ACTIVITY 2 - Arduino 2: T-Bird

Jason Harvey Lorenzo, February 16, 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 presents an Arduino UNO-based implementation of a headlight pattern inspired by the Thunderbird (T-bird) car. The pattern involves cascading LED effects and a 'braking' feature. The Arduino setup uses an LED shield with buttons A, B, and C controlling the LED 'headlights'. The cascading effect is achieved by turning on and then off individual LEDs, creating a center-to-outermost cascading effect. Button A triggers the 'braking' effect, turning on all LEDs. Buttons B and C control the left and right headlight cascading effects. The implementation uses an array to manage LED assignments and functions to control the LED patterns, providing a flexible and responsive system for controlling the LED headlight effects.**

***Index Terms*—Arduino UNO, LED Shield, T-bird headlight pattern**

I. Program Listing

This activity draws inspiration from the Thunderbird (T-bird) car headlight pattern, with the objective of implementing a similar pattern on an LED shield using an Arduino UNO. The LED shield features three built-in buttons (A, B, and C) that control the 'headlights' to follow the desired pattern. The pattern involves four LEDs all turning on and then turning off one-by-one, creating a cascading effect from the center towards the outermost LED.

When button B is pressed, the left set of LEDs will cascade, and when button C is pressed, the right set of LEDs will cascade. These cascading functions take priority over other operations. When button A is pressed, all LEDs will turn on, mimicking the effect of braking. However, if either button B or C is pressed simultaneously with button A, the corresponding cascading effect will override the 'braking' function.

A. Source Code

```
const int left[4] = {10, 11, 12, 13};
const int right[4] = {9, 8, 7, 6};
const int delayT = 150;

void setup() {
  for(int i=17; i<=19; i++)
    pinMode(i, INPUT_PULLUP);
  for(int i=6; i<=13; i++)
    pinMode(i, OUTPUT);
}

void allOn(int * lights){
  for(int i=0; i<=3; i++)
    digitalWrite(lights[i], HIGH);
}

void allOff(int * lights){
  for(int i=0; i<=3; i++)
    digitalWrite(lights[i], LOW);
}

void cascade(int * lights){
  allOn(lights);
  for(int i=0; i<=3; i++){
    delay(delayT);
    digitalWrite(lights[i], LOW);
  }
  delay(delayT);
}

void cascadeBoth(int * lights1, int * lights2){
  allOn(lights1);
  allOn(lights2);
  for(int i=0; i<=3; i++){
    delay(delayT);
    digitalWrite(lights1[i], LOW);
    digitalWrite(lights2[i], LOW);
  }
  delay(delayT);
}

void loop() {
  bool A = digitalRead(19);
  bool B = digitalRead(18);
  bool C = digitalRead(17);

  if(A==1){
    if(B==1){
      if(C==1){
        allOff(left);
        allOff(right);
      }
      else{
        allOff(left);
        cascade(right);
      }
    }
    else{
      if(C==1){
```

```
            cascade(left);
            allOff(right);
        }
        else{
            cascadeBoth(left, right);
        }
    }
  }
  else
    if(B==1){
      if(C==1){
        allOn(left);
        allOn(right);
      }
      else{
        allOn(left);
        cascade(right);
      }
    }
    else{
      if(C==1){
        cascade(left);
        allOn(right);
      }
      else{
        cascadeBoth(left, right);
      }
    }
}
```

B. How It Works

In the setup() function, pins 17, 18, and 19 are assigned as input buttons C, B, and A, respectively. To prevent high impedance in the input and enable negative logic, these pins are declared as INPUT_PULLUP. LEDs 6 to 9 constitute the right headlight, while LEDs 10 to 13 make up the left headlight; both sets are declared as outputs. To efficiently manage these values, an array is created to store the LED assignments for each headlight.

These arrays can be inputted as parameters for the following functions:

- allOn(): This function turns on the LEDs called on by the array inputted.
- allOff(): This function turns off the LEDs called on by the array inputted.
- cascade(): This function first turns on (via allOn()) the LEDs called on by the array inputted and turns them off one-by-one, after a short delay, from the center to the outermost LED.
- cascadeBoth(): This function allows for the same functionality of the cascade() function but for both sets of arrays inputted to cascade at the same time.

Within the loop() function, the digitalRead() function is used to read the inputs from buttons A, B, and C. A series of if-else statements then process the commands based on their respective states (e.g., 111, 110, etc.) and execute the corresponding pattern. This is achieved by calling the appropriate function and passing the relevant headlight array as an argument, which enables the selected headlight to perform the desired action.

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.

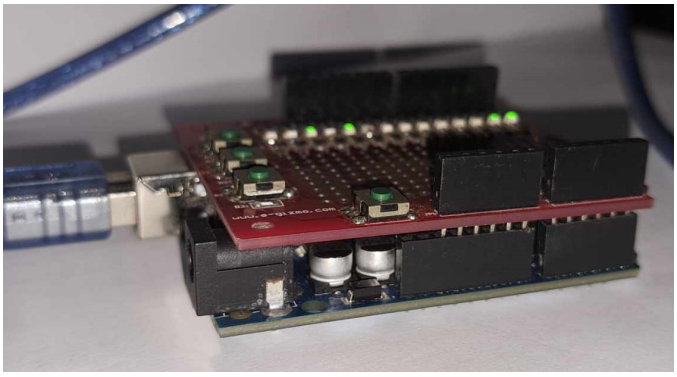

Fig. 1. Arduino Setup (Side View)

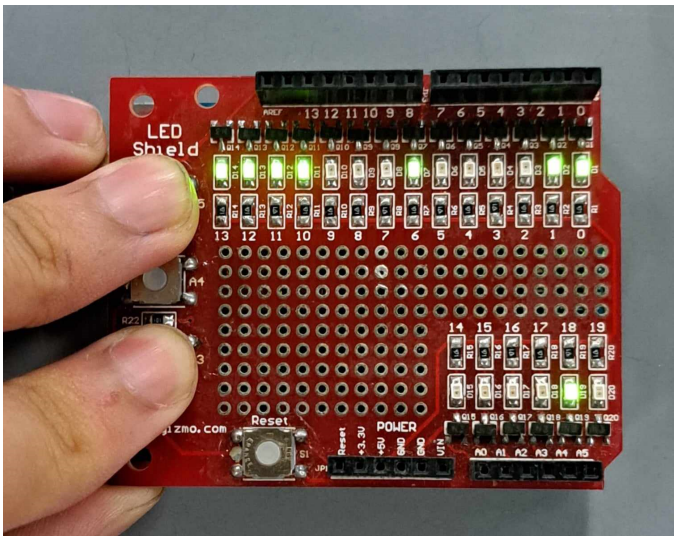

Fig. 2. T-bird Headlights Display with Input 010

Fig. 1 shows the Arduino setup in side view, while Fig. 2 shows a snapshot of the project working with user input 010 for buttons ABC.