Objective: Gain experiences programming, emphasizing use of code developed thus far in the class. This lab will involve using the button and encoder to control the LED display that was used in Lab 7. Our goal will be to control the position of a blinking LED.

1. The program in this lab is designed to enable a user to control which individual LED on a MAX7219 is lit up. In order to implement this program code from multiple previous labs had to be used and modified. The program utilized the encoder knob and button. Code was added to detect if and how long a user pressed down on the encoder knob. This was used to switch the program between its two main states, vertical and horizontal mode. This state told the program if the user was trying to switch the column position or the row position. The program utilized code for lab 7 using SPI, utilized code from lab 6 to manage the LED board. And utilized code and knowledge to manage the LCD screen. The LCD screen was used to display the mode the program was in and also the row and column position.

2.

```
#include <LedControl.h>

#include "SW_SPI.h"
#include "LedControl.h"
#include <LiquidCrystal.h>
#define MAX7219_TEST 0x0f00 //Display in test mode
#define MAX7219_BRIGHTNESS 0x0a00 // Set brightness of display
#define MAX7219_SCAN_LIMIT 0x0b00 // Set Scan limit
#define MAX7219_DECODE_MODE 0x0900 // Sets chip to accept bit patterns
#define
#define MAX7219_SHUTDOWN 0x0C00 // Code for shutdown ship


unsigned long LEDTimer; //used to measure if 200 milliseconds has passed in
the program
unsigned long ButtonTime; //this will be used to measure how long a button
has been pressed down for

LedControl lc = LedControl (A5, A3, A4, 1); //setting up the LED Control
using the ledcontrol.h file. We also set the pins A5, A3, and A4 to be used
LiquidCrystal LcdDriver(11, 9, 5, 6, 7, 8); //setting up the  LCd screen and
assigning it's pins, these are default since the arduino and screen are built
into the board

int encoderPosition = 0; //this value will keep track of the encoder position
volatile int PrevPosition = 0; //this value will track the previous position
of the encoder


int DisplayColumn; //value to track which column we are working with
int DisplayRow; // value to track witch row we are working with
int State = 0; //this int will manage what state the LCD is in. It will
either be 1, Vertical, or 2 Horizontal
```

```
int ButtonState; //used to manage the button debounce procedue


int ButtonNextState(int input) { // fucntion to service the switch switch
(ButtonState) starts the switch / case statement

switch (ButtonState){
case 1: // Idle case
  if (input == LOW) { // If the pin is reading low
    ButtonTime = millis(); // Records the time of high to low transition
    PORTB |= 0x20;
    ButtonState = 2; // This case first checks to see if the input on the pin
is Low, and if it is, it switches the state of the button to the waiting
state (state 2)
  }
  break;
case 2: // Wait
  if (input == HIGH) { // If the button has gone high
    ButtonState = 1; // Says that if the button is reading high, the program
resets back to an idle state
  }
  else if (millis() - ButtonTime >= 5) { // checks to see if 5 milliseconds
has passed
    PORTB &= ~0x20;
    ButtonState = 3; // This moves the button to a low state as described in
case 3
    encoderPosition--;
    //return 1; // This indicates the button has been pressed
    }
    break;
  case 3: // Button is Low
    if (input == HIGH) { // Reads a button press
      ButtonState = 1; // If it is pressed, the state goes back to idle

    if ( millis() - ButtonTime < 500 ) {
      return 2; // short press for horizontal switch
    }
    else {
      return 3; // long press for vertical switch
    }
    }
    break;
  }
  //return 0; // This is a return to 0 which indicates there is nothing
happening
}

void MonitorA() {
  if (digitalRead(2) == digitalRead(3)) { // checks to see if pin 2, and pin
3 are reading the same
    encoderPosition++; // if they are, this line increments the value of
encoderPostion by 1
  }
  else { // if they are not equal
    encoderPosition--;// this line decrements encoderPosition
```

```
  }
}
void MonitorB() {
  if (digitalRead(2) == digitalRead(3)) { // again checks if pin 2 and pin 3
are equal to one another
    encoderPosition--; // if they are, this decrements the position by 1
  }
  else { // if they're not equal
    encoderPosition++; // this increments encoderPosition }
  }

}
void setup() {

    pinMode(2, INPUT);
    pinMode(3, INPUT);
    pinMode(4, INPUT);
    pinMode(A3, OUTPUT);
    pinMode(A4, OUTPUT);
    pinMode(A5, OUTPUT);
    attachInterrupt(digitalPinToInterrupt(2), MonitorA, CHANGE); // Attaches
monitorA to interupt 0, and sets the mode for any change
    attachInterrupt(digitalPinToInterrupt(3), MonitorB, CHANGE); // Attaches
monitorB to interupt 1, and sets the mode for any change


    pinMode(4, INPUT); // Pin 4 is the input pin in this case
    ButtonState = 1; // Sets the program to begin in an idle state
    Serial.begin(9600); //sets the baud rate to 9600
    ButtonTime = millis(); // declares the button timer

    LcdDriver.begin(16, 2); //sets the size of the LCD screen
    LcdDriver.clear(); //clears the LCD screen
    LcdDriver.setCursor(0, 0); //sets the default position of the cursor to
0,0
    DisplayColumn = 0; //sets the default value of the column to 0
    DisplayRow = 0; //sets the defaulut value of the row to 0
    LEDTimer = millis(); //sets the LEDtimer to the number of milliseconds
since the program began
    ButtonTime = millis(); //sets the ButtonTime to the number of
milliseconds since the program began
    ButtonState = 1; //the default state of the button will be idle
    attachInterrupt(digitalPinToInterrupt(2), MonitorA, CHANGE); //attaches
monitorA to pin 2 as an interrupt
    attachInterrupt(digitalPinToInterrupt(3), MonitorB, CHANGE); //attaches
mintorB to pin3 as an interrupt

    SW_SPI_16(MAX7219_TEST + 0x01); // Turn on all the LEDs. This can be seen
when the program is being uploaded to the arduino as all the lights are on
for about 100 milliseconds while the program is going throught he dealay
    delay(100); // One time we can use a delay.
    SW_SPI_16(MAX7219_TEST + 0x00); // all LEDS off.
    SW_SPI_16(MAX7219_DECODE_MODE + 0x00); // Disable BCD mode.
    SW_SPI_16(MAX7219_BRIGHTNESS + 0x03); // Use lower intensity.
    SW_SPI_16(MAX7219_SCAN_LIMIT + 0x0f); // Scan all digits.
```

```
    SW_SPI_16(MAX7219_SHUTDOWN + 0x01); // Turn on chip.


  }
void loop() {
    int result = ButtonNextState(digitalRead(4)); //calls the buttonNextState
funcction and sends it the value on pin 4 (associated with the output of the
encoder
    if (result == 2) { //the fucntion tells us the user has set the system to
vertical mode
      Serial.println("Vertical"); //prints Vertical on the top of the screen
      State = 0; // vertical state
    }
    if (result == 3) {
      State = 1; // horizontial state
    }
    if (millis() - LEDTimer >= 200) { //checks to see if 200 milliseconds has
passed since this function was last executed
      switch (State) { //switch statements based on whether the program is in
vertical or horizontal mode
        case 0: //Vertical mode selected
          if (encoderPosition < PrevPosition) { //if the encoder position is
below the previous position
            DisplayColumn --; //decrement display column
             if (DisplayColumn <= 0) { //if the display column is less than 0
                DisplayColumn = 0; //set the value to 0
            }
          }
          else if (encoderPosition > PrevPosition) { //if the encoderposition
is greater than its last position, increase the display column
            DisplayColumn ++; //increase the dislpay column
            if (DisplayColumn >= 7) { //if the display column is greater than
7
                DisplayColumn = 7; //set the value of the displaycolumn to 7
            }
          }
          PrevPosition = encoderPosition; //set the value of prevposition to
the current encoderposition
          break; //break from the switch case
        case 1: //horizontal mode
          if (encoderPosition < PrevPosition) { //if the encoder position is
below the previous position
            DisplayRow --; //decrement display column
            if (DisplayRow <= 0) {//if the display column is less than 0
                DisplayRow = 0;//set the value to 0
            }
          }
          else if (encoderPosition > PrevPosition) {
            DisplayRow ++;
            if (DisplayRow >= 7) {
              DisplayRow = 7;
             }
          }
          PrevPosition = encoderPosition; //set prevposition to the current
position for use next loop
```

```
          break;

      }
      lc.setLed(0, DisplayRow, DisplayColumn, true); //sets the LED with the
value of the row and column to be lit up
      LcdDriver.clear(); //clear the lcd
      LcdDriver.setCursor(1, 0); //set the cursor
      LcdDriver.clear();
      if (State == 0) { //print to the LCD
        LcdDriver.print("VERTICAL");
      }
      else {
        LcdDriver.print("HORIZONTAL");
      }
        LcdDriver.print(State);
        LcdDriver.print(ButtonState);
        LcdDriver.setCursor(0, 1);
        LcdDriver.print("Row:");
        LcdDriver.print(DisplayRow);
        LcdDriver.print(" Column:");
        LcdDriver.print(DisplayColumn);

      lc.setLed(0, DisplayRow, DisplayColumn, false);
      LEDTimer += 200;//increment the LEDTIMER by 200 milliseconds
    }
  }
```

3. The program was tested through trial and error using use cases. Use cases were broken down to the smallest individual task the program could execute and built up over time once it met each individual use case. The output of the program was tested each time, both on the LED and the LCD. These outputs were also checked to see if they matched up and were accurate based on the input from the user. It is in my opinion that the testing was thorough and the program runs as intended in the design process.
4. The challenging part of this lab was the use of SW_SPI.h and the LEDControl.h files. These were mainly challenging because you are working with outside code that you did not write so time must be taken to read and understand the code so that you can work with it. Another feature that could be added to the program is a 3rd state. In this state the knob would control both the row and columns at the same time by overflowing at the end or beginning of a line. In this case the program would detect when the LED position is at the end of a row and if the knob is turned again the LED would jump down to the next row and vice versa.