California State University

EGCP 450

Spring 2023

Lab 3

Professor Michael Turi

Zaid frayeh

INTRODUCTION:

This experiment I have been advised to make a traffic light using hardware components and software, my design was based on a Finite State Machine that would switch through states and flip different LEDs on and off.

Procedure/Discussion:

First part of this lab was to make a traffic light from 2 sides west and south.
I first started by building the hardware components used :

a. Red Led x2
b. Green Led x2
c. Yellow Led x2
d. 450 ohm resistors x6
e. 10 k ohm resistors x2
f. Switches x2

I have connected the LEDs to the 450 ohm resistors to make a pull down resistors since we are making a positive logic output
The switches are also connected to the 10 k ohm resistors.
By designing the hardware in the beginning I was able to choose the ports on the MSP432 .
Port 4 was used to Output data to the LEDs the first 6 pins were used .
Port 5 was used to Input data from the switches to the msp432 pins 5.1 and 5.2 were used for the switches the switches act as sensors for the cars.
Port 1 was used to Input data from the pedestrian light the msp432 uses negative logic so I coded my port to have a pull up resistor.
Lastly port 2 was used to output the light for the pedestrian to walk or not to walk.
After figuring out which ports and pins to use everything else was a little bit more straight forward.
I have started part 1 of the lab to design the 2 way traffic light by building the finite state machine diagram and state transition diagram the diagram had 2 inputs from the switches and 6 outputs to the LEDs the design had 4 transitions which are shown below:

South Traffic Lights

P 4.0 G
P 4.1 y] output
P 4.2 R
P 5.2 Button Input

40 - 60

west Traffic Lights

P.4.3 G
P.4.4 y] output
P.4.5 R
P.5.1 Button Input

1 - 30

Input W	S	out
0	0	X
0	1	W
1	0	N
1	1	T

Bits	7	6	5	4	3	2	1	0
Go S			1	0	0	0	0	1
Wait S			1	0	0	0	1	0
Go W			0	0	1	1	0	0
Wait W			0	1	0	1	0	0

yellow S
Red W

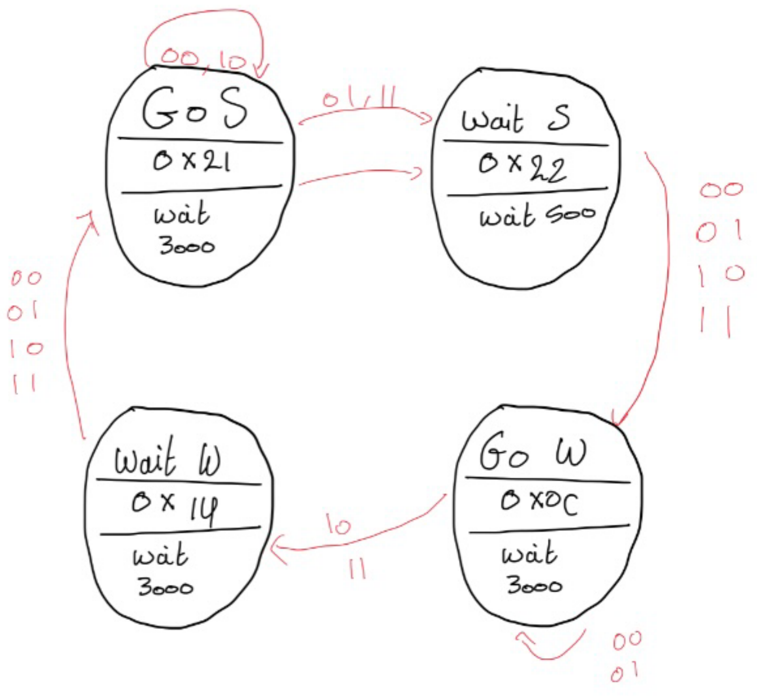
1 4

2 2

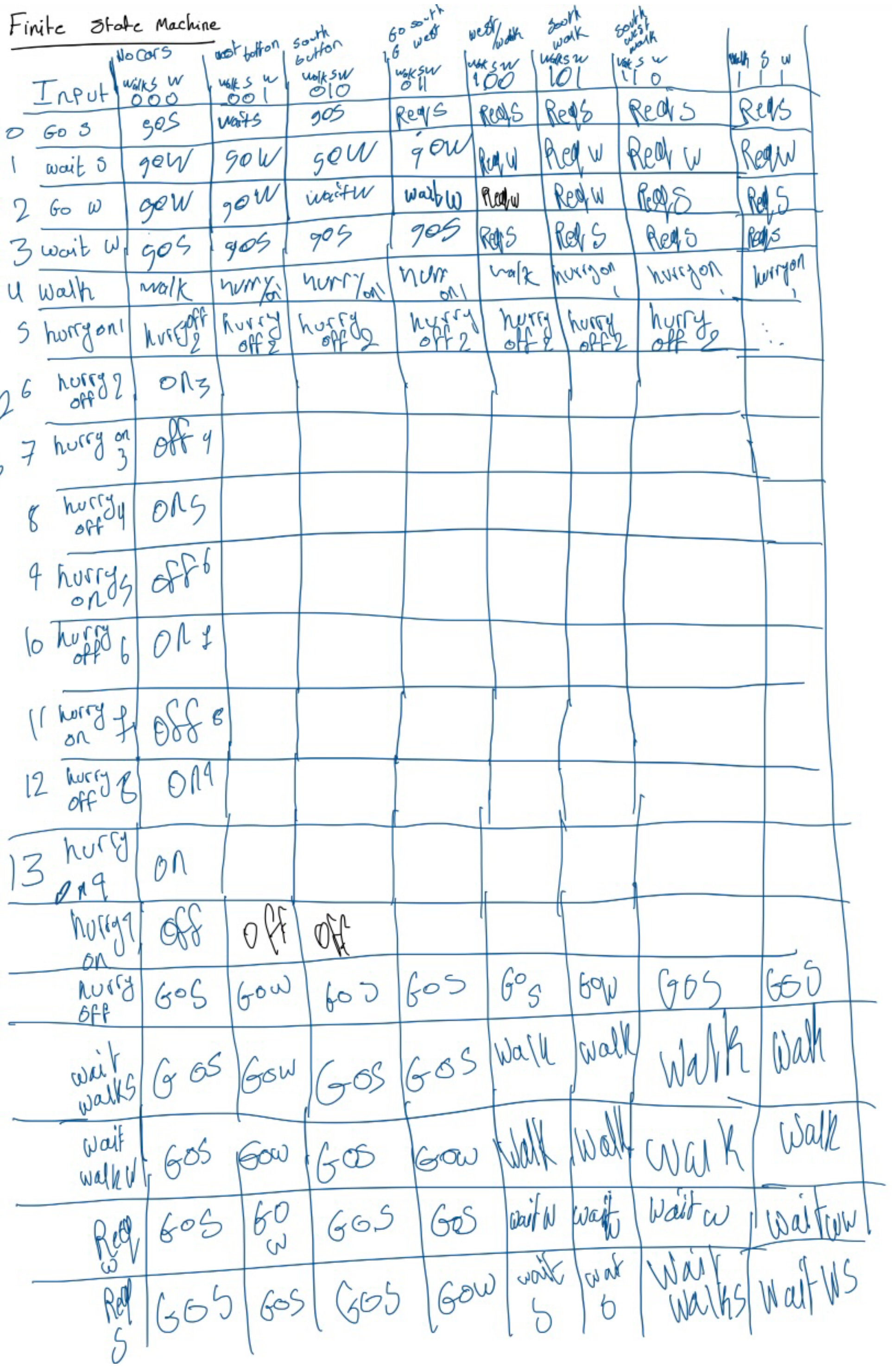
Finite State Machine

Input	No cars 00	west 01	South 10	Both 11
Go S	Go S	wait S	Go S	wait S
wait S	Go W	Go W	Go W	G W
Go W	Go W	Go W	Wait W	wait W
wait W	Go S	Go S	Go S	Go S

Table



After this part was designed and tested, I started designing part 2 of the lab which was basically add-ons from the first part of the lab.
I started with a new finite state machine diagram which had the same inputs and outputs as before but in addition to the pedestrian traffic light which had 3 inputs and 8 outputs this design is shown below:



Finite State Machine

	Input	No cars walk S W 000	west button walk S W 001	south button walk S W 010	Go south & Go west walk S W 011	west walk walk S W 100	south walk walk S W 101	south west walk walk S W 110	walk S W 1 1 1
0	Go S	goS	waitS	goS	RedS	RedS	RedS	RedS	RedS
1	wait S	goW	goW	goW	goW	RedW	RedW	RedW	RedW
2	Go W	goW	goW	waitW	waitW	RedW	RedW	RedS	RedS
3	wait W	goS	goS	goS	goS	RedS	RedS	RedS	RedS
4	walk	walk	hurry on1	hurry on1	hurry on1	walk	hurry on1	hurry on1	hurry on1
5	hurry on1	hurry off 2	hurry off 2	hurry off 2	hurry off 2	hurry off 2	hurry off 2	hurry off 2	
6	hurry off 2	on 3							
7	hurry on 3	off 4							
8	hurry off 4	on 5							
9	hurry on 5	off 6							
10	hurry off 6	on 7							
11	hurry on 7	off 8							
12	hurry off 8	on 9							
13	hurry on 9	on							
	hurry 9 on	off	off	off					
	hurry off	GoS	GoW	GoS	GoS	GoS	GoW	GoS	GoS
	wait walk S	GoS	GoW	GoS	GoS	walk	walk	walk	walk
	wait walk W	GoS	GoW	GoS	GoW	walk	walk	walk	walk
	Red W	GoS	GoW	GoS	GoS	wait W	wait W	wait W	wait W
	Red S	GoS	GoS	GoS	GoW	wait S	wait S	wait walk S	wait WS

w $\overset{s}{\text{Port}}$ 2

Bits	7 6 5 4 3 2 1 0 (B G R)	Port 4	Port 2 (G R)
Go S	1 0 0 0 0 1	21,	0 1
Wait S	1 0 0 0 1 0	22	0 1
Go W	0 0 1 1 0 0	0C	0 1
Wait W	0 1 0 1 0 0	14,	0 1
walk	0 0 1 0 0 1 0 0	24,	1 0
Hon1		24,	0 1
Hoff2		24,	0 0
Hon3		24,	0 1
Hoff4		24,	0 0
Hon5		24,	0 1
Hoff6		24,	0 0
Hon7		24,	0 1
Hoff8		24,	0 0
Hon9		24,	0 1
Hoff		24,	0 0
WWW		14,	0 1
WWS		22,	0 1
REW		0C,	0 1
RES		21,	0 1

Port 4 Port 2

As we can see the first table provides how my design works the rows show the state that the system is in and shows what states it transitions to the column shows the inputs that our system has.
The second table shows the outputs that are given to the LED.

Code:

```
#include <stdint.h>
#include "SysTick.h"
#include "msp432p401r.h"
struct State {
    uint32_t Out;
    uint32_t Outw;
    uint32_t Time; // 10 ms units
    const struct State *Next[8];
             };
typedef const struct State styp;
#define GOS     &FSM[0]
#define WAITS   &FSM[1]
#define GOW     &FSM[2]
#define WAITW   &FSM[3]
#define WALK    &FSM[4]
#define HON1    &FSM[5]
#define HOFF2   &FSM[6]
#define HON3    &FSM[7]
#define HOFF4   &FSM[8]
#define HON5    &FSM[9]
#define HOFF6   &FSM[10]
#define HON7    &FSM[11]
#define HOFF8   &FSM[12]
#define HON9    &FSM[13]
#define HON     &FSM[14]
#define HOFF    &FSM[15]
#define WAITWW  &FSM[16]
#define WAITWS  &FSM[17]
#define REQW    &FSM[18]
#define REQS    &FSM[19]

styp FSM[20] = {
 {0x21,0x01,100,{GOS,WAITS,GOS,WAITS,REQS,REQS,REQS,REQS}},  //1 gos
 {0x22,0x01,100,{GOW,GOW,GOW,GOW,REQS,REQS,REQS,REQS}},      //2 waitS
 {0x0C,0x01,100,{GOW,GOW,WAITW,WAITW,REQW,REQW,REQW,REQW}},  //3 goW
 {0x14,0x01,100,{GOS,GOS,GOS,GOS,REQW,REQW,REQW,REQW}},      //4 waitW
 {0x24,0x02,100,{WALK,HON1,HON1,HON1,WALK,HON1,HON1,HON1}},   //WALK
 {0x24,0x01,100,{HOFF2,HOFF2,HOFF2,HOFF2,HOFF2,HOFF2,HOFF2,HOFF2}},
//HURRYON1
 {0x24,0x00,100,{HON3,HON3,HON3,HON3,HON3,HON3,HON3,HON3}},
//HURRYOFF2
```

```
 {0x24,0x01,100,{HOFF4,HOFF4,HOFF4,HOFF4,HOFF4,HOFF4,HOFF4,HOFF4}},
//HURRYON3
 {0x24,0x00,100,{HON5,HON5,HON5,HON5,HON5,HON5,HON5,HON5}},
//HURRYOFF4
 {0x24,0x01,50,{HOFF6,HOFF6,HOFF6,HOFF6,HOFF6,HOFF6,HOFF6,HOFF6}},
//HURRYON5
 {0x24,0x00,50,{HON7,HON7,HON7,HON7,HON7,HON7,HON7,HON7}},
//HURRYOFF6
 {0x24,0x01,50,{HOFF8,HOFF8,HOFF8,HOFF8,HOFF8,HOFF8,HOFF8,HOFF8}},
//HURRYON7
 {0x24,0x00,50,{HON9,HON9,HON9,HON9,HON9,HON9,HON9,HON9}},   //HURRYOFF8
 {0x24,0x01,50,{HOFF,HOFF,HOFF,HOFF,HOFF,HOFF,HOFF,HOFF}},    //HURRYON9
 {0x24,0x00,50,{HON,HON,HON,HON,HON,HON,HON,HON}},            //HURRYON
 {0x24,0x01,100,{GOS,GOW,GOS,GOS,GOS,GOW,GOS,GOS}},          //HURRYOFF
 {0x14,0x01,100,{GOS,GOW,GOS,GOS,WALK,WALK,WALK,WALK}},      //WAITWALKS
 {0x22,0x01,100,{GOS,GOW,GOS,GOW,WALK,WALK,WALK,WALK}},     //WAITWALKW
 {0x0C,0x01,100,{GOS,GOW,GOS,GOS,WAITWW,WAITWW,WAITWW,WAITWW}},    //REQW
 {0x21,0x01,100,{GOS,GOS,GOS,GOW,WAITWS,WAITWS,WAITWS,WAITWS}}     //REQS
             };

void ports(void){

   //output pins
        P4SEL0 &= ~0x3F;
        P4SEL1 &= ~0x3F;
        P4DIR  |= 0x3F;
   // output led
        P2SEL0 &= ~0x03;
        P2SEL1 &= ~0x03;
        P2DS   |= 0x03;
        P2DIR  |= 0x03;
        P2OUT  |= 0x03;
   //input buttons
        P1SEL0 = 0x00;
        P1SEL1 = 0x00;
        P1DS   = 0x00;
        P1DIR  = 0x00;
        P1REN  = 0X02;
        P1OUT  = 0x02;
   //input buttons
        P5SEL0 &= ~0x06;
        P5SEL1 &= ~0x06;
        P5DIR  &= ~0x06;
}
void main(void){
    uint32_t Input;
    uint32_t Input2;
    styp *Pt;
    SysTick_Init();
    ports();
    Pt = GOS; // start state
while(1){
            P4OUT=(Pt -> Out);
            P2OUT=((Pt -> Outw)&0x03);
            SysTick_Wait10ms(Pt->Time);
             Input = P5IN&06;
            Input = Input/2;
            Input2 = ((~P1IN&0x02) <<1 );
            Pt = Pt->Next[Input2 + Input];
         }
              }
```

The first part of the code shows the defining of our states, the second part shows the transitions from state to state, the third part shows the initialization of the ports, lastly we see the main part of our code where we get the inputs and according to the we get the output from the Finite State machine, we also used the Systick timer to provide delay to the system.

In my Finite State Machine design there are 20 states there are 4 main states for the two traffic lights which I started my first program with after that I added 2 requests for the states if a pedestrian wants to request to walk and after that another 2 states which waits for the button to still be pressed which provides the 2 second delay for the press, the other states are basically walk for the pedestrians as well as hurry for the pedestrians which turns on and off the light and speeds up at the end.
This makes up a design of 20 states, in my design I used pointers and a linked data structure to achieve my finite state machine.

The only problem that I had while designing this system was with port 5 the first pin wasn't reading my data correctly, so I changed the pins and divided the value by 2 to make it appropriate for reading.

After designing the first part of the lab the second part was only a matter of designing the transition diagram and figuring out the rest of the input and output for the pedestrian, the most time-consuming part of this lab was doing the table since it was a little bit long but other than that everything was pretty straight forward.

After finishing all these parts my system was tested with all of the possible inputs, my system transitions between all of the states with no problem it transitions correctly according to the diagram I have provided.

Improvements:

Since I have started with the design on my notes and every input, output and code was translated from visual diagrams my code worked correctly and no improvements were required.

The only part of the lab that was required to be altered with is the delay from each state to state to make it appropriate for the demonstration video.

Analysis:

My design of the Finite State Machine used a moore machine since our output depends on the current state only.
If I were to use a mealy machine we would've seen a direct connection between the inputs and the output.
The data that should be collected to make sure my system works correctly is first of all the time delay that was assigned to each state and make sure the inputs are read correctly as well as the outputs if everything matches what the diagram has then this proves all of my system is working correctly.
My Finite state machine has 20 states which are discussed earlier in the report.
Fewer states may be used to but I believe I wouldn't have achieved the time delay that I wanted for the walk and hurry states
What data to check :
My design works correctly and for demonstration purposes I have reduced the delay time to demonstrate the effectiveness of my design.
My design right now the signals change if both sensors are active change every 1 second.
If one car per second arrives at each intersection, I would have an average of 2 car lined up at each intersection and again this is for demonstration purposes.
If 2 cars arrive per second, then I would have an average of 4 cars lined up at each intersection.
Crossing the street would take an average of 2 seconds and again this is for demonstration purposes.

Conclusion:

Lab 3 really helped me understand how Finite State Machine works in a real time embedded system the difference between mealy and moore machines. Timing and delay in such situations are important to be taken into consideration in these type of systems for they assure the safety for the users of the road, the safety of these users rely a lot on the timing and delay of such systems and off course the working condition of these systems.
If I were to do this experiment, I wouldn't change anything since I had everything right from the beginning no glitches or bugs were found in the system from my side of the development.

Reference:

material provided by the professor.
Students that helped me with understanding how some ports needed to be coded
David Mouser
Moniel Flores