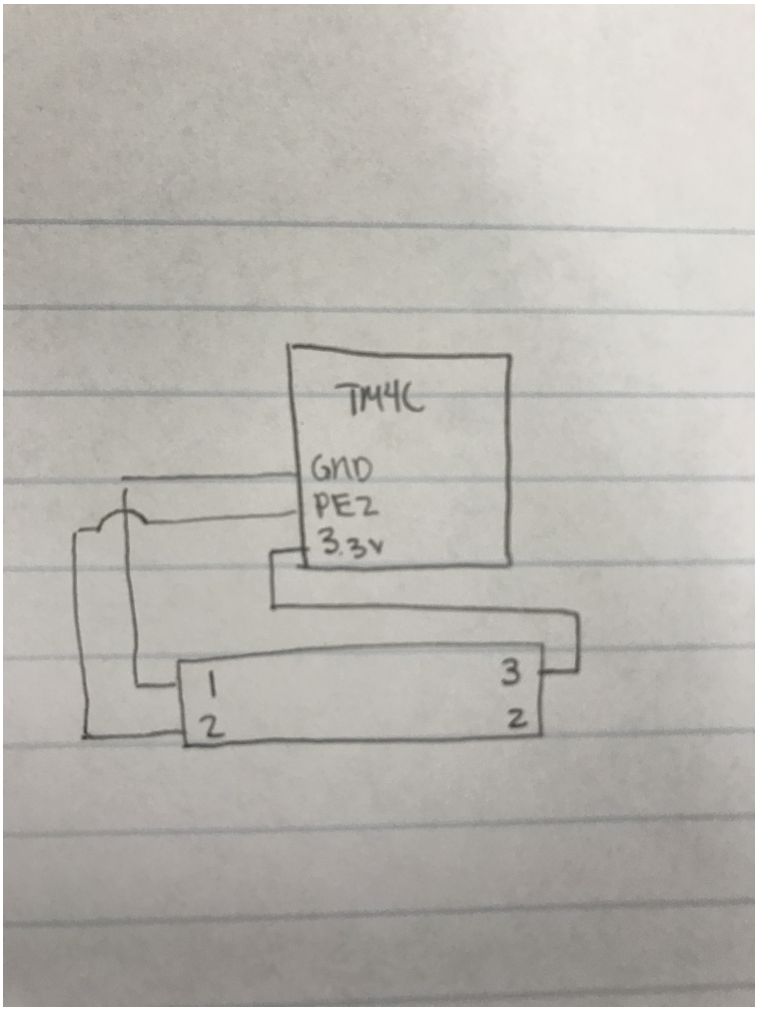
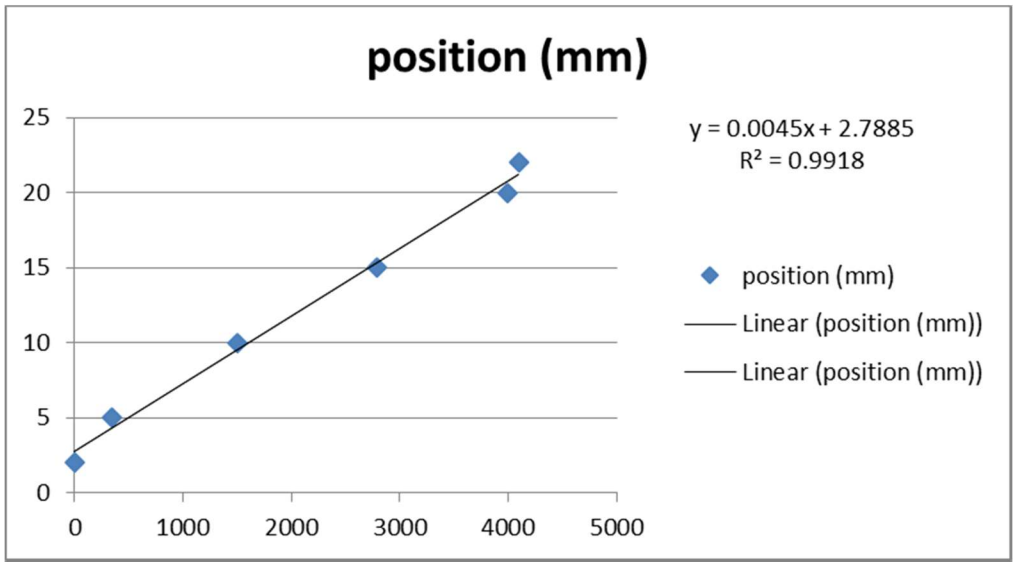
Lab 8 Deliverables

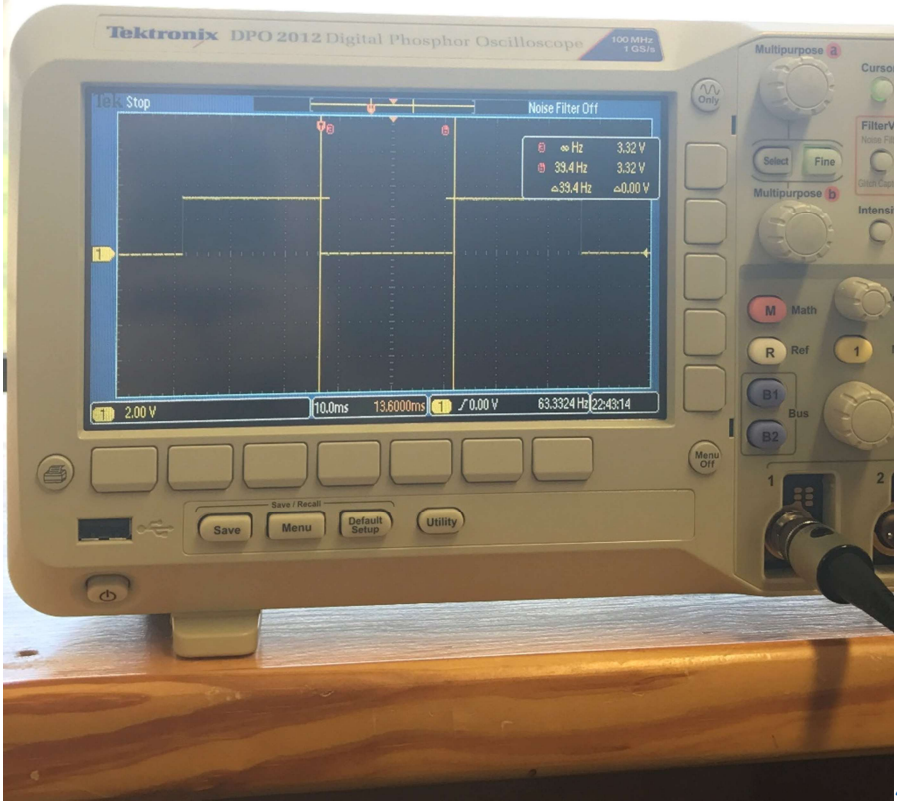
Dylan Cauwels and Andrew Han

Hand Drawn Circuit



Calibration Data





40Hz Screenshot

Time Measurement

position	fifo3	ADC sample	position (mm)	analog input
0cm		0	2	.1 mV
.5cm		340	5	.293V
1cm		1500	10	1.216V
1.5cm		2790	15	2.17V
2cm		4000	20	3.18V
2.2cm		4095	22	3.29V

Code

```
// Lab8.c
// 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: 4/5/2016

// Analog Input connected to PE2=ADC1
// displays on Sitronox ST7735
// PF3, PF2, PF1 are heartbeats


#include <stdint.h>

#include "ST7735.h"
#include "TExaS.h"
#include "ADC.h"
#include "print.h"
#include "tm4c123gh6pm.h"

//*****the first three main programs are for debugging *****
// main1 tests just the ADC and slide pot, use debugger to see data
// main2 adds the LCD to the ADC and slide pot, ADC data is on Nokia
// main3 adds your convert function, position data is no Nokia

void DisableInterrupts(void); // Disable interrupts
void EnableInterrupts(void);  // Enable interrupts
uint32_t ADCStatus = 0;
uint32_t ADCMail;
#define PF1       (*((volatile uint32_t *)0x40025008))
#define PF2       (*((volatile uint32_t *)0x40025010))
#define PF3       (*((volatile uint32_t *)0x40025020))
// Initialize Port F so PF1, PF2 and PF3 are heartbeats
void PortF_Init(void){
volatile  uint32_t delay;
  SYSCTL_RCGCGPIO_R |= 0x20;
  delay = SYSCTL_RCGCGPIO_R;
  delay = SYSCTL_RCGCGPIO_R;
  delay = SYSCTL_RCGCGPIO_R;
  delay = SYSCTL_RCGCGPIO_R;
  GPIO_PORTF_LOCK_R = 0x4C4F434B;   // unlock Port F
  GPIO_PORTF_CR_R |= 0x0E;           // allow changes to PF1-3
  GPIO_PORTF_AMSEL_R = 0x00;        // disable analog on PF
  GPIO_PORTF_PCTL_R = 0x00000000;   // PCTL GPIO on PF
  GPIO_PORTF_AFSEL_R = 0x00;        // disable alt funct on PF
  GPIO_PORTF_DIR_R |= 0x0E;
  GPIO_PORTF_DEN_R |= 0X0E;
}
uint32_t Data;        // 12-bit ADC
uint32_t Position;    // 32-bit fixed-point 0.001 cm
int main1(void){      // single step this program and look at Data
  TExaS_Init();       // Bus clock is 80 MHz
  ADC_Init();         // turn on ADC, set channel to 1
  while(1){
    Data = ADC_In();  // sample 12-bit channel 1
  }
}

int main2(void){
  TExaS_Init();       // Bus clock is 80 MHz
  ADC_Init();         // turn on ADC, set channel to 1
  ST7735_InitR(INITR_REDTAB);
  PortF_Init();
  while(1){           // use scope to measure execution time for ADC_In and LCD_OutDec
    PF2 = 0x04;       // Profile ADC
    Data = ADC_In();  // sample 12-bit channel 1
    PF2 = 0x00;       // end of ADC Profile
    ST7735_SetCursor(0,0);
    PF1 = 0x02;       // Profile LCD
    LCD_OutDec(Data);
    ST7735_OutString("    ");  // these spaces are used to coverup characters from last output
    PF1 = 0;          // end of LCD Profile
  }
```

```
  }
  uint32_t Convert(uint32_t input){
    return(((45*input)+27885)/100);
  }
  void SysTick_Init(void){
    NVIC_ST_CTRL_R = 0;
    NVIC_ST_RELOAD_R = 955238;
    NVIC_ST_CURRENT_R = 0;
    NVIC_ST_CTRL_R |= 0x07;
  }
  void SysTick_Handler(void){
    PF3 ^= 0x08;
    PF3 ^= 0x08;
    ADCMail = ADC_In();
    ADCStatus = 1;
    PF3 ^= 0x08;
  }

  int main3(void){
    TExaS_Init();          // Bus clock is 80 MHz
    ST7735_InitR(INITR_REDTAB);
    PortF_Init();
    ADC_Init();          // turn on ADC, set channel to 1
    while(1){
      PF2 ^= 0x04;       // Heartbeat
      Data = ADC_In();   // sample 12-bit channel 1
      PF3 = 0x08;        // Profile Convert
      Position = 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
    }
  }
  int main(void){
    TExaS_Init();
    ST7735_InitR(INITR_REDTAB);
    PortF_Init();
    ADC_Init();
    SysTick_Init();
    while(1){
      if(ADCStatus == 1){
        ST7735_SetCursor(0,0);
        LCD_OutFix(Convert(ADCMail));
        ST7735_OutString(" cm");
        ADCStatus = 0;
      }
    }
  }
```

```c
// ADC.c
// Runs on LM4F120/TM4C123
// Provide functions that initialize ADC0
// Last Modified: 3/6/2015
// 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

#include <stdint.h>
#include "tm4c123gh6pm.h"

// ADC initialization function
// Input: none
// Output: none
void ADC_Init(void){
      uint32_t delay;
      SYSCTL_RCGCGPIO_R |= 0x10;                  // 1) activate clock on portE
      while((SYSCTL_PRGPIO_R&0x10) != 0x10){};
      delay = SYSCTL_RCGCGPIO_R;                  // 2) extra time for clock to stabilize
      delay = SYSCTL_RCGCGPIO_R;
      //      Ain1 is on PE2
      GPIO_PORTE_DIR_R &= ~0x04;                  // 3) make PE2 input
      GPIO_PORTE_AFSEL_R |= 0x04;                 // 4) enable alternate function on PE2
      GPIO_PORTE_DEN_R &= ~0x04;                  // 5) disable digital I/O on PE2
      GPIO_PORTE_AMSEL_R |= 0x04;                 // 6) enable analog functionality on PE2

      SYSCTL_RCGCADC_R |= 0x0001;                 // 7) activate ADC0
//      while((SYSCTL_PRADC_R&0x0001) != 0x0001){};
      delay = SYSCTL_RCGCADC_R;                   // extra time for clock to stabilize
      delay = SYSCTL_RCGCADC_R;                   // extra time for clock to stabilize
      delay = SYSCTL_RCGCADC_R;                   // extra time for clock to stabilize
      delay = SYSCTL_RCGCADC_R;
      ADC0_PC_R &= ~0xF;                          // 9) clear max sample rate field
      ADC0_PC_R |= 0x1;                           //    configure for 125K samples/sec
      ADC0_SSPRI_R = 0x0123;                      // 10) Sequencer 3 is lowest priority
      ADC0_ACTSS_R &= ~0x0008;                    // 11) disable sample sequencer 3
      ADC0_EMUX_R |= 0xF000;                      // 12) seq3 is continuous trigger
      ADC0_SSMUX3_R &= ~0x000F;                   // 13) clear SS3 field
      ADC0_SSMUX3_R += 0x01;                      // set channel
      ADC0_SSCTL3_R = 0x0006;                     // 14) no TS0 D0, yes IE0 END0
      ADC0_IM_R &= ~0x0008;                       // 15) disable SS3 interrupts
      ADC0_ACTSS_R |= 0x0008;                     // 16) enable sample sequencer 3

}

//------------ADC_In------------
// Busy-wait Analog to digital conversion
// Input: none
// Output: 12-bit result of ADC conversion
uint32_t ADC_In(void){
  uint32_t data;
  ADC0_PSSI_R = 0x0008;               // 1) initiate SS3
  while((ADC0_RIS_R & 0x08) == 0){}   // 2) wait for conversion done
  data = ADC0_SSFIFO3_R & 0xFFF;      // 3) read result
  ADC0_ISC_R = 0x0008;                // 4) acknowledge completion
  return data;
}
```