ACTIVITY 8 - Arduino 8: 7-Segment Display - Counter

Jason Harvey Lorenzo, April 11, 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*—The project revolves around constructing a basic counter using an Arduino microcontroller and a multifunction shield featuring three seven-segment displays. The goal is to create an up-down counter capable of cycling from 0 to 101 and vice versa, resetting to the opposite limit upon reaching either 101 or 0. Interrupt functionality of the Arduino facilitates pausing and resuming counting, as well as toggling between counting up and down modes. A discussion of the 7-segment display section of the multifunction shield, showcasing efficient multiplexing for display control was done. Pin configurations and hardware setup are clarified, demonstrating the straightforward integration of the shield with the Arduino UNO. The program listing outlines the code implementation, utilizing an array to manage LED representations and boolean variables to regulate counting behavior. The discussion highlights the efficiency of the shield's 7-segment LED design and the software's practicality, while encouraging exploration of alternative approaches for improved efficiency and interactivity.**

***Index Terms*—Arduino, seven-segment display, interrupt function**

I. Introduction

This project entails constructing a basic counter utilizing an Arduino microcontroller and a multifunction shield equipped with three seven-segment displays. The objective is to develop an up-down counter capable of counting from 0 to 101 and vice versa, resetting to the opposite limit upon reaching either 101 or 0. Utilizing the Arduino's interrupt functionality, one interrupt function will enable pausing and resuming counting, while another will facilitate toggling between counting up and counting down modes.

II. Design Process and Discussion

Fig. 1 shows the schematic diagram of the 7-segment display section of the multifunction shield used in this activity.

As seen, each of the three 7-segment displays receives input from a common set of 7 wires, (+1 for each decimal point). However, the displays are controlled by transistors functioning as switches, effectively serving as a multiplexer to determine the active display at any given moment. This setup optimizes connectivity while maintaining the device's operational capabilities.

A. Arduino Setup

The hardware setup for this activity is straightforward. The multifunction shield is positioned directly on top of the Arduino UNO board. Power is supplied to the Arduino UNO by connecting it to a laptop using a USB cable for both power and programming. Refer to Fig. 2 for a visual representation of the Arduino setup.

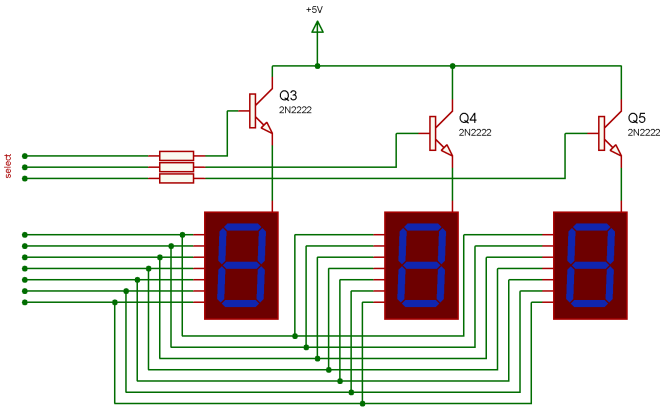

Fig. 1. 7-segment display schematic diagram of multifunction shield

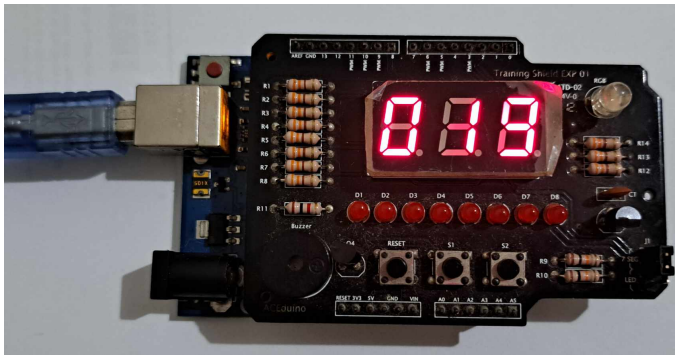

Fig. 2. Arduino setup using 7-segment displays of a multifunction shield

It's worth noting that the 7-segment display segment is grounded, as evident from the lower-left section of Fig. 2, to enable its functionality.

Pin assignments for controlling the 7-segment displays are as follows: Pin 7 controls the rightmost display, pin 8, the center one, and pin 10, the leftmost display. Pins 11 to 18 correspond to LEDs A to G, respectively, for all displays.

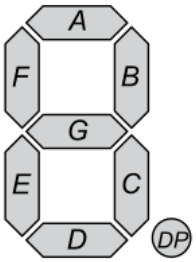

Fig. 3. 7-segment display LED configuration

Refer to Fig. 3 for a visual representation of the labeling of these LEDs on the 7-segment displays.

B. Program Listing

Listing 1 shows the code implementation for the project.

Listing 1. Arduino Counter Code

```
int digits[10][8]={
// a  b  c  d  e  f  g  .
  {1, 1, 1, 1, 1, 1, 0, 0}, // 0
  {0, 1, 1, 0, 0, 0, 0, 0}, // 1
  {1, 1, 0, 1, 1, 0, 1, 0}, // 2
  {1, 1, 1, 1, 0, 0, 1, 0}, // 3
  {0, 1, 1, 0, 0, 1, 1, 0}, // 4
  {1, 0, 1, 1, 0, 1, 1, 0}, // 5
  {1, 0, 1, 1, 1, 1, 1, 0}, // 6
  {1, 1, 1, 0, 0, 0, 0, 0}, // 7
  {1, 1, 1, 1, 1, 1, 1, 0}, // 8
  {1, 1, 1, 1, 0, 1, 1, 0}  // 9
};

int current_count = 0;
String count_as_string;

int place_val_pins[3] = {7, 8, 10};

int prev_place_val = 7;

volatile bool count_up = 1;

volatile bool count_paused = 0;

float initial_time, time_duration;

void setup() {
  for(int i=7;i<=18;i++)
    pinMode(i,OUTPUT);
  for(int i=2;i<=3;i++)
    pinMode(i,OUTPUT);
  attachInterrupt(
    digitalPinToInterrupt(2),
    pause_resume, RISING);
  attachInterrupt(
    digitalPinToInterrupt(3),
    switch_direction, RISING);
}

void switch_direction(){
  noInterrupts();  // ignore debounce
  count_up = !count_up;
}

void pause_resume(){
  noInterrupts();  /// ignore debounce
  count_paused = !count_paused;
}

void writeOn(int new_place_val){
  digitalWrite(prev_place_val, LOW);
  digitalWrite(new_place_val, HIGH);
  prev_place_val = new_place_val;
}

void displayCount(){
  initial_time = millis();
  time_duration = initial_time;
  while(time_duration-initial_time <200){
    // writes count on 7-segment display
    for(int i=0; i<=2; i++){
      writeOn(place_val_pins[i]);
      //iterate for each display(7, 8, 10)
      for(int j=0; j<=7; j++){
        digitalWrite(11+j,
        digits[count_as_string[3-i]-
        '0'][j]);
      }
      delay(5);
      time_duration = millis();
    }
  }
}

void loop() {
  if(count_paused){
    displayCount();
    return;
    // loop again without doing code below
  }

  count_as_string =
    String(1000+current_count);
  displayCount();

  if (count_up){
    current_count++;
    if (current_count>101)
      current_count=0;
  }
  else{
    current_count--;
```

```
    if (current_count<0)
      current_count=101;
  }
}
```

C. Discussion

An array was employed to efficiently manage the binary representation of the LEDs corresponding to each digit on the 7-segment displays. Additionally, boolean variables such as count_up and count_paused were initialized to regulate the counting process, enabling functionalities indicated by their respective names. These variables are manipulated by interrupt functions, allowing users to dynamically pause or resume counting and switch between counting up and down asynchronously.

To enable the presentation of a three-digit number on the 7-segment displays, the necessary pins (7, 8, and 10) controlling these displays, along with pins 11 to 18, were configured as output pins. Additionally, two functions were implemented to facilitate this process. The writeOn() function was designed to activate only one display at a time, allowing for selective control over which display is illuminated. Within the displayCount() function, this capability is leveraged to rapidly cycle through the displays from left to right (via pins 7, 8, and 10) with a brief delay of 5 milliseconds between each activation. Each digit, as represented from the array, is displayed one at a time. This rapid transition gives the illusion of simultaneous display updates, effectively tricking the eyes and brain into perceiving smooth transitions.

In the loop() function, the process of determining the number to be displayed is done. Initially, the count, represented as an integer, is established. To isolate its individual digits for display, it undergoes a conversion into a string format. Subsequently, each digit character extracted is processed by the displayCount() function, reverting it back to an integer to access the array containing the binary representation of the corresponding digit. Within this loop, boolean variables dictate the state of the counting mechanism. When the count is paused, the program solely displays the current count and returns to looping. Conversely, if the count is set upwards , it is incremented by one; otherwise, it is decremented.

III. Conclusion and Recommendation

The architecture of the 7-segment LED displays featured in the shield utilized in this activity offers a blend of efficiency and practicality, leveraging the perception of simultaneity induced by a high frame rate. By rapidly multiplexing the 7-segment displays, the number of connections were optimized, leading to a more efficient utilization of resources. However, this optimization comes at the expense of increased software complexity. Various approaches exist to achieve the required functionality outlined in this activity. While the solution presented in this paper may not be the most optimized, considering the speed of the microcontroller and effectivity of the rapid successive displaying of the digits, this solution suffices. Moreover, this solution prioritizes readability, comprehensibility of the program's logic, and rapid development. Nonetheless, the paper encourages exploration of alternative, potentially more efficient solutions, such as treating each digit independently and implementing carry mechanisms for counting up to 9 or borrow mechanisms for counting down to 0. Implementing solutions to mitigate or eliminate debounce from the interrupt inputs would also enhance the interactivity of the system.