Martin Perina

Assembly (csc-11)

Paul Conrad

12/10/2019

Final Project Documentation

This traffic lights project took me several days to complete, but I welcomed the challenge. Before beginning the project, I gathered information from class notes, code, and found a few online resources. Once I had gathered enough information, I made a small draft of what the project would look like and the steps I would take to make it. I then began building and writing code, followed by a series of tests.

First, I set up a simple circuit consisting of a long breadboard, LEDs, resistors, and a button that represented the traffic lights, crosswalk lights, and button to activate the crosswalk. I tested the circuit by applying 3 volts directly to each connection to make sure each LED worked. For the button, I connected a GPIO pin to the circuit and ran the “gpio readall” command with the button pressed and un-pressed to make sure the pin associated with the button detected a change in voltage.

Second, knowing that the LEDs all worked as intended, I wrote a program to run the green, yellow, and red LEDs for the traffic lights (for car traffic). I configured each pin for output with the GPIO pin mode. Then, I programmed each pin to be turned on and then off with a delay in between to simulate a normal traffic light.

Third, I wrote a program to run the green and red LEDs for the crosswalk signals. Once I tested the program, I added in a comparison to check if a voltage was detected upon the button being pressed. The program keeps the red LED on until the button is pressed. If the button is pressed, the red LED is turned off and the green is turned on with a delay, then the lights are reset and the red turns back on with the green off. The program loops infinitely and handles any number of button presses.

After testing both programs, I then drafted a final program that combines the traffic lights and crosswalk lights with a loop that continues unless the button is pressed. If the button is pressed a different code is executed to run the crosswalk lights. The program initially only contained three functions and contained a few minor bugs. After more testing and rewriting the program ran as intended. I then set about writing a final code that satisfied all requirements and ran more efficiently. The final program contains six functions, branching, time delays and output to the terminal for crosswalk instructions (wait, walk, walk ending). The entire code is written in blocks and contains comments for readability. The program is easily editable, allowing for a change in timing between lights if desired.

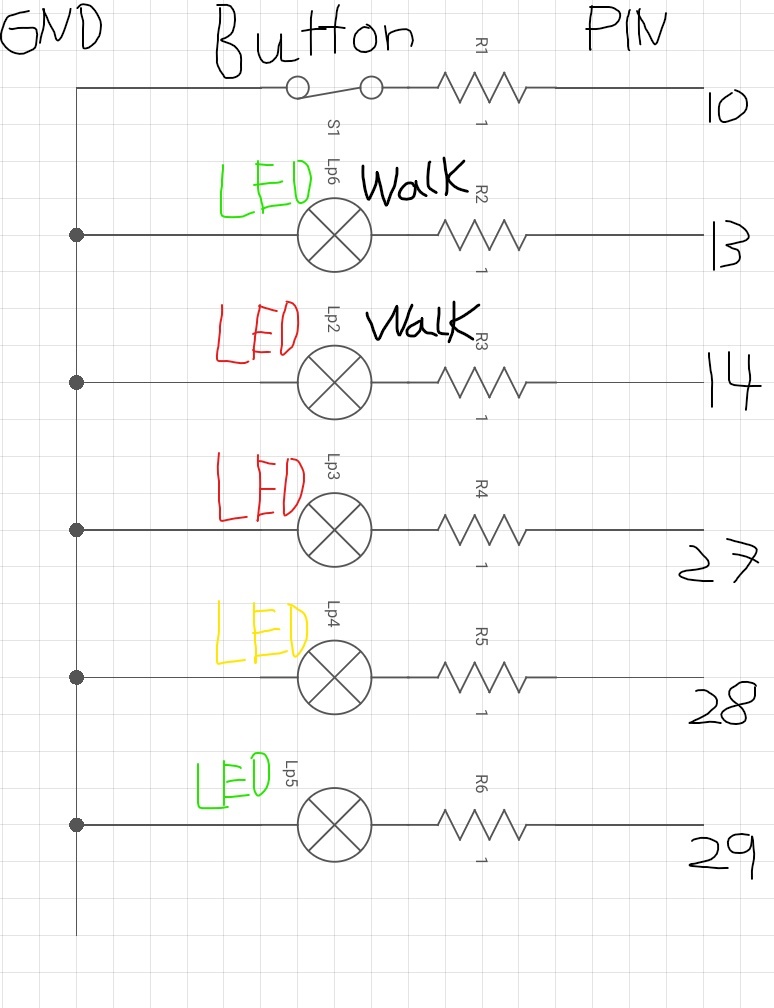
Throughout the process of making this project I used extensive testing. After writing a code or making small changes I ran the program to note exactly what had changed. When testing, I typically only made one change to the code in order to keep trouble shooting simple in order to have only one variable to deal with.

This project was an interesting and exciting challenge for me. I was happy to showcase my new skills with assembly language through writing a complex program as well as create a circuit controlled by a code I wrote. I felt very satisfied once I completed the project after struggling through the process of creating it.

I appreciated that we spent time learning the hardware of the Raspberry Pi and using assembly to program a circuit. It was a great way to extend the material in an interesting and practical way. I am planning on doing my own projects on the Raspberry Pi in the future.

**GPIO Pins that were used and Diagram:**

|  |  |  |
| --- | --- | --- |
| **LED Color** | **wiringPI #** | **Physical #** |
| Green | 29 | 40 |
| Yellow | 28 | 38 |
| Red | 27 | 36 |
| Red [crosswalk] | 14 | 23 |
| Green [crosswalk] | 13 | 21 |
| \*Button [crosswalk] | 10 | 24 |



**Pseudocode:**

all .equ directives

input, output, low, high

pins 27-29, 10,13,14

data

messages for walking

end of program message

#all functions below

main

start

MODE

set all pins for output

set button pin for input

WALK\_LIGHTS

display no\_walk message

turn on red walk light

LIGHTS

green on

delay

green off

yellow on

delay

yellow off

red on

delay

check if button pressed

yes -> branch WALK

no -> red off and b WALK\_LIGHTS

CONT\_LIGHTS

red off

branch WALK\_LIGHTS

WALK

red walk off

delay

green walk on

display walk message

delay

branch BLINK

BLINK

green and yellow on

display walk ending message

delay short

green and yellow off

green walk off

delay short

branch CONT\_LIGHTS

END

red off

red walk off

display end of program message

end program