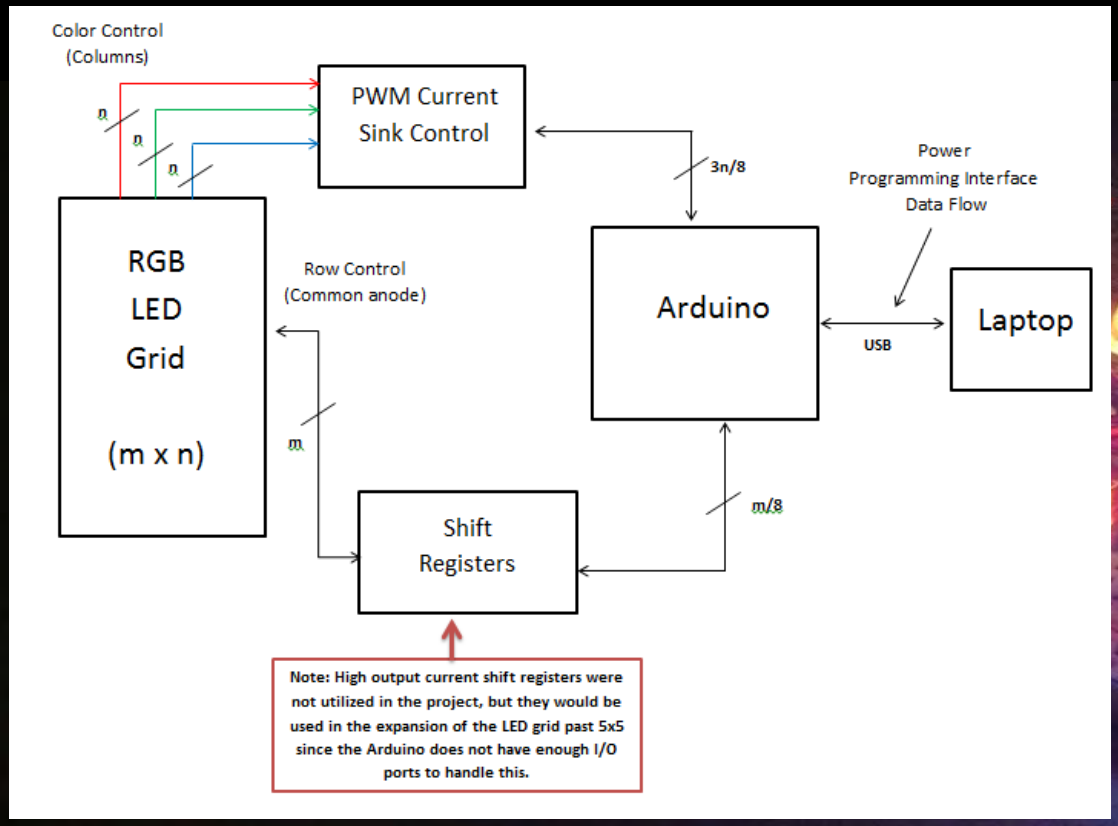
RGB LED Grid

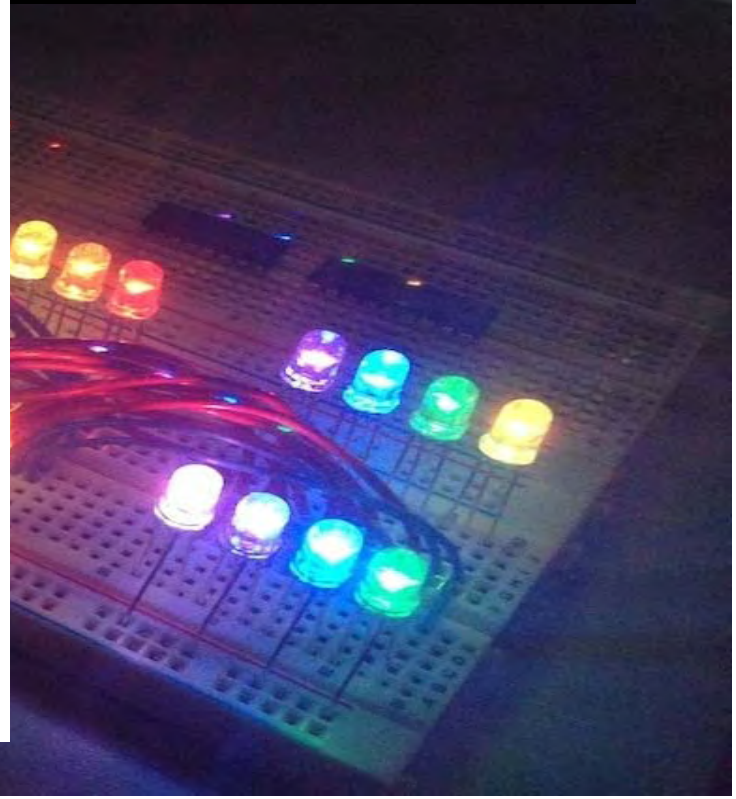
Jeremy Guiley and Nicolle Bates

Project Goals

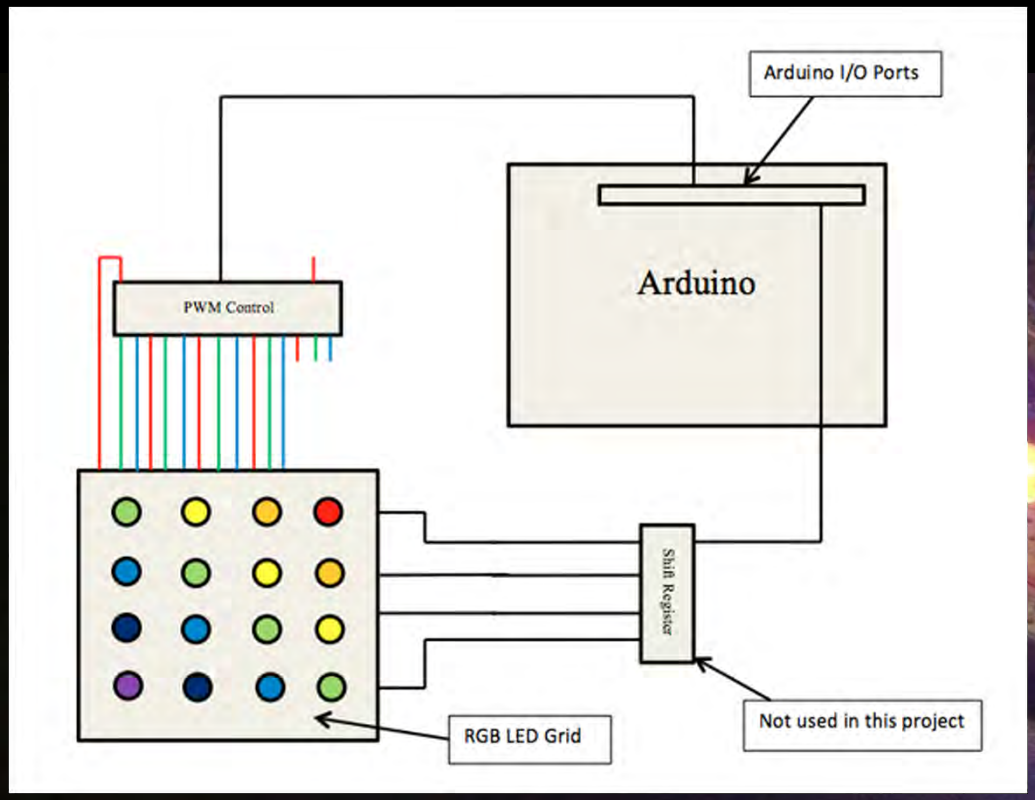
- Utilizes RGB LEDs in a grid
- Displays animations from an Arduino Uno
- Animation chosen is a rainbow display
- The LEDs are initially all illuminated
- LEDs are consistent across the diagonals
- LEDs cycle through the diagonals

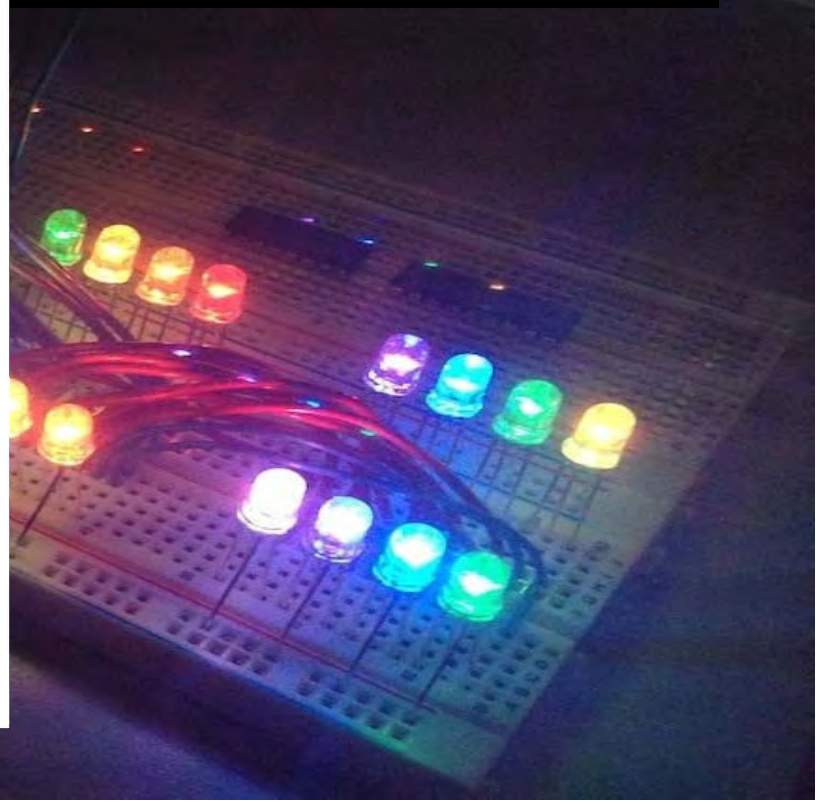
Block Diagram



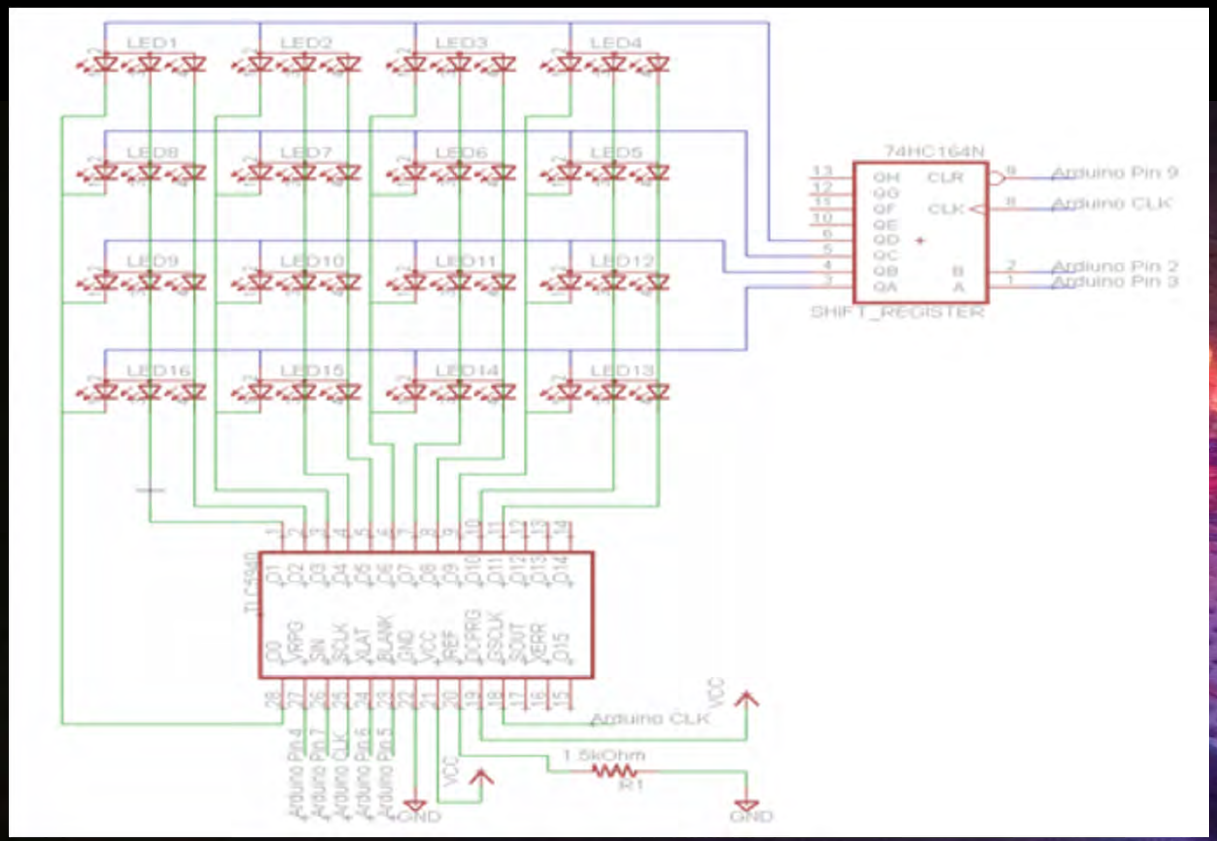


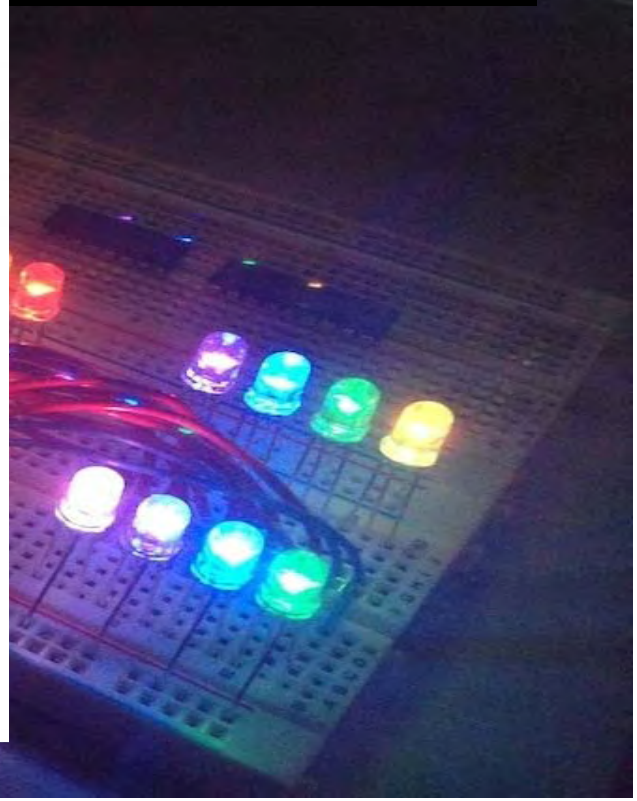
Hardware Setup



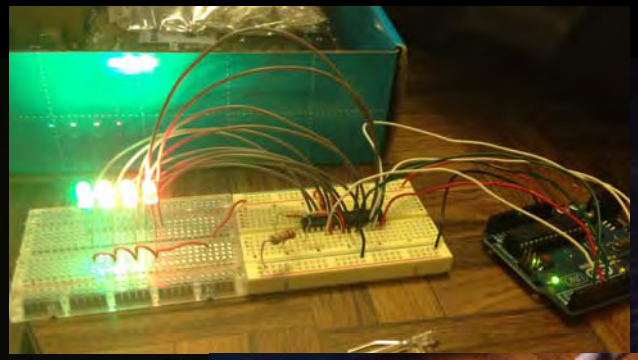


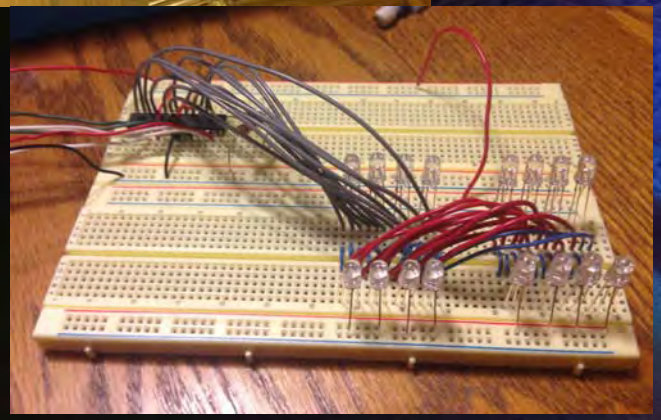
Schematic

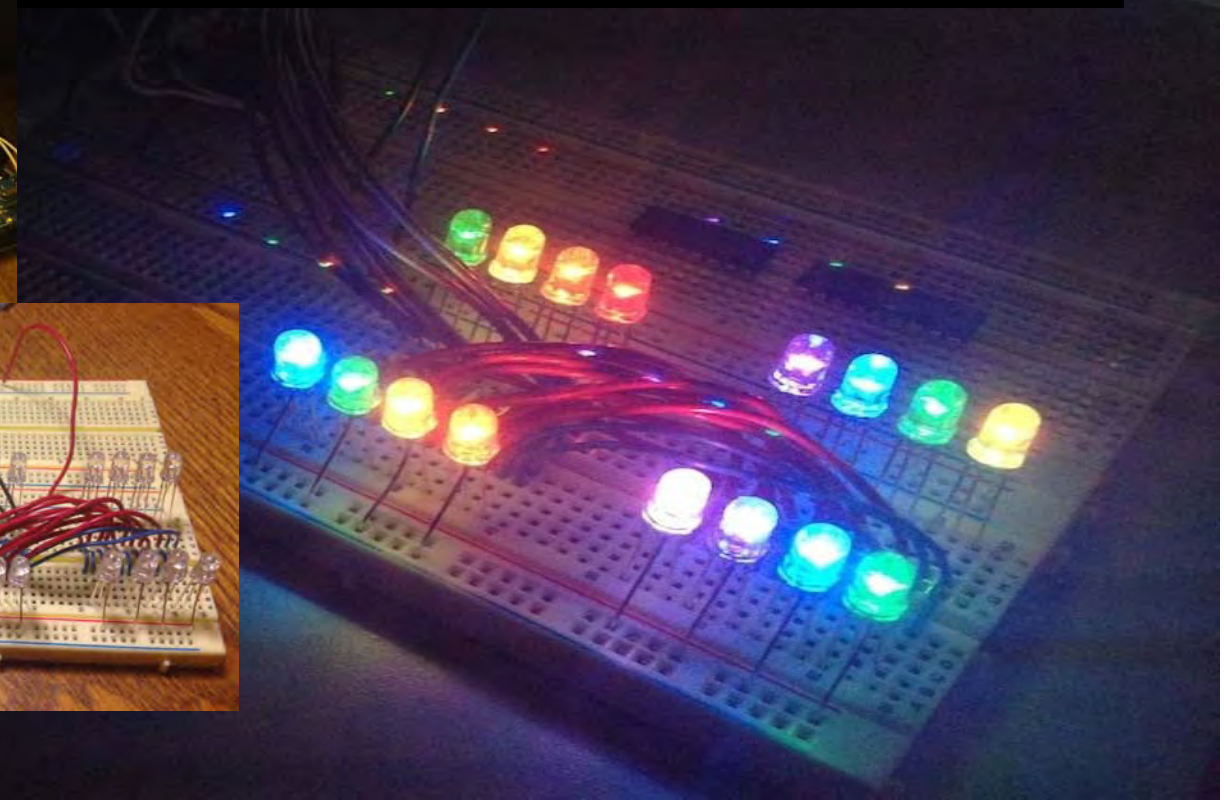

LED1
LED2
LED3
LED4
LED8
LED7
LED6
LED5
LED9
LED10
LED11
LED12
LED16
LED15
LED14
LED13
74HC164N
SHIFT_REGISTER
Arduino Pin 9
Arduino CLK
Arduino Pin 2
Arduino Pin 3
Arduino CLK
VCC
1.5kOhm
R1
GND
GND



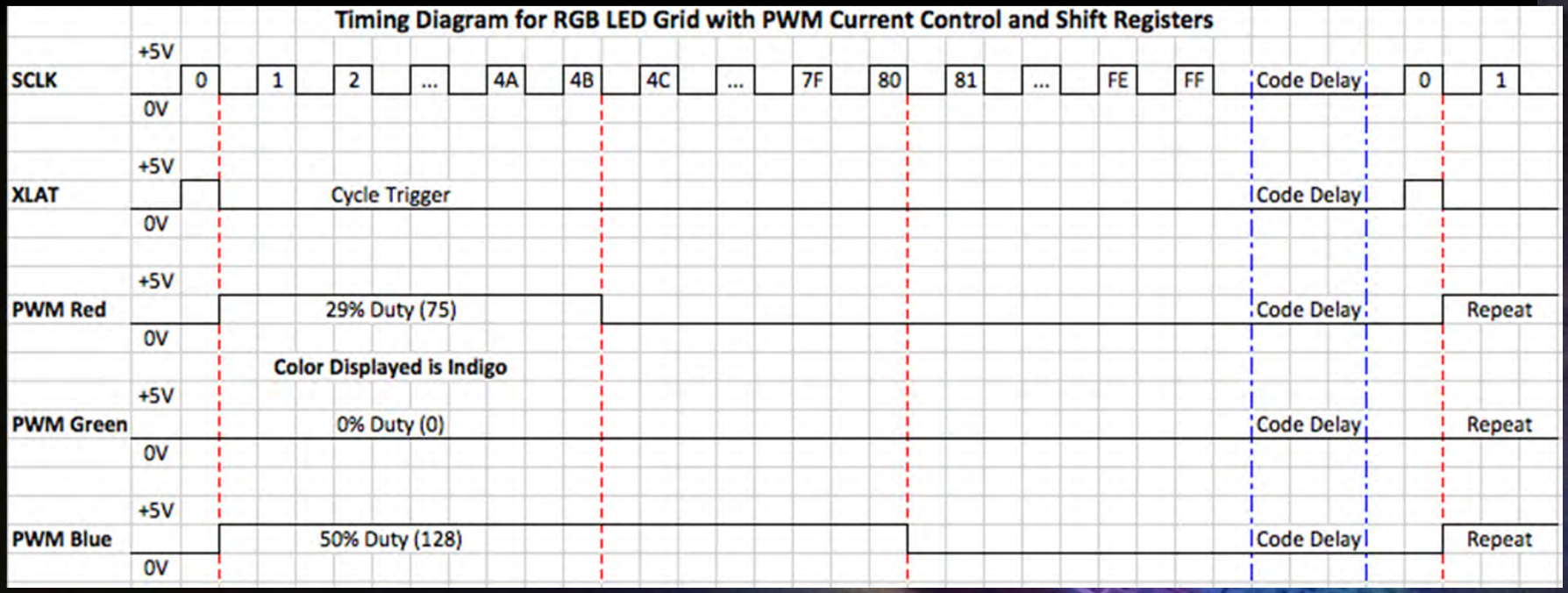
Prototyping







Timing


Timing Diagram for RGB LED Grid with PWM Current Control and Shift Registers
SCLK
+5V
0V
0
1
2
...
4A
4B
4C
...
7F
80
81
...
FE
FF
Code Delay
0
1
XLAT
Cycle Trigger
Code Delay
PWM Red
29% Duty (75)
Code Delay
Repeat
Color Displayed is Indigo
PWM Green
0% Duty (0)
Code Delay
Repeat
PWM Blue
50% Duty (128)
Code Delay
Repeat

Pseudo Code

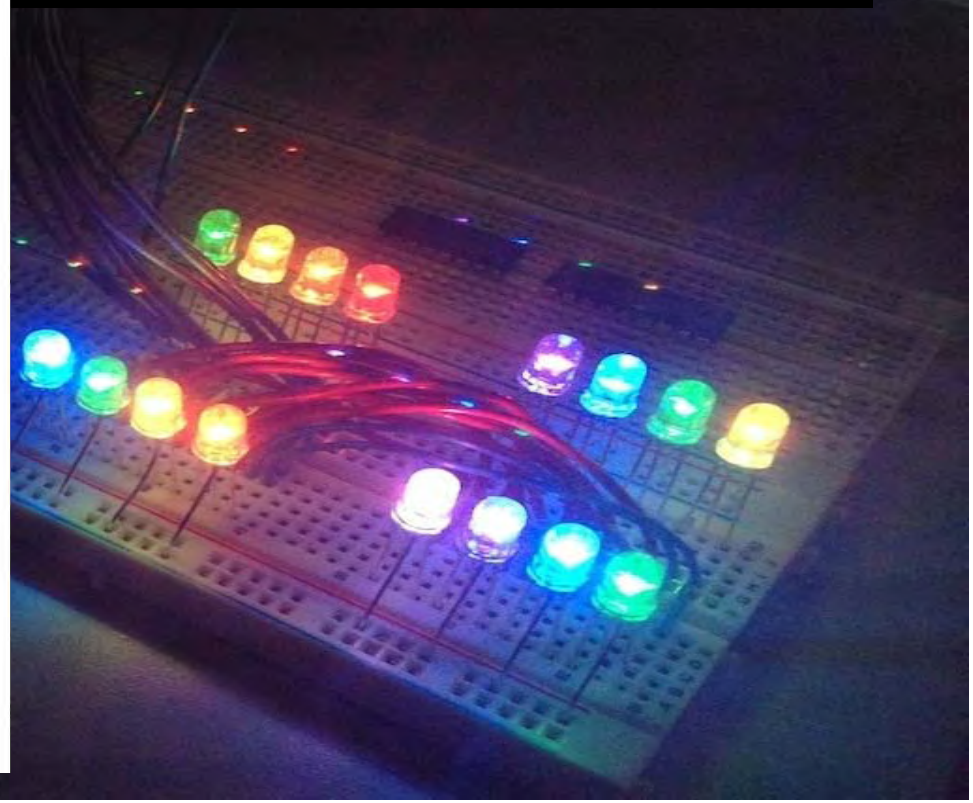
```
Initialize PWM controllers

Initialize Shift Registers {
   Start with row 1
   Continuously cycle through rows
}

Define Colors (R,G,B)
      Red
         (255,0,0)
      Blue
         (0,0,255)
      Green
         (0.255,0)
      Yellow
         (255,255,0)
      Violet
         (238,130,238)
      Orange
         (255,128,0)
      Indigo
         (75,0,130)

Define Animations
   Rainbow Static {
      Turn on all colors
      Each diagonal different rainbow
                  color in order
   }
   Rainbow Cycle {
      Start with red (upper right corner)
      Wait 5 seconds
      Cycle through color diagonals each 5
                  seconds
      }

Main {
   Call Initializations
   Call Animation Sequence
      Static Rainbow
      Wait about 30 seconds
      Cycle Rainbow (Diagonals) for about 5
                  seconds each
}
```

Global Variables

```
//GLOBAL VARIABLES
static int dotBits = 6;                  //bits in dot correction value
static int tlcBits = 12;                 //bits in TLC value
int sendBit = 0, sendByte = 0;           //bits and bytes sent thru SPRI
int outputs = 16;                        //number of TLC outputs
int tlcData = (tlcBits*outputs/8);
int full = 4095, half = 2048, off = 0;   //for colors
byte sendBytes[24];                      //bytes for SPI = tlcData
//ANIMATION VARIABLES
int columns = 4;                         //# of columns in LED grid
int rows = 4;                            //# of rows in LED grid
int start = 0, cstart = 1;               //cstart is first output used from
//the TLC, start is an animation variable being initiliazed (don't change)
```

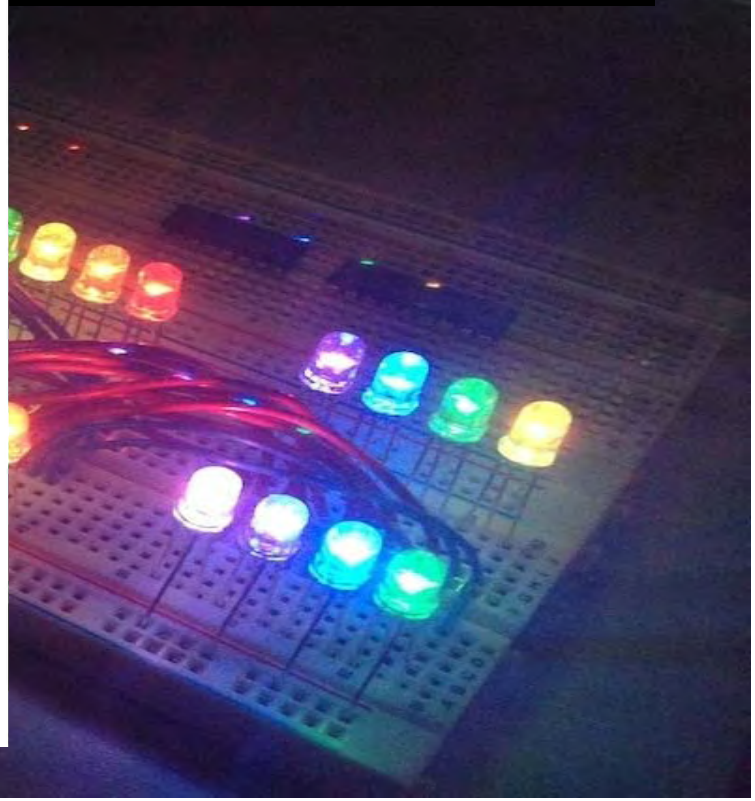
Setup

```
//SYSTEM INITIALIZATION
void setup(){
  noInterrupts();

  pinMode(2,OUTPUT);      //XLAT
  pinMode(3,OUTPUT);      //GSCLK
  pinMode(4,OUTPUT);      //VPRG
  pinMode(6,OUTPUT);      //row 1
  pinMode(7,OUTPUT);      //row 2
  pinMode(8,OUTPUT);      //row 3
  pinMode(9,OUTPUT);      //row 4
  pinMode(11,OUTPUT);     //SIN
  pinMode(13,OUTPUT);     //SPCLK

  //must choose MSBFIRST as stated in tlc data sheet
  SPI.setBitOrder(MSBFIRST);      //Most significant bit first
  SPI.setDataMode(SPI_MODE0);     //rising, clock low, default 4MHz

  for(int i = 0; i < tlcData; i++)    //Clear sendBytes
    sendBytes[i] = 0;

  DOT();                          //Initialize dot correction data

  for(int i = 0; i < outputs; i++)    //clear tlc
    TLC(i,0);

  TCCR2A = 0x12;        //setup GSCLK hardwired to pin 3
  TCCR2B = 0x01;        //GSCLK prescaler = 1 => 16 MHz

  pinMode(5,OUTPUT);    //BLANK
}
```



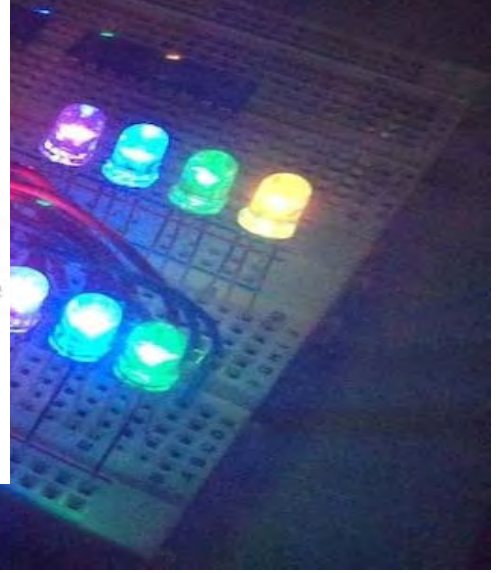
DOT

```
//DOT CORRECTION
void DOT(){
  PORTD |= 0x10;        //VPRG high
  //dot correction for all channels (red, green, blue are common)
  //this code can be adjusted if the application requires dot correction
  //for blue, or another color. the value is only set to max for simplicity
  sendByte = 0;
  sendBit = 0;
  for(int out = 0; out < outputs; out++){
    for(int outBit = 0; outBit < dotBits; outBit++){
      if(sendBit == 8){
        sendByte++;
        sendBit = 0;
      }
  //set dot correction to full bright
      bitSet(sendBytes[sendByte],sendBit);
      sendBit++;
    }
  }
  SPI.begin();
  for(int i = sendByte; i >= 0; i--)  //because SPI must be MSB first
    SPI.transfer(sendBytes[i]);
  PORTD |= 0x04;
  PORTD &= ~0x04;                 //XLAT data in
  PORTD &= ~0x10;                 //VPRG low
}
```



TLC

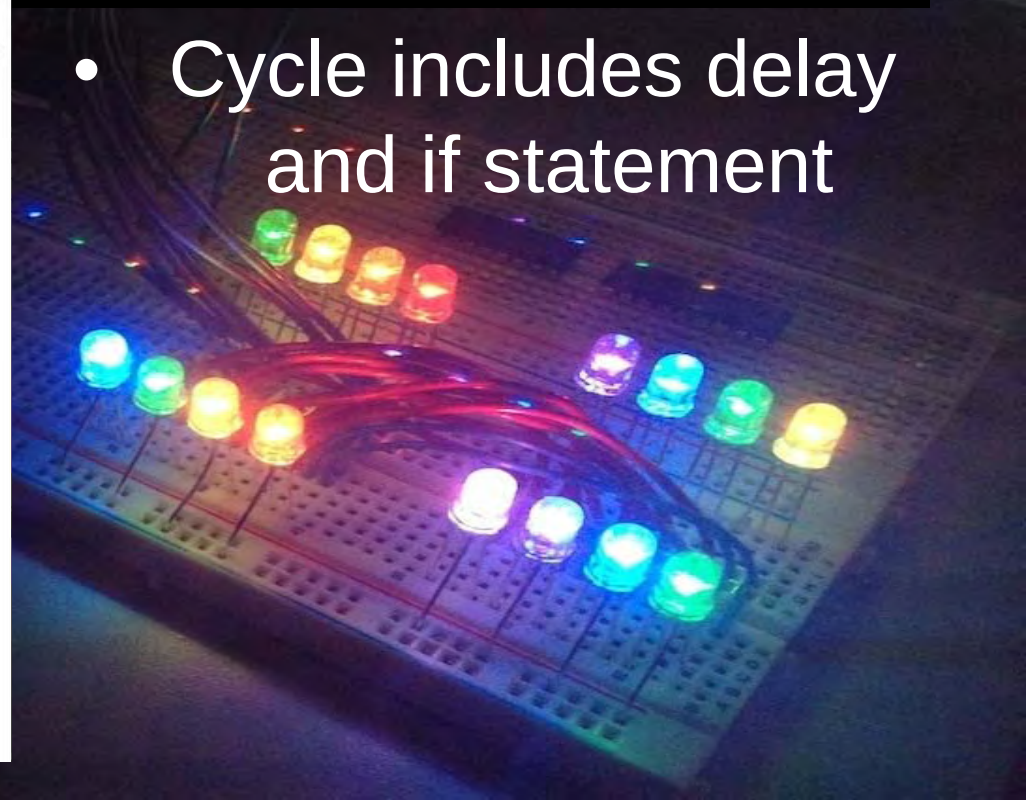
```
//TLC OUTPUT CONTROL
//difficulty is sending a series of 12-bit signals as 8-bit bytes
void TLC(int channel, int bright){
  sendBit = 0;
  if(channel % 2)       //check for odd channel
  sendBit = 4;          //for odd channels we start in the middle of a byte
  sendByte = int(channel*tlcBits/8);    //starting byte
  for(int outBit = 0; outBit < tlcBits; outBit++){
    if(sendBit == 8){
      sendByte++;       //next byte
      sendBit = 0;
    }
    if(bitRead(bright, outBit))          //for bright = 1, set bits in sendByte
    bitSet(sendBytes[sendByte], sendBit);
    else                                 //for bright = 0, clear bits in sendByte
    bitClear(sendBytes[sendByte], sendBit);
    sendBit++;
  }
}
```



Animation

```
//ANIMATIONS
//array of pointers to the color functions for other animations
void (*rainbow[4][4])(int column) = {{RED, ORANGE, YELLOW, GREEN},
                                     {ORANGE, YELLOW, GREEN, BLUE},
                                     {YELLOW, GREEN, BLUE, INDIGO},
                                     {GREEN, BLUE, INDIGO, VIOLET}};

void staticRainbow(void){
  start = 0;
  for(int i = 0; i < rows; i++){
    if(i == 1){
      PORTB &= ~0x02;
      PORTD |= 0x40;
    }else if(i == 2){
      PORTD &= ~0x40;
      PORTD |= 0x80;
    }else if(i == 3){
      PORTD &= ~0x80;
      PORTB |= 0x01;
    }else if(i == 0){
      PORTB &= ~0x01;
      PORTB |= 0x02;
    }
    for(int j = 0; j < columns; j++)
      (*rainbow[i][j])(cstart+j*3);
    if((start == rows - 1) || (start == 7))
      start = 0;
    else
      start++;
    sendData();
    for(int i = 0; i < outputs; i++)    //clear tlc
      TLC(i,0);
  }
}
```

- Cycle includes delay and if statement

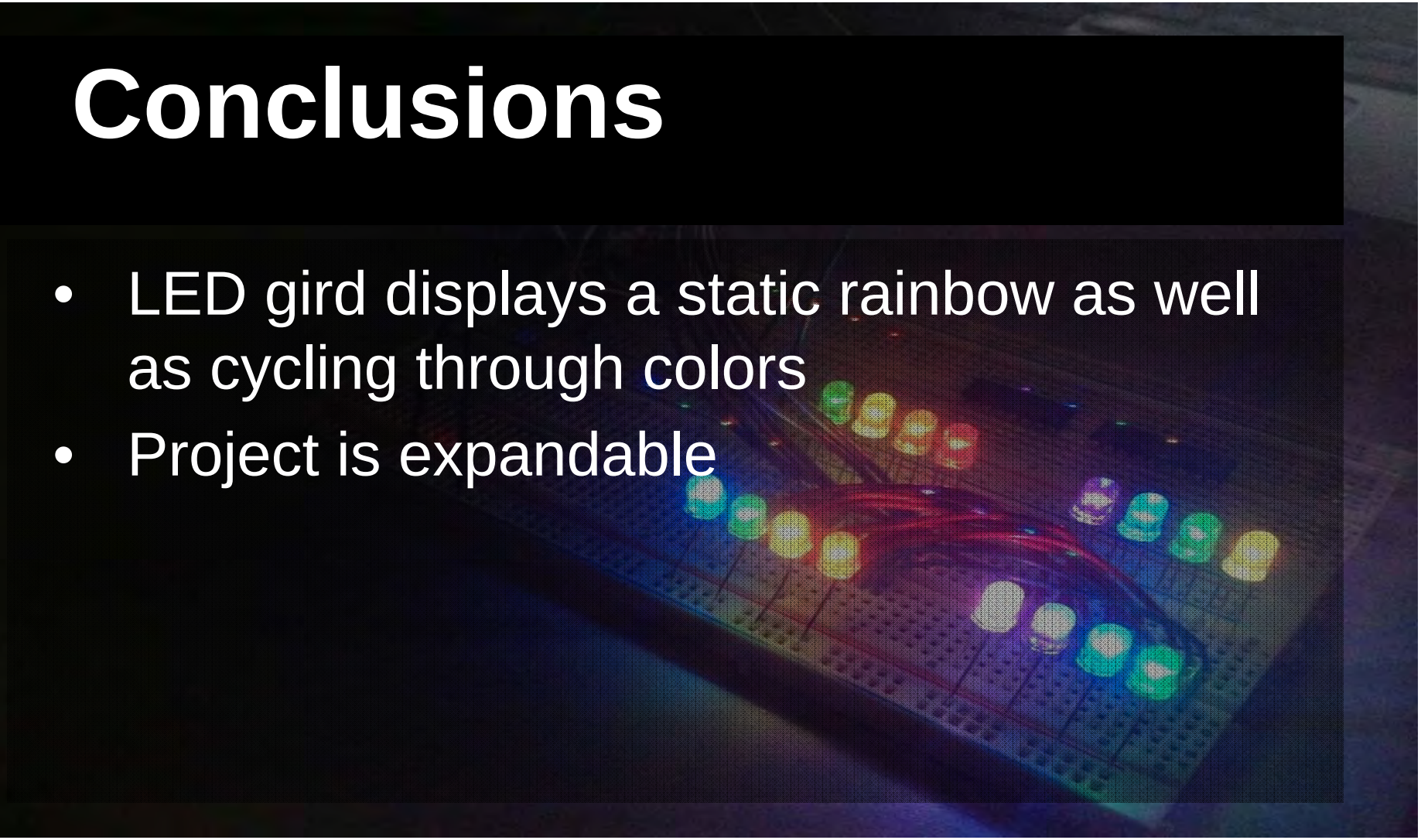


Lessons Learned

- Do not use a Mac for Arduino interfacing
- Matrices make life easier
- Arduino interrupts can be complicated

Conclusions

- LED gird displays a static rainbow as well as cycling through colors
- Project is expandable



Questions?