CS 241 Lab 06
(LED Matrix)

Benjamin Stream
Solomon Himelbloom

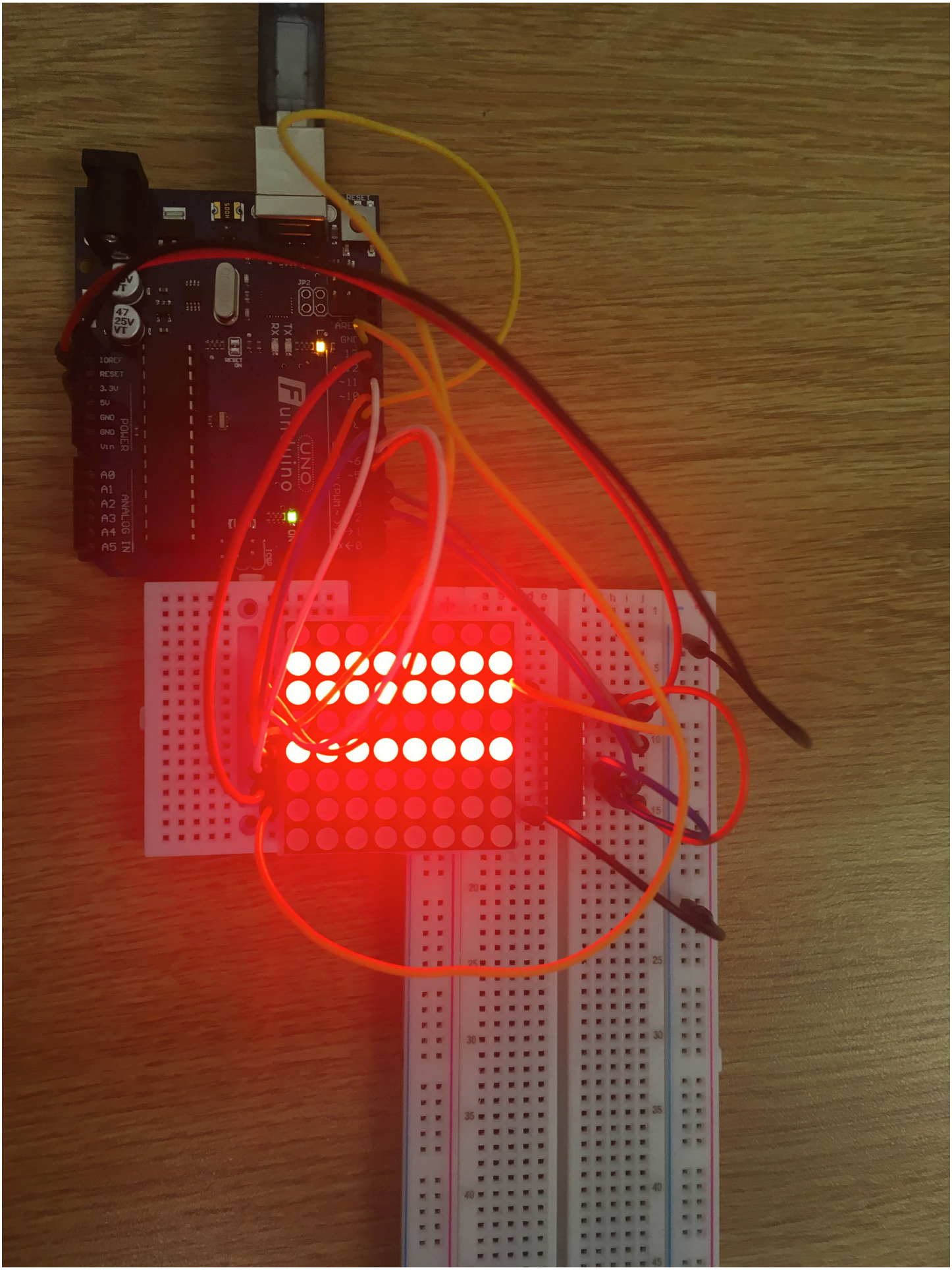
February 22, 2021

1 Answers to Questions

- *What does your final smiley face look like? (Include a photo.)*
- See the attached image. We were not able to get the exact smiley face requested but instead experimented with logging a variety of shapes (lines, boxes, etc.) to the LED matrix.
- *How similar is the photo of the display to how it looks to the human eye? Why is that?*
- A photo displayed is different from what is visible from the naked human eye. This effect is because our code and the LED combination can give the unique appearance that the lights are static (see the "persistence of vision in circuits" lecture).
- *How many microseconds does it take the Arduino to display one of your static patterns? (You can use the benchmark function from the last lab, or call micros() before and after displaying a pattern.)*
- After implementing the micros function to record at the start of our pattern-changing loop, we found that it took

299520 microseconds (the difference between trials) to display one of our static patterns. Note that we used a delay of 1000 milliseconds under each loop to step through each adjustment of the pattern-making process.

- *Timing glitches can cause some dim red fuzz around your images. If you wanted to reduce this fuzz...*
- *...could changing how you access the shift register clock pins help? Why or why not?*
- This process would not help because you are only using two pins. Since data needs to be collected and a delay for data to input into the register, changes concerning the clock pins would not change the efficiency.
- *...could direct PORT access during the shift register write or Arduino pin write help? Why or why not?*
- This could help, as it allows for us to change the pins directly connected to the LED matrix, thus tightening up any unnecessary time delay. To accomplish this, we would need to join the array to a direct port access string, similar to what we did in a previous lab.
- *Include your smiley face Arduino code in the write-up. (As usual, clarity and comments count!)*
- See the attached code snippet.


IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
UNO
47
25V
VT
TX
RX
RESET
ON

2 Appendix

2.1 Source Code

```
/// These constants give the bit numbers for each row and column.
///  (Mostly to give nice names for row/column "rc" values.)
const int dataPin = 2, clockPin = 3;
typedef enum {
  c0 = 0, c1, c2, c3, c4, c5, c6, c7,
  r0 = 8, r1, r2, r3, r4, r5, r6, r7,
} rc_t;

/// The shift register's output pins are hardwired to the 8x8's top pins
//  This table lets you look up the row/col bit numbers
//   that need to be fed into the shift register:
const int nshift_bits = 9;
const rc_t rc_for_shift[nshift_bits] = {r0, c3, c5, r3, c0, r1, c6, c7, 0};
// old array: const rc_t rc_for_shift[nshift_bits] = {c7, c6, r1, c0, r3, c5, c3, r0};
/// These are Arduino pins hardwired into the 8x8's bottom pins:
const int nhardwire = 8;
const char pin_for_hardwire[nhardwire] = {4, 5, 6, 7, 8, 9, 10, 11};
const rc_t rc_for_hardwire[nhardwire] = {r4, r6, c1, c2, r7, c4, r5, r2};

/// Each bit of this 16-bit datatype stores a value
/// for the rows (bits 15..8) and columns (bits 7..0)
/// for one fixed pattern on the LED matrix.
/// These are static patterns that can be shown without any row scanning.
typedef unsigned short LED_rc_bits_t;

/// Extract one bit from an rc_bits matrix (for a digitalWrite)
inline bool get_rc_bit(LED_rc_bits_t rc_bits, rc_t bit) {
  return (rc_bits >> bit) & 1;
}

const int npatterns = 6 + 8 + 8;

LED_rc_bits_t patterns[npatterns] = {

  //    0=on    1=on
  //     row   column
  //  7654321076543210
  0b1111111100000000, // all LEDs off
  0b0000000011111111, // all LEDs on
  0b0111111010000001, // corner LEDs on
  0b1011110101000010, // corner inset-1
  0b1101101100100100, // corner inset-2
  0b1110011100011000, // center 4

  0b0000000010000000, // right-to-left col scan (dark background)
  0b0000000001000000,
  0b0000000000100000,
  0b0000000000010000,
  0b0000000000001000,
  0b0000000000000100,
  0b0000000000000010,
  0b0000000000000001,

  0b0111111111111111, // bottom-to-top row scan (dark background)
  0b1011111111111111,
  0b1101111111111111,
```

```
  0b1110111111111111,
  0b1111011111111111,
  0b1111101111111111,
  0b1111110111111111,
  0b1111111011111111,
};

void setup() {
  Serial.begin(9600);

  // Pins 4,5,6,7,8,9,10,11
  for (int i; i <= nhardwire; i++) {
    pinMode(pin_for_hardwire[i], OUTPUT);
  }
  pinMode(clockPin, OUTPUT); // Pin 3
  pinMode(dataPin, OUTPUT); // Pin 2
}
void clockFunction(bool regInput) {
  digitalWrite(clockPin, 0); // clock low prepares for data
  digitalWrite(dataPin, regInput);
  digitalWrite(clockPin, 1); // rising clock accepts data
}

void loop() {

  // sets design pattern; refer to array
  // time delay = 1000 ms --> adjust for persistence of vision.
  for (int currentPattern = 0; currentPattern < npatterns; currentPattern++) {
    // Log time for static patterns.
    unsigned long time;
    time = micros();

    Serial.print("Current Pattern: ");
    Serial.print(currentPattern);
    Serial.print("\n");

    for (int currentBit = 0; currentBit < nshift_bits; currentBit++) {
      //8+1 times
      clockFunction(get_rc_bit(patterns[currentPattern], rc_for_shift[currentBit]));
      delay(1000);

      Serial.print("Current Bit: ");
      Serial.print(currentBit);
      Serial.print("\n");
    }

    for (int currentPin = 0; currentPin < nhardwire; currentPin++) {
      digitalWrite(pin_for_hardwire[currentPin],
                   get_rc_bit(patterns[currentPattern], rc_for_hardwire[currentPin]));
      delay(1000);

      Serial.print("Current Pin: ");
      Serial.print(currentPin);
      Serial.print("\n");
    }

    // Display static image benchmark time.
    Serial.print("Time: ");
    Serial.println(time);
  }

}
```