**Traffic Light**

##### **Name** : Laurentiu Fratila

##### **University**: Lucian Blaga University of Sibiu, Faculty of Engineering

##### **Main of field study**: Computer Engineering and Information Technology

##### **Specialization**: Computer Science and Computer Engineering

## Introduction

For this project I will use a Texas Instruments TM4C123GXL LaunchPad, three 10kΩ resistors, six LEDs and six 470Ω resistors and three switches.

## Technologies used

For software developing I used Keil uVision4 developed by armKeil.

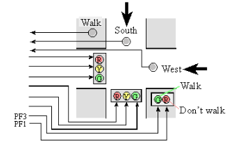
https://www.keil.com/download/

## Purpose

This starter project has the objective of learning of real-time synchronization by designing a finite state machine controller.

## Objectives

Consider a 4 corner intersection:



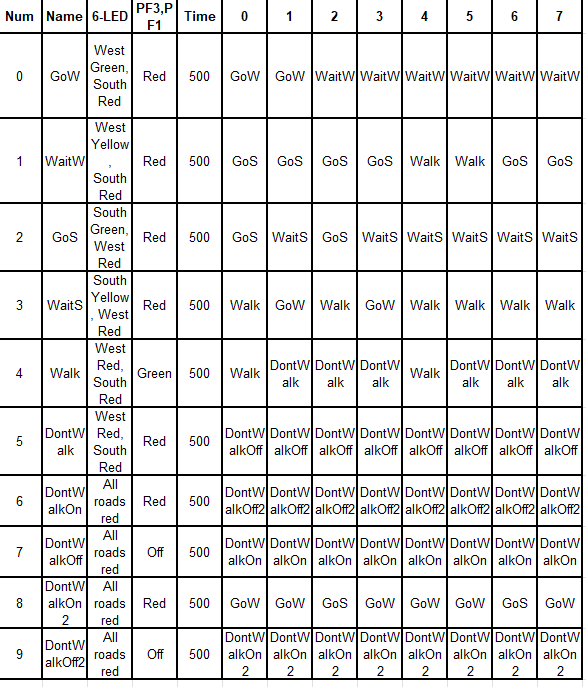
There are two one-way streets are labeled South (cars travel South) and West (cars travel West). There are three inputs to your LaunchPad, two are car sensors, and one is a pedestrian sensor. The South car sensor will be true (3.3V) if one or more cars are near the intersection on the South road. Similarly, the West car sensor will be true (3.3V) if one or more cars are near the intersection on the West road. The Walk sensor will be true (3.3V) if a pedestrian is present and he or she 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 sensor until the FSM recognizes the presence of the pedestrian.

In this simple system, if the walk sensor is +3.3V, there is pedestrian to service, and if the walk sensor is 0V, there are no people who wish to walk. The walk sensor and walk light will service pedestrians who wish 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.

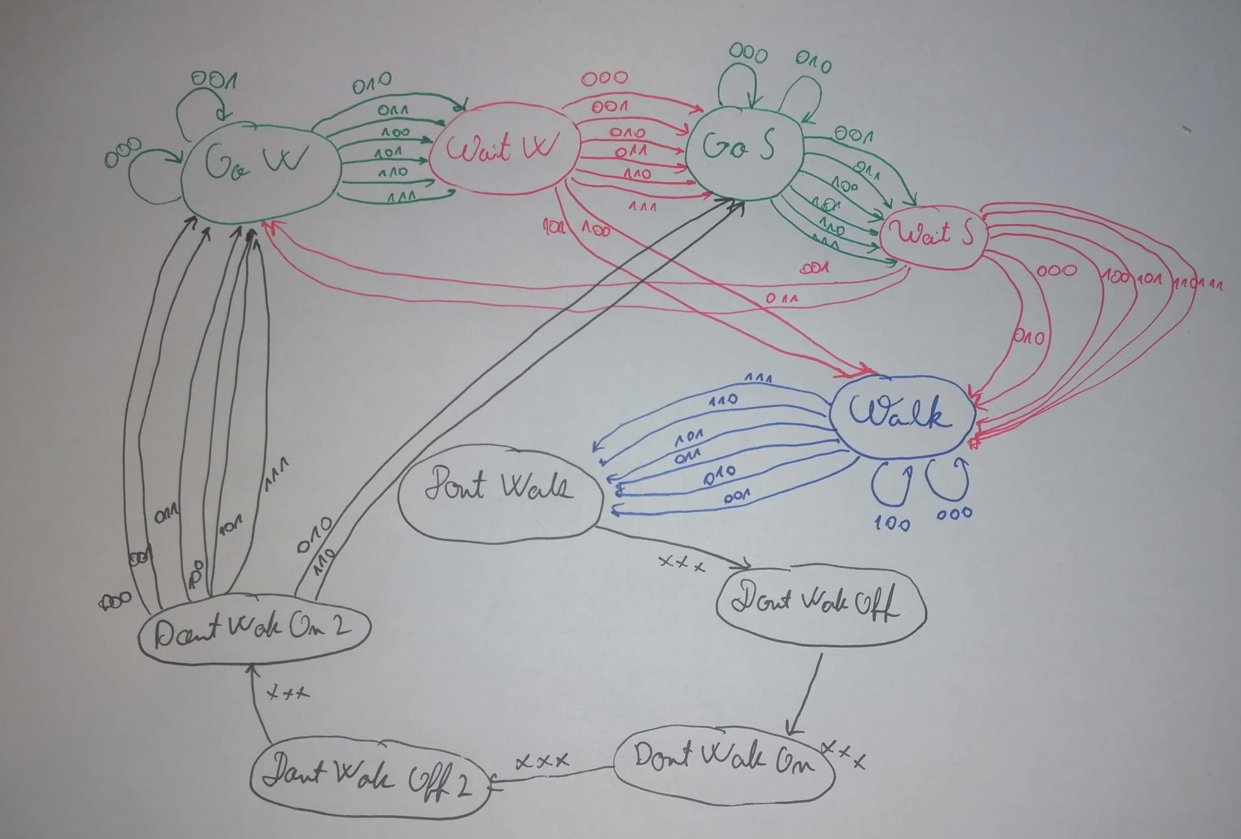
Traffic should not be allowed to crash. There should not be a green or yellow on one road at the same time there is a green or  yellow LED on the other road. In other words, while traffic is flowing in one direction, there should be a red light in the other direction.

## Functionality

Building a state transition table. A state transition table has exactly the same information as the state transition graph, but in tabular form. Let's begin with the format of the table. There will be a column for the state number; we will number the states sequentially from 0. The next two columns will define the output patterns for six traffic lights and two walk lights. The next column is the time to wait with this output. The last eight columns will be the next states for each possible input pattern. Recall the FSM controller will 1) output, 2) wait, 3)  input, and 4) change to the next state depending on the current state and the input.



State Transition Graph (hand written)



With the inputs:

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | |  |
| 0 | 0 | 0 | No one |
| 0 | 0 | 1 | Cars West |
| 0 | 1 | 0 | Cars South |
| 0 | 1 | 1 | Cars W and S |
| 1 | 0 | 0 | Pedestrian |
| 1 | 0 | 1 | Pedestrian and Car W |
| 1 | 1 | 0 | Pedestrian and Car S |
| 1 | 1 | 1 | Pedestrian, Car W, and Car S |

For detecting the direction from where cars come in intersection I have used 3 switches which have the role of a sensor. SW1(connected to pin PE0) will take action for the cars which travel West, SW2(connected to pin PE1) will take action for the cars which travel South, and SW3 will take action for Pedestrians and outputs on the launchpad integrated LED. 