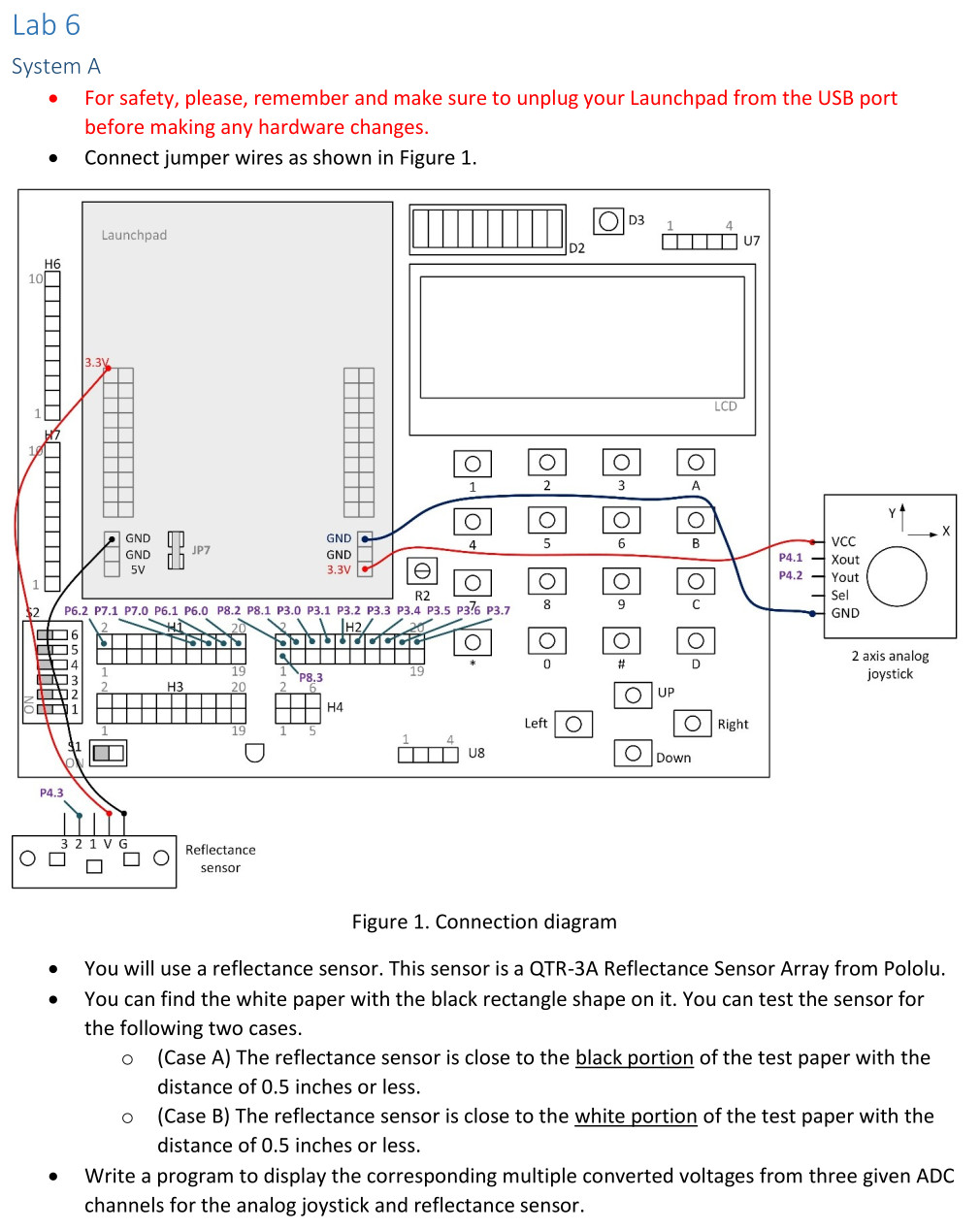
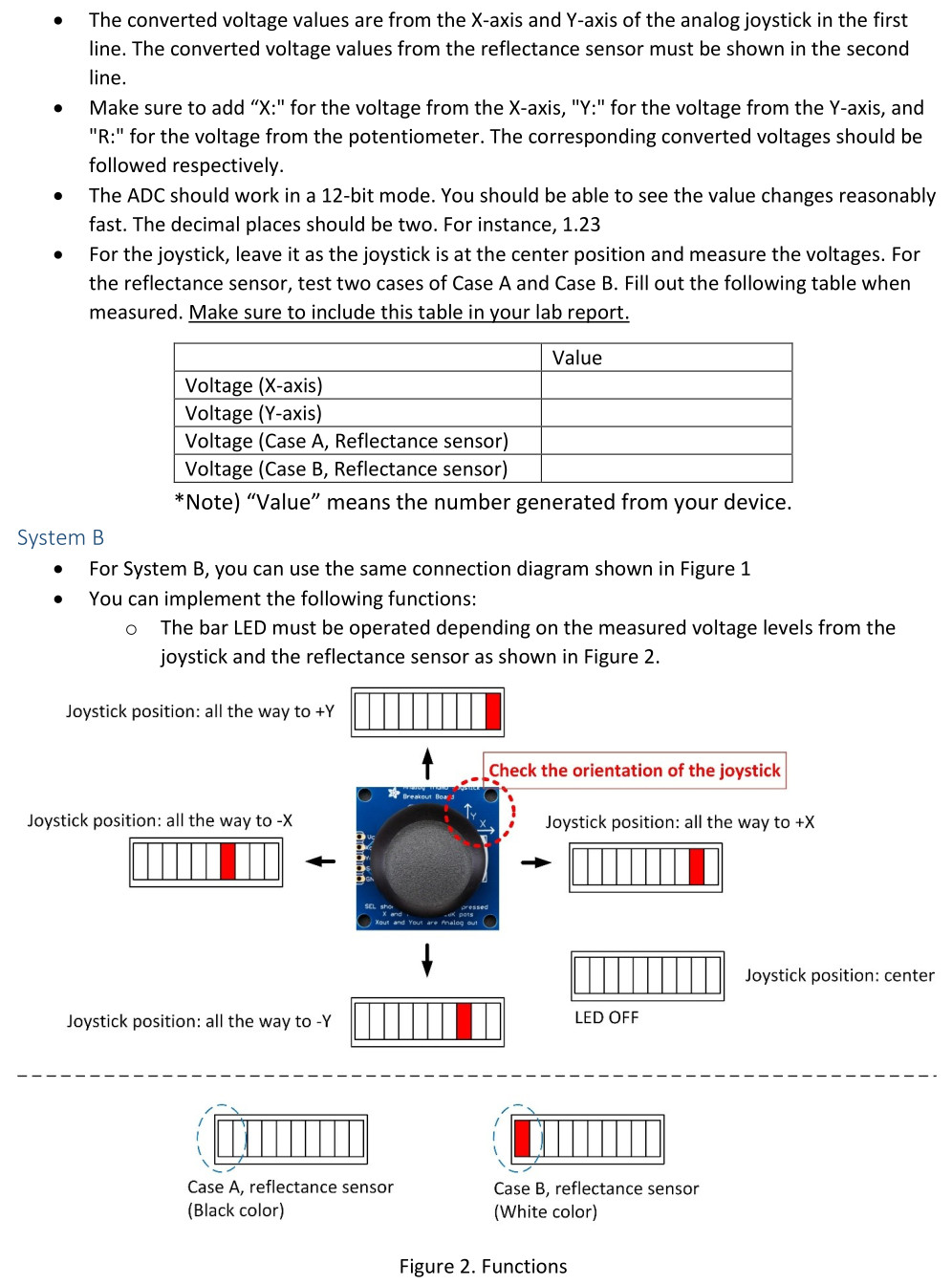
**Prompt from Canvas:**





**Simplified prompt:**

You will use a reflectance sensor. This sensor is a QTR-3A Reflectance Sensor Array from Pololu.

You can find the white paper with the black rectangle shape on it. You can test the sensor for the following two cases.

(Case A) The reflectance sensor is close to the black portion of the test paper with a distance of 0.5 inches or less.

(Case B) The reflectance sensor is close to the white portion of the test paper with a distance of 0.5 inches or less.

Write a program to display the corresponding multiple converted voltages from three given ADC channels for the analog joystick and reflectance sensor.

The converted voltage values are from the X-axis and Y-axis of the analog joystick in the first line. The converted voltage values from the reflectance sensor must be shown in the second line.

Make sure to add "X:" for the voltage from the X-axis, "Y:" for the voltage from the Y-axis, and "R:" for the voltage from the potentiometer. The corresponding converted voltages should be followed respectively.

The ADC should work in a 12-bit mode. You should be able to see the value changes reasonably fast. The decimal places should be two. For instance, 1.23

For the joystick, leave it as the joystick is at the center position and measure the voltages. For the reflectance sensor, test two cases of Case A and Case B. Fill out the following table when measured. Make sure to include this table in your lab report.

|  |  |
| --- | --- |
|  | Value |
| Voltage (X-axis) (Right/Left) | 3.29/0 |
| Voltage (Y-axis) (Up/Down) | 3.29/0 |
| Voltage (Case A – Black, Reflectance sensor) | 2.70 |
| Voltage (Case B – White, Reflectance sensor) | 0.19 |

\*Note: “Value” means the number generated from your device.

**Example of a program from a previous lab (That might help):**

#include <msp430.h>

unsigned int adc\_raw[3]; // array

int main(void) {

WDTCTL = WDTPW | WDTHOLD; // hold watchdog timer

PM5CTL0 &= ~LOCKLPM5; // clear LOCKLPM5 bit

P1DIR |= 0x01; // output direction (P1.0)

P4SEL1 |= 0x0E; // alternate function (A9, A10, A11)

P4SEL0 |= 0x0E; // alternate function (A9, A10, A11)

ADC12CTL0 = ADC12SHT0\_6 | ADC12MSC | ADC12ON; // ADC CTL0 set up

ADC12CTL1 = ADC12SHP | ADC12CONSEQ\_1; // ADC CTL1 set up

ADC12CTL2 = ADC12RES\_2; // 12-bit conversion

// multiple sample conversion, sequence of channel mode

ADC12MCTL0 = ADC12INCH\_9;

ADC12MCTL1 = ADC12INCH\_10;

ADC12MCTL2 = ADC12INCH\_11 | ADC12EOS; // end of sequence

while(1) {

ADC12CTL0 |= ADC12ENC | ADC12SC; // ADC, Start conversion

while ((ADC12IFGR0 & BIT2)==0); // flag check

adc\_raw[0]=ADC12MEM0; // read ADC

adc\_raw[1]=ADC12MEM1;

adc\_raw[2]=ADC12MEM2;

P1OUT ^= 0x01; // toggle (P1.0)

\_\_delay\_cycles(25000); // delay

}

return 0;

}

**Example of a program from a previous lab that uses the LCD screen (That might help):**

#include <msp430.h>

void LCD\_command(unsigned char);

void LCD\_write(unsigned char);

void LCD\_init(void);

unsigned int adc\_raw;

double volts;

int i;

/\*\*

\* main.c

\*/

void main(void)

{

WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

PM5CTL0 &= ~LOCKLPM5;

ADC12CTL0 = ADC12SHT0\_6 | ADC12ON;

ADC12CTL1 = ADC12SHP;

ADC12CTL2 = ADC12RES\_2;

ADC12MCTL0 = ADC12INCH\_9;

ADC12IER0 |= ADC12IE0;

TA1CTL = TASSEL\_2 | MC\_1 | TACLR; //timer A

P6DIR |= 0x01;

P3DIR |= 0xFF;

P3OUT &= ~0xFF;

P8DIR |= 0x0E;

P8OUT |= 0x0E;

P4SEL1 |= BIT1;

P4SEL0 |= BIT1;

TA1CCR0 = 0;

\_\_enable\_interrupt();

while(1) {

ADC12CTL0 |= ADC12ENC | ADC12SC;

volts = (((double)(adc\_raw)) / 4095) \* 3.3;

\_\_delay\_cycles(5000);

LCD\_init(); //initialize LCD

LCD\_command(0x01); //clear display

LCD\_command(0x00); //put at the first spot

\_\_delay\_cycles(5000);

LCD\_write('N');

LCD\_write('i');

LCD\_write('k');

LCD\_write('e');

LCD\_write('s');

LCD\_write('h');

LCD\_write(' ');

LCD\_write('N');

LCD\_write('a');

LCD\_write('i');

LCD\_write('r');

LCD\_command(0xC0);

LCD\_write('0' + (int)volts);

LCD\_write('.');

LCD\_write('0' + ((int)(volts \* 10) % 10));

LCD\_write('0' + ((int)(volts \* 100) % 10));

if (volts < 1.00) {

TA1CCR0 =0;

for( i=0; i < 3000; i++){

if ((TA1CCTL0 & CCIFG) !=0){

P6OUT ^= 0x01;

TA1CCTL0 &= ~CCIFG;

}

}

} else if (volts >= 1.00 && volts < 2.00) {

TA1CCR0 =956;

for( i=0; i < 3000; i++){

if ((TA1CCTL0 & CCIFG) !=0){

P6OUT ^= 0x01;

TA1CCTL0 &= ~CCIFG;

}

}

} else if (volts >= 2.00 && volts < 3.00) {

TA1CCR0 = 851;

for( i=0; i < 3000; i++){

if ((TA1CCTL0 & CCIFG) !=0){

P6OUT ^= 0x01;

TA1CCTL0 &= ~CCIFG;

}

}

} else if (volts >= 3.00 && volts <= 3.30) {

TA1CCR0 = 758;

for( i=0; i < 3000; i++){

if ((TA1CCTL0 & CCIFG) !=0){

P6OUT ^= 0x01;

TA1CCTL0 &= ~CCIFG;

}

}

} else {

TA1CCR0 =0;

for( i=0; i < 3000; i++){

if ((TA1CCTL0 & CCIFG) !=0){

P6OUT ^= 0x01;

TA1CCTL0 &= ~CCIFG;

}

}

}

}

}

#pragma vector = ADC12\_B\_VECTOR

\_\_interrupt void ADC12\_ISR(void) {

if ((ADC12IFGR0 & BIT0) != 0) {

adc\_raw = ADC12MEM0;

}

}

void LCD\_command(unsigned char in) {

P3OUT = in;

P8OUT &= ~BIT3;

P8OUT &= ~BIT2;

P8OUT |= BIT1;

\_\_delay\_cycles(200);

P8OUT &= ~BIT1;

}

void LCD\_write(unsigned char in) {

P3OUT = in;

P8OUT |= BIT3;

P8OUT &= ~BIT2;

P8OUT |= BIT1;

\_\_delay\_cycles(200);

P8OUT &= ~BIT1;

}

void LCD\_init() {

P8OUT &= ~BIT1;

\_\_delay\_cycles(15000);

LCD\_command(0x30);

\_\_delay\_cycles(300);

LCD\_command(0x30);

\_\_delay\_cycles(300);

LCD\_command(0x30);

\_\_delay\_cycles(300);

LCD\_command(0x38);

LCD\_command(0x10);

LCD\_command(0x0F);

LCD\_command(0x06);

LCD\_command(0x01);

\_\_delay\_cycles(3000);

}

**Example of empty program:**

#include <msp430.h>

/\*\*

\* main.c

\*/

int main(void)

{

WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

return 0;

}

**Connections:**

MSP430FR5994 Launchpad Connections

16x2 LCD screen Connections

P8.3 connects to RS

P8.2 connects to R/W

P8.1 connects to E

P3.0 connects to DB1

P3.1 connects to DB1

P3.2 connects to DB2

P3.3 connects to DB3

P3.4 connects to DB4

P3.5 connects to DB5

P3.6 connects to DB6

P3.7 connects to DB7

LED Connections

P6.0 connects to LED1

P6.1 connects to LED2

P7.0 connects to LED3

P7.1 connects to LED4

P6.2 connects to LED5

Reflectance Sensor Connections

G port of sensor connects to the GND port

V port of sensor connects to 3.3V port

2 port of sensor connects to P4.3

2 Axis Analog Joystick Connections

GND port of joystick connects to the GND port

VCC port of joystick connects to 3.3V port

Xout port of joystick connects to P4.1

Yout port of joystick connects to P4.2