Su22-ENGR-40M-01 Lab 3a

Jannah Sabic El-Rayess

TOTAL POINTS

70 / 70

QUESTION 1

1 Lab Complete 15 / 15

✓ **- 0 pts Correct**

- 9 pts 15% penalty for late submission

QUESTION 2

2 L1 Code 5 / 5

✓ **- 0 pts Correct**

- 0.5 pts Didn't deactivate anode or cathode pins

QUESTION 3

3 L2 Code 5 / 5

✓ **- 0 pts Correct**

- 0.5 pts Pattern condition case reversed

- 1 pts error in code

QUESTION 4

Analysis 10 pts

4.1 A1 5 / 5

✓ **- 0 pts Correct**

- 5 pts Incomplete Answer

- 2 pts Slightly Incorrect

- 3 pts Attempted answer but incorrect

- 1 pts Need to explicitly say change time proportional to amount of total time it could be on

4.2 A2 5 / 5

✓ **- 0 pts Correct**

- 2 pts Should be delay relative to other LEDS. The relative part is critical

- 2 pts Code doesn't set LEDs based on total time LED could be on (i.e. uses delay to change brightness)

- 5 pts Incorrect / Not attempted

- 3 pts Incomplete

- 0.2 pts Minor error (see comments)

- 1 pts Code incomplete or vague explanation

QUESTION 5

Style 30 pts

5.1 Soldering 15 / 15

+ 5 pts plus

✓ **- 0 pts check plus**

- 5 pts check

- 10 pts minus

5.2 Build Quality 15 / 15

+ 5 pts plus

✓ **- 0 pts check plus**

- 5 pts check

- 10 pts minus

QUESTION 6

6 Cleanup 5 / 5

✓ **- 0 pts Correct**

1 Lab Complete **15 / 15**

✓ **- 0 pts** **Correct**

- 9 pts 15% penalty for late submission

```
// everylight
/* Starter code to blink every LED using the simplest possible iteration
 * through anode and cathode pins.
 *
 * ENGR 40M
 * July 2018
 */

// Define arrays for the anode (+) and cathode (-) wire pins.
// Your pins will probably be different.
// Remember that analog pins (A0, A1, ...) can also act as digital.
const byte ANODE_PINS[8] = {13, 12, 11, 10, 9, 8, 7, 6};
const byte CATHODE_PINS[8] = {A3, A2, A1, A0, 5, 4, 3, 2};

void setup() {
  // In this function, you need to do two things:
  //  1. Configure all 8 anode (+) and all 8 cathode (-) pins to outputs
  //  2. Turn all 16 pins "off" (does this mean HIGH or LOW?)

  // Here's part 1, as an example (you can use this):
  for (byte i = 0; i < 8; i++) {
    pinMode(ANODE_PINS[i], OUTPUT);
    pinMode(CATHODE_PINS[i], OUTPUT);
  }

  // TODO: Do part 2 (turn all 16 pins off).
  // (Hint: a 'for' loop to loop through the arrays above might help,
  //  or you could even add to the same loop above.)
  for (byte i = 0; i < 8; i++) {
    digitalWrite(ANODE_PINS[i], HIGH);
    digitalWrite(CATHODE_PINS[i], HIGH);
  }

}

void loop() {
  // TODO: In this function you need to write code that flashes each LED one by one.
  // Hint: Here's some suggested pseudocode:
  //   for each anode (+) wire
  //       activate anode (+) wire (does this mean HIGH or LOW?)
  //       for each cathode (-) wire
  //           activate cathode (-) wire (does this mean HIGH or LOW?)
  //           wait a bit
  //           deactivate cathode (-) wire
  //       end for
  //       deactivate anode (+) wire
  //   end for
```

```
  for (byte i = 0; i < 8; i++) {
    digitalWrite(ANODE_PINS[i], LOW);
    for (byte j = 0; j < 8; j++) {
      digitalWrite(CATHODE_PINS[j], LOW);
      delay(100);
      digitalWrite(CATHODE_PINS[j], HIGH);
    }
    digitalWrite(ANODE_PINS[i], HIGH);
  }
}
```

2 L1 Code 5 / 5

✓ **- 0 pts** **Correct**

- 0.5 pts Didn't deactivate anode or cathode pins

```
//display
/* LED array test code
 *
 * Reads (x,y) co-ordinates from the Serial Monitor and toggles the state of
 * the LED at that co-ordinate. The co-ordinates are specified as "x y", e.g.
 * "1 2", followed by a newline. Invalid co-ordinates are rejected.
 *
 * You need to fill in all the places marked TODO.
 *
 * == Setting up the Serial Monitor ==
 * The Serial Monitor must be configured (bottom-right corner of the screen) as:
 *   - Newline (for the line ending)
 *   - Baud rate 115200
 *
 * ENGR 40M
 * July 2018
 */

// Arrays of pin numbers. Fill these in with the pins to which you connected
// your anode (+) wires and cathode (-) wires.
const byte ANODE_PINS[8] = {13, 12, 11, 10, 9, 8, 7, 6};
const byte CATHODE_PINS[8] = {A3, A2, A1, A0, 5, 4, 3, 2};

void setup() {
  // TODO: configure all anode (+) and cathode (-) wires to outputs
  // TODO: turn "off" all the LEDs
  // Hint: You did the same thing in everylight.ino.

  // Here's part 1, as an example (you can use this):
  for (byte i = 0; i < 8; i++) {
    pinMode(ANODE_PINS[i], OUTPUT);
    pinMode(CATHODE_PINS[i], OUTPUT);
  }

  // TODO: Do part 2 (turn all 16 pins off).
  // (Hint: a 'for' loop to loop through the arrays above might help,
  //  or you could even add to the same loop above.)
  for (byte i = 0; i < 8; i++) {
    digitalWrite(ANODE_PINS[i], HIGH);
    digitalWrite(CATHODE_PINS[i], HIGH);
  }

  // Initialize serial communication
  // (to be read by Serial Monitor on your computer)
  Serial.begin(115200);
  Serial.setTimeout(100);
}
```

```
/* Function: display
 * -----------------
 * Runs through one multiplexing cycle of the LEDs, controlling which LEDs are
 * on.
 *
 * During this function, LEDs that should be on will be turned on momentarily,
 * one row at a time. When this function returns, all the LEDs will be off
 * again, so it needs to be called continuously for LEDs to be on.
 */
void display(byte pattern[8][8]) {
  // TODO: You need to fill in this function.
  // Here's some suggested pseudocode:
  //   for each anode (+/row) wire
  //     for each cathode (-/column) wire
  //       look up in pattern whether this LED should be on or off
  //       if LED should be on, activate cathode (-) wire, else deactivate it
  //     end for
  //     activate anode (+) wire
  //     wait a short time (hint: try delayMicroseconds())
  //     deactivate anode (+) wire
  //   end for
  for (byte i = 0; i < 8; i++) {
    for (byte j = 0; j < 8; j++) {
      if (pattern[i][j] == 1) {
        digitalWrite(CATHODE_PINS[j], LOW);
      }
      else {
        digitalWrite(CATHODE_PINS[j], HIGH);
      }
    }
    digitalWrite(ANODE_PINS[i], LOW);
    delayMicroseconds(50);
    digitalWrite(ANODE_PINS[i], HIGH);
  }
}

void loop() {
  // You shouldn't need to edit this function during lab 3a.

  // use 'static' so that it retains its value between successive calls of loop()
  static byte ledOn[8][8];

  byte x = 0;
  byte y = 0;
  static char message[60];
```

```
  if (Serial.available()) {
    // Parse the values from the serial string
    x = Serial.parseInt();
    y = Serial.parseInt();

    // Check for input validity
    if (Serial.read() != '\n') {
      Serial.println("invalid input - check that line ending is set to \"Newline\"; input must
be two numbers");
      return;
    }
    if (x < 0 || x > 7 || y < 0 || y > 7) {
      sprintf(message, "x = %d, y = %d -- indices out of bounds", x, y);
      Serial.println(message);
      return;
    }

    // Print to Serial Monitor to give feedback about input
    sprintf(message, "x = %d, y = %d", x, y);
    Serial.println(message);

    // Toggle the LED state
    ledOn[x][y] = !ledOn[x][y];
  }

  // This function gets called every loop
  display(ledOn);
}
```

3 L2 Code 5 / 5

✓ **- 0 pts** **Correct**

- 0.5 pts Pattern condition case reversed

- 1 pts error in code

4 Analysis

The display function that you created in the lab has only two states for each LED: on or off. It would be nice if we could store brightness values in the LED pattern, and have the code create lights of different intensity. Sixteen brightness levels would probably be good enough for most displays.

We could change the brightness by varying the current, but it's difficult to change the current flowing through the diodes. There's only a small range of current for which the LEDs will turn on without blowing out. We'll have to vary something else to set the brightness.

A1: Without changing any hardware, what property could you change to set the brightness of each LED? (Think about what determines the relative LED brightness.)

Depending on the desired brightness, each LED would be on for a certain amount of time relative to the entire cycle of turning all the LEDs on. If I wanted a LED to be half as bright as the brighest it can be, I would have it be on for half of the total time. Ex: turning on an LED for 8 secs/16 secs will be half as bright as an LED that was on for 16 secs/16 secs.

A2: Write pseudocode for the `display()` function, showing how you would modify it for variable brightness. You should start with the code you've already written. Assume that `pattern[8][8]` returns a number between 0 (off) and 15 (fully on) representing the brightness for the LED specified.

There are a number of easy of mistakes to make on this problem. One common mistake is to set a different delay time for each cathode wire- think about why this doesn't directly change relative brightness. If you want to be sure your method works, you should try it on your array!

```
for (k=0; k<16; k++) {
 for (byte i = 0; i < 8; i++) {
   for (byte j = 0; j < 8; j++) {
     if (pattern[i][j]  > k) {
       digitalWrite(CATHODE_PINS[j], LOW);
     }
     else {
       digitalWrite(CATHODE_PINS[j], HIGH);
     }
   }
   digitalWrite(ANODE_PINS[i], LOW);
   delayMicroseconds(50);
   digitalWrite(ANODE_PINS[i], HIGH);
 }
}
```

4.1 A1 5 / 5

✓ **- 0 pts** **Correct**

- 5 pts Incomplete Answer

- 2 pts Slightly Incorrect

- 3 pts Attempted answer but incorrect

- 1 pts Need to explicitly say change time proportional to amount of total time it could be on

4 Analysis

The display function that you created in the lab has only two states for each LED: on or off. It would be nice if we could store brightness values in the LED pattern, and have the code create lights of different intensity. Sixteen brightness levels would probably be good enough for most displays.

We could change the brightness by varying the current, but it's difficult to change the current flowing through the diodes. There's only a small range of current for which the LEDs will turn on without blowing out. We'll have to vary something else to set the brightness.

A1: Without changing any hardware, what property could you change to set the brightness of each LED? (Think about what determines the relative LED brightness.)

Depending on the desired brightness, each LED would be on for a certain amount of time relative to the entire cycle of turning all the LEDs on. If I wanted a LED to be half as bright as the brighest it can be, I would have it be on for half of the total time. Ex: turning on an LED for 8 secs/16 secs will be half as bright as an LED that was on for 16 secs/16 secs.

A2: Write pseudocode for the `display()` function, showing how you would modify it for variable brightness. You should start with the code you've already written. Assume that `pattern[8][8]` returns a number between 0 (off) and 15 (fully on) representing the brightness for the LED specified.

There are a number of easy of mistakes to make on this problem. One common mistake is to set a different delay time for each cathode wire- think about why this doesn't directly change relative brightness. If you want to be sure your method works, you should try it on your array!

```
for (k=0; k<16; k++) {
 for (byte i = 0; i < 8; i++) {
   for (byte j = 0; j < 8; j++) {
     if (pattern[i][j]  > k) {
       digitalWrite(CATHODE_PINS[j], LOW);
     }
     else {
       digitalWrite(CATHODE_PINS[j], HIGH);
     }
   }
   digitalWrite(ANODE_PINS[i], LOW);
   delayMicroseconds(50);
   digitalWrite(ANODE_PINS[i], HIGH);
 }
}
```

4.2 A2 5 / 5

✓ **- 0 pts** **Correct**

- 2 pts Should be delay relative to other LEDS. The relative part is critical

- 2 pts Code doesn't set LEDs based on total time LED could be on (i.e. uses delay to change brightness)

- 5 pts Incorrect / Not attempted

- 3 pts Incomplete

- 0.2 pts Minor error (see comments)

- 1 pts Code incomplete or vague explanation

5.1 Soldering 15 / 15

+ **5 pts** plus

✓ **- 0 pts check plus**

- 5 pts check

- 10 pts minus

5.2 Build Quality 15 / 15

+ **5 pts** plus

✓ **- 0 pts check plus**

- 5 pts check

- 10 pts minus

6 Cleanup 5 / 5

✓ - 0 pts Correct