ACTIVITY 3 - Arduino 3: Analog Input

Jason Harvey Lorenzo, February 23, 2024
ENGG 122.02, Department of Electronics, Computer, and Communications Engineering
School of Science and Engineering, Ateneo de Manila University
Quezon City, 1108, Philippines
jason.lorenzo@obf.ateneo.edu

***Abstract*—This paper explores the utilization of Arduino's analogRead() function to regulate LED blinking. A voltage is generated via a basic voltage divider, facilitated by a potentiometer connected to 5V. The quantity of blinking LEDs corresponds directly to the detected voltage. It presents the source code, explaining the conversion of analog to digital values and the subsequent LED activation based on the detected voltage. Computationally, linear algebra principles are applied to derive equations for converting digital to analog voltage and determining LED pin numbers based on interval boundaries.**

***Index Terms*—Arduino UNO, analog input, voltage divider, blinking LEDs**

I. Program Listing

This activity utilizes Arduino's analogRead() function to control the blinking of LEDs. A voltage is produced through a basic voltage divider formed with a potentiometer connected to 5V. The quantity of blinking LEDs corresponds to the voltage detected. The function of the final outcome is outlined in Table I.

TABLE I
LED Pins blinking based on input signal

Input Signal (Volts)	LED Pins Blinking
0 to 0.5	None
0.5 to 1	4
1 to 1.5	4 to 5
1.5 to 2	4 to 6
2 to 2.5	4 to 7
2.5 to 3	4 to 8
3 to 3.5	4 to 9
3.5 to 4	4 to 10
4 to 4.5	4 to 11
4.5 to 4.9	4 to 12
4.9 to 5	4 to 13

A. Source Code

```
int digitalVolt;
float analogVolt;

void setup() {
  for(int i=4; i<=13; i++)
    pinMode(i, OUTPUT);
  pinMode(A0, INPUT);
}

void loop() {
  digitalVolt = analogRead(A0);
  analogVolt = (5./1023.)*digitalVolt;

  for(float i=0.5; i<analogVolt; i+=0.5){
    digitalWrite((2*i+3), HIGH);
  }

  if(analogVolt>4.9)
    digitalWrite(13, HIGH);

  delay(200);

  for(int i=4; i<=13; i++)
    digitalWrite(i, LOW);

  delay(100);
}
```

B. How It Works

In the setup() function, pins 4 to 13 were configured as OUTPUT pins, while pin A0 was designated as the input pin. The potentiometer's middle pin, connected to A0, produces a voltage ranging from 0 to 5V depending on its adjustment. The analogRead() function converts this analog input to a digital value within the range of 0 to 1023, stored in the digitalVolt variable as illustrated in the code.

In the loop() function, analogWrite() is employed to set the value of digitalVolt. To facilitate the use of analog values in Table I, this digital value is converted to an analog equivalent using the formula:

$$analogVolt = \frac{5}{1023} \times digitalVolt \tag{1}$$

Further discussion on the derivation of this formula is presented in Section II. Following the computation of the analog voltage, a for loop is utilized to iterate over the boundaries of the input signal intervals, incrementing by 0.5 until reaching 4.5. Each iteration checks if the current boundary is less than the detected analog voltage. If so, it turns on the corresponding LED pin using the formula:

$$LED\ pin = 2 \times (interval\ boundary) + 3 \tag{2}$$

The for loop terminates once the interval boundary in the current iteration is less than the detected voltage, avoiding unnecessary iterations through all possible interval boundaries.

Given the specification in Table I that pin 13 also blinks at 4.9 to 5V, and the continuous incrementation of 0.5 doesn't reach exactly 4.9, an additional statement was included to address this requirement. Subsequently, after turning on the necessary LEDs, a delay is introduced to allow their observation. Following this, the LEDs are turned off in preparation for the next iteration of the loop() function.

II. Computations

In linear algebra, finding the equation of a line given two points is a fundamental concept. Utilizing the point-slope form ($y - y_1 = m(x - x_1)$) and the slope formula ($m = \frac{y_2 - y_1}{x_2 - x_1}$), the following equation can be derived:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \tag{3}$$

Considering analogVolt as y and digitalVolt as x, the linear equation is formed with two points: (0, 0) and (1023, 5). Substituting these into eq. 3 yields the formula in eq. 1:

$$\begin{aligned} analogVolt - 0 &= \frac{5 - 0}{1023 - 0}(digitalVolt - 0) \\ analogVolt &= \frac{5}{1023} \times digitalvolt \end{aligned} \tag{4}$$

However, since both digital and analog voltages start at zero, eliminating the need for a y-intercept, a simpler method is to solve the problem as a ratio and proportion:

$$\begin{aligned} \frac{analogVolt}{5} &= \frac{digitalVolt}{1023} \\ analogVolt &= \frac{5}{1023} \times digitalvolt \end{aligned} \tag{5}$$

The linear algebra concept, on the other hand, becomes useful in deriving eq. 2 to determine the LED pin number (4 to 13) from interval boundaries 0.5 to 4.5. With the points (0.5, 4) and (4.5, 12), substitution into eq. 3 yields:

$$\begin{aligned} LED\ pin - 4 &= \frac{12 - 4}{4.5 - 0.5}(interval\ boundary - 0.5) \\ LED\ pin - 4 &= \frac{2}{1}(interval\ boundary) - 1 \\ LED\ pin &= 2 \times interval\ boundary + 3 \end{aligned} \tag{6}$$

III. Arduino Setup

As seen in 1, the Arduino setup involves an Arduino UNO microcontroller with an LED shield mounted on top of it, plugged with the appropriate pin connections. The potentiometers left pin is connected to 5V while its right pin to GND. Lastly, the middle pin is connected to A0. A USB cord was also used to power and upload the program with a laptop.

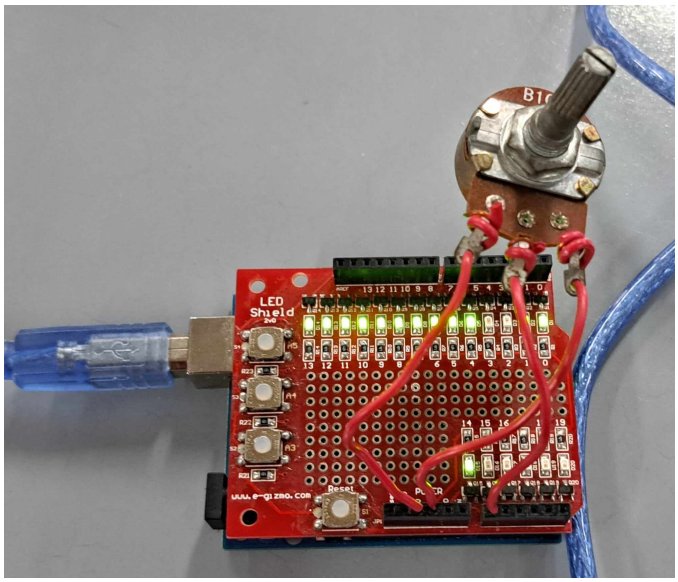

Fig. 1. Arduino Setup with LED Shield and Potentiometer