Electrical, Computer & Energy Engineering
UNIVERSITY OF COLORADO BOULDER

ESM Lab Grading Report

Course Number: 3
Module: 2: Lab 4 on Strain Gauges

Lab Report Date 15/11/2018

Student Name(s) **Poorn Mehta and Rushi James Macwan**

Each Section of this Lab Report Counts for 20 points. Points will be allocated as follows:
20 points: section fully meets requirements of this rubric
15 points: section mostly meets requirements of this rubric
10 points: section meets roughly half the requirements of this rubric
5 points: section does not meet requirements, but shows a weak attempt
0 points: section blank

Goal: The purpose of this lab is to learn how to interface a strain gauge to a microprocessor such as the Cypress PSoC Chip. A secondary goal is to learn how to mitigate the negative effects of the input offset voltage of an amplifier.

Background: Strain gauges can be used to directly measure the strains in a machine or structure. They can also be used a component in a weight measuring instrument called a load cell. They are difficult to mechanically attach to structures, as you must glue them in a prescribed way for them to get accurate readings. Rather than make you glue your own strain gauges, we have specified a strain gauge kit from the web site www.sparkfun.com for you in your lab. This way, you can focus on the circuit and software associated with accurately measuring the strain gauge signal, rather than the production methods of gluing the gauge itself.

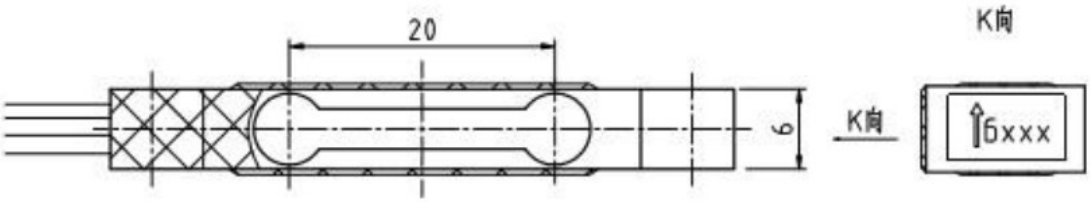
If you were to blindly measure the amplified signal of the strain gauge through the PGA, you would introduce the error of the input offset voltage of the PGA. Therefore, we have created an extra circuit using a DAC, Opamp, three resistors in order to null out this voltage. The compensated signal is then fed to the ADC. When you create your PSoC software to execute the nulling process, make sure to avoid touching putting any force on the strain gauges. Also, make sure that the ambient temperature at which you null the input offset voltage is within a couple of degrees C of the ambient temperature that you take your strain gauge readings. The video supplied with this lab gives you the appropriate instructions.

In this lab we want you to read the amplified strain gauge data on your PC. The UART (Universal Asynchronous Receiver Transmitter) in the schematic is required to communicate with the terminal emulation program that you run on

your PC. By using it, you avoid the need to use an LCD in your circuit. Windows will create a virtual COM port employing the same USB connection that you use to power the board. This COM port is the printed circuit connection, not the J6 micro-USB connector. You need to go into the Windows Control Panel to see which COM port Windows uses. You must also install and run a terminal emulation program on your PC. We recommend the use of a free program called Tera Term. (Go https://download.cnet.com/Tera-Term/3000-2094_4-75766675.html for one site that offers a free download.) You must make sure that the communication parameters in Tera Term match those in the UART.

For this lab, you will display the amplified strain gauge data on Tera Term. You create a strain in your Sparkfun kit by anchoring one end of the metal beam to your lab bench or work table, and ***lightly*** pulling down with one or two fingers on the other end. Then you should see a voltage output reading on your PC screen that corresponds to the amplified bridge offset signal.

The Sparkfun kit has one end (right side in this picture) for lightly anchoring it to your table, and the other end where the wires come out. You can even hold down the anchored end with one hand, and lightly push down on the other end.



Note the specifications of the kit. We ordered the 100g model. Note the excitation voltage to the beam (5V nominal, but get exact data), and the output reading on your PC in volts. Then, calculate how many grams of force the strain gauges are reading. (You should be reading a value between 0 and 100 grams). You should be able to do all of the calculations in software, so that your PC directly shows the grams of force, as opposed to a raw voltage.

Specifications:				
capacity	g	100, 150, 200	300,500,750	1000,1500
rated output	mV/V	0.6 ± 0.15	0.7 ± 0.15	1.0 ± 0.15
safe overload	%FS	150		
ultimate overload	%FS	200		
excitation voltage	Vdc	≤6		

(A) Functional demonstration of your circuit to our TA. In this exercise, you schedule an appointment with your TA to show that your hardware functions as designed. For the Strain Gauge lab, this will involve the following steps:

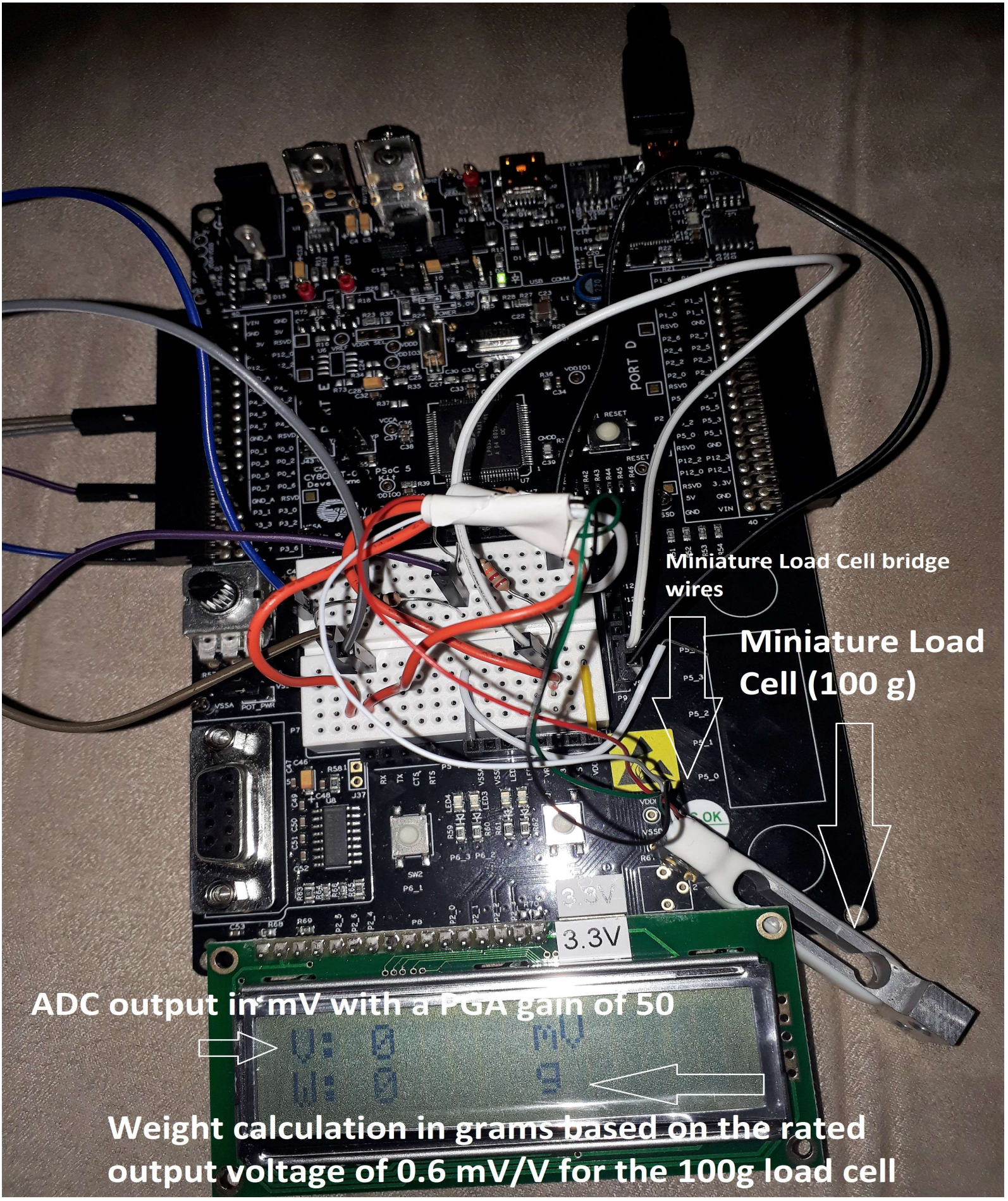
1. Show that all hardware is in place, and that pushing down ***lightly*** on the beam gets you a voltage output to read on your PC.
2. Show that the grams of force you read correlates to how hard you are pushing down on the beam. You should get close to zero grams when you have no force on the beam. How close depends on the accuracy of your nulling circuit that nulls the effect of input offset voltage. When

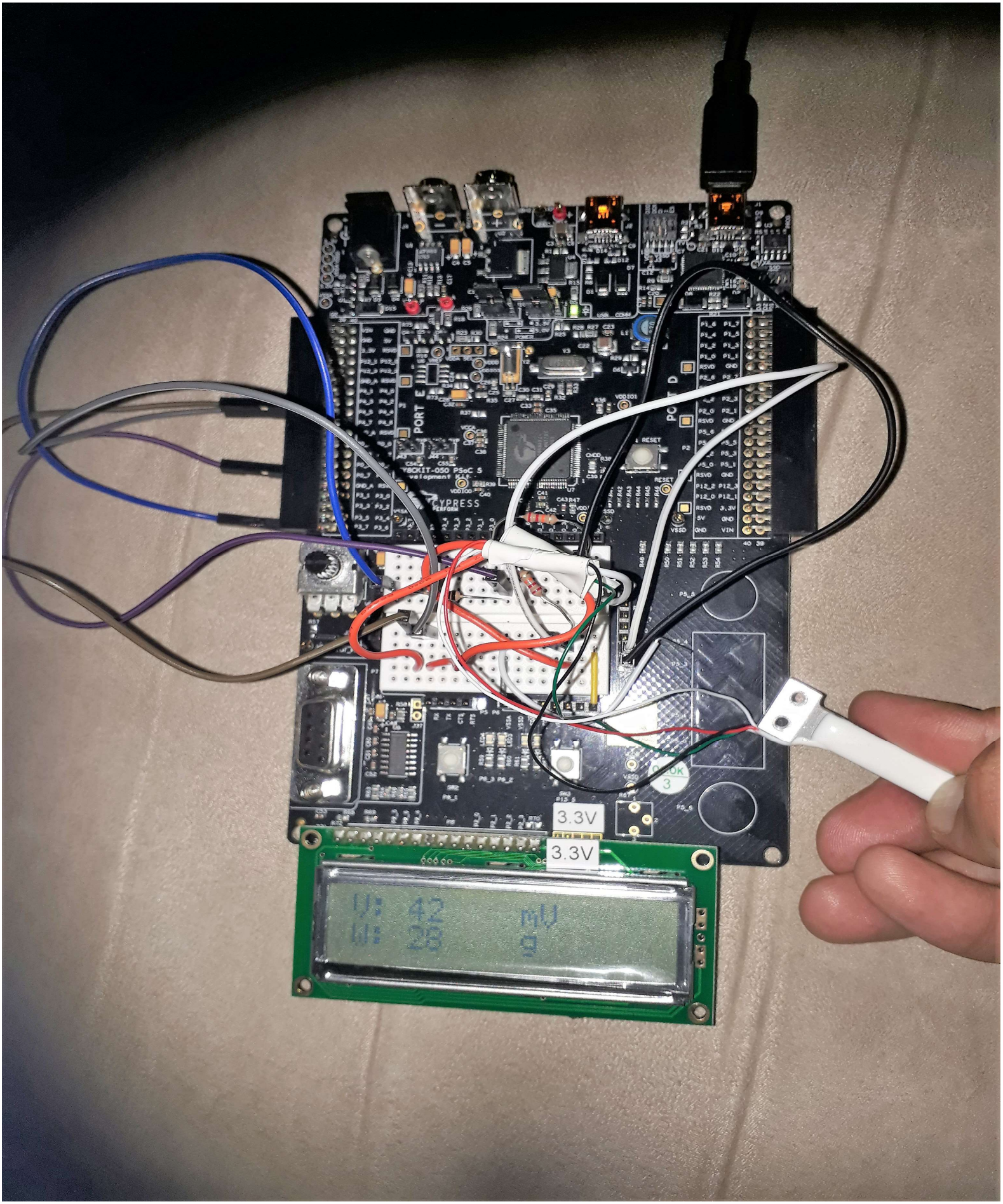
you push down harder, you should get closer to 100 g's. ***Do not push as hard as you can***. The ultimate overload is only 200%, which means that pushing with 200 g's force (~ 1 pound) can damage the strain gauges.

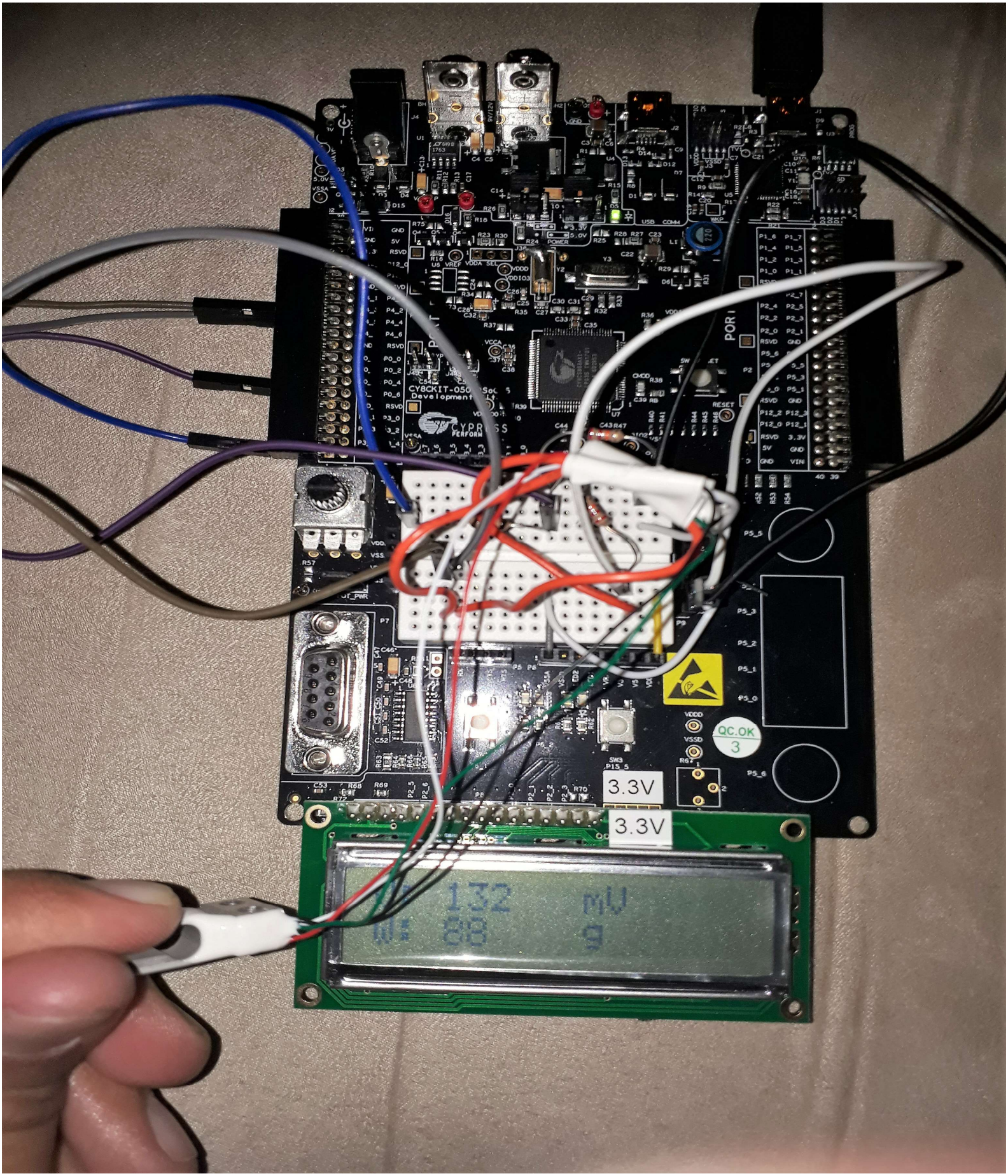
If you are an on-campus student, then show your circuit to one of our TA's during office hours.

If you are a distance learning student, make an online appointment with your TA to demonstrate your work via Zoom meeting or other Web-based meeting tool. You can use the camera on your laptop PC or suitable plug-in webcam (Logitech etc.) to demonstrate a working circuit.

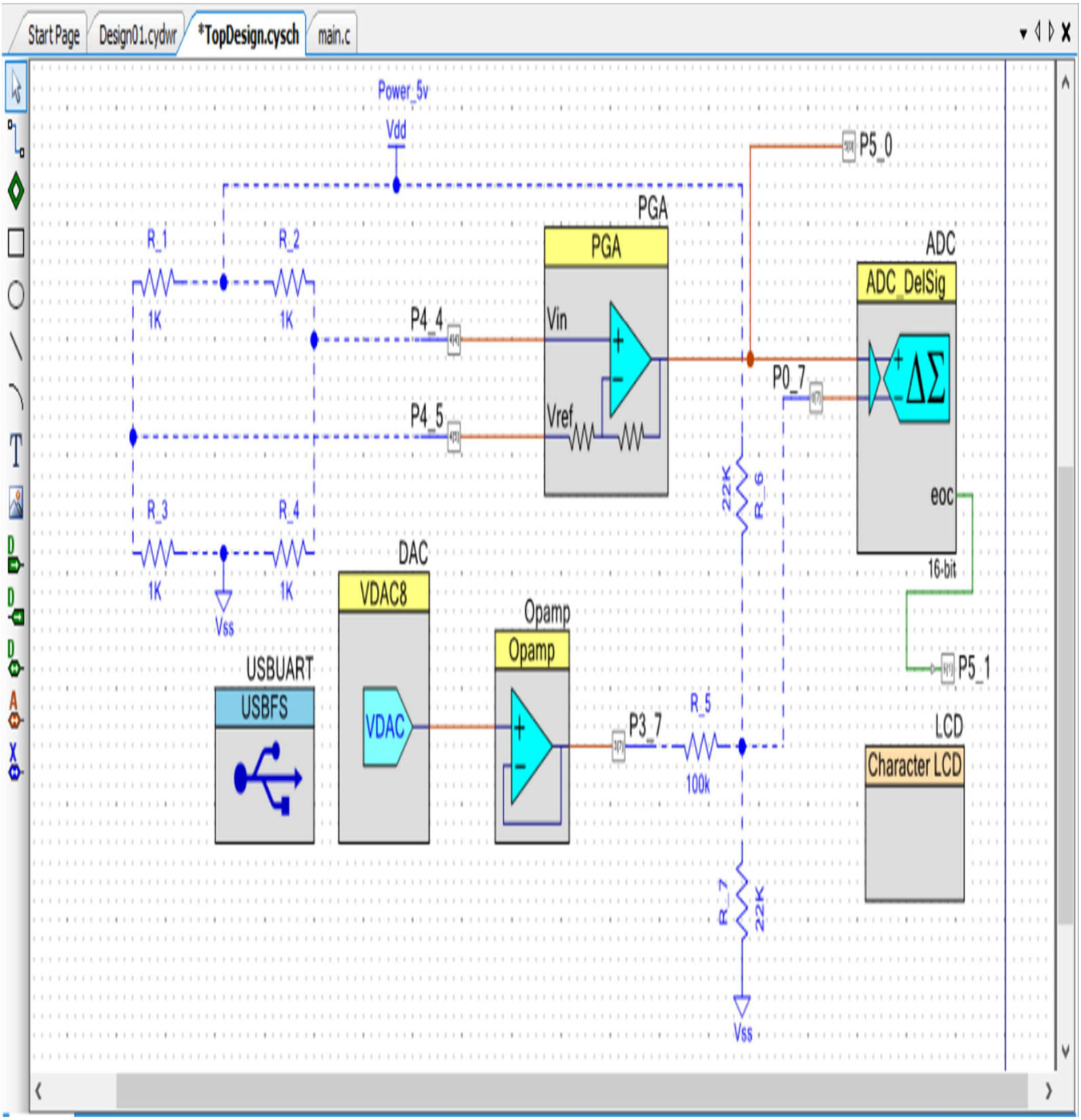
(B) Place photos here of your hardware setup, including PSoC board, connections to PC and/or nScope, wiring, and components. Label all components.

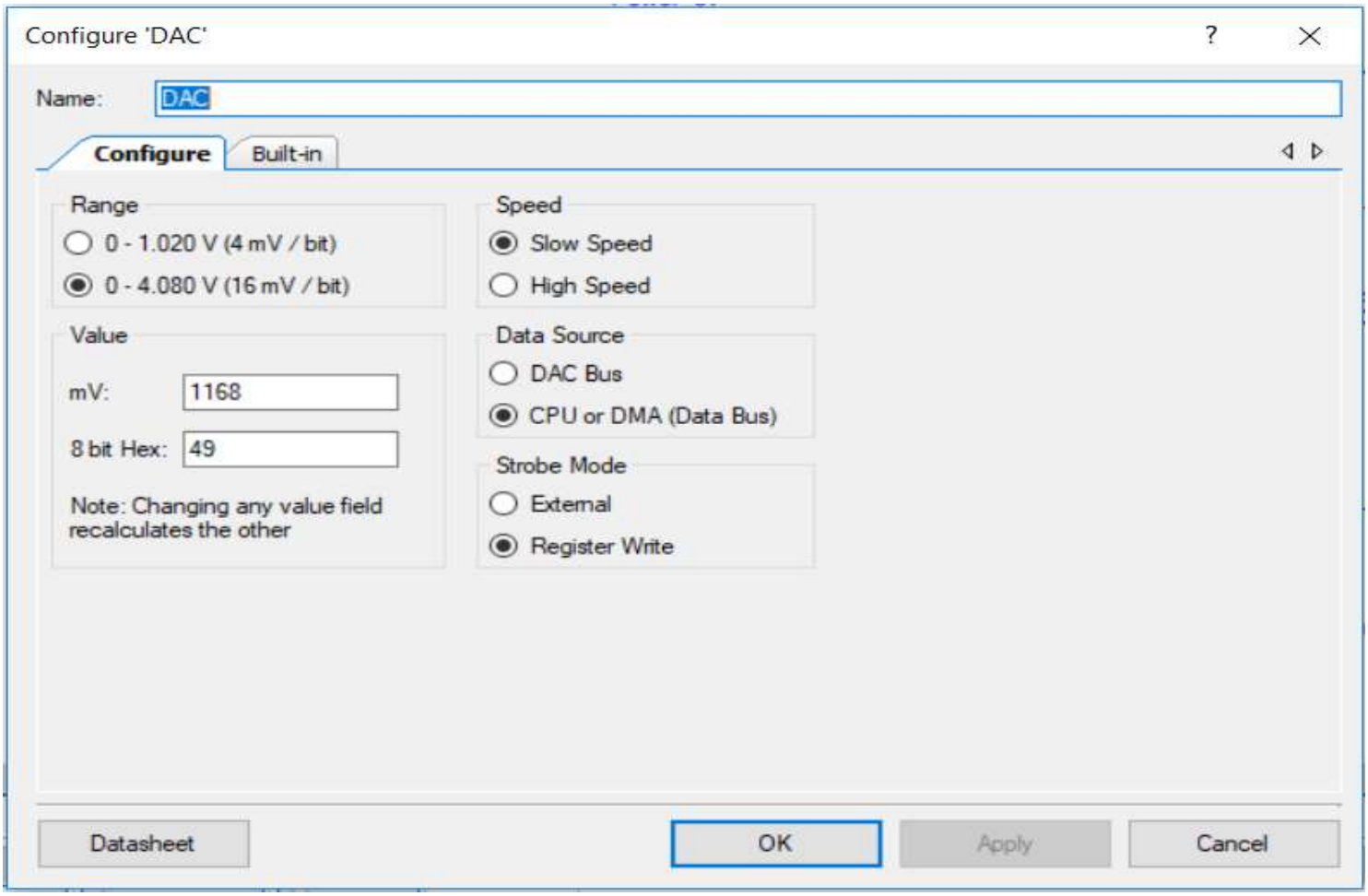

Miniature Load Cell bridge wires
Miniature Load Cell (100 g)
3.3V
ADC output in mV with a PGA gain of 50
Weight calculation in grams based on the rated output voltage of 0.6 mV/V for the 100g load cell

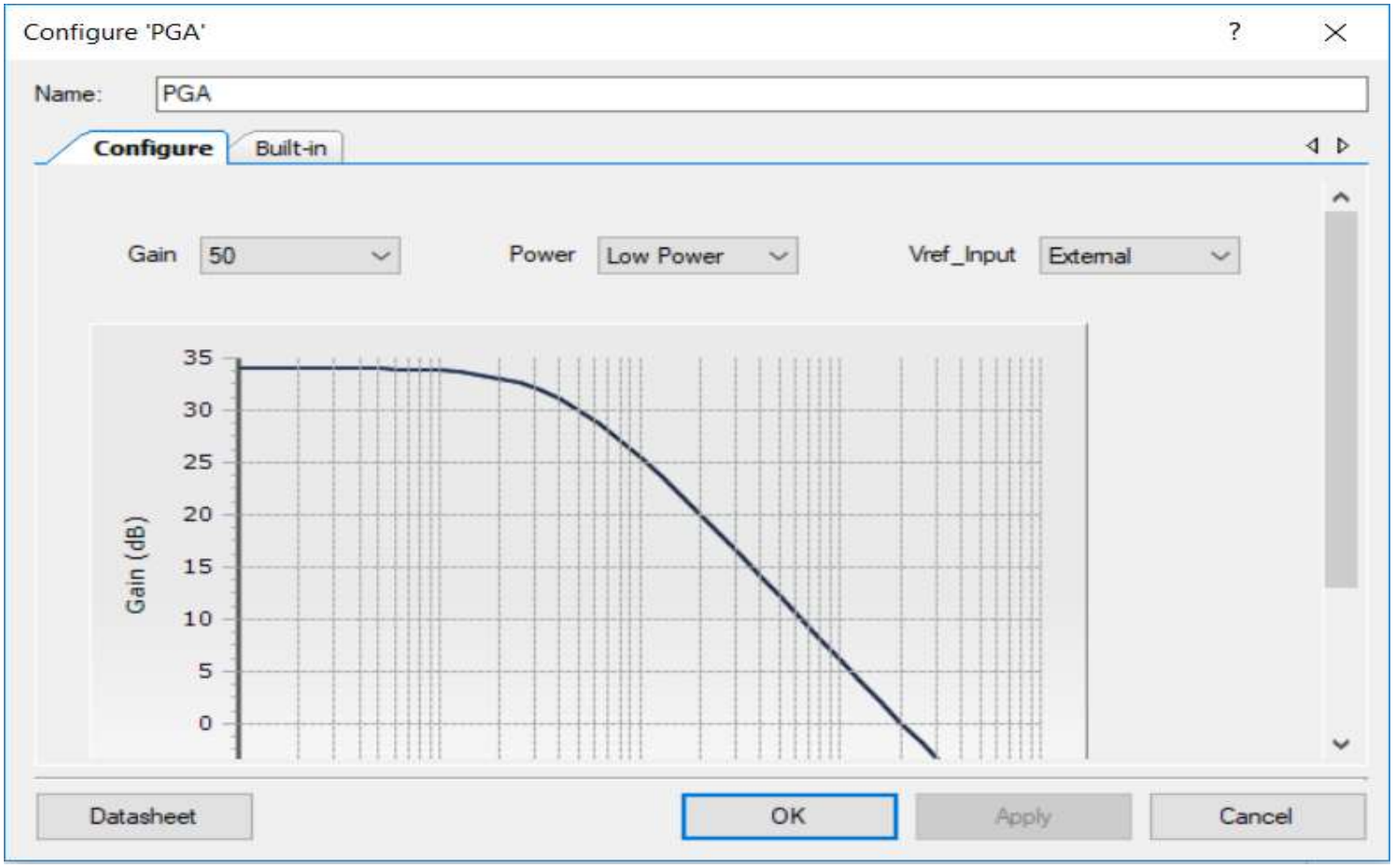

3.3V
3.3V
U: 42 mV
I: 28 9


3.3V
3.3V
132
mV
88
g
QC.OK
3

(C) Place complete PSoC schematic here. This schematic must include internal components from the PSoC board (amplifier, MUX, etc.), as well as external components (thermistor, resistor, display).




Configure 'DAC'
Name:
DAC
Configure
Built-in
Range
0 - 1.020 V (4 mV / bit)
0 - 4.080 V (16 mV / bit)
Value
mV:
1168
8 bit Hex:
49
Note: Changing any value field recalculates the other
Speed
Slow Speed
High Speed
Data Source
DAC Bus
CPU or DMA (Data Bus)
Strobe Mode
External
Register Write
Datasheet
OK
Apply
Cancel


Configure 'PGA'
Name:
PGA
Configure
Built-in
Gain
50
Power
Low Power
Vref_Input
External
35
30
25
20
15
10
5
0
Gain (dB)
Datasheet
OK
Apply
Cancel

(D) Complete PSoC software. This software must include calls to all internal functions, appropriate comments, and functional code that you included. We will not grade you on the exact syntax and structure, as there are numerous ways to structure the code and still provide a force reading. Instead, we will grade you on the completeness of the code relative to using the appropriate PSoC functions to gather the necessary data.

PGA GAIN = 50, DAC 8-bit Hex Value: 49 (Range: 0 – 4.080 V (16mV/bit))

```
/* ========================================
 * Poorn Mehta and Rushi James Macwan
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include "project.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

/*******************UART PART STARTS*********************/

char print[6], i;

#if defined (__GNUC__)
    /* Add an explicit reference to the floating point printf library
*/
    /* to allow usage of the floating point conversion specifiers. */
    /* This is not linked in by default with the newlib-nano library.
*/
    asm (".global _printf_float");
#endif

#define USBFS_DEVICE    (0u)

/* The buffer size is equal to the maximum packet size of the IN and
OUT bulk
* endpoints.
*/
#define USBUART_BUFFER_SIZE (64u)

    uint8_t buffer[USBUART_BUFFER_SIZE];
    uint8_t buffer2[USBUART_BUFFER_SIZE];

/*****************UART PART ENDS************************/

void display(char *ptr, char ID)
{
    for(i = 0; i < 10; i ++)
    {
        if(ID == 0)     buffer[i] = ' ';
```

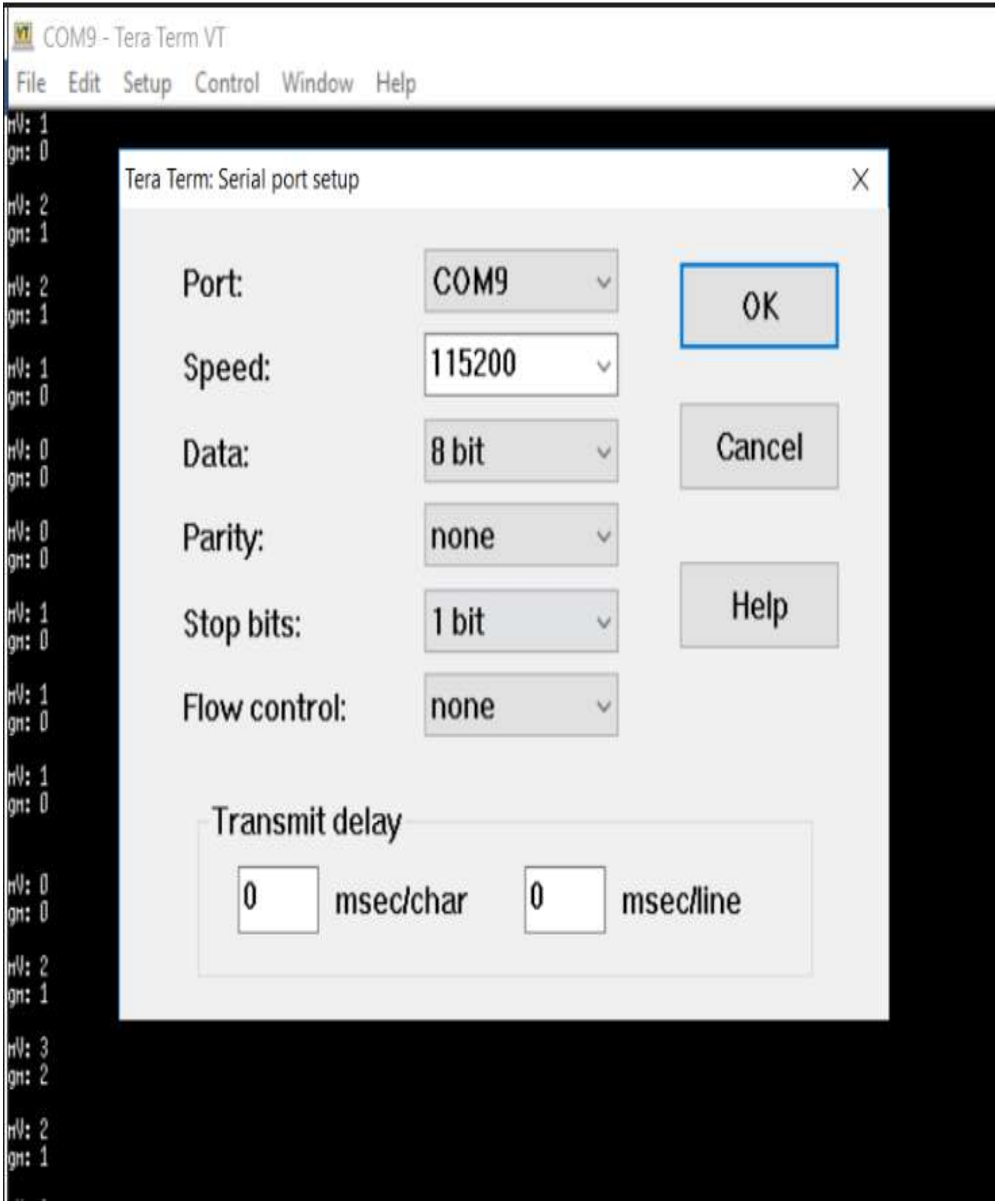
```
        else    buffer2[i] = ' ';
    }
     for(i = 0; *ptr != 0; i ++, ptr ++)
    {
        if(ID == 0)     buffer[i] = (uint8_t)(*ptr);
        else        buffer2[i] = (uint8_t)(*ptr);
    }
}

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    ADC_Start();
    PGA_Start();
    DAC_Start();
    Opamp_Start();
    LCD_Start();
    LCD_ClearDisplay();
    int16_t adc_out;
    int16_t adc_mv;
    int16_t adc_mv_neg;
    uint16_t weight;

    /************************UART PART BEGINS***************/
    uint16_t count;

    uint8_t volts[5] = {"mV: "};
    uint8_t grams[5] = {"gm: "};
    USBUART_Start(USBFS_DEVICE, USBUART_5V_OPERATION);
    /************************UART PART  ENDS*****************/

    while(1)
    {
        LCD_Position(0,0);
        LCD_PrintString("V: ");
        adc_out = ADC_Read16(); // Direct method to read the signed 16-
bit ADC output
        adc_mv = ADC_CountsTo_mVolts(adc_out);
        adc_mv_neg = 65536 - adc_mv;
        weight = ((adc_mv * 2) / 3); // Using the equation:
((adc_mv/50) / 3 mV)* 100) for finding weight in grams

        /************************UART PART BEGINS***************/
        /* Host can send double SET_INTERFACE request. */
        if (0u != USBUART_IsConfigurationChanged())
        {
            /* Initialize IN endpoints when device is configured. */
            if (0u != USBUART_GetConfiguration())
            {
                /* Enumeration is done, enable OUT endpoint to receive
data
                 * from host. */
                USBUART_CDC_Init();
            }
        }
        /************************UART PART  ENDS*****************/

        /* Conditions for output/display */
```

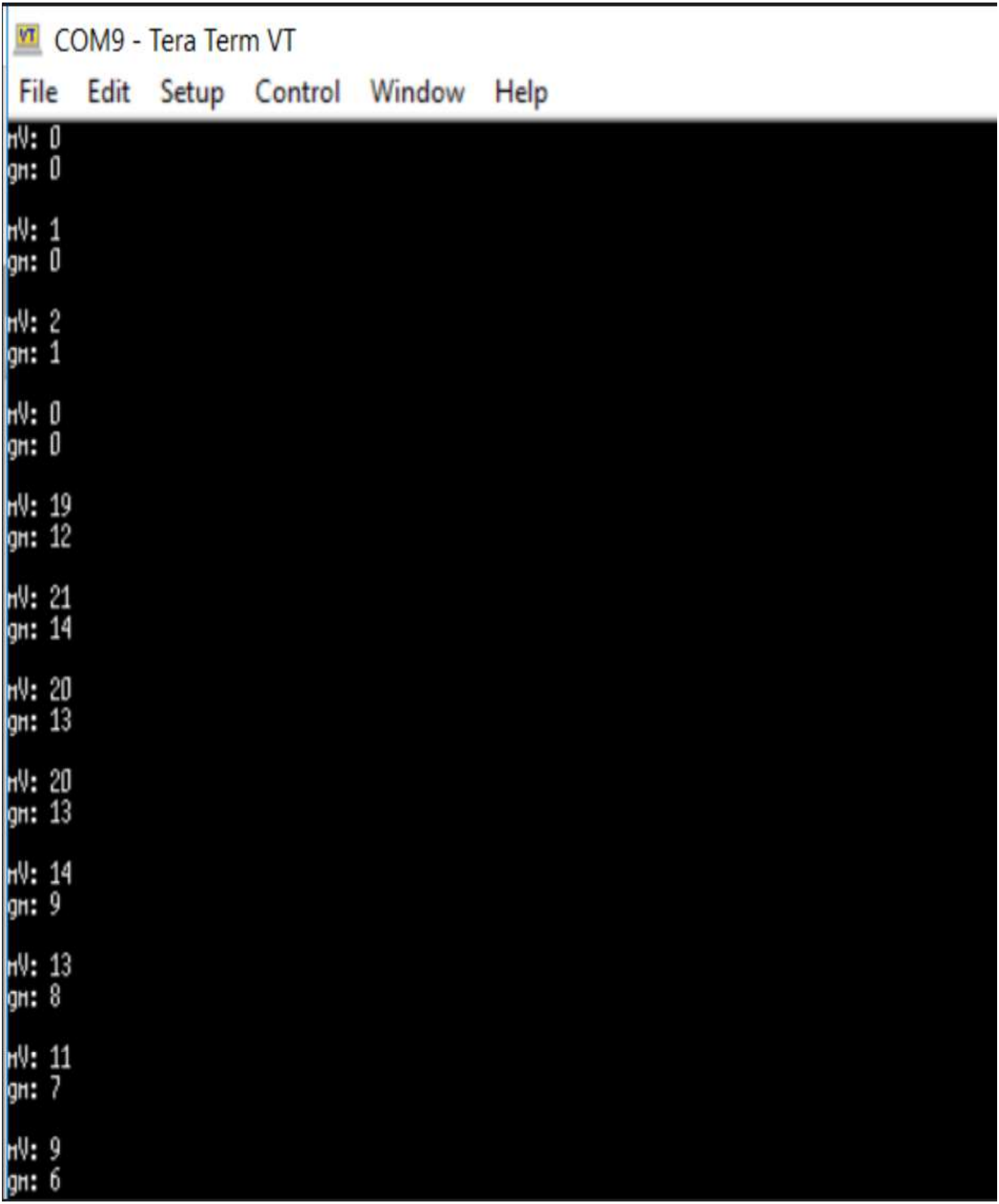
```
        if(adc_mv >= 0)
        {
            LCD_Position(1,0);
            LCD_PrintString("W: ");
            LCD_Position(1,3);
            LCD_PrintNumber(weight);

            if(adc_mv <= 9)
            {
                LCD_Position(0,3);
                LCD_PrintNumber(adc_mv);
                LCD_Position(0,4);
                LCD_PrintString("    ");
            }
            if(adc_mv >= 10 && adc_mv < 100)
            {
                LCD_Position(0,3);
                LCD_PrintNumber(adc_mv);
                LCD_Position(0,5);
                LCD_PrintString("   ");
            }
            if(adc_mv >= 100 && adc_mv <1000)
            {
                LCD_Position(0,3);
                LCD_PrintNumber(adc_mv);
                LCD_Position(0,6);
                LCD_PrintString("  ");
            }
            if(adc_mv >= 1000)
            {
                LCD_Position(0,3);
                LCD_PrintNumber(adc_mv);
                LCD_Position(0,7);
                LCD_PrintString(" ");
            }
            LCD_Position(0,9);
            LCD_PrintString("mV");
            LCD_Position(1,9);
            LCD_PrintString("g");

            /*UART Part*/
            if (0u != USBUART_GetConfiguration())
            {
                itoa(adc_mv, print, 10);
                display(print, 0);
                itoa(weight, print, 10);
                display(print, 1);
                while (0u == USBUART_CDCIsReady());
                USBUART_PutData(volts, 5);
                while (0u == USBUART_CDCIsReady());
                USBUART_PutData(buffer, 10);
                while (0u == USBUART_CDCIsReady());
                USBUART_PutCRLF();
                while (0u == USBUART_CDCIsReady());
                USBUART_PutData(grams, 5);
                while (0u == USBUART_CDCIsReady());
                USBUART_PutData(buffer2, 10);
                while (0u == USBUART_CDCIsReady());
```

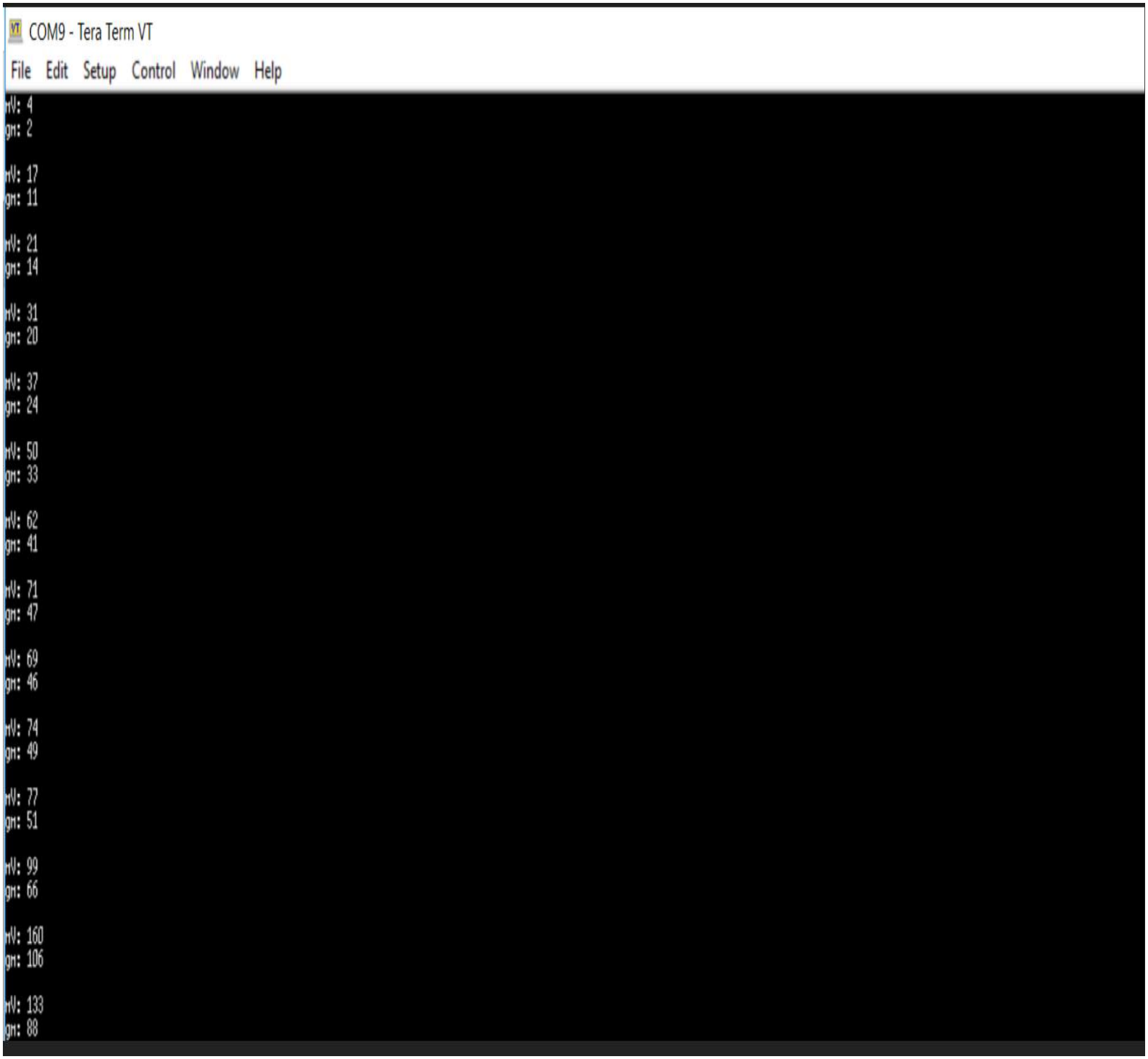
```
                USBUART_PutCRLF();
                while (0u == USBUART_CDCIsReady());
                USBUART_PutCRLF();
            }
        }
        if(adc_mv < 0)
        {
            LCD_Position(1,0);
            LCD_PrintString("W: ");
            LCD_Position(1,3);
            LCD_PrintString("0");

            /*Conditions for the negative values of ADC output */
            if(adc_mv_neg == 0)
            {
                LCD_Position(0,3);
                LCD_PrintNumber(adc_mv_neg);
                LCD_Position(0,4);
                LCD_PrintString(" ");
            }
            if(adc_mv_neg <= 9)
            {
                LCD_Position(0,3);
                LCD_PrintString("-");
                LCD_Position(0,4);
                LCD_PrintNumber(adc_mv_neg);
                LCD_Position(0,5);
                LCD_PrintString("    ");
            }
            if(adc_mv_neg >= 10 && adc_mv_neg < 100)
            {
                LCD_Position(0,3);
                LCD_PrintString("-");
                LCD_Position(0,4);
                LCD_PrintNumber(adc_mv_neg);
                LCD_Position(0,6);
                LCD_PrintString("   ");
            }
            if(adc_mv_neg >= 100 && adc_mv_neg <1000)
            {
                LCD_Position(0,3);
                LCD_PrintString("-");
                LCD_Position(0,4);
                LCD_PrintNumber(adc_mv_neg);
                LCD_Position(0,7);
                LCD_PrintString("  ");
            }
            if(adc_mv_neg >= 1000)
            {
                LCD_Position(0,3);
                LCD_PrintString("-");
                LCD_Position(0,4);
                LCD_PrintNumber(adc_mv_neg);
                LCD_Position(0,8);
                LCD_PrintString(" ");
            }
            LCD_Position(0,9);
            LCD_PrintString("mV");
```

```
            LCD_Position(1,9);
            LCD_PrintString("g");

            /* UART Part */
            if (0u != USBUART_GetConfiguration())
            {
                while (0u == USBUART_CDCIsReady());
                USBUART_PutData(volts, 5);
                while (0u == USBUART_CDCIsReady());
                USBUART_PutString("-");
                while (0u == USBUART_CDCIsReady());
                itoa(adc_mv_neg, print, 10);
                display(print, 0);
                while (0u == USBUART_CDCIsReady());
                USBUART_PutData(buffer, 10);
                while (0u == USBUART_CDCIsReady());
                USBUART_PutCRLF();
                while (0u == USBUART_CDCIsReady());
                USBUART_PutData(grams, 5);
                while (0u == USBUART_CDCIsReady());
                USBUART_PutString("0");
                while (0u == USBUART_CDCIsReady());
                USBUART_PutCRLF();
                while (0u == USBUART_CDCIsReady());
                USBUART_PutCRLF();
            }
        }
        CyDelay(1000);
        LCD_ClearDisplay();
    }
}

/* [] END OF FILE */
```

E) Place screenshots from your PC showing the Tera Term settings, as well as the display on the Tera Term screen of your force output of the bridge.


COM9 - Tera Term VT
File Edit Setup Control Window Help
mV: 1
gn: 0
mV: 2
gn: 1
mV: 2
gn: 1
mV: 1
gn: 0
mV: 0
gn: 0
mV: 0
gn: 0
mV: 1
gn: 0
mV: 1
gn: 0
mV: 1
gn: 0
mV: 0
gn: 0
mV: 2
gn: 1
mV: 3
gn: 2
mV: 2
gn: 1
Tera Term: Serial port setup
Port:
COM9
Speed:
115200
Data:
8 bit
Parity:
none
Stop bits:
1 bit
Flow control:
none
OK
Cancel
Help
Transmit delay
0
msec/char
0
msec/line


COM9 - Tera Term VT
File Edit Setup Control Window Help
mV: 0
gm: 0
mV: 1
gm: 0
mV: 2
gm: 1
mV: 0
gm: 0
mV: 19
gm: 12
mV: 21
gm: 14
mV: 20
gm: 13
mV: 20
gm: 13
mV: 14
gm: 9
mV: 13
gm: 8
mV: 11
gm: 7
mV: 9
gm: 6


COM9 - Tera Term VT
File Edit Setup Control Window Help
mV: 4
gm: 2
mV: 17
gm: 11
mV: 21
gm: 14
mV: 31
gm: 20
mV: 37
gm: 24
mV: 50
gm: 33
mV: 62
gm: 41
mV: 71
gm: 47
mV: 69
gm: 46
mV: 74
gm: 49
mV: 77
gm: 51
mV: 99
gm: 66
mV: 160
gm: 106
mV: 133
gm: 88