CG1111 Engineering Principles and Practice I

Arduino Workshop

Challenge Activity for Week 6 Studio 2

Studio Objectives:

1. Learn to translate a schematic diagram to a breadboard prototype
2. Develop neat bread-boarding techniques
3. Develop a simple Arduino-based application

Materials:

- Breadboard, connecting wires, Arduino Uno, USB cable
- 5% tolerance resistors (12 x 220 Ω per group)
- 2 kΩ potentiometer and trimming tool (1 set per group)
- LEDs (5 per group – any colour)
- Common-Anode 7-segment LED (1 per group)

Introduction:

In Week 6's Arduino workshop you have been introduced to the Arduino Uno Microcontroller and some of its basic features. The code that you develop makes extensive use of pre-built libraries that simplify the underlying hardware interface. It is now very easy to start developing an embedded application using such development tools.

To challenge you slightly, we are now going to build a simple project that will also test your HW prototyping skills, as well as basic SW development techniques. A snapshot of the circuit is shown below in Figure 1.

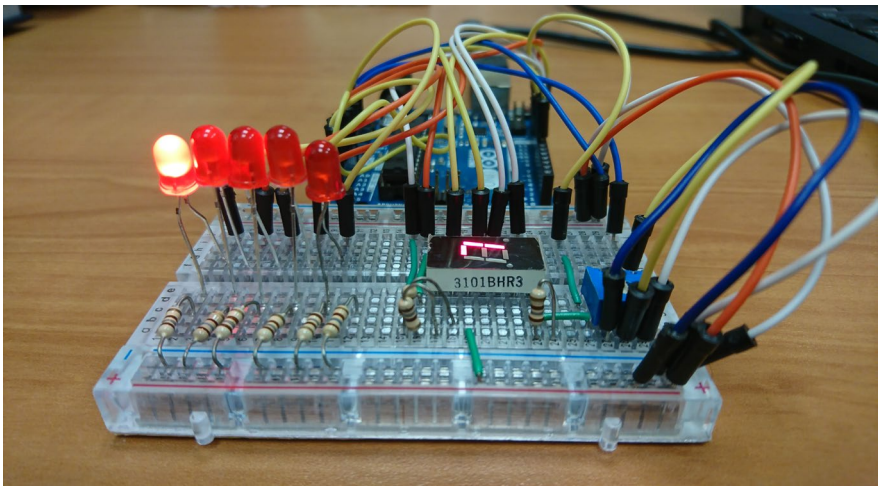

Figure 1: Prototype Layout

A video demonstration of the project can be found here https://youtu.be/5vlvMFCa6mc. We will develop this project progressively over the next three activities. Code snippets are provided to help you get started. Apply the C coding techniques that you have learned in your CS1010 to complete the project.

Activity 1: Running LED

In this activity you are going to focus on developing the HW and SW to create the LED running effect. The connections between the Uno and the LEDs are shown below in Figure 2.

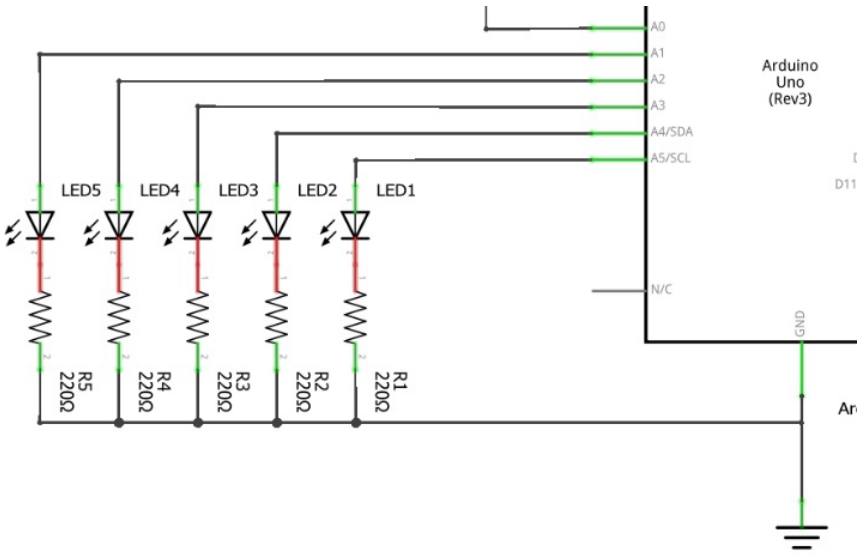

Figure 2: LED Connections to Arduino

Code Snippet:

```
void setup() {
      pinMode(A1, OUTPUT);
      …
      // Add configuration for the other pins
}
```

In your `loop()` function, you may want to implement loops together with function calls to light-up the appropriate LED. It is important to add an appropriate delay after a LED is turned ON, so as to see the desired effect.

Activity 2: Seven-Segment LED

In this activity, you are being introduced to a popular (used to be) display device, which is the seven segment display, commonly written as "7-Seg". This device is packaged with 7 LED's with TWO possible configurations, either, common-anode or common-cathode. In this activity, we are going to be using the Common-Anode (CA) type. The layout of the pins is shown in Figure 3.

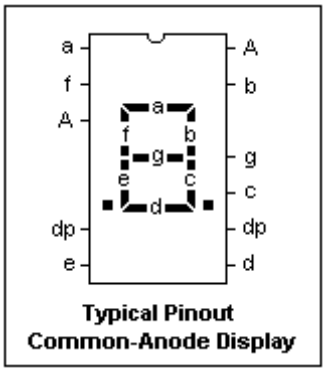

Figure 3: Common-Anode 7-Seg Display

The internal configuration of the LED's is shown in Figure 4 (without the decimal point [dp] pins). It can be seen that the terminology 'Common-Anode' means that the anode pin of the all the LED's in the display are connected together, and hence "common". This implies that in order to TURN-ON any LED, we need to supply a logic '0' or 0 Volts to that LED's cathode pin, which is connected to the Arduino Uno.

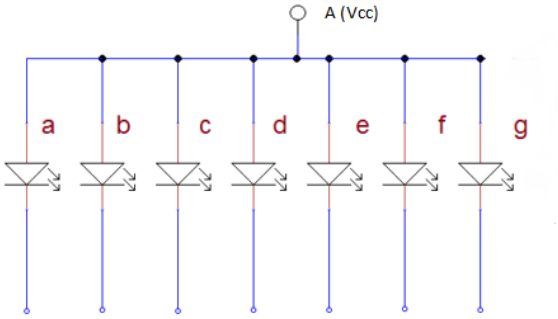

Figure 4: Common-Anode LED Configuration

Looking at Figure 3 and 4, we can see that, if we want to display a value of 2, we will need LED's a, b, g, e and d to be turned ON, which means writing a value of '0' to those pins.

NOTE: You must remember to connect the Common-Anode pin, labelled 'A' (in Figure 3) to a +5V source. This +5V source can be obtained from the Arduino board. There are two pins labelled as 'A'. Only one of these 'A' pins needs to be connected to 5 V.

Step 1: HW Design

Wire the 7-Seg to the Uno based on the circuit diagram given in Figure 5. It is important to insert a current-limiting resistor to the cathode of each LED before connecting it to the Uno. This is to prevent a high current from passing through and damaging the 7-Seg. Figure 6 shows the actual mapping of the LED's to the various pins.

You DO NOT need to follow the exact mapping shown. You need to ensure that the pins a-to-g of the 7-Seg are connected to any of the Digital I/O pins (except D0 and D1) of the Microcontroller.

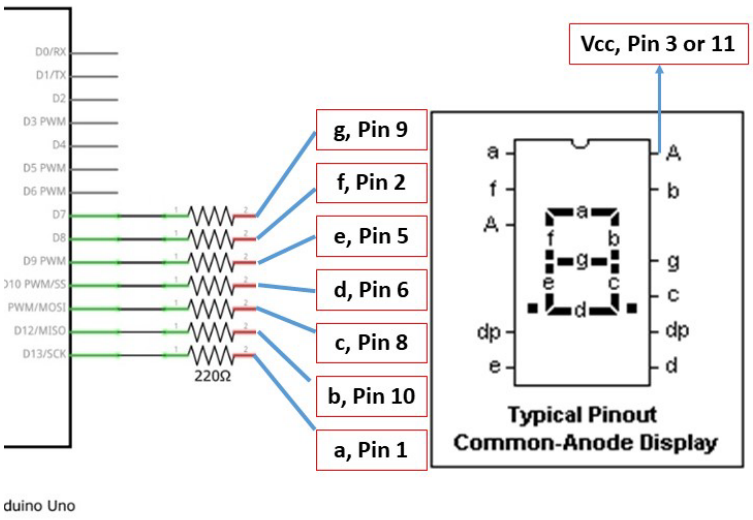

Figure 5: 7-seg Connections to the Arduino Uno (Example)

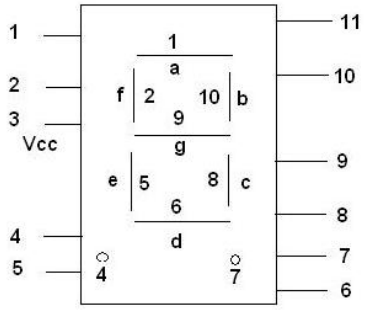

Figure 6: Mapping of LED's to the pin numbers

Step 2: SW Design

You can choose to create a new sketch program or carry on from the code in the earlier activity.

In your `setup()`, you must configure the direction of the pins as OUTPUT.

In your `loop()`, you can update the 7-Seg to display a new value every 1 sec to see that the numbers are being displayed correctly.

You can consider creating a function as shown below. The function can have one input that indicates the number to be displayed. Within the function, you can use either if-else statements or switch-case statements to set the appropriate values for the various LED's.

```
void sseg_display(char num)
{
  switch (num)
  {
    case 1:
      digitalWrite(…);
      // Logic values for other LED's.
      break;
}
```

Your program needs to count and display values from 1 to 9 and then roll-over back to 1 to start counting again.

IMPORTANT:

Refer to Figure 3 for the correct mapping of the 7-Seg pins to their appropriate LED's.

Activity 3: Potentiometer & ADC

In this activity, we are going to interface the potentiometer to the Uno and obtain the potentiometer value based on its voltage. The circuit is shown in Figure 7.

It can be seen that both ends of the potentiometer are connected to +5V and 0V. The middle pin will give us a voltage that is proportional to the potentiometer setting.

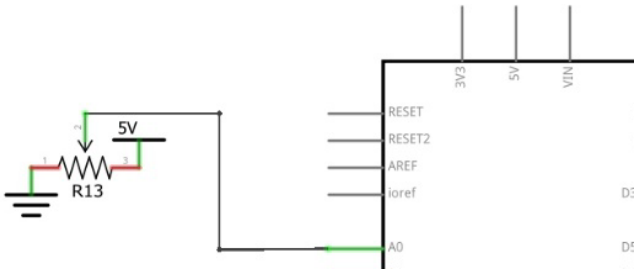

Figure 7: Potentiometer connection to the Arduino Uno

Pin A0 is one of the pins that have built-in ADC capability. The ADC value can be obtained by using the code shown below:

```
int adc_val;

adc_val = analogRead(A0);
```

You can observe the values obtained from the ADC by using a Serial.Print().

In setup(), include this line

```
Serial.begin(9600);
```

In loop(), include this line:

```
Serial.println(adc_val);
```

In your Arduino development tool, select Tools -> Serial Monitor to observe the data being sent through the serial terminal.

To observe the effect of the change in the potentiometer value, you can use this ADC result to change the rate at which the 7-Seg updates its display, in Activity 2.

Activity 4: Final Integration

In this final stage you are going to integrate the HW and SW from the earlier activities. The final circuit is shown below in Figure 8.

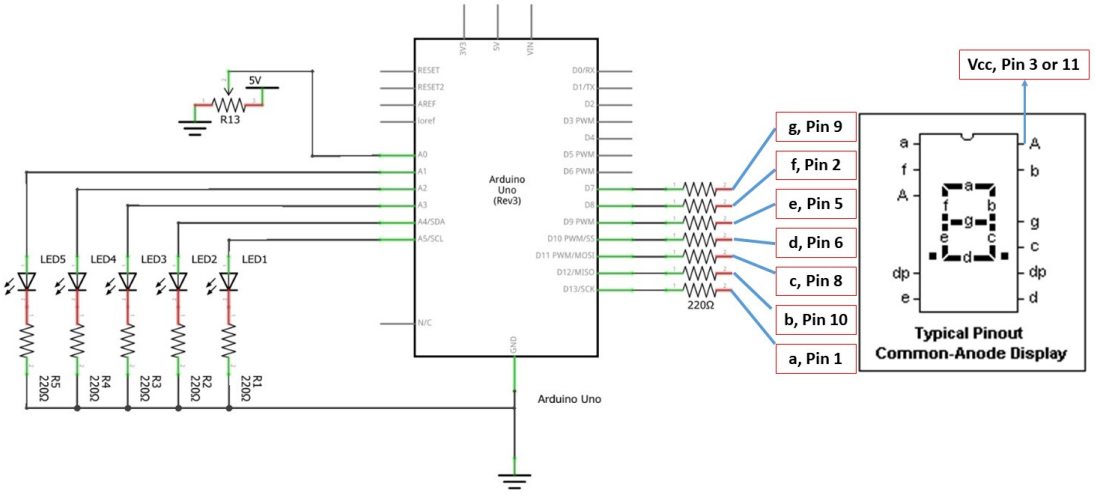

Figure 8: Final Layout

In your `loop()`, you must update the 7-Seg display each time the Running LED completes one cycle (Right->Left->Right). This can be accomplished using your own function. At the same time, the rate at which the Running LED's change is determined by the Potentiometer setting.

OBSERVATION POINT:

You will notice that the ADC value is in the range of 0 to 1023. Using those values directly as the delay's value may not be suitable; the delay will be too small or too large. Think of how you can rescale the values to a range that is more suitable.

Demonstrate your final HW and SW to your Tutor. If the SW works as expected and the HW is neatly arranged, you stand to win a small token for your effort. ☺

THE END