Led Light Controller

Jason Medeiros, Luan Nguyen and Naveen Reddy

CMPE 30 Fall 2017 , Lab Section 2

Abstract

This report will describe how we used the SJ One board to program an led light strip to change colors on button presses and when the board reached a specified temperature. We will include diagrams in addition to writings to better describe the project.

I. Introduction

We used the SJ One board and created a state machine to make an LED light strip change colors across different conditions. We use the GPIO (General Purpose Input Output) pins in order to send the right signals to the board appliances. 3 MOSFET transistors were used to control the current that would be sent from the board to the light strip. Jumper wires and a breadboard were used to connect the light strip to the board. The GPIO pins were used to specify when we wanted to essentially switch on the current going to each color of the LED strip.

II. Design Methodology

In our design, we utilized jumper wires, 3 MOSFET transistors, an LED light strip, and a battery holder that output 12 volts. In order to make these parts change the colors of the light strip with our board, we connected the battery to the light strip, the light strip to the transistors, and the

transistors to both the ground as well as the GPIO pins on the board. The transistor connection was the most complex, as it had three parts that function differently. MOSFET transistors have a gate, source, and drain. The gate was connected to the GPIO pins in the board, to allow the programmed board to switch the transistors on and off. The source was connected to the ground of the breadboard to create a full circuit. The drain was connected to the LED strip, so that each transistor would switch a different color on and off. In the design of our program, we used the buttons on the SJ One board to activate different states that we created. The states were placed within a "while loop" so the program could be repeated and executed in the order we desired. Bool statements were used to activate individual conditions for a case. Specific colors were activated by setting certain pins to high and low voltages (i.e. "myPin.setHigh"()). The states we created were Start, temperature_sensor, Light, and Idle. The "Start" case simply started the program. Button 0 on the board transitioned to the "temperature_sensor" case. This case activated the temperature sensor located on the board, and the LED display displayed the current temperature the sensor was reading. Once the sensor reached 87 degrees Fahrenheit (specified by us), the light strip would flash blue. Button 1 on the board transitioned to the "Light" case. This case removed the flashing blue light, and kept a solid red. Button 2 on the board transitioned to the "idle" case and changed the light strip's color to green. Button 3 on the board remained in the "idle" case and changed the color of the strip to red. Button 0 on the board could then be pressed again to transition from "idle" to "Light" or Button 1 could transition from "idle" to "temperature sensor"; which effectively allowed us to repeat the program and continue to transition from case to case (as opposed to getting stuck in idle). For more information, refer to

Appendix A for the Circuit Diagram, Appendix B for the State Machine Diagram, and Appendix C for the Source Code.

A. Parts List

Non-addressable RGB LED light strip

12V 8 x AA battery clip slot holder

3 N Channel MOSFET

Breadboard

Jumper Cables

B. Schematics

Refer to the appendices for diagrams/schematics.

III. Testing Procedures

1. We studied our lab lecture concerning this project and used the diagram drawn on the board as our basis for wiring our LED strip and MOSFETs to our SJ One board.
2. We programmed our project from start to finish, and then edited it for a longer period of time to remove the errors when building the program.
3. Despite no errors, our project didn't work as desired and we studied and troubleshot our circuit wiring and the program's code.
4. The TA, Kevin Chan, and the Lab Instructor, Charley Abboud helped and guided us in wiring our project correctly as well as editing our code so it would function as desired.
5. We went through all the buttons verifying the project worked correctly, and fine polished the code so everything worked ideally as we desired.

IV. Testing Results

Nothing seemed to work until Kevin and Charley helped us to troubleshoot our code. We were pretty close to a functioning project, but we made a couple mistakes with the wiring to our MOSFETs and our code required a few extra lines as well as 5-8 modifications. We were given an example on how to program a specific instruction, and then we would apply it to the rest of the program on our own.

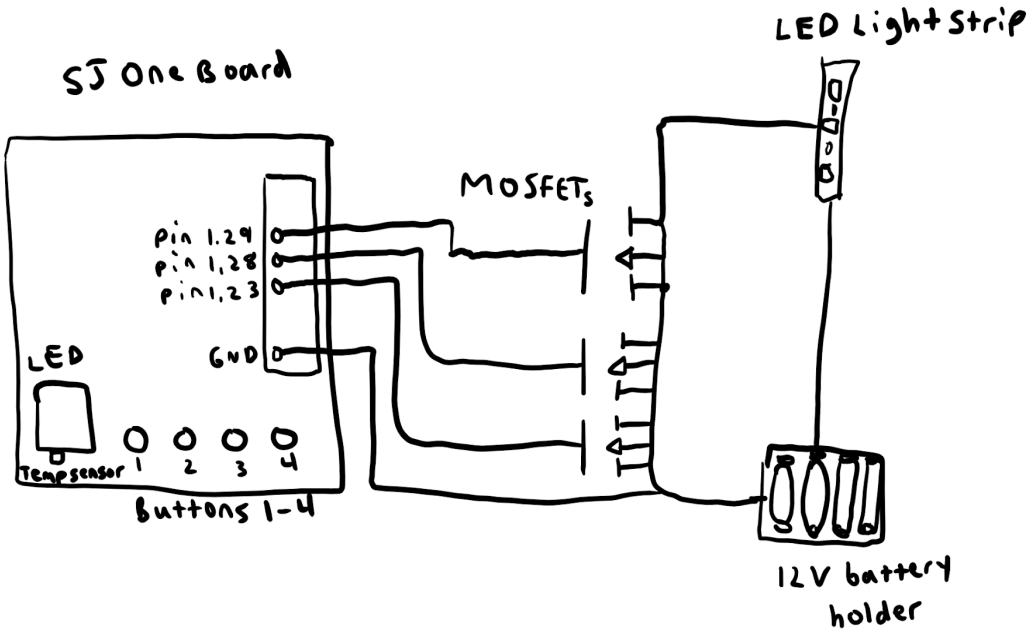
V. Conclusion

As stated above, we designed a program and circuit that would change the colors of an LED light strip upon button pushes on the SJ One board and when high enough temperatures were released. We learned a great deal from this project, including wiring our own circuit and programming an IO board to control the circuit. We got some practice working with a breadboard and the components used on it, as well as applying a program to an external piece of hardware. We encountered a good number of problems, but we learned from our mistakes and ended up understanding a lot more. The first problem we encountered was with our wiring. We thought it was correct, but Kevin showed us how we set up the circuit backwards due to a mixup of the locations of the source and drain on the MOSFETs. Second, we had some unknown problems in our code. Charley and Kevin throughout lab helped us discover the problems and learn from them. We learned that we had to set a pin to low before the color could be changed (we were trying to change colors, but weren't getting desired colors due to having all the other pins set to high still). We also learned that the SJ One board buttons were mislabeled (Labeled 0-3 instead

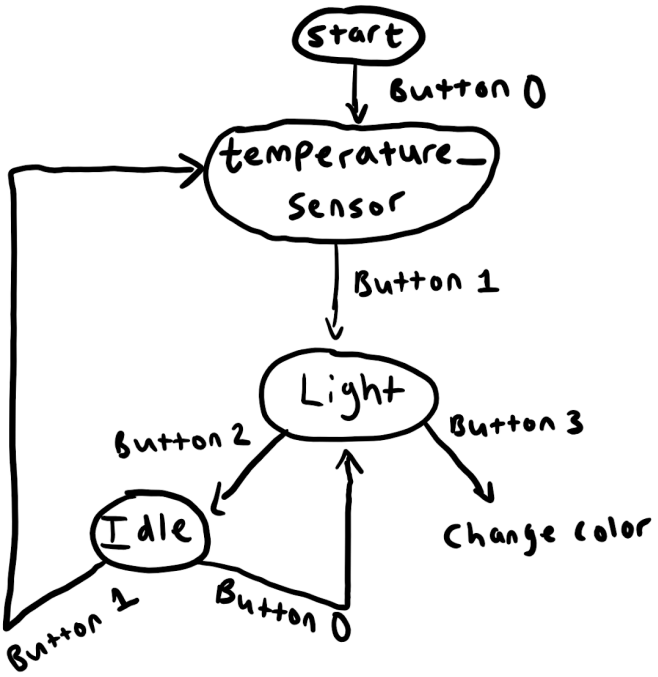
of 1-4), so we changed our “get switch” functions accordingly. Additionally, we didn’t know we had to declare all our GPIO pins as outputs. We only had one declared as an output previously, which caused only one color on the light strip to work. Lastly, we learned that a button could be reused/reassigned in different cases. We wanted to be able to transition from idle back to one of our other two cases, but didn’t think we could since we already used all the buttons. So we assigned more switches in the cases so the user wouldn’t get stuck at any one case in the program. In the way of optimizing our now functioning code, we listed different pins in each case to see which colors we liked better to show up. We also reduced the temperature requirement for the light strip to turn on because all of our hands were cold and we were having trouble getting the temperature sensor on the board to reach the required temperature.

VI. Appendices and References

Appendix A: Circuit Diagram



Appendix B: State Machine Diagram



Appendix C: Source Code

```
#include <stdio.h>

#include <stdbool.h>

#include "io.hpp"

#include "utilities.h"

#include "gpio.hpp"

void setTemperature(int*temperature, bool *hotEnough);

void setTemperature(int*temperature, bool *hotEnough){

	*temperature = TS.getFarenheit();

	LD.setNumber(*temperature);

	if (*temperature > 86)

	{

		*hotEnough = true;

			LE.on(1);

		}

		else

		{

			LE.setAll(0);

		}

	}

int main(void) {

	/* Without this, printf() will not work */
```

```
                setvbuf(stdout, 0, _IONBF, 0);
                setvbuf(stdin, 0, _IONBF, 0);
typedef enum {
        start, idle, light, temperature_sensor
} myStateType;
myStateType currentState = start
}
bool hotEnough = false;
GPIO myPin(P1_29);
myPin.setAsOutput();
GPIO myPin1(P1_28);
myPin1.setAsOutput();
GPIO myPin2(P1_23);
myPin2.setAsOutput();
int temperature;
        bool sw0_pressed = SW.getSwitch(0);
        bool sw1_pressed = SW.getSwitch(1);
        bool sw2_pressed = SW.getSwitch(2);
        bool sw3_pressed = SW.getSwitch(3);
while(1) {
delay_ms(200);
                sw0_pressed = SW.getSwitch(1);
```

```
        sw1_pressed = SW.getSwitch(2);
        sw2_pressed = SW.getSwitch(3);
        sw3_pressed = SW.getSwitch(4);
switch(currentState){
case start:
        myPin.setLow();
                myPin1.setLow();
                myPin2.setLow();
        printf("Current state is start\n");
if(sw0_pressed)
{
        currentState = temperature_sensor;
        printf("Going to temperature\n");
        LE.setAll(0);
}
break;
case temperature_sensor:
        printf("Current state is temperature\n");
        setTemperature(&temperature, &hotEnough);
        if(hotEnough)
        {
                myPin2.setHigh();
```

```
            delay_ms(200);
            myPin2.setLow();//LED flashes blue
        }
        else{
            myPin2.setLow();
        }
        if (sw1_pressed)
        {
            currentState = light;

            printf("Going to light\n");
        }
        break;


case light:
        printf("Current state is light\n");
        if (sw2_pressed)
        {
            currentState = idle;
            myPin1.setHigh();
            delay_ms(200);
```

```
//LED is green

                printf("Going to idle\n");
            }

                else if (sw3_pressed){
                    myPin.setHigh();
                    delay_ms(200);
                    myPin1.setLow();
                    //LED is red
                }

            break;
        case idle:
            printf("Current state is idle\n");
            if (sw0_pressed)
            {
                currentState = light;
                LE.setAll(0);
            }
            else if (sw1_pressed){
                currentState = temperature_sensor;
```

```
                }


                break;
        default:
                printf("Error\n");


    }
    }
}
```