Principles of Engineering

Mini-Project 1 - Build Your Own Bike Light

Repository Link: `https://github.com/notwaltervilla/PIE-Mini-Project-1`

Section 1
Walter Villa & Tolulope Oshinowo

Due: September 16, 2021

Summary

The following report aims at introducing the principles of circuits and embedded software in a project where students are able to build a bike light with a push-down button, LED's, and a potentiometer. The bike light must be able to change different modes with a press of the push-down button, as well as vary brightness using a potentiometer.

Introduction

In this mini-project, we were tasked to create a bike light that was capable of switching between several modes of flashing, as well as incorporating a potentiometer to modify the brightness of the LED's in real time.

Materials

- 5 LED's (Blue, Green, Yellow, Red, and White)
- 3 Resistors - 300 Ω
- 1 Push Down button
- 1 Pull-down resistor for button (10k Ω)
- 1 Potentiometer
- 1 Arduino Uno R3
- Several Wires

Method - Part 1

In the first part of the lab, we were tasked to build a bike light using the following schematic:

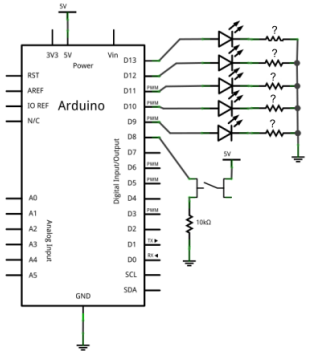

Figure 1.0 - Schematic provided by the teaching team.

To complete our drawing, we needed to compute the correct resistance in order to complete the circuit schematic. To do this, we made a simple sketch, shown in Figure 1.1, to conceptualize what values were known in order to find the missing resistance values.

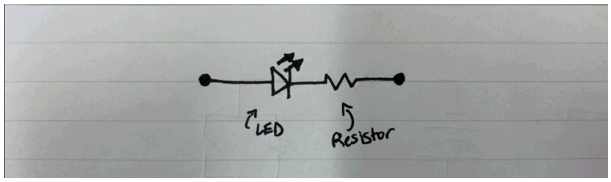

Figure 1.1 - Sketch that was used to conceptualize circuit diagram.

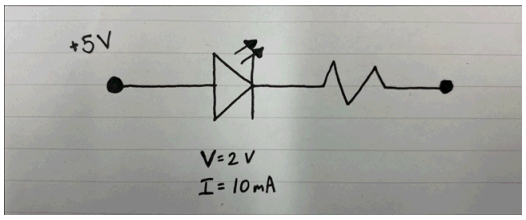

Figure 1.2 - Filled in known values to our circuit diagram.

From our schematic and our data sheet, we were able to figure out values for the input voltage, as well as the voltage drop and current through the LED. The LED data was found on the data spec sheet for the LED provided in the assignment. We filled these values into our sketch, shown in Figure 1.2.

Because the LED's and resistors in the schematic were connected in series, we knew that due to Kirchhoff's circuit laws, the sum of the voltage drops across each component of the circuit would be equal to the voltage supply of the circuit: 5V. With this, we found that the voltage drop through each resistor would come out to be 3V. Additionally, because the circuit is in series, we are able to say that the current throughout the circuit is the same, meaning that the current through the resistor is also equivalent to 10mA. Using Ohm's Law, we were able to find that the correct resistance to use was 300Ω, as shown in Figure 1.3.

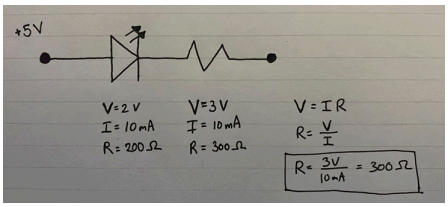

Figure 1.3 - Evaluating the resistor values using Ohm's Law.

Once we were able to calculate the values from Figure 1.3, we assembled our circuit, following the schematics from Figure 1.0. Our final product is shown in Figure 1.4.

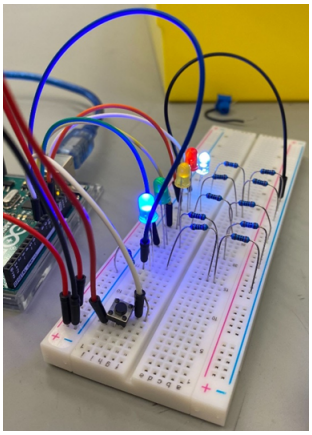

Figure 1.4 - Picture of our final circuit for Part 1 of Mini-Project 1.

With the circuit built, we decided to program the different modes for the lights. According to the Mini Project 1 instructions we needed to have at least three parallel LED's that were fed signals from the Arduino to operate. In addition to multiple LED's, we needed them collectively to operate in five different modes: all on, all off, all blinking, "bouncing" (like KITT's light bar from *Knight Rider*), and our own custom mode that we called "zone" which could be thought of as two mirrored "bouncing" patterns.

To accomplish this task we first had to allocate which pins on the Arduino would correspond to the correct LED's. Based on the original schematic in figure 1.0, we had five LED's with digital signals fed from ports D13 through D9. From there we made sure that we had all the necessary circuit configurations referenced in the `setup()` function of our code; this

included referencing each of the digital pins we used as output signal pins. Alongside the `setup()` function we had three variables (`ledState`, `currentButtonState`, and `lastButtonState`) initialized to help with cycling through each of the illumination modes. With the Arduino code properly set up we then went on to write out helper functions to handle the illumination modes with a series of `digitalWrite()` and `delay()` function calls to generate each pattern. With all of this working we were then able to add an 'if' statement and a 'switch' statement to the `main()` function. The 'if' statement handled determining what to do when the button was pressed (based on differences between `currentButtonState` and `lastButtonState`), cycling the `ledState` value between 0 and 4; and the 'switch' statement handled determining which helper function needed to be invoked based on the current value of `ledState`. After we had fleshed out all of this code our bike light worked like a charm and we were done with part one of mini-project 1. It was now time to add in our analog input: the potentiometer.

Method - Part 2

In Part 2, we were tasked with introducing a potentiometer into our circuit in order to modify some part of our output, whether that be speed, brightness, or anything that could affect the LED's. Our instinctive approach was to use the potentiometer in order to control the brightness of the LED's during each different type of light mode. For this, we consulted outside resources in order to understand how a potentiometer worked with respect to the Arduino system. Figure 2.0 is a diagram that we used to reverse engineer how the potentiometer would be implemented on our circuit. Understanding how a potentiometer was integrated in a circuit allowed us to modify our circuit in order to read its output signal to then modify our program.

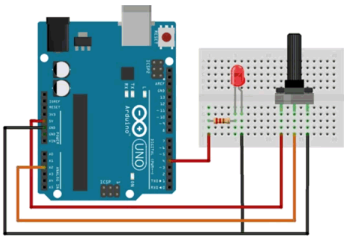

Figure 2.0 - Schematic we used to reverse engineer how potentiometers worked in relation with Arduino.

(Source: https://coolcomponents.co.uk/blogs/news/how-to-use-a-potentiometer-with-an-arduino)

Our initial step with using the potentiometer was to connect the Analog Pin 0 on the Arduino to the Signal Pin on the potentiometer with an intention to see if we could get different resistance readings through the Serial Monitor. Next, we integrated the potentiometer into our current circuit by connecting its power to 5V, its ground to ground, and keeping the signal coming from Pin 0. Once our serial monitor read values while also not affecting our different LED modes via our push-button, we started to investigate ways in which we could use the changes in the resistance values to modify the brightness of our LED's.

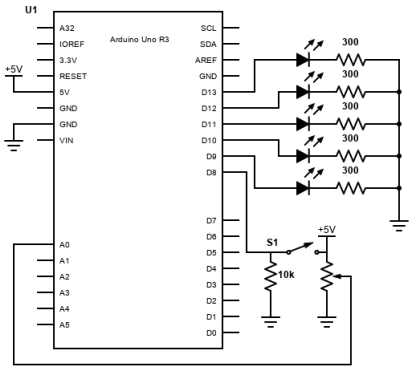

Figure 2.1 - Schematic of our final circuit after Part 2.

We found `analogWrite()`, a function that writes an analog value (PWM wave) to a pin (commonly used to vary brightness or speed), to be very useful when it came to finalizing our program during this part of the lab. We replaced our original

`digitalWrite()` function calls in our helper functions with `analogWrite()` in order for it to vary the output on the different LED pins.

Conclusion

In conclusion, this was a great project to begin to understand and feel comfortable with integrated systems in engineering. We were both fairly new to working with Arduino, and we believe that we learned equally in building functional circuits and programming embedded software.

Walter: In this project, I started to become a lot more comfortable with building circuits and embedded software. I enjoyed reading documentation and notes on what certain electrical and software components did. I am eager to do future projects with Tolu, and I am also excited to be more creative with the next way in which we approach the next project.

Tolu: During this mini-project I had the opportunity to think about the most intuitive way to build circuits. I have worked with Arduino in the past but this was a great refresher for me and let me create something that I would envision myself using. Working with Walter was a pleasure and I am thrilled to see what's next.

References

- https://coolcomponents.co.uk/blogs/news/how-to-use-a-potentiometer-with-an-arduino
- https://olin.instructure.com/courses/298/files/38408?wrap=1
- https://www.digikey.com/en/products/detail/kingbright/WP7113ID/1747663