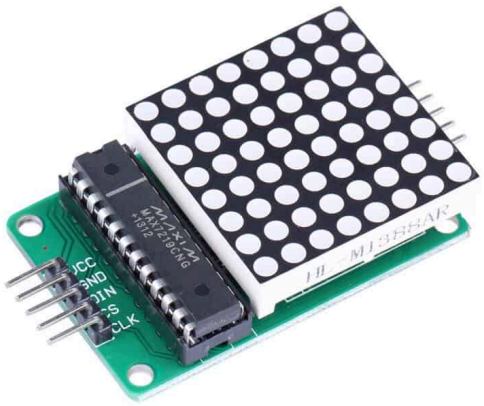
SPI 8x8 LED Matrix CSE/ECE 474 Spring 2024

Ishaan Bhimani, Revised by Vikram Iyer University of Washington

This background document is intended for working with an 8x8 LED Matrix that uses SPI communication with a MAX7219 chip. This part was included in the Elegoo MEGA 2560 Starter Kit, and is seen in the figure below. Your part may or may not have the MAX7219 IC visible, but the pins should be the same.

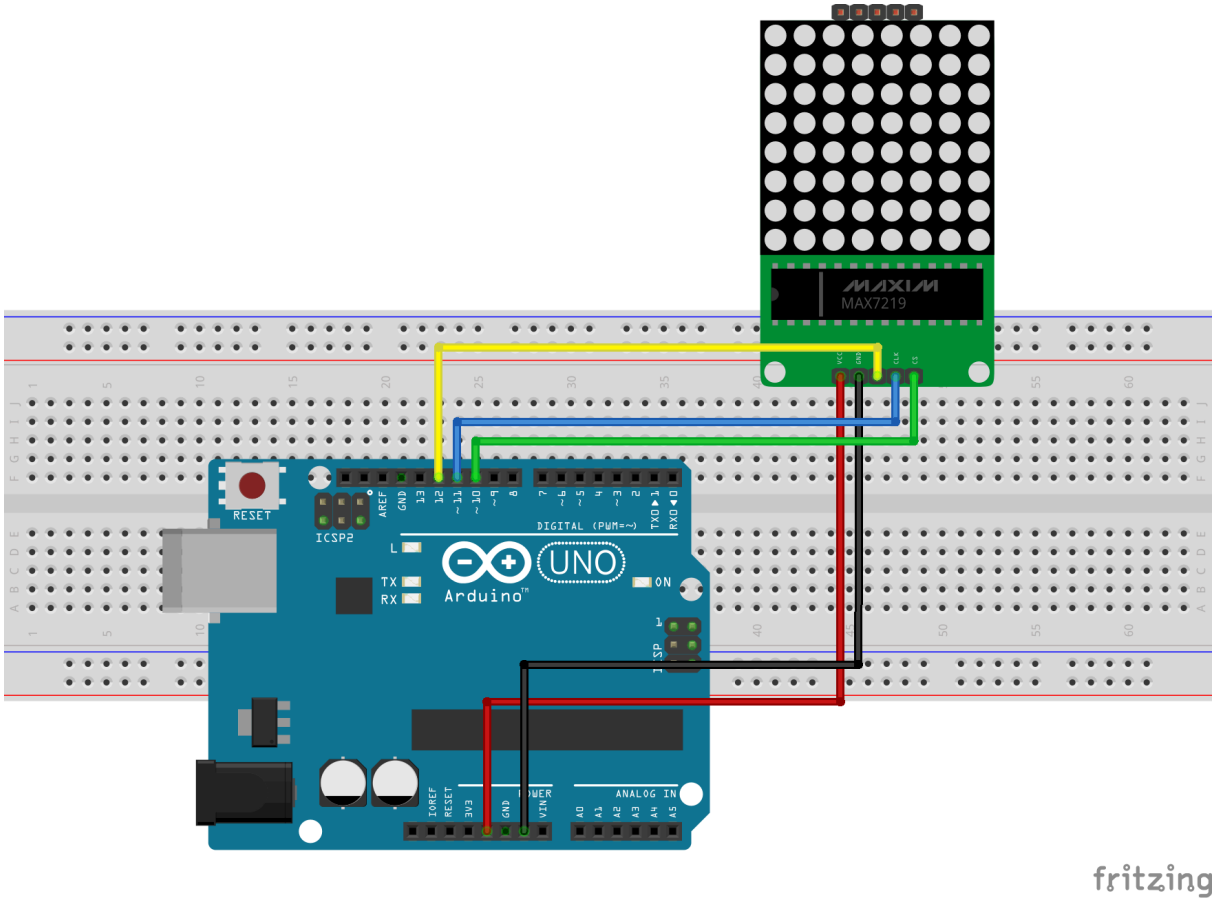


8x8 LED Dot Matrix with MAX7219 IC

This part uses Serial Peripheral Interface (SPI) to change the dots on the matrix. For Lab 2, you are not required to learn SPI, but if you are interested, you can look here [LINK].

In order to interface with the document, we have provided a function called spiTransfer(). `spiTransfer()` takes an int row and byte data. The row corresponds to a row of dots in the LED matrix, and the data's bit representation will indicate the dots in the row that are on. The indices of bits set to 1 in data are the same as the index of dots in the row which are on. That means: spiTransfer(0, 0b11100111) will light up all but the two middle dots in row 0. **Since it will be unfeasible (or very painful) to do the tasks we have asked you to complete using just the spiTransfer() function, we expect you to write your own functions to easily manipulate the LEDs in the dot matrix.**

When using the `spiTransfer()` function, you need to do some important setup. Firstly, set up your hardware. Vcc should connect to 5V, and GND should connect to GND on your Arduino. The other 3 pins need to be connected to digital GPIO pins. An image of this can be seen below:



Software Setup:
Code demonstrating the software setup below is posted in the Lab 2 folder:
https://courses.cs.washington.edu/courses/cse474/24sp/assignments/lab2/LED_Matrix.ino

Define the relevant macros at the top of your .ino file. The macros for these commands are:

```
#define OP_DECODEMODE  8
#define OP_SCANLIMIT   10
#define OP_SHUTDOWN    11
#define OP_DISPLAYTEST 14
#define OP_INTENSITY   10
```

Also, have a global variable `byte spidata[2];` Note: you only need 2 bytes for one device, but 16 bytes will not affect functionality.

In the `setup()` function of your code, you should set all of the digital pins to OUTPUT mode, and set the pin for CS to HIGH. **You should do this by accessing registers directly, not using any Arduino functions.**

Next, run the spiTransfer() commands below. This is still in the setup() function.

```
  spiTransfer(OP_DISPLAYTEST,0);
  spiTransfer(OP_SCANLIMIT,7);
  spiTransfer(OP_DECODEMODE,0);
  spiTransfer(OP_SHUTDOWN,1);
```

Make sure you've added our spiTransfer() function to your code, with an appropriate function prototype at the top of your function. Feel free to move relevant code into another file or a header file for organizational purposes.

Lastly, test out your hardware. The code below should turn on 4 rows of LEDs, and then turn off the LEDs. States shown in pictures below.

```
void loop(){
  int j = 0;
  int i = 0;
  for (j = 0; j < 8; j++){ //for each row, set the LEDs
    spiTransfer(j, 0b10100101);
  }
  delay(500);
  for (i = 0; i < 8; i++){ //for each row, clear the LEDs
    spiTransfer(i, 0b00000000);
  }
  delay(500);
}
```