Electrical, Computer & Energy Engineering
UNIVERSITY OF COLORADO BOULDER

ESM Lab Grading Report

Course Number: 3
Module: 4: Lab 3 on Rotary Sensors

Lab Report Date ***29th October 2018***

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

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 two different rotary position sensors (a potentiometer and the EN11 rotary encoder) to a microprocessor such as the Cypress PSoC Chip. A secondary goal is to observe and compensate for switch bounce.

Background: Rotary position encoders can take many forms. Perhaps the simplest type is a potentiometer such the one labeled "R56" on your PSoC5 dev kit. It is simply a resister whose value varies as the device is rotated. One obvious downside of this design is the limited range of rotation- just one turn in this case.

In contrast, the EN11 rotary encoder has no such limitation- it can be turned indefinitely. However, measuring angular position using this device requires some electronics and possibly some software.

The EN11 encoder is basically a set of cross coupled mechanical switches. Like all switches, the contacts will physically bounce and exhibit noise. This complicates the interface design because the PSoC microprocessor is fast enough to respond to every noise spike. Careful filtering is important for reliable operation.

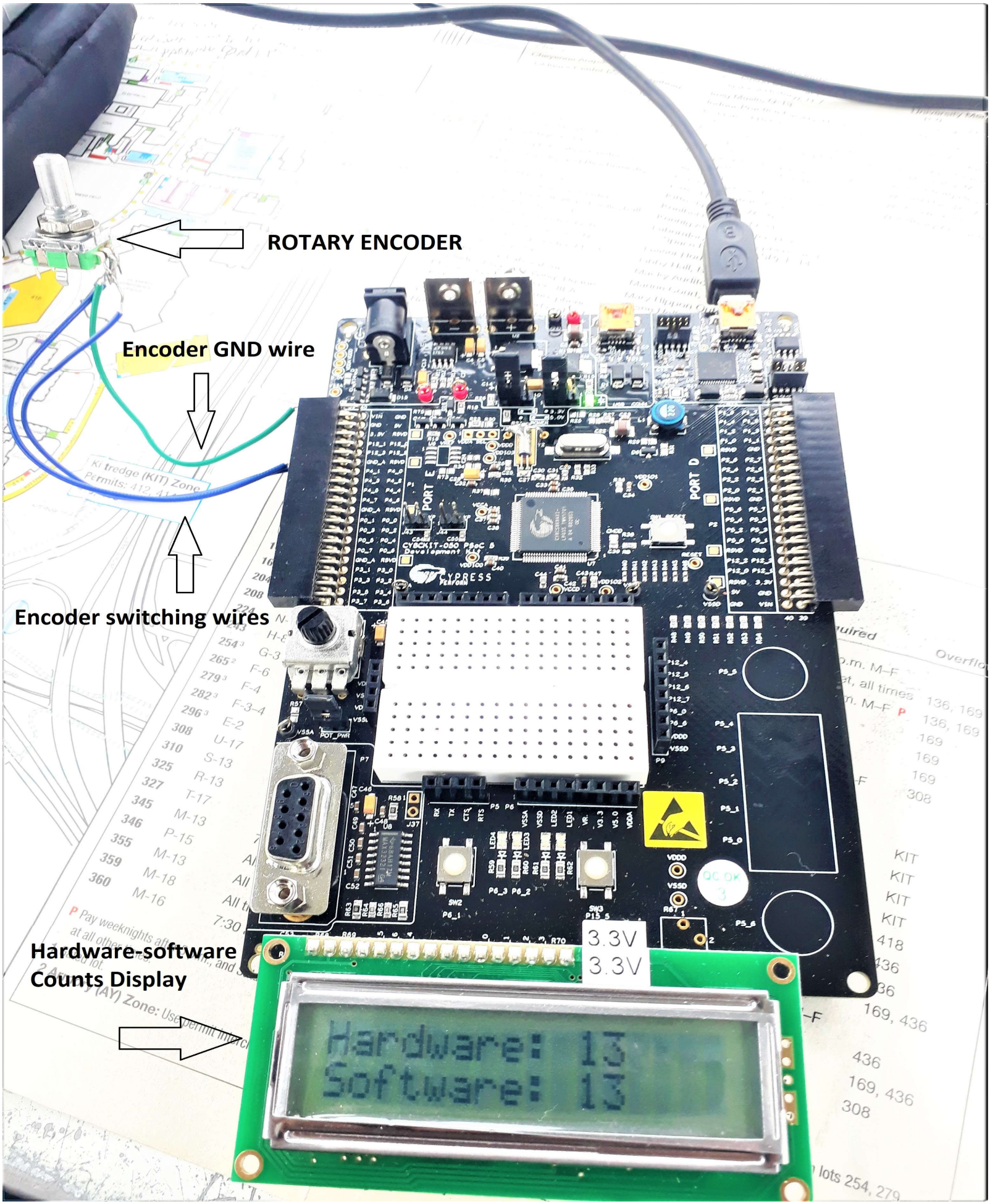
(A) Functional demonstration of your circuit to our TA. In this exercise, you schedule an appointment with your TA to show that your hardware functions as designed. For the Rotary Sensors lab, this will involve the following steps:

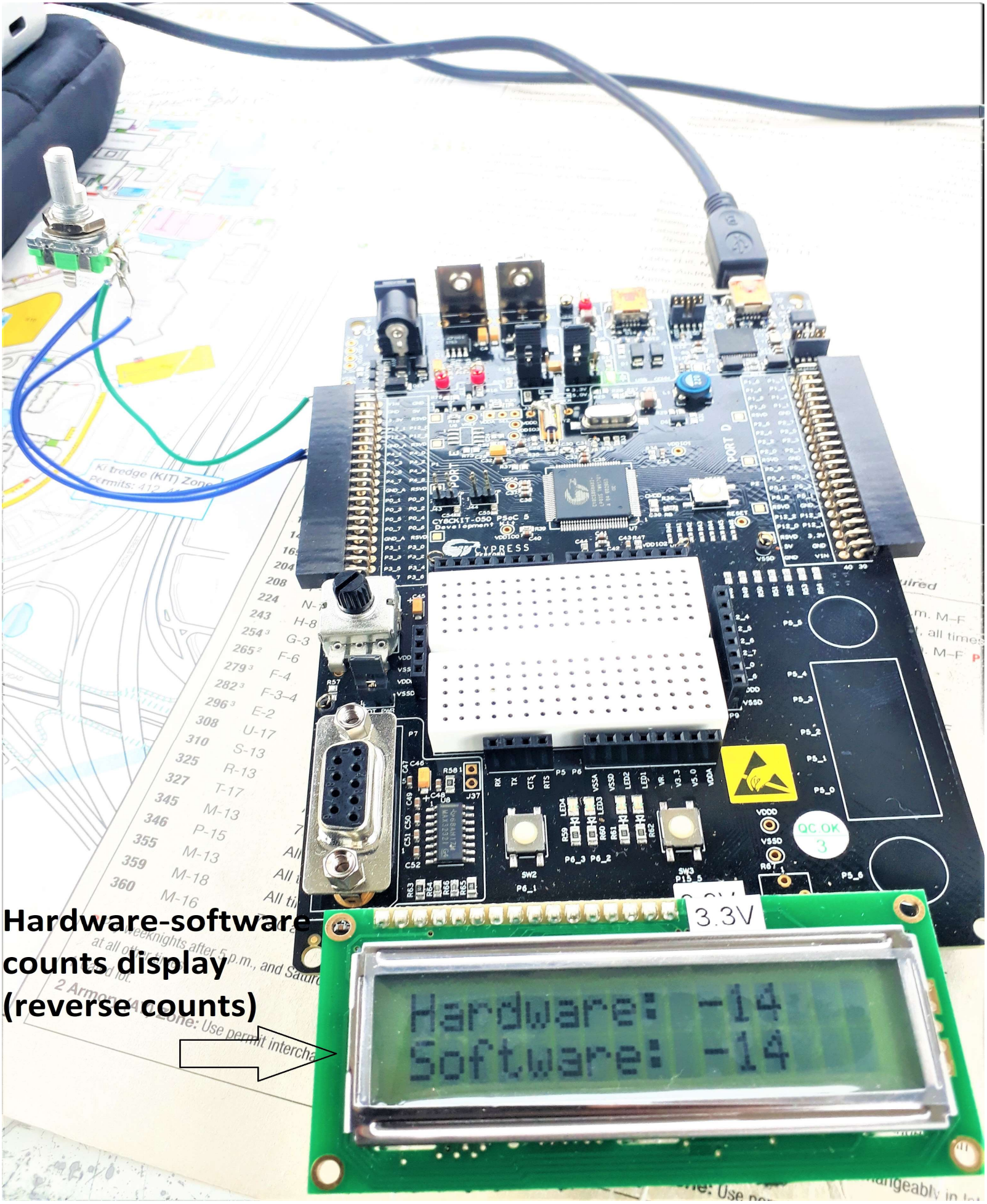
1. Show that all hardware is in place, and that rotating the knob of the switch changes which of the edges of your signals on the channels of your scope is the leading one. (See red and green scope traces in the video). You should also demonstration evidence in the scope screens of signal bounce.
2. Alter which of your scope traces is the leading one by rotating the rotary switch counterclockwise and clockwise.
3. Demonstrate that your PSoC software is acting as the digital filter described in the video for this lab. Show the switch response on either your oscilloscope or your nScope and on the LCD.
4. Use the quadrature encoder circuit (QuadDec) within the PSoC microprocessor to count the number of rotary switch counts when you manually rotate the switch for 20 times. You should get exactly 20, as this hardware is very accurate.
5. Write your own custom software to count the number of pulses, using the necessary digital filtering to avoid counting switch bounce. See how many pulses you count when you turn the rotary switch for 20 clicks. Count those clicks by hand up to 20. You may or may not count to 20 switch pulses with this software. It depends on how good a job you do with your digital filtering.

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

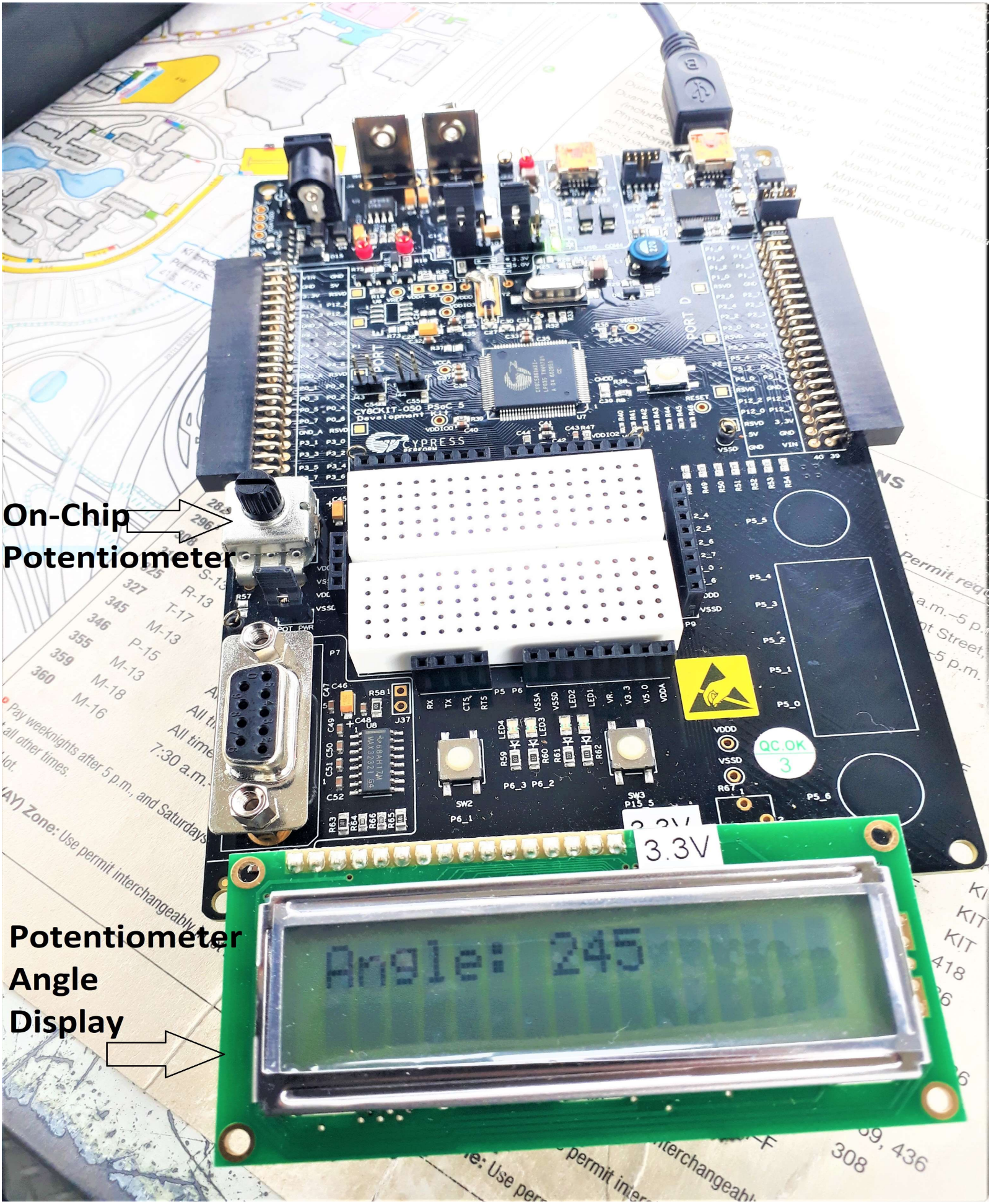
If you are a distance learning student, make an online appointment with your TA to demonstrate your work via Zoom meeting or other Web-based meeting tool. You can use the camera on your laptop PC or suitable plug-in webcam (Logitech etc.) to demonstrate a working circuit.

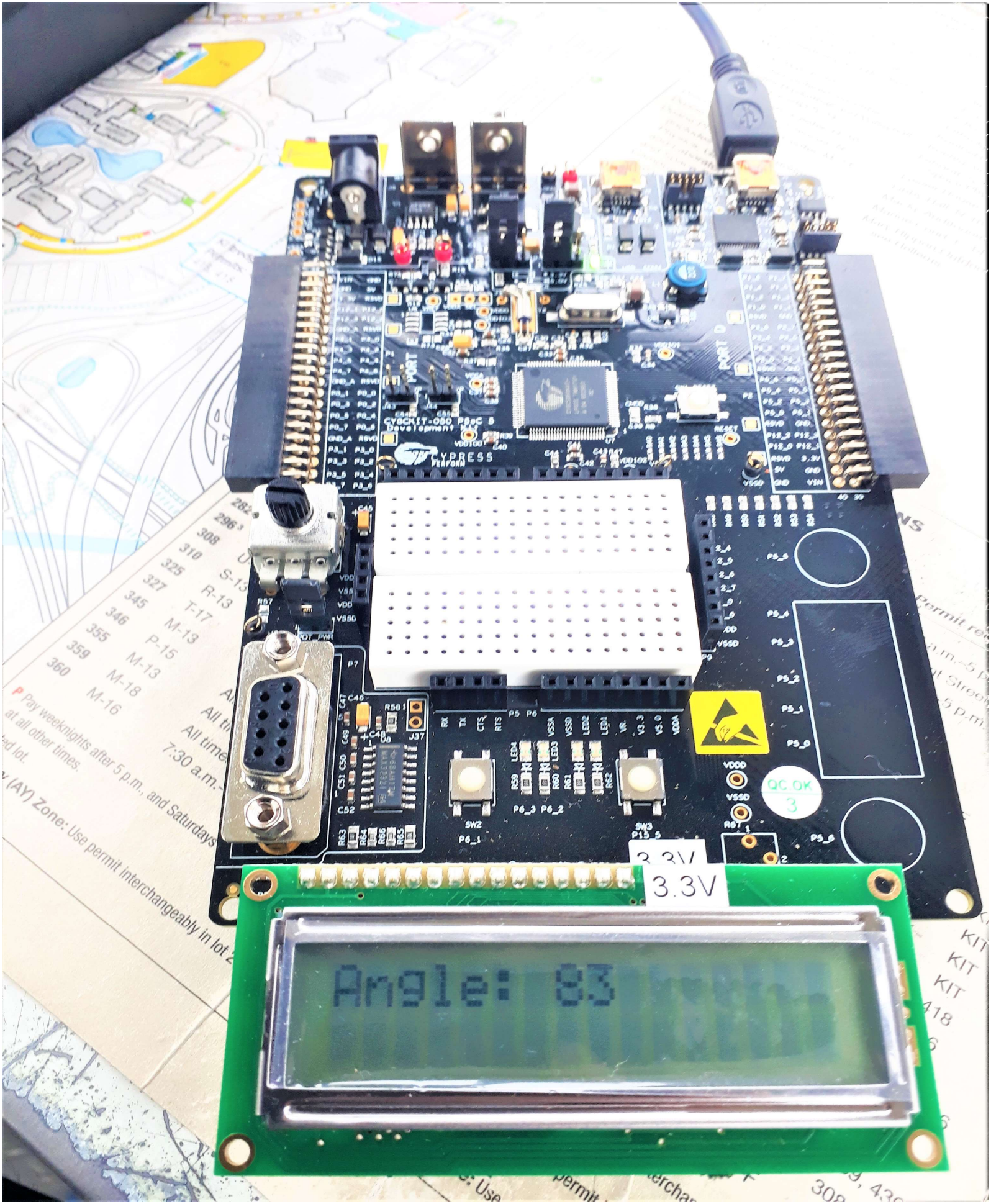
(B) Place photos here of your hardware setup, including PSoC board, connections to Oscilloscope or nScope, wiring, LCD Display, components. Label all components.


ROTARY ENCODER
Encoder GND wire
Encoder switching wires
Hardware-software
Counts Display
Hardware: 13
Software: 13
3.3V
3.3V

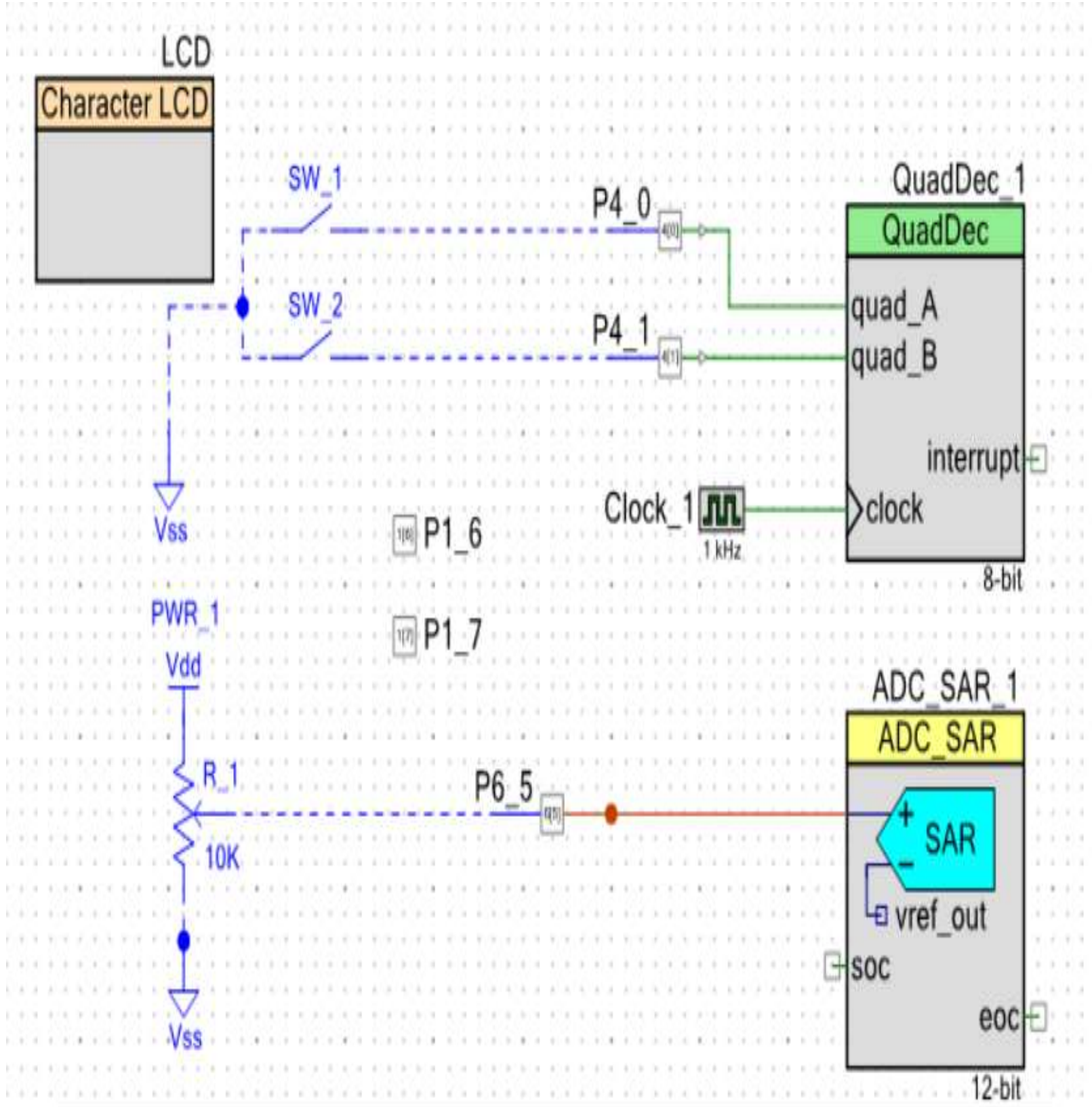


Hardware-software counts display (reverse counts)


On-Chip
Potentiometer
Potentiometer
Angle
Display
Angle: 245
3.3V
QC OK
3


Angle: 83
3.3V
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).



(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 the temperature measurement function. Instead, we will grade you on the completeness of the code relative to using the appropriate PSoC functions to gather the necessary data.

```
/* ========================================
 *
 * 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"

#define Rotary_Mode      0
#define POT_Mode         1
#define Mode     Rotary_Mode

void rotary(void)
{
    QuadDec_1_Start();
    LCD_Start();
    LCD_Position(0,0);
    LCD_PrintString("Software: ");
    int8_t soft_counter, hard_counter, s2;
    uint8_t pin0_status, pin1_status, update;
    uint32_t loop_counter = 0;
    s2 = 0;
    soft_counter = 0;
    hard_counter = 0;
    update = 1;

    /*Pins P1_6 and P1_7 are used to calculate the critical loop times
*/

    while(1)
    {
        loop_counter +=1;
        pin1_status = P4_1_Read();
        pin0_status = P4_0_Read();
        if(pin1_status == 0)
        {
            //raise one pin
            P1_6_Write(1);
            do{
                pin0_status = P4_0_Read();
            }while(pin0_status != 0);

            do{
```

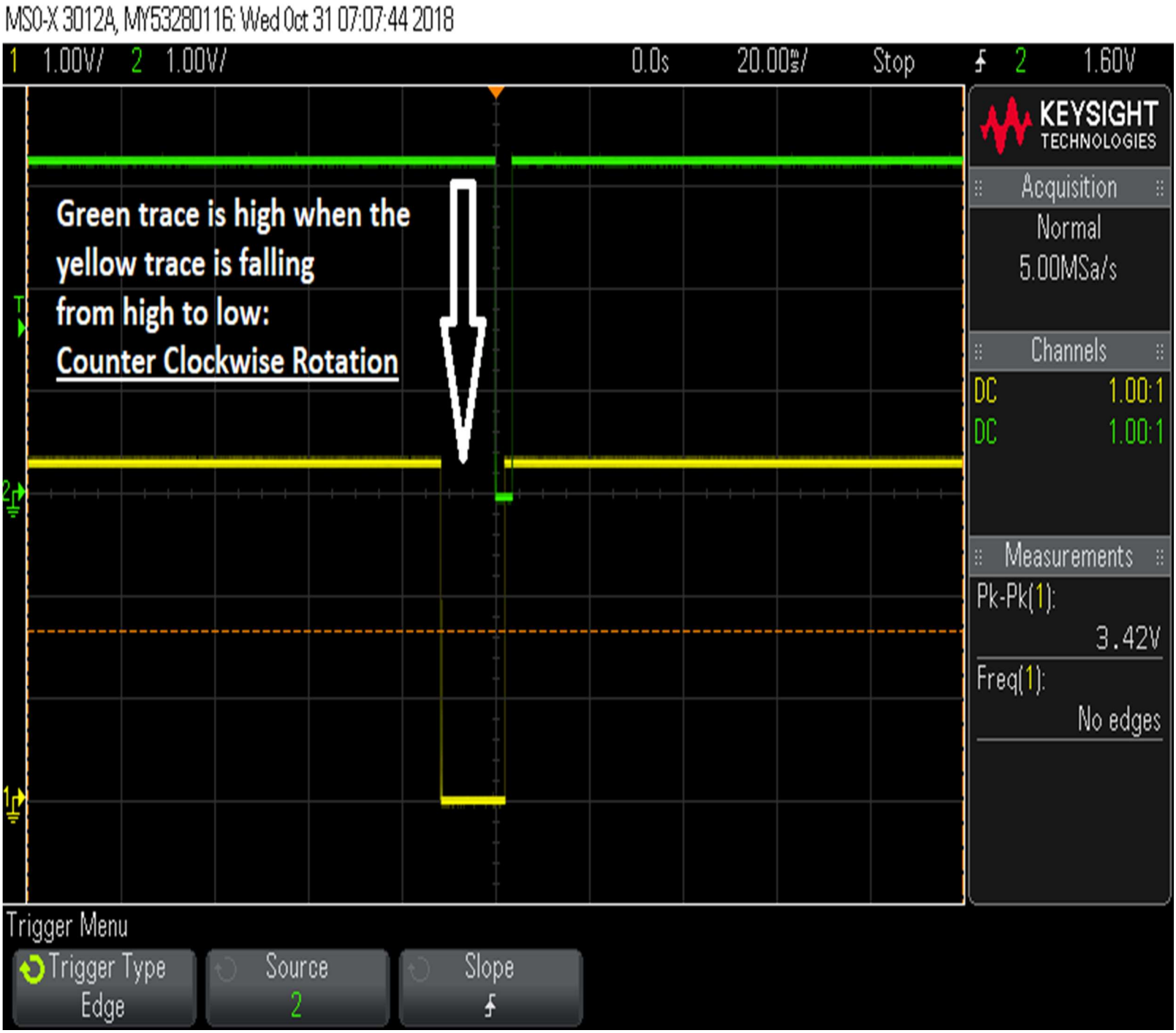
```
                pin0_status = P4_0_Read();
                CyDelay(1);
            }while(pin0_status != 1);
            if(s2 > 0)
            {
                s2 -= 1;
            }
            else
            {
                soft_counter += 1;
            }
            update = 1;
            //clear that pin, measure time
            P1_6_Write(0);
        }
        else if(pin0_status == 0)
        {
            //raise one pin
            P1_6_Write(1);
            do{
                pin1_status = P4_1_Read();
            }while(pin1_status == 1);

            do{
                pin0_status = P4_0_Read();
                CyDelay(1);
            }while(pin0_status != 1);
            if(soft_counter < 1)
            {
                s2 += 1;
            }
            else
            {
                soft_counter -= 1;
            }
            update = 1;
            //clear that pin, measure time
            P1_6_Write(0);
        }
        hard_counter = QuadDec_1_GetCounter();
        if(update == 1)
        {
            //raise pin 2
            P1_7_Write(1);
            update = 0;
            loop_counter = 0;
            LCD_ClearDisplay();
            LCD_Position(0,0);
            LCD_PrintString("Hardware: ");
            LCD_Position(0,10);
            if(hard_counter < 0)
            {
                LCD_PrintString("-");
                hard_counter &= ~(1 << 7);
                hard_counter = 128 - hard_counter;
            }
            LCD_PrintNumber(hard_counter);

```

```
            LCD_Position(1,0);
            LCD_PrintString("Software: ");
            LCD_Position(1,10);
            if(s2 > 0)
            {
                LCD_PrintString("-");
                LCD_PrintNumber(s2);
            }
            else
            {
                LCD_PrintNumber(soft_counter);
            }
            //clear that pin
            P1_7_Write(0);
        }
    }
}

void pot(void)
{
    uint16_t pot1,pot2,pot3;
    ADC_SAR_1_Start();
    while(1)
    {
        //raise 1 pin
        P1_6_Write(1);
        ADC_SAR_1_StartConvert();
        ADC_SAR_1_IsEndConversion(ADC_SAR_1_WAIT_FOR_RESULT);
        //clear that pin
        P1_6_Write(0);
        pot1 = ADC_SAR_1_GetResult16();
        pot2 = ADC_SAR_1_CountsTo_mVolts(pot1);
        pot3 = (pot2*270) / 5000;
        //raise second pin
        P1_7_Write(1);
        LCD_ClearDisplay();
        LCD_Position(0,0);
        LCD_PrintString("Angle: ");
        LCD_Position(0,7);
        LCD_PrintNumber(pot3);
        //clear that pin
        P1_7_Write(0);
        CyDelay(100);
    }
}

int main(void)
{
    if(Mode)
    {
        pot();
    }
    else
    {
        rotary();
    }
}
```

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

E) Place screenshots from your oscilloscope or nScope showing critical loop times or signal outputs. For this lab you should show screen shots of the switch being rotated both clockwise and counterclockwise, as shown in the sample screen shot below. Then you can discuss how the leading or trailing edges of the relative signals indicate which way the switch was rotated.

The Counter-Clockwise and Clockwise Rotation Clicks can be seen from the below Screenshots taken from the DSO:




MSO-X 3012A, MY53280116: Wed Oct 31 07:07:32 2018
1 1.00V/ 2 1.00V/
0.0s
20.00ms/
Stop
2 1.60V
KEYSIGHT TECHNOLOGIES
Yellow trace is high when the green trace is falling from high to low: Clockwise Rotation
Acquisition
Normal
5.00MSa/s
Channels
DC 1.00:1
DC 1.00:1
Measurements
Pk-Pk(1):
3.42V
Freq(1):
No edges
Trigger Menu
Trigger Type
Edge
Source
2
Slope