Lab 3: Mini Traffic Light System Report

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

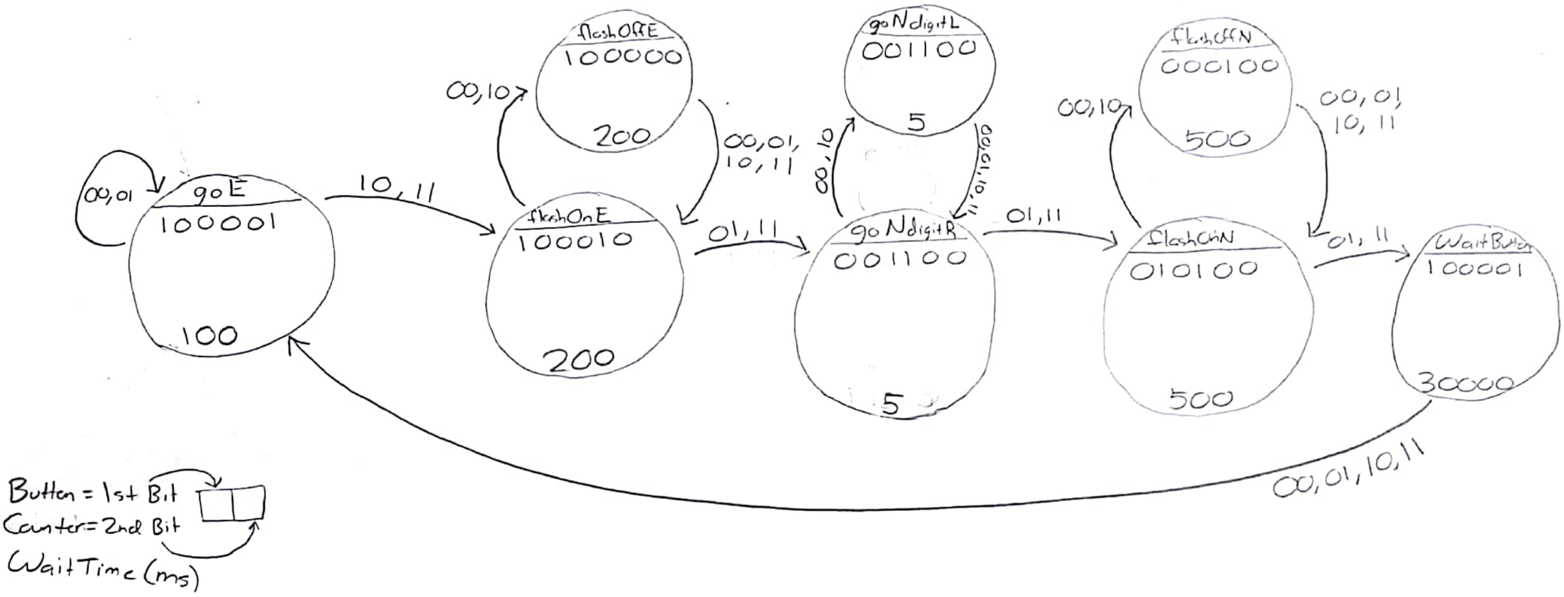
Thinking and Exploring

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.



goE
100001
100
00,01
10,11
flashOnE
100010
200
flashOffE
100000
200
00,10
00,01,10,11
01,11
goNdigitL
001100
5
goNdigitR
001100
5
00,10
00,01,10,11
01,11
flashOffN
000100
500
flashOnN
010100
500
00,10
00,01,10,11
01,11
WaitButton
100001
30000
00,01,10,11
Button = 1st Bit
Counter= 2nd Bit
WaitTime (ms)

```c
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <stdint.h>
#include <unistd.h>

#define INPUT  0
#define OUTPUT 1

#define goE        0
#define flashOnE   1
#define flashOffE  2
#define goN        3
#define flashOnN   4
#define flashOffN  5
#define waitButton 6

int buttonValue = 0;
uint32_t counter = 0;

typedef struct State
{
    uint32_t out[6];
    uint32_t time;
    uint32_t next[4];
} State;

State FSM[7] = {
    {{1, 0, 0, 0, 0, 1}, 100, {goE, goE, flashOnE, flashOnE}},
    {{1, 0, 0, 0, 1, 0}, 200, {flashOffE, goN, flashOffE, goN}},
    {{1, 0, 0, 0, 0, 0}, 200, {flashOnE, flashOnE, flashOnE, flashOnE}},
    {{0, 0, 1, 1, 0, 0}, 1000, {goN, flashOnN, goN, flashOnN}},
    {{0, 1, 0, 1, 0, 0}, 500, {flashOffN, waitButton, flashOffN, waitButton}},
    {{0, 0, 0, 1, 0, 0}, 500, {flashOnN, flashOnN, flashOnN, flashOnN}},
    {{1, 0, 0, 0, 0, 1}, 30000, {goE, goE, goE, goE}},
};

void clockWrite(int time){
    printf("Display: %d\n\n", time);
}

int main()
{
    int currState;
    uint32_t inputs = 0;

    sleep(1);
    currState = goE;
    while (1)
    {
        printf("Current state: ");
        if (currState == 0)
        {
            printf("Go east\n");
        }
        else if (currState == 1)
        {
            printf("Flash on east\n");
        }
        else if (currState == 2)
        {
            printf("Flash off east\n");
        }
        else if (currState == 3)
        {
            printf("Go north\n");
```

```
        }
        else if (currState == 4)
        {
            printf("Flash on north\n");
        }
        else if (currState == 5)
        {
            printf("Flash off north\n");
        }
        else if (currState == 6)
        {
            printf("Wait for button\n");
        }

        printf("\nTraffic Light Red: %d\n", FSM[currState].out[0]);
        printf("Traffic Light Blue: %d\n", FSM[currState].out[1]);
        printf("Traffic Light Green: %d\n", FSM[currState].out[2]);
        printf("Pedestrian Light Red: %d\n", FSM[currState].out[3]);
        printf("Pedestrian Light Blue: %d\n", FSM[currState].out[4]);
        printf("Pedestrian Light Green: %d\n\n", FSM[currState].out[5]);
        printf("Counter: %d\n\n", counter);

        if (currState == goN)
        {
            clockWrite(20 - counter);
            counter++;
        }
        if (currState == flashOnN)
        {
            clockWrite(10 - counter);
            counter++;
        }

        sleep(FSM[currState].time/1000);

        if (currState == flashOnE)
        {
            counter++;
        }

        inputs = 0;
        if (currState == goE) // Simulated button press
        {
            buttonValue = 1;
        }
        if (buttonValue)
        {
            inputs += 2;
        }
        if (currState == goE) // Simulated button release
        {
            buttonValue = 0;
        }

        if (counter == 10)
        {
            inputs += 1;
            counter = 0;
        }

        printf("Inputs: %d\n\n", inputs);
        printf("----------------------------------------------\n\n");
        currState = FSM[currState].next[inputs];
    }
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <stdint.h>

#define INPUT  0
#define OUTPUT 1

#define goE        0
#define flashOnE   1
#define flashOffE  2
#define goNdigitR  3
#define goNdigitL  4
#define flashOnN   5
#define flashOffN  6
#define waitButton 7

double counter = 0.00; // Used for flashing the LED's and segment display the appropriate amount of times

typedef struct State
{
    uint32_t out[6];
    uint32_t time;
    uint32_t next[4];
} State;

State FSM[8] = {
    {{1, 0, 0, 0, 0, 1}, 100, {goE, goE, flashOnE, flashOnE}},
    {{1, 0, 0, 0, 1, 0}, 200, {flashOffE, goNdigitR, flashOffE, goNdigitR}},
    {{1, 0, 0, 0, 0, 0}, 200, {flashOnE, flashOnE, flashOnE, flashOnE}},
    {{0, 0, 1, 1, 0, 0}, 5, {goNdigitL, flashOnN, goNdigitL, flashOnN}},
    {{0, 0, 1, 1, 0, 0}, 5, {goNdigitR, goNdigitR, goNdigitR, goNdigitR}},
    {{0, 1, 0, 1, 0, 0}, 500, {flashOffN, waitButton, flashOffN, waitButton}},
    {{0, 0, 0, 1, 0, 0}, 500, {flashOnN, flashOnN, flashOnN, flashOnN}},
    {{1, 0, 0, 0, 0, 1}, 30000, {goE, goE, goE, goE}},
};

void rightDigit(double time) // Prints the right digit on the dual 14-segment display
{
    digitalWrite(40, 1); // Adjust the ground pins appropriately
    digitalWrite(12, 0);

    if (time >= 9 && time < 10) // Print the number
    {
        digitalWrite(35, 1);
        digitalWrite(37, 1);
        digitalWrite(38, 1);
        digitalWrite(32, 1);
        digitalWrite(22, 1);
        digitalWrite(7, 0);
        digitalWrite(36, 1);
        digitalWrite(18, 1);
    }
    else if (time >= 8 && time < 9)
    {
        digitalWrite(35, 1);
        digitalWrite(37, 1);
        digitalWrite(38, 1);
        digitalWrite(32, 1);
        digitalWrite(22, 1);
        digitalWrite(7, 1);
        digitalWrite(36, 1);
        digitalWrite(18, 1);
    }
    else if (time >= 7 && time < 8)
```

```
    {
        digitalWrite(35, 0);
        digitalWrite(37, 1);
        digitalWrite(38, 1);
        digitalWrite(32, 1);
        digitalWrite(22, 0);
        digitalWrite(7, 0);
        digitalWrite(36, 0);
        digitalWrite(18, 0);
    }
    else if (time >= 6 && time < 7)
    {
        digitalWrite(35, 1);
        digitalWrite(37, 1);
        digitalWrite(38, 0);
        digitalWrite(32, 1);
        digitalWrite(22, 1);
        digitalWrite(7, 1);
        digitalWrite(36, 1);
        digitalWrite(18, 1);
    }
    else if (time >= 5 && time < 6)
    {
        digitalWrite(35, 1);
        digitalWrite(37, 1);
        digitalWrite(38, 0);
        digitalWrite(32, 1);
        digitalWrite(22, 1);
        digitalWrite(7, 0);
        digitalWrite(36, 1);
        digitalWrite(18, 1);
    }
    else if (time >= 4 && time < 5)
    {
        digitalWrite(35, 1);
        digitalWrite(37, 0);
        digitalWrite(38, 1);
        digitalWrite(32, 1);
        digitalWrite(22, 0);
        digitalWrite(7, 0);
        digitalWrite(36, 1);
        digitalWrite(18, 1);
    }
    else if (time >= 3 && time < 4)
    {
        digitalWrite(35, 0);
        digitalWrite(37, 1);
        digitalWrite(38, 1);
        digitalWrite(32, 1);
        digitalWrite(22, 1);
        digitalWrite(7, 0);
        digitalWrite(36, 1);
        digitalWrite(18, 1);
    }
    else if (time >= 2 && time < 3)
    {
        digitalWrite(35, 0);
        digitalWrite(37, 1);
        digitalWrite(38, 1);
        digitalWrite(32, 0);
        digitalWrite(22, 1);
        digitalWrite(7, 1);
        digitalWrite(36, 1);
        digitalWrite(18, 1);
    }
    else if (time >= 1 && time < 2)
```

```
    {
        digitalWrite(35, 0);
        digitalWrite(37, 0);
        digitalWrite(38, 1);
        digitalWrite(32, 1);
        digitalWrite(22, 0);
        digitalWrite(7, 0);
        digitalWrite(36, 0);
        digitalWrite(18, 0);
    }
    else if (time >= 0 && time < 1)
    {
        digitalWrite(35, 1);
        digitalWrite(37, 1);
        digitalWrite(38, 1);
        digitalWrite(32, 1);
        digitalWrite(22, 1);
        digitalWrite(7, 1);
        digitalWrite(36, 0);
        digitalWrite(18, 0);
    }
}

void leftDigit(double time) // Prints the left digit on the dual 14-segment display
{
    if (time >= 10 && time < 20)
    {
        digitalWrite(12, 1);
        digitalWrite(40, 0);

        // Left side digit 1
        digitalWrite(35, 0);
        digitalWrite(37, 0);
        digitalWrite(38, 1);
        digitalWrite(32, 1);
        digitalWrite(22, 0);
        digitalWrite(7, 0);
        digitalWrite(36, 0);
        digitalWrite(18, 0);
    }
    else if (time < 10)
    {
        // No digit on left side
        digitalWrite(12, 0);
        digitalWrite(40, 1);
    }
}

int main()
{
    if(wiringPiSetupPhys() == -1){
        exit(1);
    }
    int currState;
    int externalButtonValue;
    uint32_t inputs = 0;

    // Initialize ports and timer
    pinMode(16, INPUT);  //Button
    pinMode(11, OUTPUT); //Red
    pinMode(13, OUTPUT); //Blue
    pinMode(15, OUTPUT); //Green
    pinMode(29, OUTPUT); //Red
    pinMode(31, OUTPUT); //Blue
    pinMode(33, OUTPUT); //Green

```

```
    pinMode(35, OUTPUT); //Top left
    pinMode(37, OUTPUT); //Top
    pinMode(38, OUTPUT); //Top right
    pinMode(32, OUTPUT); //Bottom right
    pinMode(22, OUTPUT); //Bottom
    pinMode(7, OUTPUT);  //Bottom left
    pinMode(36, OUTPUT); //Middle left
    pinMode(18, OUTPUT); //Middle right

    pinMode(40, OUTPUT); //Ground for Left Digit
    pinMode(12, OUTPUT); //Ground for Right Digit

    digitalWrite(11, 0);
    digitalWrite(13, 0);
    digitalWrite(15, 0);
    digitalWrite(29, 0);
    digitalWrite(31, 0);
    digitalWrite(33, 0);

    digitalWrite(35, 0);
    digitalWrite(37, 0);
    digitalWrite(38, 0);
    digitalWrite(32, 0);
    digitalWrite(22, 0);
    digitalWrite(7, 0);
    digitalWrite(36, 0);
    digitalWrite(18, 0);

    delay(1000); // Shut both pins off for a second to recognize program reset
    currState = goE; // Initial state
    while (1)
    {
        // Light up the LED's correspondent to the current state
        digitalWrite(11, FSM[currState].out[0]);
        digitalWrite(13, FSM[currState].out[1]);
        digitalWrite(15, FSM[currState].out[2]);
        digitalWrite(29, FSM[currState].out[3]);
        digitalWrite(31, FSM[currState].out[4]);
        digitalWrite(33, FSM[currState].out[5]);

        // Print the remaining time on the 14-segment display if necessary
        if (currState == goNdigitR)
        {
            rightDigit(10 - counter);
            counter += 0.01;
        }
        else if (currState == goNdigitL)
        {
            leftDigit(20 - counter);
        }
        else if (currState == flashOnN)
        {
            rightDigit(9 - counter);
            counter += 1;
        }

        // Wait for how long the state is supposed to delay for
        delay(FSM[currState].time);

        // Turn off segment display when the time is up
        if(currState == flashOnN && counter == 10)
        {
            digitalWrite(35, 0);
            digitalWrite(37, 0);
            digitalWrite(38, 0);
            digitalWrite(32, 0);
```

```
            digitalWrite(22, 0);
            digitalWrite(7, 0);
            digitalWrite(36, 0);
            digitalWrite(18, 0);
        }

        // Increment the flash count when the traffic light is flashing
        if (currState == flashOnE)
        {
            counter++;
        }

        // Use the inputs to determine which state to go to next
        inputs = 0;
        externalButtonValue = digitalRead(16);
        if (externalButtonValue)
        {
            inputs += 2;
        }
        if (counter >= 10) // The counter will always be at 10 when it's ready to go to a new state
        {
            inputs += 1;
            counter = 0;
        }

        // Go to the next state (it might loop back to the same state)
        currState = FSM[currState].next[inputs];
    }
}
```