Austin Petersen, Ayden Dauenhauer

Prof. Wolfe

ELEN 120L Tuesday 2:15-5:00

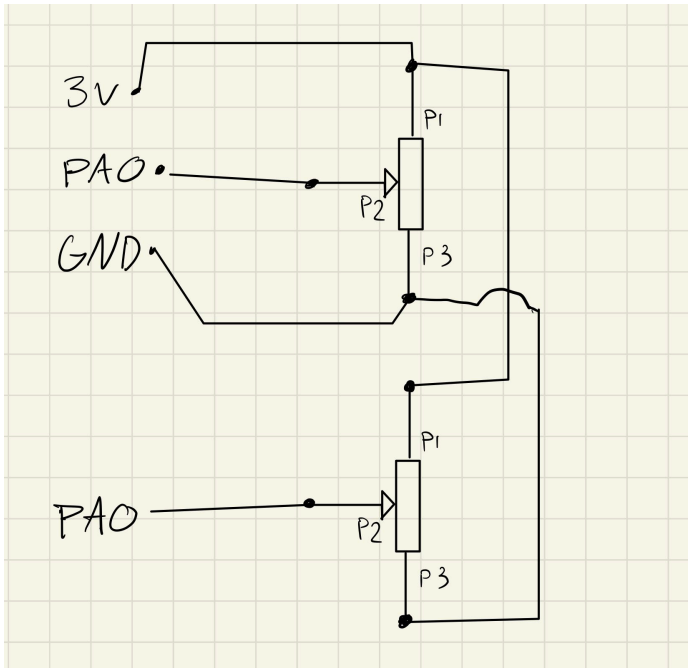
November 14, 2023

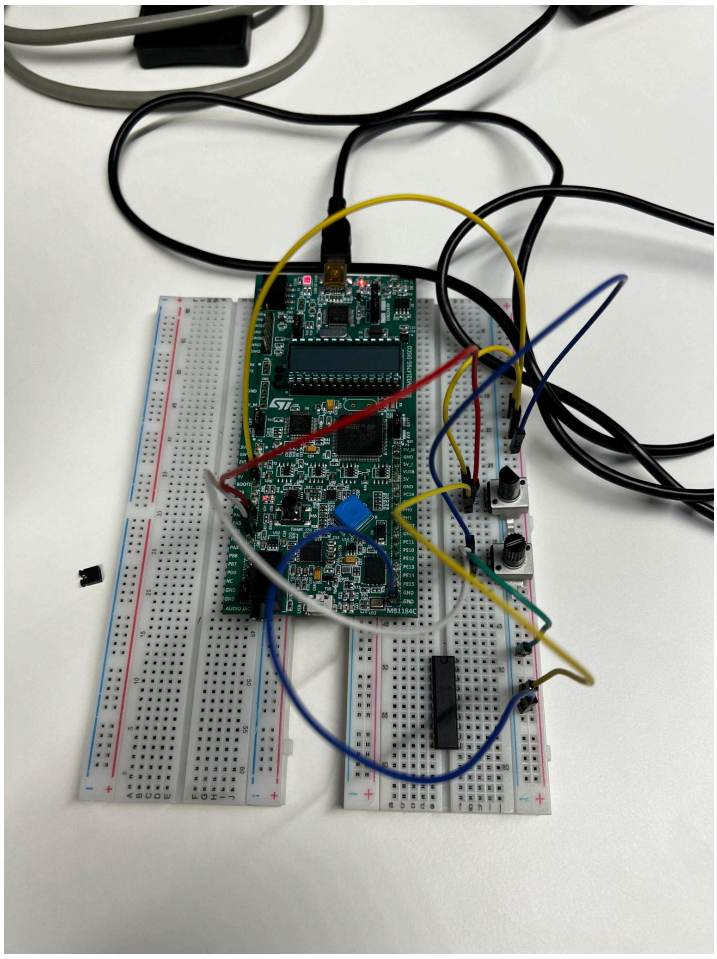
Report 7

Problem 1:

```
108
109 adc_read    PROC        ;return ADC channel 6 value in r0
110             EXPORT  adc_read
111             ldr r0, =(ADC1_BASE+ADC_CR)
112             ldr r1, [r0]
113             ldr r3, =0x8000003f
114             bic r1, r3
115             orr r2, r1, #ADC_CR_ADSTART
116             str r2, [r0]
117 loop        ldr r0, =(ADC1_BASE+ADC_CSR)
118             ldr r1, [r0]
119             cmp r1, #ADC_CSR_EOC_MST
120             beq loop
121             ldr r1, =(ADC1_BASE+ADC_DR)
122             ldr r0, [r1]
123         bx      lr
124             ENDP
125             ALIGN
126             END
```

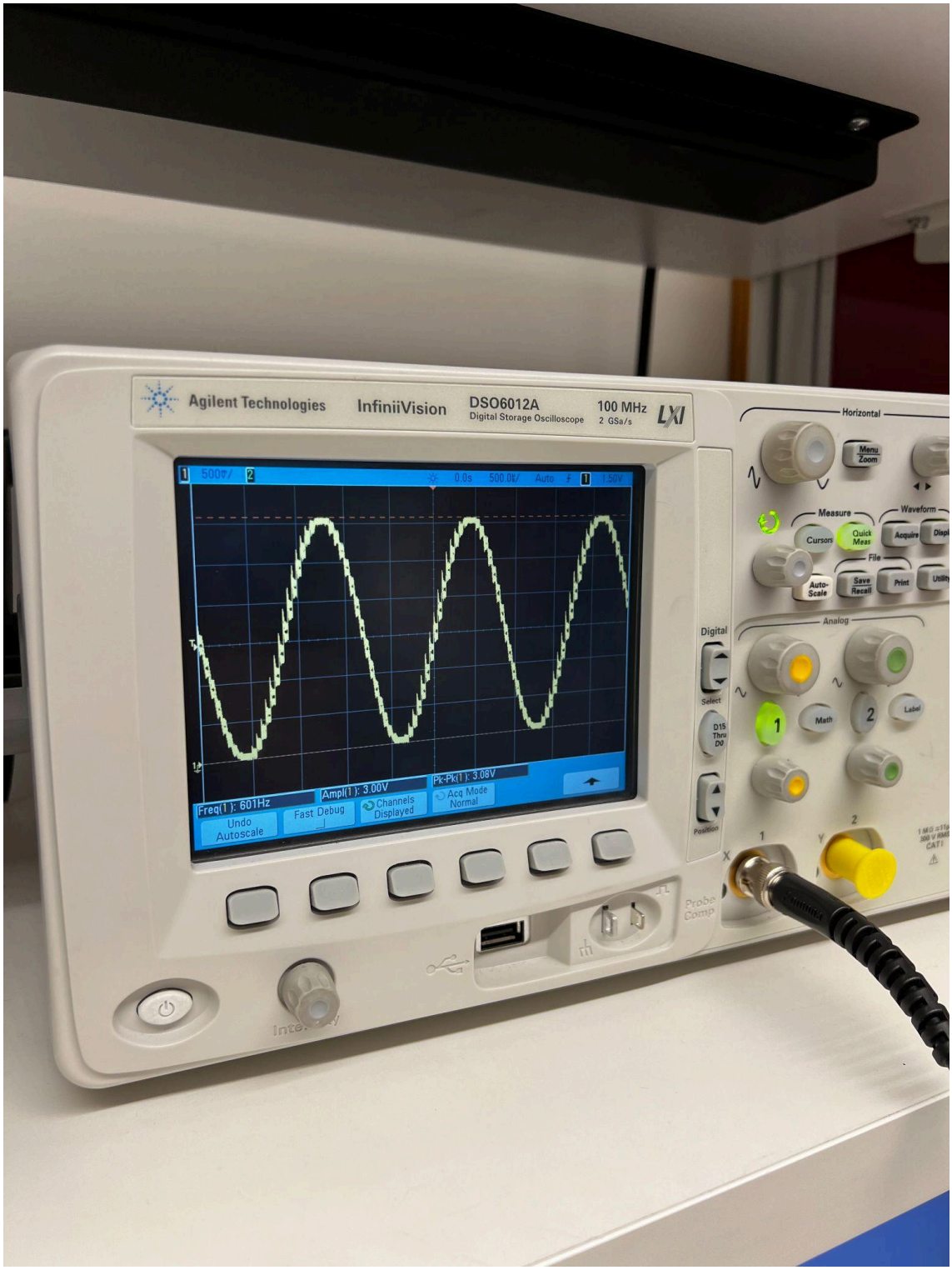
```
;*******************************************************************************



    INCLUDE core_cm4_constants.s        ; Load Constant Definitions
    INCLUDE stm32l476xx_constants.s

    INCLUDE leds.h

    INCLUDE adc.h


            AREA    main, CODE, READONLY
            EXPORT  __main
            ENTRY

__main  PROC

; Add code here to configure the proper GPIO port to drive the red LED.
; You may use the routines provided in leds.s
        ldr r0, =RCC_AHB2ENR_GPIOBEN
        bl      portclock_en
        ldr r0, =GPIOB_BASE
        ldr r1, =GPIO_MODER_MODER2_0
        bl      port_bit_pushpull


        bl      adc_init
loop    bl      adc_read
        cmp     r0, #2048
        blt     roff
        bl      red_on


roff    bl red_off
        b loop
; Add and/or modify code here to repeatedly read the A/D converter and turn the red LED on if
; the reading is greater than or equal to 2048 and turn off the red LED is the reading is less than that.

endless b       endless
        ENDP




            ALIGN
            AREA    myData, DATA, READWRITE

            ALIGN


    END

```


3v
PA0
GND
P1
P2
P3
P1
PA0
P2
P3



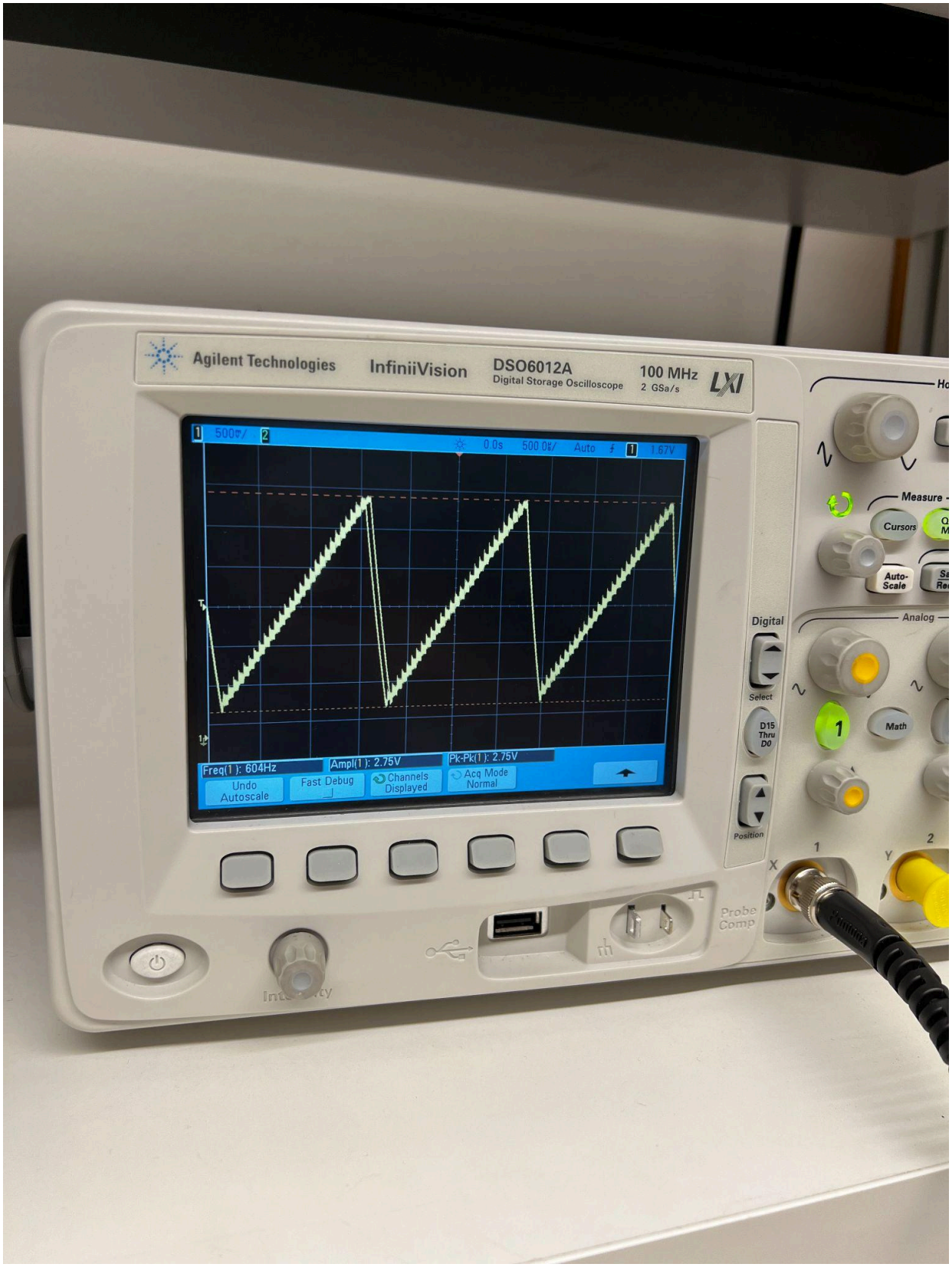
Problem 2:

```
45
46  TIM2_IRQHandler PROC
47          EXPORT  TIM2_IRQHandler
48          push    {lr}
49          ldr     r0,=phase           ;get a pointer to the current phase
50          ldr     r1,=sintbl          ;Get pointer to waveform table
51          bl      get_tblval
52          bl      dac_set
53          ldr     r1,=phaseinc        ;load phase increment
54          ldr     r0,=phase           ;reload last phase value
55          bl      update_phase
56          pop     {lr}
57          ldr     r2,=(TIM2_BASE+TIM_SR)  ;reset pending interrupt for TIM2
58          mov     r1,#~TIM_SR_UIF
59          str     r1,[r2]
60          dsb
61          bx      lr
62          ENDP
63
64  calc_phaseinc   PROC
65                      ; To calculate the phaseinc, take the new frequency (w)/sampling freq.(w0) * 1024
66                      ; to avoid precision issues - we will keep phase in 16ths then divide at the last minute
67                      ; w arrives in r0; phase increment returned in r0
68                      ; works from about 2Hz to sampling freq./2
69                      ; Assumes a wave table size of 1024 and a phase iterator scaled up by 16
70
71          ldr r1, =sample_freq
72          lsl r0, #14
73          udiv r0, r0, r1
74
75          bx      lr
76          ENDP
77
78  update_phase    PROC
79                      ;recieves a pointer to phase in r0 and a pointer to phaseinc in r1
80                      ;adds phaseinc to phase
81          ldr r2, [r0]
82          ldr r3, [r1]
83          add r3, r2
84          bic r3, #0x4000
85          str r3, [r0]
86
87          bx      lr
88          ENDP
89
90  get_tblval      PROC
91                      ;recieves a pointer to phase in r0 and a pointer to a wave table in r1
92                      ;Assume the wave table is 1024 entries; 16-bits each
93                      ;Assume the phase value is in 16ths.
94                      ;Return the sample in r0
95
96          ldr r2, [r0]
97          lsr r2, #4
98          lsl r2, #1
99          add r1, r2
100         ldrh r0, [r1]
101
```



Problem 3:
Same code as problem 2 except:

```
TIM2_IRQHandler PROC
        EXPORT  TIM2_IRQHandler
        push    {lr}
        ldr     r0,=phase           ;get a pointer to the current phase
        ldr     r1,=sawtbl          ;Get pointer to waveform table
        bl      get_tblval
        bl      dac_set
        ldr     r1,=phaseinc        ;load phase increment
        ldr     r0,=phase           ;reload last phase value
        bl      update_phase
        pop     {lr}
        ldr     r2,=(TIM2_BASE+TIM_SR)  ;reset pending interrupt for TIM2
        mov     r1,#~TIM_SR_UIF
        str     r1,[r2]
        dsb
        bx      lr
        ENDP
```


Agilent Technologies
InfiniiVision
DSO6012A
Digital Storage Oscilloscope
100 MHz
2 GSa/s
LXI
Freq(1): 604Hz
Ampl(1): 2.75V
Pk-Pk(1): 2.75V
Undo Autoscale
Fast Debug
Channels Displayed
Acq Mode Normal
Measure
Cursors
Auto-Scale
Digital
Select
D15 Thru D0
Position
Analog
Math
Probe Comp

Problem 4:

```
__main  PROC

        ldr     r0,=test_freq
        bl      calc_phaseinc       ;compute the phase increment value (phaseinc)
        ldr     r2,=phaseinc
        str     r0,[r2]             ;store the phase increment value in memory

        bl      dac_init            ;initialize dac
        bl      tim2_init           ;initialize timer interrupt
        ldr     r0,=sample_per      ;set output rate to 20KHz
        bl      tim2_freq

        bl adc_init
loop    bl adc_read
        ldr r1, =gain
        str r0, [r1]
        b loop

endless b       endless
        ENDP
```

```
get_tblval      PROC
                    ;recieves a pointer to phase in r0 and a pointer to a wave table in r1
                    ;Assume the wave table is 1024 entries; 16-bits each
                    ;Assume the phase value is in 16ths.
                    ;Return the sample in r0
        push {r4}
        ldr r2, [r0]
        lsr r2, #4
        lsl r2, #1
        add r1, r2
        ldrh r0, [r1]

        ldr r3, =gain
        ldr r4, [r3]
        mul r0, r4
        lsr r0, #12
        pop {r4}

        bx      lr
        ENDP
```