

# **Lab 3: Mini Traffic Light System Report**

Tyler Nardecchia  
Yerania Hernandez  
CSCE 462 - 502  
October 1st, 2017

## Thinking and Exploring

- 1) *Share your experience on differences between a simulator and the actual implementation in 75-100 words.*

When we simulated our stoplight finite state machine, at every state iteration we would print the value of the current state as well as the value of each of the 6 LED pins (3 for each light). This helped indicate that the software logic of our finite state machine worked correctly, and we now just needed to implement the hardware. The difference of the actual implementation was that now we had to use functions such as `pinMode` and `digitalWrite` in order to communicate with the GPIO pins, and we also had to use `digitalRead` in order to wait for the button to be pressed before going through the pedestrian crossing process.

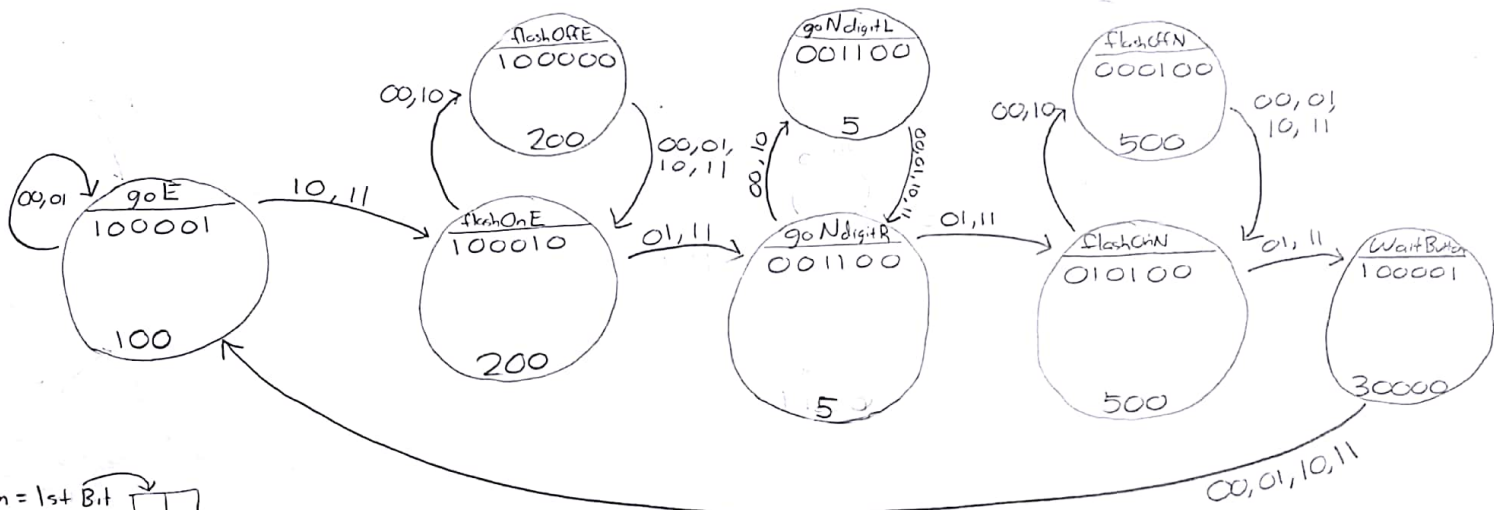
- 2) *Can we possibly have street-crossings for cars without traffic-signals like what we have now in the future? If yes, how can we implement that?*


Yes, it could be possible to have street-crossings for cars without traffic-signals, but we believe this can only be done with self-driving cars in the road. Safety hazards will be high if these traffic signals were simply removed and people were responsible to monitor the crossing of the road. This experiment has been conducted, named “Naked Streets”, where street lights are removed from certain roads and it forces the driver to slow down and carefully proceed considering there are no rules or direction about what could happen next. However, this is only done in low-traffic streets and definitely brings a safety hazard for pedestrians trying to cross roads, especially since you are simply trusting that the driver will do what is common sense and courteous. With autonomous driving cars though, this issue could be resolved and avoid the human hazard that could occur. As a matter of fact, MIT has conducted research in this field and named it the “slot-based system”. Similar to air control, vehicles will be passing through roads on a slot availability base, meaning that they will only be able to take that specific intersection when the slot is ready. As a result, it adjusts the speed of every vehicle that enters an intersection and allows them to pass one another at the precise time that the slot is available. It might take awhile before we reach this state considering there is a mix of human-driven cars and autonomous-driven cars in the road now, but if it ever becomes possible that there only be autonomous vehicles on roads, then the advantage of such a system could bring a new form of traffic flow along with eliminating risk for pedestrians.

- 3) *Consider you want to have similar waiting countdown display arrangement across the zebra crossing. How is your design going to change?*

If we were to develop another pedestrian light across the zebra crossing, we would have to implement another button, LED, and dual 14-segment display. Since the two pedestrian lights and counters would always be displaying the same thing, we could use

the same wires as that of the original light to power the second light, which would conserve the use of additional GPIO's. However, since the button of the original light should be independent from that of the second light, an additional input GPIO would need to be used to consider the second button. Whenever the cars have the green signal (and it's been 30 seconds since the last pedestrian crossing), the input for both buttons would need to be constantly checked, and if at least one of them is being pressed at any given time, you turn the car stoplight to blue (yellow in real-world terms) and disregard any other button pressings for the length of the pedestrian crossing process. Overall, this implementation would not be too demanding.



Button = 1st Bit   
 Counter = 2nd Bit  
 Wait Time (ms)

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <wiringPi.h>
4  #include <stdint.h>
5  #include <unistd.h>
6
7  #define INPUT 0
8  #define OUTPUT 1
9
10 #define goE      0
11 #define flashOnE 1
12 #define flashOffE 2
13 #define goN      3
14 #define flashOnN 4
15 #define flashOffN 5
16 #define waitButton 6
17
18 int buttonValue = 0;
19 uint32_t counter = 0;
20
21 typedef struct State
22 {
23     uint32_t out[6];
24     uint32_t time;
25     uint32_t next[4];
26 } State;
27
28 State FSM[7] = {
29     {{1, 0, 0, 0, 0, 1}, 100, {goE, goE, flashOnE, flashOnE}},
30     {{1, 0, 0, 0, 1, 0}, 200, {flashOffE, goN, flashOffE, goN}},
31     {{1, 0, 0, 0, 0, 0}, 200, {flashOnE, flashOnE, flashOnE, flashOnE}},
32     {{0, 0, 1, 1, 0, 0}, 1000, {goN, flashOnN, goN, flashOnN}},
33     {{0, 1, 0, 1, 0, 0}, 500, {flashOffN, waitButton, flashOffN, waitButton}},
34     {{0, 0, 0, 1, 0, 0}, 500, {flashOnN, flashOnN, flashOnN, flashOnN}},
35     {{1, 0, 0, 0, 0, 1}, 30000, {goE, goE, goE, goE}},
36 };
37
38 void clockWrite(int time){
39     printf("Display: %d\n\n", time);
40 }
41
42 int main()
43 {
44     int currState;
45     uint32_t inputs = 0;
46
47     sleep(1);
48     currState = goE;
49     while (1)
50     {
51         printf("Current state: ");
52         if (currState == 0)
53         {
54             printf("Go east\n");
55         }
56         else if (currState == 1)
57         {
58             printf("Flash on east\n");
59         }
60         else if (currState == 2)
61         {
62             printf("Flash off east\n");
63         }
64         else if (currState == 3)
65         {
66             printf("Go north\n");

```

```

67     }
68     else if (currState == 4)
69     {
70         printf("Flash on north\n");
71     }
72     else if (currState == 5)
73     {
74         printf("Flash off north\n");
75     }
76     else if (currState == 6)
77     {
78         printf("Wait for button\n");
79     }
80
81     printf("\nTraffic Light Red: %d\n", FSM[currState].out[0]);
82     printf("Traffic Light Blue: %d\n", FSM[currState].out[1]);
83     printf("Traffic Light Green: %d\n", FSM[currState].out[2]);
84     printf("Pedestrian Light Red: %d\n", FSM[currState].out[3]);
85     printf("Pedestrian Light Blue: %d\n", FSM[currState].out[4]);
86     printf("Pedestrian Light Green: %d\n\n", FSM[currState].out[5]);
87     printf("Counter: %d\n\n", counter);
88
89     if (currState == goN)
90     {
91         clockWrite(20 - counter);
92         counter++;
93     }
94     if (currState == flashOnN)
95     {
96         clockWrite(10 - counter);
97         counter++;
98     }
99
100    sleep(FSM[currState].time/1000);
101
102    if (currState == flashOnE)
103    {
104        counter++;
105    }
106
107    inputs = 0;
108    if (currState == goE) // Simulated button press
109    {
110        buttonValue = 1;
111    }
112    if (buttonValue)
113    {
114        inputs += 2;
115    }
116    if (currState == goE) // Simulated button release
117    {
118        buttonValue = 0;
119    }
120
121    if (counter == 10)
122    {
123        inputs += 1;
124        counter = 0;
125    }
126
127    printf("Inputs: %d\n\n", inputs);
128    printf("-----\n\n");
129    currState = FSM[currState].next[inputs];
130 }
131 }

```

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <wiringPi.h>
4  #include <stdint.h>
5
6  #define INPUT 0
7  #define OUTPUT 1
8
9  #define goE      0
10 #define flashOnE 1
11 #define flashOffE 2
12 #define goNdigitR 3
13 #define goNdigitL 4
14 #define flashOnN 5
15 #define flashOffN 6
16 #define waitButton 7
17
18 double counter = 0.00; // Used for flashing the LED's and segment display the
    appropriate amount of times
19
20 typedef struct State
21 {
22     uint32_t out[6];
23     uint32_t time;
24     uint32_t next[4];
25 } State;
26
27 State FSM[8] = {
28     {{1, 0, 0, 0, 0, 1}}, 100, {goE, goE, flashOnE, flashOnE}},
29     {{1, 0, 0, 0, 1, 0}}, 200, {flashOffE, goNdigitR, flashOffE, goNdigitR}},
30     {{1, 0, 0, 0, 0, 0}}, 200, {flashOnE, flashOnE, flashOnE, flashOnE}},
31     {{0, 0, 1, 1, 0, 0}}, 5, {goNdigitL, flashOnN, goNdigitL, flashOnN}},
32     {{0, 0, 1, 1, 0, 0}}, 5, {goNdigitR, goNdigitR, goNdigitR, goNdigitR}},
33     {{0, 1, 0, 1, 0, 0}}, 500, {flashOffN, waitButton, flashOffN, waitButton}},
34     {{0, 0, 0, 1, 0, 0}}, 500, {flashOnN, flashOnN, flashOnN, flashOnN}},
35     {{1, 0, 0, 0, 0, 1}}, 30000, {goE, goE, goE, goE}},
36 };
37
38 void rightDigit(double time) // Prints the right digit on the dual 14-segment display
39 {
40     digitalWrite(40, 1); // Adjust the ground pins appropriately
41     digitalWrite(12, 0);
42
43     if (time >= 9 && time < 10) // Print the number
44     {
45         digitalWrite(35, 1);
46         digitalWrite(37, 1);
47         digitalWrite(38, 1);
48         digitalWrite(32, 1);
49         digitalWrite(22, 1);
50         digitalWrite(7, 0);
51         digitalWrite(36, 1);
52         digitalWrite(18, 1);
53     }
54     else if (time >= 8 && time < 9)
55     {
56         digitalWrite(35, 1);
57         digitalWrite(37, 1);
58         digitalWrite(38, 1);
59         digitalWrite(32, 1);
60         digitalWrite(22, 1);
61         digitalWrite(7, 1);
62         digitalWrite(36, 1);
63         digitalWrite(18, 1);
64     }
65     else if (time >= 7 && time < 8)

```

```
66 {
67     digitalWrite(35, 0);
68     digitalWrite(37, 1);
69     digitalWrite(38, 1);
70     digitalWrite(32, 1);
71     digitalWrite(22, 0);
72     digitalWrite(7, 0);
73     digitalWrite(36, 0);
74     digitalWrite(18, 0);
75 }
76 else if (time >= 6 && time < 7)
77 {
78     digitalWrite(35, 1);
79     digitalWrite(37, 1);
80     digitalWrite(38, 0);
81     digitalWrite(32, 1);
82     digitalWrite(22, 1);
83     digitalWrite(7, 1);
84     digitalWrite(36, 1);
85     digitalWrite(18, 1);
86 }
87 else if (time >= 5 && time < 6)
88 {
89     digitalWrite(35, 1);
90     digitalWrite(37, 1);
91     digitalWrite(38, 0);
92     digitalWrite(32, 1);
93     digitalWrite(22, 1);
94     digitalWrite(7, 0);
95     digitalWrite(36, 1);
96     digitalWrite(18, 1);
97 }
98 else if (time >= 4 && time < 5)
99 {
100     digitalWrite(35, 1);
101     digitalWrite(37, 0);
102     digitalWrite(38, 1);
103     digitalWrite(32, 1);
104     digitalWrite(22, 0);
105     digitalWrite(7, 0);
106     digitalWrite(36, 1);
107     digitalWrite(18, 1);
108 }
109 else if (time >= 3 && time < 4)
110 {
111     digitalWrite(35, 0);
112     digitalWrite(37, 1);
113     digitalWrite(38, 1);
114     digitalWrite(32, 1);
115     digitalWrite(22, 1);
116     digitalWrite(7, 0);
117     digitalWrite(36, 1);
118     digitalWrite(18, 1);
119 }
120 else if (time >= 2 && time < 3)
121 {
122     digitalWrite(35, 0);
123     digitalWrite(37, 1);
124     digitalWrite(38, 1);
125     digitalWrite(32, 0);
126     digitalWrite(22, 1);
127     digitalWrite(7, 1);
128     digitalWrite(36, 1);
129     digitalWrite(18, 1);
130 }
131 else if (time >= 1 && time < 2)
```



```

132     {
133         digitalWrite(35, 0);
134         digitalWrite(37, 0);
135         digitalWrite(38, 1);
136         digitalWrite(32, 1);
137         digitalWrite(22, 0);
138         digitalWrite(7, 0);
139         digitalWrite(36, 0);
140         digitalWrite(18, 0);
141     }
142     else if (time >= 0 && time < 1)
143     {
144         digitalWrite(35, 1);
145         digitalWrite(37, 1);
146         digitalWrite(38, 1);
147         digitalWrite(32, 1);
148         digitalWrite(22, 1);
149         digitalWrite(7, 1);
150         digitalWrite(36, 0);
151         digitalWrite(18, 0);
152     }
153 }
154
155 void leftDigit(double time) // Prints the left digit on the dual 14-segment display
156 {
157     if (time >= 10 && time < 20)
158     {
159         digitalWrite(12, 1);
160         digitalWrite(40, 0);
161
162         // Left side digit 1
163         digitalWrite(35, 0);
164         digitalWrite(37, 0);
165         digitalWrite(38, 1);
166         digitalWrite(32, 1);
167         digitalWrite(22, 0);
168         digitalWrite(7, 0);
169         digitalWrite(36, 0);
170         digitalWrite(18, 0);
171     }
172     else if (time < 10)
173     {
174         // No digit on left side
175         digitalWrite(12, 0);
176         digitalWrite(40, 1);
177     }
178 }
179
180 int main()
181 {
182     if(wiringPiSetupPhys() == -1){
183         exit(1);
184     }
185     int currState;
186     int externalButtonValue;
187     uint32_t inputs = 0;
188
189     // Initialize ports and timer
190     pinMode(16, INPUT); //Button
191     pinMode(11, OUTPUT); //Red
192     pinMode(13, OUTPUT); //Blue
193     pinMode(15, OUTPUT); //Green
194     pinMode(29, OUTPUT); //Red
195     pinMode(31, OUTPUT); //Blue
196     pinMode(33, OUTPUT); //Green
197

```

```

198 pinMode(35, OUTPUT); //Top left
199 pinMode(37, OUTPUT); //Top
200 pinMode(38, OUTPUT); //Top right
201 pinMode(32, OUTPUT); //Bottom right
202 pinMode(22, OUTPUT); //Bottom
203 pinMode(7, OUTPUT); //Bottom left
204 pinMode(36, OUTPUT); //Middle left
205 pinMode(18, OUTPUT); //Middle right
206
207 pinMode(40, OUTPUT); //Ground for Left Digit
208 pinMode(12, OUTPUT); //Ground for Right Digit
209
210 digitalWrite(11, 0);
211 digitalWrite(13, 0);
212 digitalWrite(15, 0);
213 digitalWrite(29, 0);
214 digitalWrite(31, 0);
215 digitalWrite(33, 0);
216
217 digitalWrite(35, 0);
218 digitalWrite(37, 0);
219 digitalWrite(38, 0);
220 digitalWrite(32, 0);
221 digitalWrite(22, 0);
222 digitalWrite(7, 0);
223 digitalWrite(36, 0);
224 digitalWrite(18, 0);
225
226 delay(1000); // Shut both pins off for a second to recognize program reset
227 currState = goE; // Initial state
228 while (1)
229 {
230     // Light up the LED's correspondent to the current state
231     digitalWrite(11, FSM[currState].out[0]);
232     digitalWrite(13, FSM[currState].out[1]);
233     digitalWrite(15, FSM[currState].out[2]);
234     digitalWrite(29, FSM[currState].out[3]);
235     digitalWrite(31, FSM[currState].out[4]);
236     digitalWrite(33, FSM[currState].out[5]);
237
238     // Print the remaining time on the 14-segment display if necessary
239     if (currState == goNdigitR)
240     {
241         rightDigit(10 - counter);
242         counter += 0.01;
243     }
244     else if (currState == goNdigitL)
245     {
246         leftDigit(20 - counter);
247     }
248     else if (currState == flashOnN)
249     {
250         rightDigit(9 - counter);
251         counter += 1;
252     }
253
254     // Wait for how long the state is supposed to delay for
255     delay(FSM[currState].time);
256
257     // Turn off segment display when the time is up
258     if(currState == flashOnN && counter == 10)
259     {
260         digitalWrite(35, 0);
261         digitalWrite(37, 0);
262         digitalWrite(38, 0);
263         digitalWrite(32, 0);

```

```
264         digitalWrite(22, 0);
265         digitalWrite(7, 0);
266         digitalWrite(36, 0);
267         digitalWrite(18, 0);
268     }
269
270     // Increment the flash count when the traffic light is flashing
271     if (currState == flashOnE)
272     {
273         counter++;
274     }
275
276     // Use the inputs to determine which state to go to next
277     inputs = 0;
278     externalButtonValue = digitalRead(16);
279     if (externalButtonValue)
280     {
281         inputs += 2;
282     }
283     if (counter >= 10) // The counter will always be at 10 when it's ready to go to
a new state
284     {
285         inputs += 1;
286         counter = 0;
287     }
288
289     // Go to the next state (it might loop back to the same state)
290     currState = FSM[currState].next[inputs];
291 }
292 }
```