also in pseudo-digital mode: analogWrite(pinX,0) writes 0V to pinX and analogWrite(pinY,255) writes 5V to pinY

highway

highway

crossroad

crossroad

R Y G

R Y G



R Y G

Fig 1: Simplified traffic control system

**Part B: Traffic light with pedestrians crossing**

Now add the following feature to your control system as shown in Fig 2:

* Provide a ‘Walk Request’ button for pedestrians wishing to cross the highway. One button on each side of the highway is sufficient. You can use the momentary push-button switches.
* Whenever a group of pedestrians needs to cross, they press the button. In the next traffic light change cycle, all traffic lights stay red for a pre-defined large time step (ideally determined by the volume of pedestrians, but here you can assume it to be a fixed parameter). A separate light provided for the pedestrian’s ‘walk signal’ is on during this time, allowing them to cross. Use a separate (white?) LED to indicate ‘Pedestrian – GO’ state.
* You can assume that pedestrians only wish to cross the highway, not the crossroad. So two ‘walk request’ buttons and the corresponding ‘pedestrian – GO’ signal are sufficient.
* The pedestrian walk request signal comes asynchronously – you need to detect and ‘record’ it. You will need a little bit of extra circuitry for this. The walk request button push arrives as a pulse which needs to be stored as a logic level. The conversion can be done using a D register IC (see page 4 for notes on usage of a D register). Note that the signal must be ‘cleared’ after the request has been served.

highway

highway

crossroad

crossroad

R Y G

R Y G

R Y G

Fig 2: pedestrians matter too!

Ped. Walk request

Ped. Walk request

Ped. GO

Ped. GO

R

Y

G

Note on Human Police controlling traffic at junctions:

* As you will see on examining your FSM diagram the police-person is following the state transitions, so he/she is also implementing an FSM, except their state transitions are subject to human error.
* So far we have studied a specific type of FSM: a deterministic FSM. In these FSM’s, the transition from one state to another is always definite and deterministic based on preset rules.
* A second type of FSM is a ‘non-deterministic FSM’ where the transitions between states happen with some probabilistic variation.
* The traffic police-person is an example of a non-deterministic FSM, since he/she is liable to make mistakes. Many modern computer games like first-person-shooters (Quake and Doom) use an FSM as the core algorithm for their characters. The better ones implement a non-deterministic FSM to make the characters behave in a more human manner.

**FSM State Diagram Part A:**

 A complete example diagram with state transition table is given below. This state diagram is for control of traffic light at a highway intersection (no cross-road) and a pedestrian walk request.



**(2 marks for part A; 3 marks for Part B)**

some simplifications of the state transitions can be made based on the specifications of the problem statement here. Full marks can only be given if this level of detail is presented in the state diagram and state transition table.

**FSM State Diagram Part B: (with circuitry for pedestrian walk request)**

**Notes on usage of a D register IC (74LS175)**



One 74LS175 IC contains 4 D registers (pinout diagram given above).

1. Effectively, at the rising edge of every clock cycle, the D register records the value of the input D to the output Q. Apart from the rising clock edges the register is insensitive to the value of D.

 /4

(Part A)

Code+Ckt

/15

 /2

(Part A)

State Diagram

 /6

(Part B)

Code+Ckt

 /3

(Part B)

State Diagram

1. The stored value in the register can be cleared by supplying an ‘active-low’ CLR signal on pin 1. The overbar on the CLR signal in the pinout diagram indicates that it is an ‘active low’ signal. Normally, the input to this pin must be kept in logic 1. When you want clear the contents of the register, you pull the value of this pin 1 to logic 0.