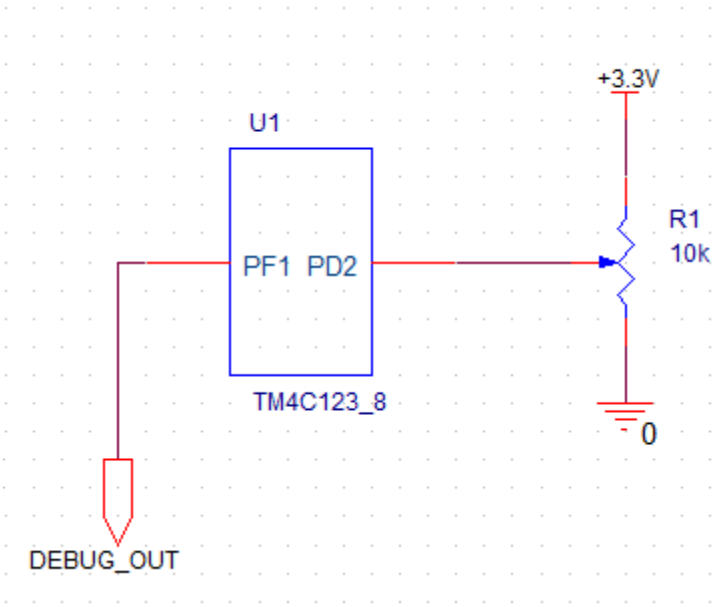
# Lab 8 Deliverables

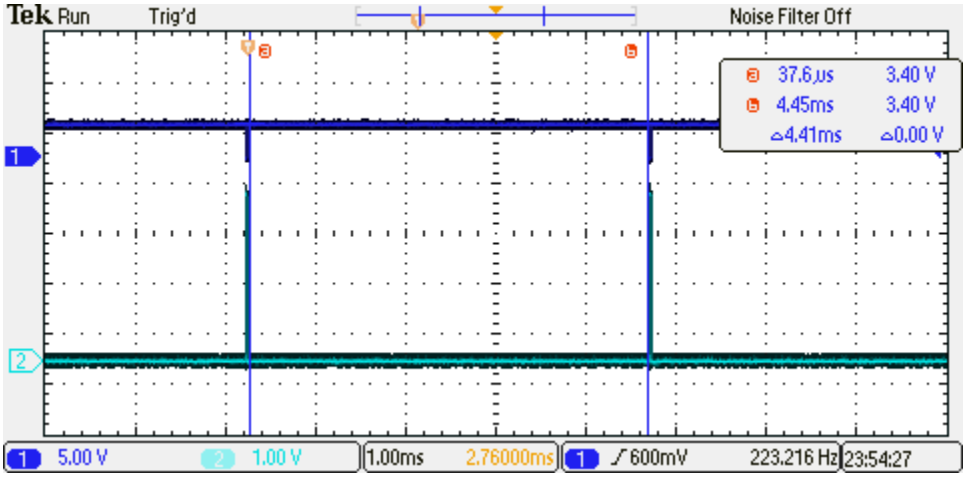
Eric Chen & Dennis Liu

1. Circuit Diagram

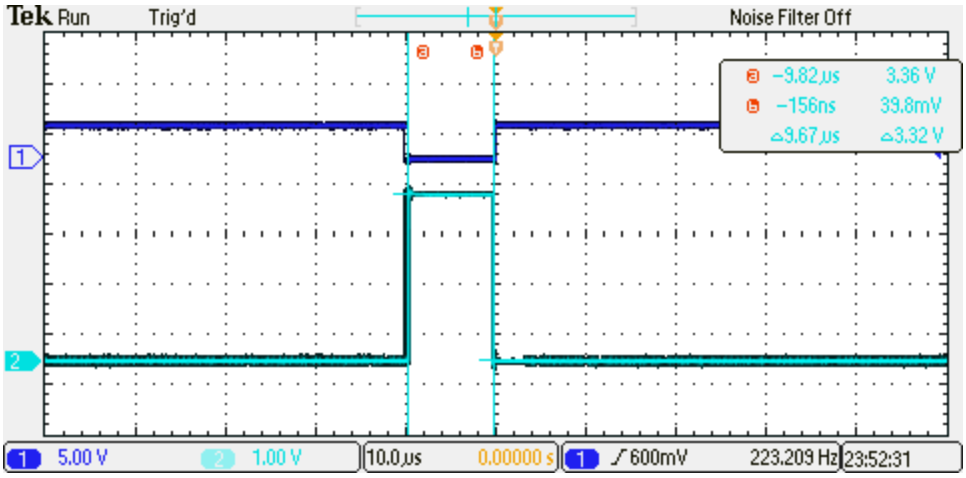



2. Time measurements/Photo for ADC/LCD execution time

LCD measurement: 4.48 ms




ADC_IN() measurement: 9.8 microseconds

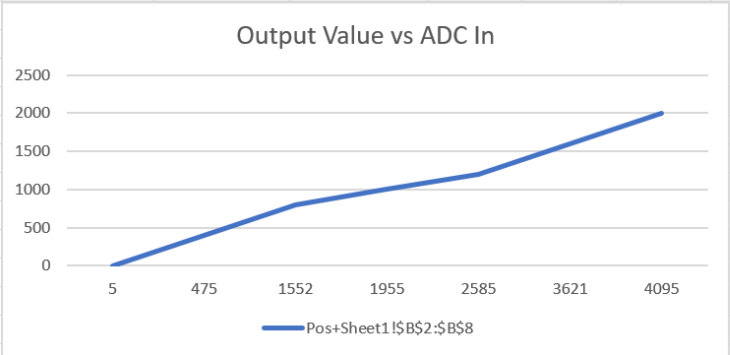



3. Calibration Data

| Position (mm) | Output Val | Voltage | ADC Value |
|---|---|---|---|
| 0 | 0 | 0 | 5 |
| 4 | 400 | 0.3812 | 475 |
| 8 | 800 | 1.278 | 1552 |
| 10 | 1000 | 1.6104 | 1955 |
| 12 | 1200 | 2.1235 | 2585 |
| 16 | 1600 | 2.9276 | 3621 |
| 20 | 2000 | 3.2959 | 4095 |

| Regression Statistics | |
|---|---|
| Multiple R | 0.99287517 |
| R Square | 0.985801103 |
| Adjusted R Square | 0.982961323 |
| Standard Error | 89.17052527 |
| Observations | 7 |

| | Coefficients | Standard Error | t Stat | P-value | Lower 95% | Upper 95% | ower 95.0% | pper 95.0% |
|---|---|---|---|---|---|---|---|---|
| Intercept | 88.96050237 | 59.38735214 | 1.497970513 | 0.19441 | -63.6995 | 241.6206 | -63.6995 | 241.6206 |
| Slope | 0.44633794 | 0.023955843 | 18.6316939 | 8.2E-06 | 0.384757 | 0.507918 | 0.384757 | 0.507918 |



## 4. Code

```
void PortF_Init(void){
        volatile uint32_t delay;
        SYSCTL_RCGCGPIO_R |= 0x20;
        delay = 100;
        delay++;
        GPIO_PORTF_DIR_R |= 0x0E;
        GPIO_PORTF_DEN_R |= 0x0E;
}

void SysTick_Init(void){
        // write this
        NVIC_ST_CTRL_R = 0;
        NVIC_ST_RELOAD_R = 0x145855;
        NVIC_ST_CURRENT_R = 0;

        NVIC_ST_CTRL_R = 0x7;
}
```

```
uint32_t Convert(uint32_t input){
  return input*weight+bias;
}
void SysTick_Handler(void){
  GPIO_PORTF_DATA_R ^=0x02;
  GPIO_PORTF_DATA_R ^=0x02;
  ADCstatus = 1;
  ADCmail = ADC_In();
}

void IO_Touch(void) {
  uint8_t state;
  while(1){
    state = GPIO_PORTF_DATA_R;
    state &= 0x10;
    if(GPIO_PORTF_DATA_R == 0){
      DelayIms(20);
      state = GPIO_PORTF_DATA_R;
      state &= 0x10;
      if(GPIO_PORTF_DATA_R==0x10){
        return;
      }
    }
  }
```

```
float mean(void){
  int32_t sum=0;
  for(int i=0;i<7;i++){
    sum+=adcval[i];
  }
  return sum/7;
}
float variance(int32_t mean){
  int32_t var=0;
  for(int i=0;i<7;i++){
    var+=(adcval[i]-mean)*(adcval[i]-mean);
  }
  return var;
}
float covar(int32_t mean){
  int32_t covar=0;
  for(int i=0;i<7;i++){
    covar+=(adcval[i]-mean)*(dist[i]-ymean);
  }
  return covar;
}
```

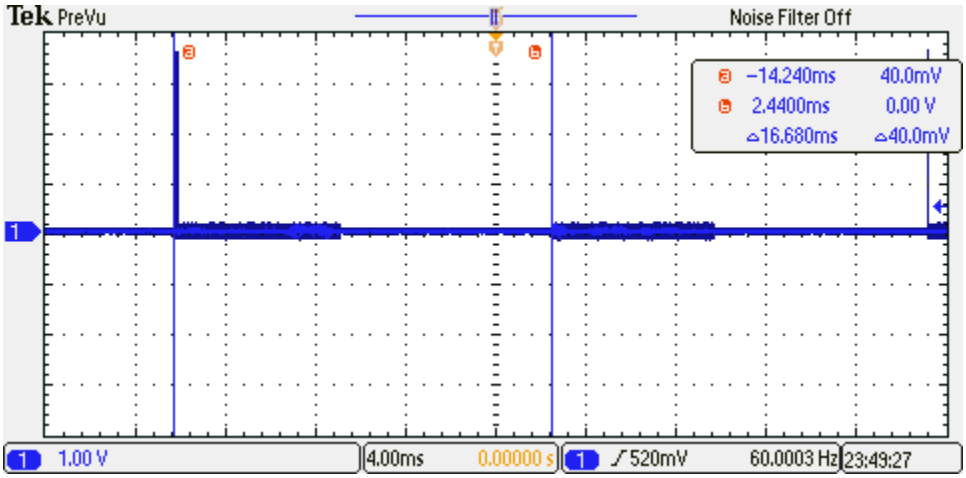
```
int main(void){
  PortF_Init();
  TExaS_Init();
  // your Lab 8
  EnableInterrupts();
  SysTick_Init();
  ADC_Init();
  ST7735_InitR(INITR_REDTAB);
  AutoCal();
  ADCstatus = -1;
  while(1){
    while(ADCstatus<0){};
    LCD_OutFix(Convert(ADCmail));
    ST7735_SetCursor(0,0);
    ADCstatus = -1;
  }
}
```

```
void ADC_Init(void){
        SYSCTL_RCGCGPIO_R |= 0x9;
        uint8_t i=0;
        i++;
        i++;

        GPIO_PORTD_DIR_R &= ~0x04;      // 2) make PD2 input
  GPIO_PORTD_AFSEL_R |= 0x04;     // 3) enable alternate fun on PD2
  GPIO_PORTD_DEN_R &= ~0x04;      // 4) disable digital I/O on PD2
  GPIO_PORTD_AMSEL_R |= 0x04;     // 5) enable analog fun on PD2
  SYSCTL_RCGCADC_R |= 0x01;       // 6) activate ADC0
  uint32_t delay = SYSCTL_RCGCADC_R;       // extra time to stabilize
  delay = SYSCTL_RCGCADC_R;       // extra time to stabilize
  delay = SYSCTL_RCGCADC_R;       // extra time to stabilize
  delay = SYSCTL_RCGCADC_R;
  ADC0_PC_R = 0x01;               // 7) configure for 125K
  ADC0_SSPRI_R = 0x0123;          // 8) Seq 3 is highest priority
  ADC0_ACTSS_R &= ~0x0008;        // 9) disable sample sequencer 3
  ADC0_EMUX_R &= ~0xF000;         // 10) seq3 is software trigger
  ADC0_SSMUX3_R = (ADC0_SSMUX3_R&0xFFFFFFF0)+5;  // 11) Ain9 (PD2)
  ADC0_SSCTL3_R = 0x0006;         // 12) no TS0 D0, yes IE0 END0
  ADC0_IM_R &= ~0x0008;           // 13) disable SS3 interrupts
  ADC0_ACTSS_R |= 0x0008;
}
```

```
uint32_t ADC_In(void){
        uint32_t data;
        ADC0_PSSI_R = 0x8;
        while((ADC0_RIS_R & 0x08)==0){};
        data = ADC0_SSFIFO3_R & 0xFFF;
        ADC0_ISC_R = 0x08;
  return data; // remove this, replace with real code
}
```

```
void AutoCal(void){
  dist[0]=0;
  dist[1]=400;
  dist[2]=800;
  dist[3]=1000;
  dist[4]=1200;
  dist[5]=1600;
  dist[6]=2000;

  adcval[0]=5;
  adcval[1]=475;
  adcval[2]=1552;
  adcval[3]=1955;
  adcval[4]=2885;
  adcval[5]=3621;
  adcval[6]=4095;


  ST7735_OutString("Start AutoCal.\nSet slider to 0\nthen press SW1");
  IO_Touch();
  adcval[0]=ADC_In();


  ST7735_SetCursor(0,0);
  ST7735_FillScreen(0);
  ST7735_OutString("Set slider to 4mm.\nPress SW1 to confirm");
  IO_Touch();
  adcval[1]=ADC_In();

  ST7735_SetCursor(0,0);
  ST7735_FillScreen(0);
  ST7735_OutString("Set slider to 8mm.\nPress SW1 to confirm");
  IO_Touch();
  adcval[2]=ADC_In();

  ST7735_SetCursor(0,0);
  ST7735_FillScreen(0);
  ST7735_OutString("Set slider to 10mm.\nPress SW1 to confirm");
  IO_Touch();
  adcval[3]=ADC_In();

  ST7735_SetCursor(0,0);
  ST7735_FillScreen(0);
  ST7735_OutString("Set slider to 12mm.\nPress SW1 to confirm");
  IO_Touch();
  adcval[4]=ADC_In();

  ST7735_SetCursor(0,0);
  ST7735_FillScreen(0);
  ST7735_OutString("Set slider to 16mm.\nPress SW1 to confirm");
  IO_Touch();
  adcval[5]=ADC_In();

  ST7735_SetCursor(0,0);
  ST7735_FillScreen(0);
  ST7735_OutString("Set slider to 20mm.\nPress SW1 to confirm");
  IO_Touch();
  adcval[6]=ADC_In();
  ST7735_SetCursor(0,0);
  ST7735_FillScreen(0);

  float mn = mean();
  float var = variance(mn);
  float cv = covar(mn);
  float internal_w = cv/var;
  weight = cv/var;
  bias = ymean-weight*mn;
  ST7735_OutString("Calibration complete\nSW1 to continue");

  IO_Touch();
  ST7735_SetCursor(0,0);
  ST7735_FillScreen(0);
  return;
```

5. 60 Hz Verification




6. Accuracy Data and Calculations (units in mm)

| True Position | Measured Position | Error (True – Measured) |
|---|---|---|
| 0 | 0.8 | -0.8 |
| 8 | 7.8 | 0.2 |
| 10 | 9.98 | 0.02 |
| 16 | 16.82 | -0.82 |
| 20 | 19.16 | 0.84 |