ACTIVITY 1 - Arduino 1: DEC to BIN

Jason Harvey Lorenzo, February 2, 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 papers explores a project using Arduino UNO and an LED shield to display binary representations of integers from 0 to 1023. The Arduino code efficiently converts decimals to binary using an array of the first ten powers of 2, subtraction and comparison operators. The project demonstrates a display of binary patterns on the LED shield, providing a practical understanding of Arduino programming concepts such as digitalWrite(), for loops, and if-else statements.**

***Index Terms*—Arduino UNO, LED Shield, Decimal-to-Binary**

I. Program Listing

The Arduino code is designed to use an Arduino UNO microcontroller in conjunction with an LED shield to generate and display binary representations of decimal numbers from 0 to 1023. Each LED on the shield corresponds to a binary bit, allowing the user to visually observe the binary representation of the internally generated decimal numbers.

A. Source Code

```
int bit_level[] = {512, 256, 128,
64, 32, 16, 8, 4, 2, 1};

void setup() {
  for(int i=2; i<=11; i++)
    pinMode(i, OUTPUT);
}

void dectobin(int dec){
  for(int i=0; i<10; i++){
    if(dec==0){
      digitalWrite(i, LOW);
    }
    else if(dec>=bit_level[i]){
      digitalWrite(11-i, HIGH);
      dec-=bit_level[i];
    }
    else{
      digitalWrite(11-i, LOW);
    }
  }
}

void loop() {
  for (int i=0; i<=1023; i++){
    dectobin(i);
    delay(50);
  }
}
```

B. How It Works

The setup() function is responsible for initializing the pins (and LEDs) to be used by the LED shield. In this case, pins 2 to 11 are configured as outputs, enabling the control of the corresponding LEDs for binary representation.

The core of the code lies in the dectobin() function, which efficiently converts decimal values to binary and controls the LED display. The traditional method involves repeated division by 2, tracking remainders to build the binary representation. However, this code optimizes the process by leveraging the fact that the maximum decimal value is 1023.

The function uses an array called bit_level to store powers of 2 (512, 256, 128, ..., 1), representing each bit's place value in binary. This eliminates the need for recalculating powers of 2 in the loop, trading some memory for processing efficiency.

The for loop iterates through each bit, determining whether the LED corresponding to that bit should be turned on or off based on the provided decimal input. Starting with the most significant bit (10th bit), the code checks if the decimal value is greater than or equal to the stored power of 2. If true, it turns on the corresponding LED and subtracts the power of 2 from the decimal value, otherwise it turns the LED off without subtracting. This process continues for each bit until the entire binary representation is displayed. The initial if statement ensures that any subsequent LEDs are turned off even if the decimal value reaches zero before the last bit.

Let's take $dec = 640$ as an example. Here's how the function will work:

- Since the decimal value $dec = 640$ is greater than 512, the 10th LED is turned on and 512 is subtracted to 640 which results to 128. dec is then set to be 128 ($dec = 128$).
- Since 128 is less than 256, the 9th LED is turned off and the loop proceeds to the next bit_level.
- Since 128 is equal to 128, the 8th LED is turned on and subtracting 128 from dec results to it becoming zero ($dec = 0$).
- Since $dec = 0$ for the rest of the iterations, all other LEDs will be turned off by the first if statement.

The loop() function continuously cycles through decimal values from 0 to 1023 (effectively generating the integers),

invoking the dectobin() function to update and display the binary representation on the LEDs. A brief delay of 50 milliseconds ensures a visible and comprehensible transition between numbers.

II. Arduino Setup

The Arduino setup involves an Arduino UNO microcontroller with an LED shield mounted on top of it, plugged with the appropriate pin connections. A USB cord was also used to power and upload the program with a laptop.

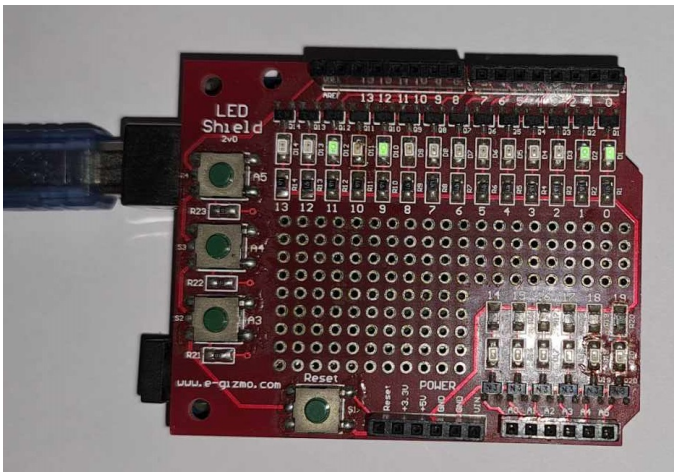

Fig. 1. Arduino Setup (Top View)

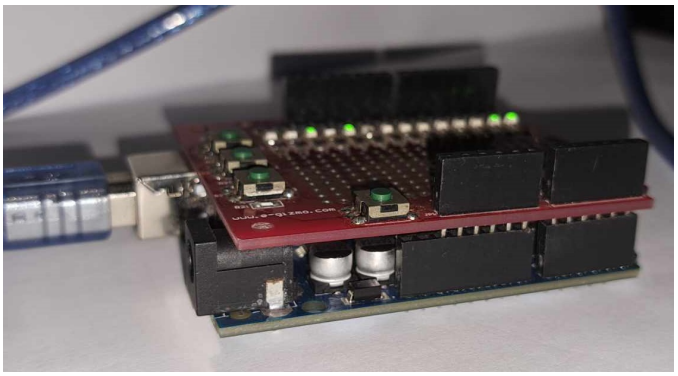

Fig. 2. Arduino Setup (Side View)

Fig. 1 shows the setup in top view, while Fig. 2 shows the side view.