SANTA CLARA UNIVERSITY
Electrical and Computer Engineering Department

ELEN 120 – Embedded Computing Systems

Lab 6 – Serial Communications

Dylan Thornburg, Sal Martinez, & Genevieve (Gigi) Patmore

Andrew Wolfe

Assignment: In this assignment, you will connect a 60-LED light strip to the Discovery board and use the SPI protocol to make it light up.

Learning Objective: Learn how a serial communications port operates and how to implement a simple serial port in software.

Lab Overview:

The LED strip I have provided is from Pololu. https://www.pololu.com/product/3089

A company called DONGGUANG OPSCO OPTOELECTRONICS CO., LTD makes a chip called the SK9822. It includes red/green/blue LEDs and a digital controller for those LEDs that uses the SPI protocol. Other companies mount these onto flex circuits and place them in a reasonably waterproof clear tube – generally 5M worth using this chip. Pololu cuts this into 1M strips and solders a connector and power wires on them and provides some directions and software – but not the software you need .

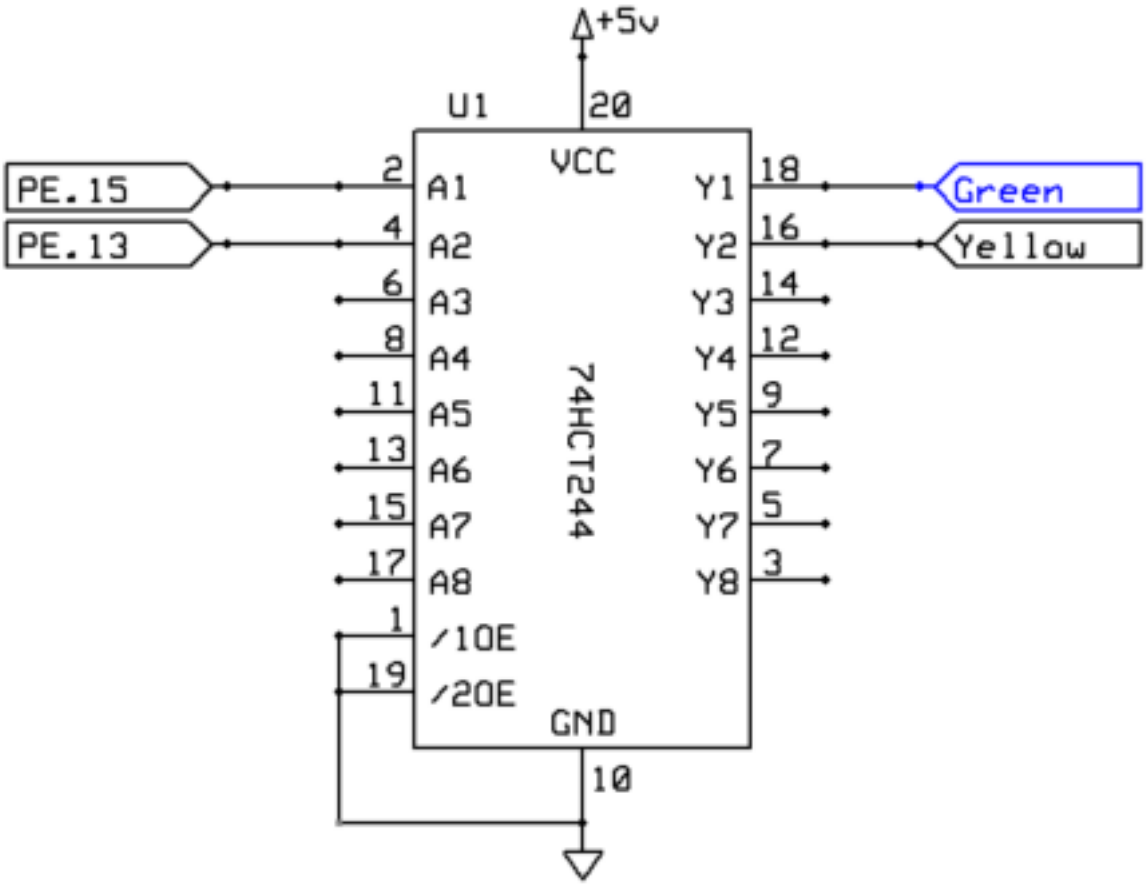
I have provided you with the datasheet for the SK9822. Review it and use it to develop this project. I have also copied the directions from Pololu. They recommend a small change to the protocol. I have used the SK9822 datasheet protocol without any problems – but you may want to try both.

We are going to use 2 GPIO pins to create an SPI output port. Pin E.13 will be the SPI clock. Pin E.15 will be the SPI Data Out. These will drive the LED chips. The LED will be powered by 5V. The GPIO signals should be configured as push-pull outputs even though this will only provide 3.3V. You need to convert these signals to 5V with the converter chip. Use the proto board and wires.

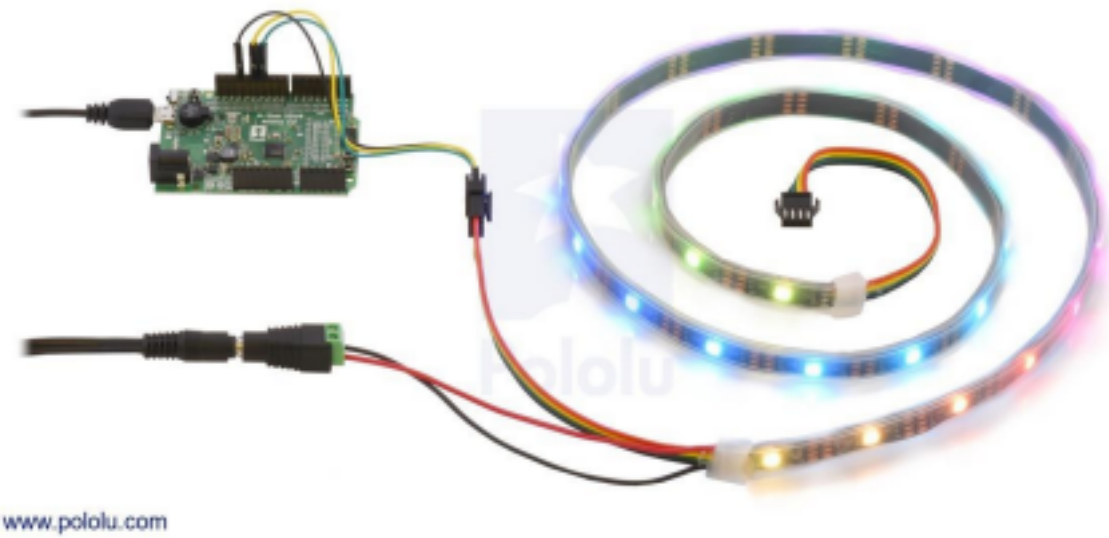
A schematic drawing is provided here. Review the 74HCT244 data sheet carefully and make sure you know which pins on the chip correspond to which pin numbers. Put the chip into your proto board straddling the center valley. You may need to straighten pins to get them to fit in the holes. Connect DISCO board pins PE.13 and PE.15 to the 74HCT244 chip as shown. Connect ground on the DISCO board to pins 1, 19, and 10 on the 74HCT244 chip. Connect 5V on the DISCO board to pin 20 on the 74HCT244 chip.

Connect the green and yellow LED wires to the 74HCT244 chip as shown. Make sure you can identify pin 1 properly. Connect DISCO board pins PE.13 and PE.15 to the 74HCT244 chip as shown. Connect ground on the DISCO board to pins 1, 19, and 10 on the 74HCT244 chip and to

the black wire from your LED strip. Connect 5V on the DISCO board to pin 20 on the 74HCT244 chip. The remaining pins can be unconnected. To connect up the LED strip, use the connector with the male pins to connect 3 of the wires to your Discovery board. Use female male jumper cables to connect to the proto-board.



Do not connect the red wire on the LED strip to your Discovery board or proto-board. The picture has a different board and no level converter– but the idea is the same.



The protocol for the SK9822 works as follows:

- You are communicating with the first chip. A header word alerts the chip that data is coming.

- The first chip grabs the next 32 bits as its new lighting command.

- The remaining bits are passed on to the next chip. It will grab the next 32 bits as its lighting command and pass on the rest, etc.

- Each set of passed-on bits is delayed by ½ clock cycle. Because of this you need to add on some extra data at the end. The SK9822 recommends 0xFFFFFFFF. This worked fine for me. (It may not work for more than 64 LEDs). Pololu has an alternate recommendation.

It took me more than a couple of hours to write and debug the SPI software for the GPIO pins, so I am not going to ask you to do that during lab. I have gone over this code in class. I have provided you with a file rgb60_redact.s that includes 3 routines:

- spisw_init - Initializes Port E pins 13 and 15 to work as an SPI output port • spi8 -

  Sends a byte over SPI, MSB first

- spi32 - Sends a 32-bit word over SPI, MSB first

You can use this code in your program. Except, whoops, I forgot to give you most of the code for spisw_init and spi32. Guess you need to write it.

Do not set the brightness for the LEDs above 50% - it is too bright for indoors and will make your head spin.

****Disclaimer for photos: final LED is broken hence why it doesn't light up (confirmed by Professor Wolfe)**

Problem 1

Complete at least spisw_init and write code in main to light up the first LED in the string red. Demo for the TA and take a picture for your report. (The LED next to it might inadvertently light up even if you do everything right)

main:

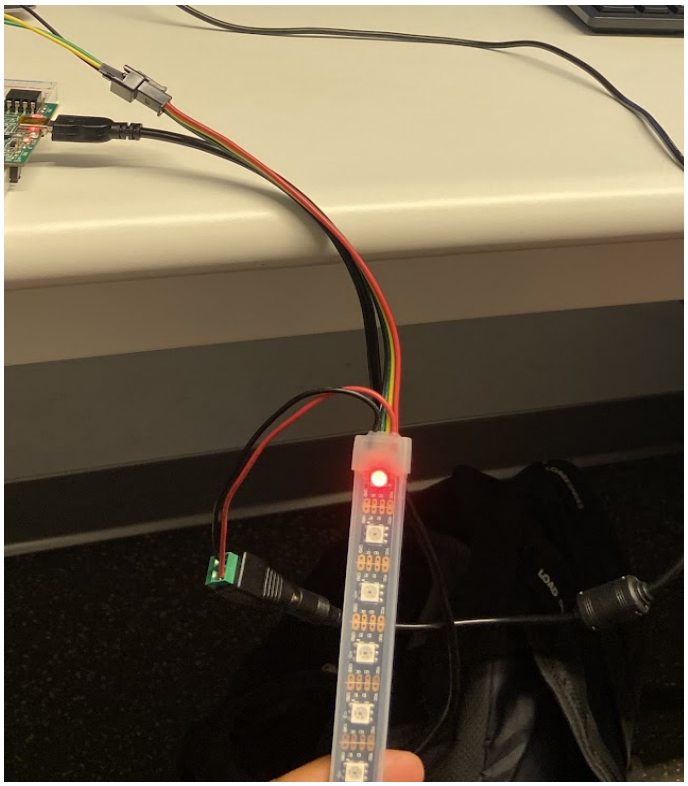
```
********** (C) Andrew Wolfe*****************************************
 @file    main_hw_proto.s
 @author  Andrew Wolfe
 @date    August 18, 2019
 @note
        This code is for the book "Embedded Systems with ARM Cortex-M
        Microcontrollers in Assembly Language and C, Yifeng Zhu,
        ISBN-13: 978-0982692639, ISBN-10: 0982692633 as used at Santa Clara University
*******************************************************************************

        INCLUDE core_cm4_constants.s            ; Load Constant Definitions
        INCLUDE stm32l476xx_constants.s
        INCLUDE jstick.h
        INCLUDE leds.h
        INCLUDE rgb60_redact.h

                AREA    main, CODE, READONLY
                EXPORT      __main
                ENTRY

__mainPROC
                bl spisw_init
                mov r8, #15
                ldr r0, =0
                bl spi32
                ldr r0, =0xed0000ff
                bl spi32
                ldr r0, =0
                bl spi32


                ALIGN
            ENDP

        END
```

Rgb60 (is not changed for the rest of the lab):

```
        INCLUDE core_cm4_constants.s            ; Load Constant Definitions
        INCLUDE stm32l476xx_constants.s
        INCLUDE leds.h
        AREA    main, CODE, READONLY



;Utility routines for the 60-LED SK9822 LED strip
spisw_init      PROC ;Initialize Port E pins 13/15 as outputs to use as a software SPI port.
                     ;Try push-pull outputs at 3.3V
                     ;Pin 13 is sclk, pin 15 is Dout
                     ;Data is clocked into the RGB strip on the rising edge of sclk

EXPORT          spisw_init

                push {lr}
                ldr r0, =RCC_AHB2ENR_GPIOEEN
                bl portclock_en
                ldr r1, =(GPIOE_BASE + GPIO_MODER)
                ldr r2, [r1]
                bic r2, #(0xcc << (2*15))
                orr r2, #(0x88 << (2*15))
                str r2, [r1]
                ldr r0, =GPIOE_BASE
                ldr r1, =GPIO_MODER_MODER13_0
                bl port_bit_pushpull
                ldr r0, =GPIOE_BASE
                ldr r1, =GPIO_MODER_MODER15_0
                bl port_bit_pushpull
                pop {lr}
                bx      lr

                ENDP
```

```
spi8        PROC                ;send 8 bits out the SPI port - MSB first
                                ;send out the low 8 bits of r0
                                ;sclk starts low and ends low
EXPORT      spi8

            mov r1,#8
            ldr r2,=(GPIOE_BASE+GPIO_BSRR)
            push    {r4,r5,r6}
            ldr r3,=GPIO_BSRR_BS_13
            ldr r4,=GPIO_BSRR_BR_13
            ldr r5,=GPIO_BSRR_BS_15
            ldr r6,=GPIO_BSRR_BR_15

spi8_1      tst r0, #0x80
            streq r6,[r2]
            strne r5,[r2]
            str r3,[r2]
            str r4,[r2]
            lsl r0,#1
            subs r1,#1
            bne     spi8_1
            pop{r4,r5,r6}
            bx      lr

            ENDP



spi32       PROC  ;send 32 bits out the SPI port - MSB first
                                ;send out the 32 bits of r0
                                ;sclk starts low and ends low
EXPORT      spi32

            mov r1,#32
            ldr r2,=(GPIOE_BASE+GPIO_BSRR)
            push{r4,r5,r6}
            ldr r3,=GPIO_BSRR_BS_13
            ldr r4,=GPIO_BSRR_BR_13
            ldr r5,=GPIO_BSRR_BS_15
            ldr r6,=GPIO_BSRR_BR_15
```

```
spi32_1         tst r0,#0x80000000
                streq r6,[r2]
                strne r5,[r2]
                str r3,[r2]
                str r4,[r2]
                lsl r0,#1
                subs r1,#1
                bne     spi32_1
                pop{r4,r5,r6}
                bx      lr

                ENDP

        ALIGN

END
```

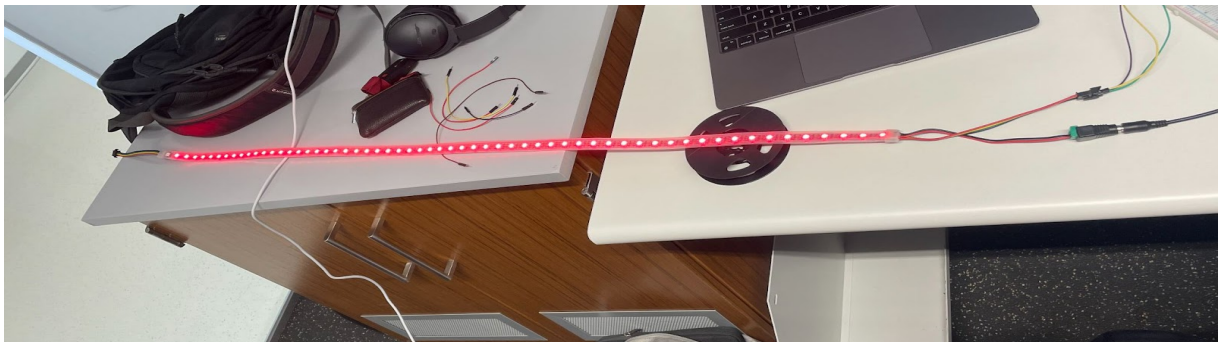


Problem 2

If you have not already written code for spi32, complete that code and use it to make all 60 LEDs red. Demo for the TA and take a picture for your report.

Rgb60:

```
(C) Andrew Wolfe
@file    main_hw_proto.s
@author  Andrew Wolfe
@date    August 18, 2019
@note
        This code is for the book Embedded Systems with ARM Cortex-M
        Microcontrollers in Assembly Language and C, Yifeng Zhu,
        ISBN-13 978-0982692639, ISBN-10 0982692633 as used at Santa Clara University


        INCLUDE core_cm4_constants.s            ; Load Constant Definitions
        INCLUDE stm32l476xx_constants.s
        INCLUDE jstick.h
        INCLUDE leds.h
        INCLUDE rgb60_redact.h



                AREA    main, CODE, READONLY
                EXPORT      __main
                ENTRY

__mainPROC
                mov r7, #60
                bl spisw_init
                mov r8, #15
                ldr r0, =0
                bl spi32

loop        ldr r0, =0xed0000ff
                bl spi32
                subs r7, #1
                bne loop
                ldr r0, =0x00000000
                bl spi32


                ALIGN
            ENDP

        END
```



Problem 3

Copy the project or the code into a new project. Now light up all 60 LEDs in the pattern red/blue/green/white. Demo for the TA and take a picture for your report.

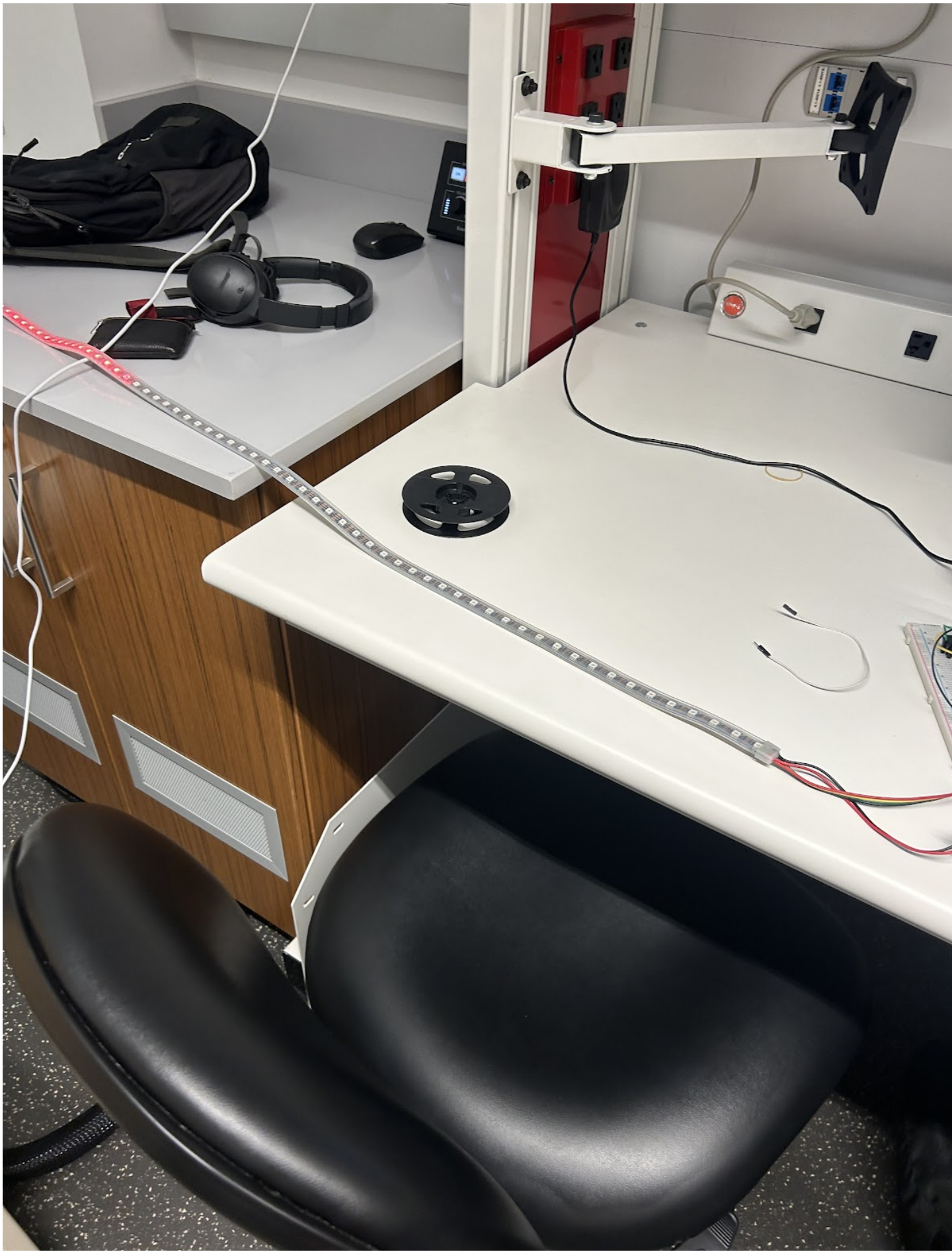
Main:

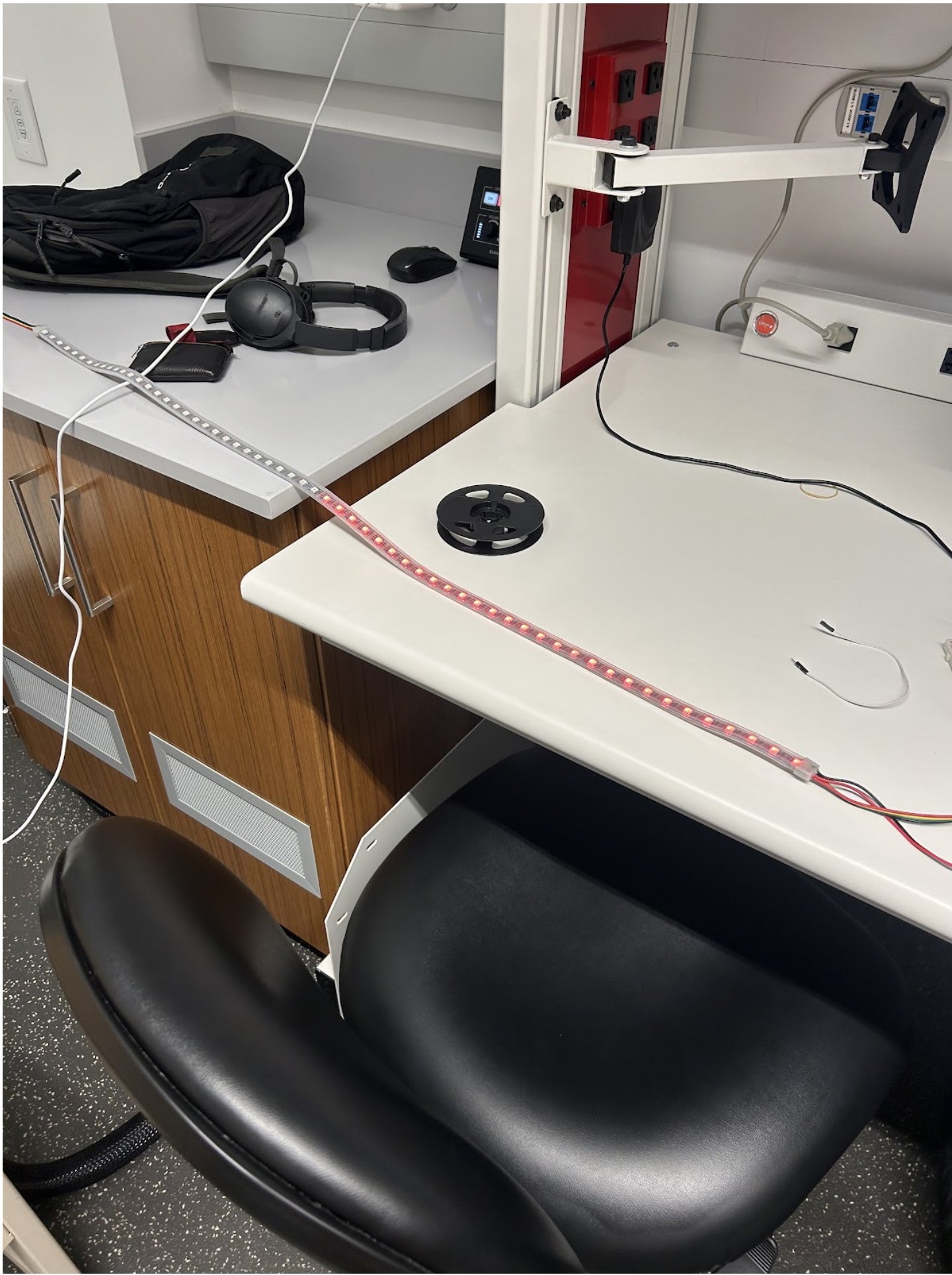
```
******************** (C) Andrew Wolfe **********************************
@file    main_hw_proto.s
@author  Andrew Wolfe
@date    August 18, 2019
@note
        This code is for the book "Embedded Systems with ARM Cortex-M
        Microcontrollers in Assembly Language and C, Yifeng Zhu,
        ISBN-13: 978-0982692639, ISBN-10: 0982692633 as used at Santa Clara
University
*****************************************************************************

        INCLUDE core_cm4_constants.s            ; Load Constant Definitions
        INCLUDE stm32l476xx_constants.s
        INCLUDE jstick.h
        INCLUDE leds.h
        INCLUDE rgb60_redact.h

                AREA    main, CODE, READONLY
                EXPORT      __main
                ENTRY

__mainPROC
                mov r7, #60
                bl spisw_init
                mov r8, #15
                ldr r0, =0
                bl spi32
```

```
loop        ldr r0, =0xe40000ff
            bl spi32
            ldr r0, =0xe4ff0000
            bl spi32
            ldr r0, =0xe400ff00
            bl spi32
            ldr r0, =0xe4ffffff
            bl spi32
            subs r7, #4
            bne loop
            ldr r0, =0x00000000
            bl spi32


            ALIGN
        ENDP

    END
```

Problem 4

Now make it do something really cool that changes over time. It should definitely be cooler than the group next to you. Demo for the TA and take a picture for your report.





MAIN:

```
;******************** (C) Andrew Wolfe
*****************************************
; @file    main_hw_proto.s
; @author  Andrew Wolfe
; @date    August 18, 2019
; @note
;          This code is for the book "Embedded Systems with ARM Cortex-M
;          Microcontrollers in Assembly Language and C, Yifeng Zhu,
;          ISBN-13: 978-0982692639, ISBN-10: 0982692633 as used at Santa Clara University
;*************************************************************************
**



        INCLUDE core_cm4_constants.s        ; Load Constant Definitions
        INCLUDE stm32l476xx_constants.s
        INCLUDE jstick.h
        INCLUDE leds.h
        INCLUDE rgb60_redact.h



                AREA    main, CODE, READONLY
                EXPORT      __main
                ENTRY

__mainPROC
                bl spisw_init
loop        ldr r0, =0x00000000
                bl spi32
                ldr r9, =60
cond        ldr r0, =0xe40000ff
                bl spi32
                bl delay
                sub r9, #1
                cmp r9, #0
                bne cond
                ldr r0, =0x00000000
                bl spi32
                bl clear
                b loop

            ENDP

clear PROC
                ldr r9, =60
```

```
                ldr r0, =0x00000000
                push {lr}
                bl spi32
cond2           ldr r0, =0xe00f0000
                bl spi32
                bl delay
                subs r9, #1
                bne cond2
                ldr r0, =0x00000000
                bl spi32
                pop {lr}
                bx lr
        ENDP

delay PROC
                ldr r0, =0x4000
loop1   sub r0, #1
                cmp r0, #0
                bne loop1
                bx lr
                ENDP
        END
```