Lab 1: Intro to MATLAB and Arduino

Roberto Melendez, Christopher Felix, Juliana Facchini, Yeani Mitropoulos

Florida International University, Biomedical Engineering Department, 10555 W Flagler St, Miami, FL 33174

1. Introduction

The focus of this lab was to illustrate the use of LEDs, potentiometers, and photoresistors in creating a Morse code blinker and analyzing the correlation between LED brightness and light intensity detected by a photoresistor. Moreover, by using MATLAB, we were able to program an LED to blink out the first letter of our group number in Morse code, control the LED's brightness using a potentiometer, and analyze how a photoresistor detects the varying brightness levels.

Key concepts:

- **Morse Code**: A method of encoding text using a sequence of short and long signals, known as "dots" and "dashes". Each letter of the alphabet is translated into a specific sequence of these signals. For this experiment, we encoded the first letter of our group number, "t", which was represented by the blinking of an LED on the Arduino.
- **LED Brightness Control**: A potentiometer allows for the adjustment of voltage. We were able to effectively dim the LED from full brightness (1) to no brightness (0), simulating fading light intensity.
- **Photoresistor**: A light-sensitive resistor whose resistance decreases as the intensity of light it is exposed to increases, allowing the system to measure how much light the LED emits based on its brightness.

The software tool used to implement the functionality was the Arduino Integrated Development Environment, which facilitates the programming of the microcontroller to handle the LED blink and the potentiometer's input.

2. Methods and Materials

Materials:

- Arduino Uno board
- LED (light-emitting diode)
- Potentiometer
- Photoresistor (LDR - Light Dependent Resistor)
- Breadboard and jumper wires
- Resistors (220 ohms for LED, 10k ohms for photoresistor)
- Arduino Integrated Development Environment (IDE)
- 5V power supply

❖ **Setting up the Morse Code LED Blink:**

a. The LED was connected to pin 13 of the Arduino.
b. The potentiometer was connected to an analog input pin which allowed for the

control of the LED's brightness. The wiper pin of the potentiometer was connected to the analog input, while the other two pins were connected to 5V and GND respectively.
c. The program was written using the Arduino IDE to generate the Morse code for the first letter of our group number, “t”, which blinked as “-”.
d. The potentiometer’s reading ranged from 0 to 1023. It was mapped to control the brightness of the LED, with 1023 corresponding to full brightness and 0 corresponding to no brightness.

❖ **Setting up the Photoresistor:**

a. The photoresistor was connected in a voltage divider configuration with a 10k ohm resistor. One leg was connected to 5V, the other to the analog input pin, and the middle node to the 10k ohm resistor, which was grounded.
b. The photoresistor’s resistance would change based on the brightness of the LED, and its voltage was read by the analog input pin.

❖ **Programming the Arduino:**

a. The program’s goal was to read the potentiometer's value and adjust the PWM signal sent to the LED to either dim or brighten it.
b. At the same time, the photoresistor's voltage was monitored, and its output was plotted to illustrate how the photoresistor's voltage corresponds to the changes in the LED’s brightness.

Data Collection:

- The potentiometer was turned, dimming and brightening the LED from its full intensity to a dim state.
- The photoresistor's voltage was recorded at various stages of the potentiometer’s adjustments to determine how the change in LED brightness affected the detected light intensity.

3. Results

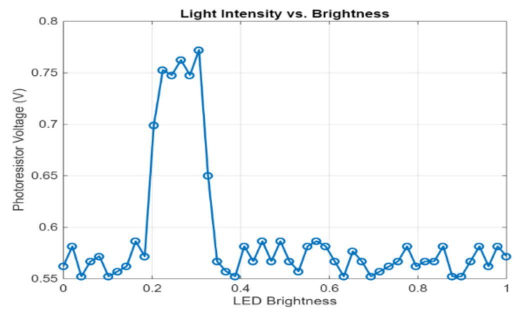

Figure 1 - Correlation between Voltage and Brightness

The relationship between the LED brightness and the voltage output of the photoresistor is depicted in Figure 1. The x-axis represents the LED brightness as controlled by the potentiometer, ranging from 0 (no brightness) to 1 (full brightness). The y-axis represents the voltage measured from the photoresistor, which signifies the intensity of light detected.

The graph shows a significant increase in the photoresistor's voltage as the LED brightness is adjusted from 0 to approximately 0.3. This steep increase demonstrates the photoresistor's sensitivity to increasing light intensity, which corresponds to the rising brightness of the LED. The voltage peaks at around 0.75 V when the LED brightness reaches approximately 0.3. Beyond this point, the voltage sharply declines, illustrating that once the brightness diminished, so does the voltage.

For LED brightness levels greater than 0.4, the photoresistor's voltage stabilizes around 0.55–0.6 V, with minor fluctuations. This constant response indicates that the photoresistor is detecting minimal or no additional changes in light intensity at these higher brightness levels.

These results align with the expected behavior of the photoresistor, which is designed to detect changes in light intensity.

4. Discussion

The results of this experiment demonstrate a clear relationship between LED brightness and the voltage output of the photoresistor. As expected, increasing the LED brightness resulted in a higher voltage reading from the photoresistor and vice versa. Figure 1 further corroborates this relationship, showing that brighter LED output consistently corresponded to higher detected voltage. This outcome is intuitive, as more light reaching the photoresistor naturally leads to a greater change in resistance and, consequently, an increase in voltage. No major discrepancies were observed, confirming the expected behavior of the components used.

One of the main challenges encountered during the experiment was correctly connecting the jumper wires, LED, and resistors on the breadboard to ensure proper circuit functionality. Initially, we encountered issues surrounding proper connection of wires with the Arduino pins and breadboard rows, which prevented the LED from turning on. Troubleshooting involved checking for loose connections, verifying component placement, and systematically adjusting the wiring until the circuit functioned correctly.

https://youtu.be/zkt8DwNkIOU?si=NycIy5T0VVp62BrF

https://youtu.be/bBnLFCIltTA?si=zxi-NkqVulCdBmJB

5. References

Ghassaei, Amanda. “Beginner Arduino.” *Autodesk Instructables*, Autodesk, Inc., 18 Oct. 2017, www.instructables.com/Beginner-Arduino/.