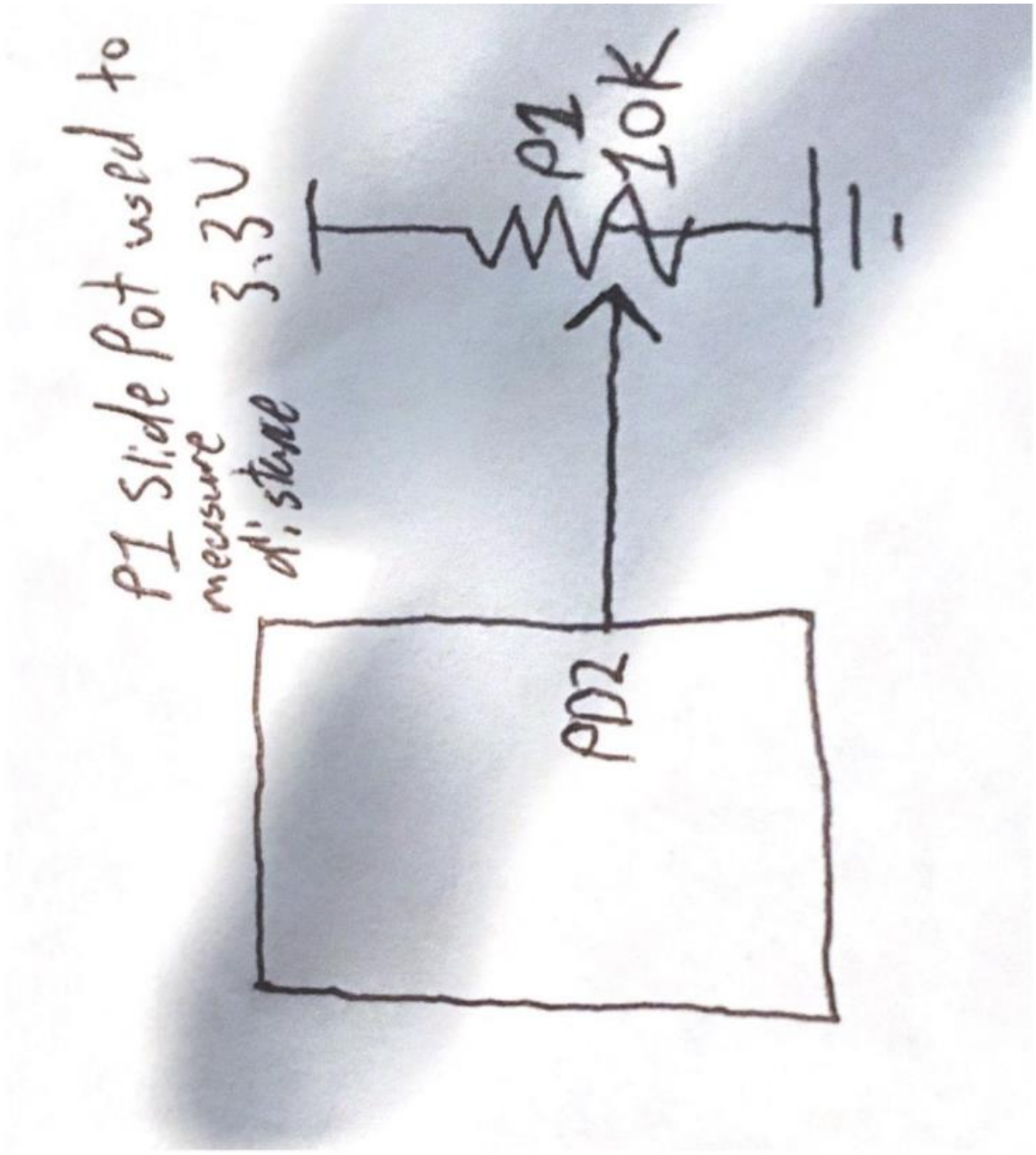
Deliverables:

1)



2)

Time to Sample In uS: 129
Time to Print in uS: 4841

3)

Position	Analog input	ADC sample	Correct Fixed-point	Measured Fixed-point Output
0.10 cm	0.2	390	10	20
0.40 cm	0.4	800	40	40
0.80 cm	.79	1560	80	790
1.20 cm	1.28	2500	120	127
1.40 cm	1.51	2940	140	150

4)

```
// Lab8.cpp
// Runs on LM4F120 or TM4C123
// Student names: change this to your names or look very silly
// Last modification date: change this to the last modification date or look very silly
// Last Modified: 1/17/2020

// Specifications:
// Measure distance using slide pot, sample at 60 Hz
// maximum distance can be any value from 1.5 to 2cm
// minimum distance is 0 cm
// Calculate distance in fixed point, 0.01cm
// Analog Input connected to PD2=ADC0 (OG 5)
// displays distance on Sitronox ST7735
// PF3, PF2, PF1 are heartbeats (use them in creative ways)
// must include at least one class used in an appropriate way

#define Main //Replace with Main, MainOne, MainTwo, or MainThree depending on current test
```

```
#define CONVERSION_NUMERATOR 210
#define CONVERSION_DENOMINATOR 4096
#define CONVERSION_OFFSET 10

#include <stdint.h>
#include "../inc/tm4c123gh6pm.h"
#include "PLL.h"
#include "ST7735.h"
#include "TExaS.h"
#include "PLL.h"
#include "SlidePot.h"
#include "print.h"

SlidePot Sensor(CONVERSION_NUMERATOR,CONVERSION_DENOMINATOR, CONVERSION_OFFSET);

extern "C" void DisableInterrupts(void);
extern "C" void EnableInterrupts(void);
extern "C" void SysTick_Handler(void);

#define PF1      (*((volatile uint32_t *)0x40025008))
#define PF2      (*((volatile uint32_t *)0x40025010))
#define PF3      (*((volatile uint32_t *)0x40025020))
#define PF4      (*((volatile uint32_t *)0x40025040))
// **************SysTick_Init*********************
// Initialize Systick periodic interrupts
// Input: interrupt period
//        Units of period are 12.5ns
//        Maximum is 2^24-1
//        Minimum is determined by length of ISR
```

```
// Output: none
void SysTick_Init(unsigned long period){
  //*** students write this ******
        NVIC_ST_CTRL_R = 0;
        NVIC_ST_RELOAD_R = period;
        NVIC_ST_CURRENT_R = 0;
        NVIC_ST_CTRL_R = 7;
}

// Initialize Port F so PF1, PF2 and PF3 are heartbeats
void PortF_Init(void){
  //*** students write this ******
        SYSCTL_RCGCGPIO_R |= 0x20;
        __asm__ {
                NOP
                NOP
        }
        GPIO_PORTF_DIR_R |= 0x04;
        GPIO_PORTF_DEN_R |= 0x04;

}

void toggleHearBeat(void) {
        GPIO_PORTF_DATA_R ^= 0x04;
}

uint32_t Data;      // 12-bit ADC
uint32_t Position;  // 32-bit fixed-point 0.01 cm
```

```
#ifdef MainOne
int main(void){     // single step this program and look at Data
  DisableInterrupts();
  TExaS_Init();     // start scope set system clock to 80 MHz
  ADC_Init();       // turn on ADC, PD2, set channel to 5
  EnableInterrupts();
  while(1){
    Data = ADC_In();  // sample 12-bit channel 5, PD2
  }
}
#endif

#ifdef MainTwo
uint32_t time0,time1,time2,time3;
uint32_t ADCtime,OutDectime; // in usec
int main(void){
  TExaS_Init();    // Bus clock is 80 MHz
  NVIC_ST_RELOAD_R = 0x00FFFFFF; // maximum reload value
  NVIC_ST_CURRENT_R = 0;        // any write to current clears it
  NVIC_ST_CTRL_R = 5;
  ADC_Init();       // turn on ADC, set channel to 5
  ADC0_SAC_R = 4;   // 16-point averaging, move this line into your ADC_Init()
  ST7735_InitR(INITR_GREENTAB);
  while(1){         // use SysTick
    time0 = NVIC_ST_CURRENT_R;
    Data = ADC_In();  // sample 12-bit channel 5
    time1 = NVIC_ST_CURRENT_R;
    ST7735_SetCursor(0,0);
    time2 = NVIC_ST_CURRENT_R;
```

```
    LCD_OutDec(Data);
    ST7735_OutString("    ");  // spaces cover up characters from last output
    time3 = NVIC_ST_CURRENT_R;
    ADCtime = ((time0-time1)&0x0FFFFFF)/80;      // usec
    OutDectime = ((time2-time3)&0x0FFFFFF)/80; // usec
  }
}

#endif

#ifdef MainThree
int main(void){
  DisableInterrupts();
  TExaS_Init();        // Bus clock is 80 MHz
  ST7735_InitR(INITR_GREENTAB);
  PortF_Init();
  ADC_Init();       // turn on ADC, PD2, set channel to 5
  EnableInterrupts();
  while(1){
    PF2 ^= 0x04;      // Heartbeat
    Data = ADC_In();  // sample 12-bit channel 5, PD2
    PF3 = 0x08;       // Profile Convert
    Position = Sensor.Convert(Data);
    PF3 = 0;          // end of Convert Profile
    PF1 = 0x02;       // Profile LCD
    ST7735_SetCursor(0,0);
    LCD_OutDec(Data); ST7735_OutString("    ");
    ST7735_SetCursor(6,0);
    LCD_OutFix(Position);
```

```
  PF1 = 0;        // end of LCD Profile
 }
}
#endif



#ifdef Main
// final main program to create distance meter
int main(void){
  //*** students write this ******

 DisableInterrupts();
 TExaS_Init();  // bus clock at 80 MHz
 ST7735_InitR(INITR_GREENTAB);
 ADC_Init();      // turn on ADC, PD2, set channel to 5
          SysTick_Init(8000000);
 PortF_Init();

 // more initializations
 EnableInterrupts();

 while(1){
  Sensor.Sync(); // wait for semaphore
  // can call Sensor.ADCsample, Sensor.Distance, Sensor.Convert as needed
  uint32_t distance = Sensor.Distance();
                    ST7735_SetCursor(0,0);
                    LCD_OutFix(distance);
                    ST7735_OutString(" cm");
 }
```

```
}
#endif

void SysTick_Handler(void){ // every sample
   //*** students write this ******
// should call ADC_In() and Sensor.Save
            toggleHearBeat();
 Sensor.Save(ADC_In());
}

// SlidePot.cpp
// Runs on LM4F120/TM4C123
// Provide functions that initialize ADC0 and use a slide pot to measure distance
// Created: 3/28/2018
// Student names: change this to your names or look very silly
// Last modification date: change this to the last modification date or look very silly

#define Simulator //change to Hardware or Simulator depending on environment

#include <stdint.h>
#include "SlidePot.h"
#include "../inc/tm4c123gh6pm.h"

// ADC initialization function
// Input: none
// Output: none
// measures from PD2, analog channel 5
void ADC_Init(void){
//*** students write this ******
```

```
    SYSCTL_RCGCADC_R |= 0x01; //enable ADC0
    SYSCTL_RCGCGPIO_R |= 1<<3; //enable Port D

    __nop();
    __nop(); //wait for clock

    GPIO_PORTD_DIR_R |= 1<<2;
    GPIO_PORTD_AFSEL_R |= 1<<2;
    GPIO_PORTD_DEN_R &= ~(1<<2);
    GPIO_PORTD_AMSEL_R |= 1<<2;

    ADC0_PC_R &= ~(0x0F);
    ADC0_PC_R |= 0x01;
    ADC0_SSPRI_R = 0x0123;
    ADC0_ACTSS_R &= ~0x0008;
    ADC0_EMUX_R &= ~0xF000;
    ADC0_SSMUX3_R &= ~0x000F;
    ADC0_SSMUX3_R += 5;
    ADC0_SSCTL3_R = 0x0006;
    ADC0_IM_R &= ~0x0008;
    ADC0_ACTSS_R |= 0x0008;

    #ifdef Hardware
    ADC0_SAC_R = 0x06;
    #endif

    //copied from textbook
}
```

```
//------------ADCIn------------
// Busy-wait Analog to digital conversion
// Input: none
// Output: 12-bit result of ADC conversion
// measures from PD2, analog channel 5
uint32_t ADC_In(void){
//*** students write this ******

        uint32_t result;
        ADC0_PSSI_R = 0x0008;
        while((ADC0_RIS_R&0x08)==0){};
        result = ADC0_SSFIFO3_R&0xFFF;
        ADC0_ISC_R = 0x0008;
        return result;

}

// constructor, invoked on creation of class
// m and b are linear calibration coeffients
SlidePot::SlidePot(uint32_t mT, uint32_t mD, uint32_t b){
//*** students write this ******
// initialize all private variables
// make slope equal to m and offset equal to b
        slopeNumerator = mT;
        slopeDenominator = mD;
        offset = b;
}
```

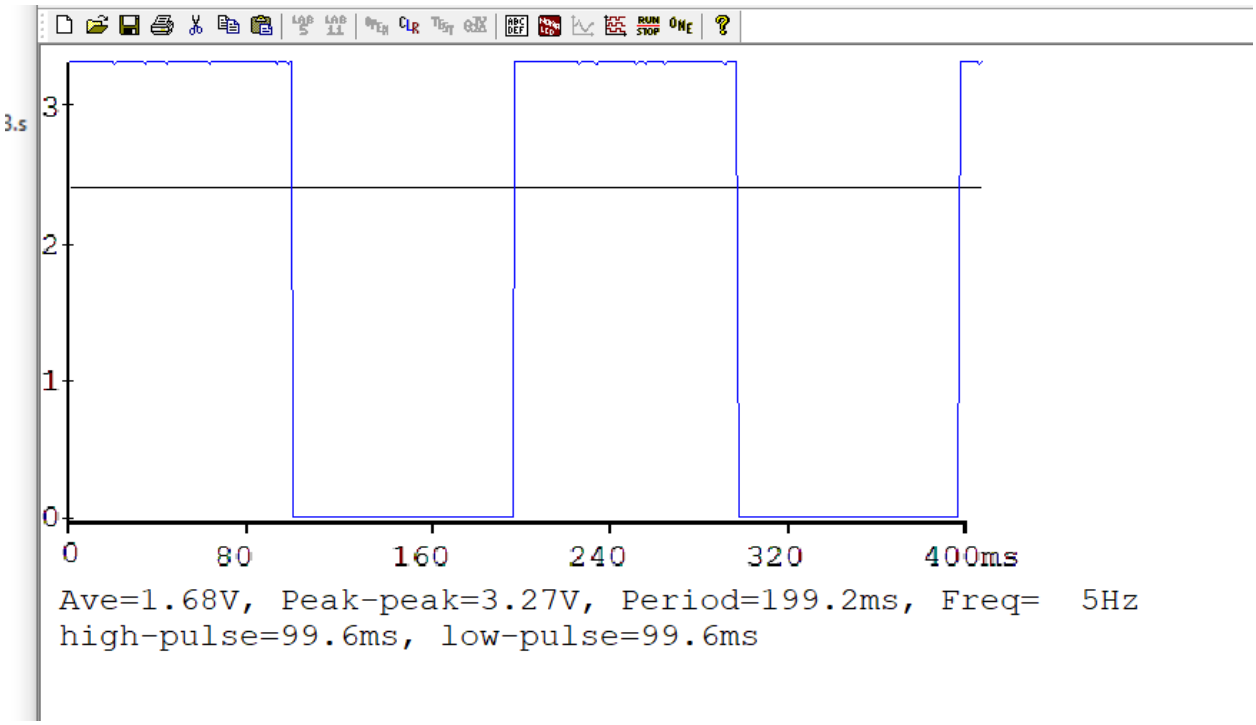
```
void SlidePot::Save(uint32_t n){
//*** students write this ******
// 1) save ADC sample into private variable
// 2) calculate distance from ADC, save into private variable
// 3) set semaphore flag = 1
        data = n;
        distance = Convert(data);
        flag = 1;
}
uint32_t SlidePot::Convert(uint32_t n){
 //*** students write this ******
 // use calibration data to convert ADC sample to distance
 return (slopeNumerator*n - offset )/slopeDenominator;
        //TODO: Verify overflow? Idk why but the lab manual says to watch out for it here
}

void SlidePot::Sync(void){
// 1) wait for semaphore flag to be nonzero
// 2) set semaphore flag to 0
        while(flag==0);
        flag = 0;
}

uint32_t SlidePot::ADCsample(void){ // return ADC sample value (0 to 4095)
 //*** students write this ******
 // return last calculated ADC sample
 return data; // replace this with solution
}
```

```
uint32_t SlidePot::Distance(void){  // return distance value (0 to 2000), 0.001cm
  //*** students write this ******
  // return last calculated distance in 0.001cm
  return distance; // replace this with solution
}
```

5)



6)

Average accuracy (with units in cm) = .105cm

True position x_{ti}	Measured Position x_{mi}	Error $x_{ti} - x_{mi}$
.1	.2	-.1

.4	.4	0
.8	.79	.01
1.2	1.51	-.31

Table 8.2. Accuracy results of the position measurement system.