Group #12 **Fall ECGR 4101/5101 Lab #05 Report** **10/06/2022**
Naseeruddin Lodge
nlodge@uncc.edu
Srikar Thakkallapally
sthakkal@uncc.edu

Lab Objective:

The objective of this lab is to write a code in Code Composer Studio (CCS) that will read the value of 3-axis accelerometer and convert them to gravity values on –3G to +3G scale. Both the raw values and the G values are displayed on a quad digit 7-segment display. All the raw values and G values are shown in 3 second delays generated by timer. On-board button utilizes interrupts to help switch between raw values and G values.

Commentary and Conclusion:

A lot of accelerometer related issues occurred during this lab. Many different methods were used to make sure the accelerometer was working as intended. First multiple different ports were used to make sure the accelerometer consistently. Then a potentiometer and the quad digit 7 segment LED was used to make sure that the ADC conversion was going smoothly. The second problem occurred was to find the method to convert the ADC values into G values. After many trials and errors, we were able to derive a formula that takes in a ADC value and convert into G values. The most challenging part was to get timers and interrupts right. For timers we had to test multiple different methods and finally we figured out that the best method was to declare a flag for timer and then use that for the delay. For button interrupts, after multiple trials we figured out that the best way to go about is to use bit masking to prevent button from changing states and it worked out at the end. Overall, it was a very challenging project and taught us how to use timers and interrupts efficiently.

Button	Pins
On-board button	P1.3

Table 1 – On-board button connection

Quad digit 7-segment display	Pins
A	P2.0
B	P2.1
C	P2.2
D	P2.3
E	P2.4
F	P2.5
G	P2.6

DP	P2.7
D1	P1.0
D2	P1.1
D3	P1.2
D4	P1.4

Table 2 - Quad digit 7-segment display connections

Accelerometer	Pins
Vcc	Vcc
X-axis	P1.7
Y-axis	P1.6
Z-axis	P1.5
GND	GND

Table 3 – Accelerometer connections

Figures:

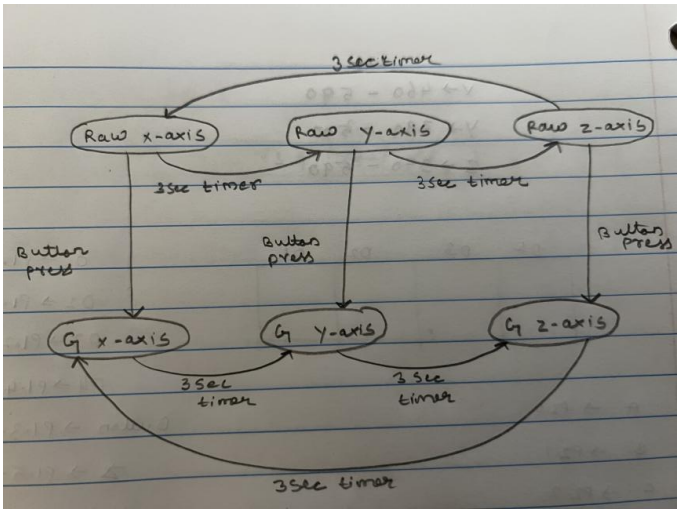

Figure 1 – FSM for the program

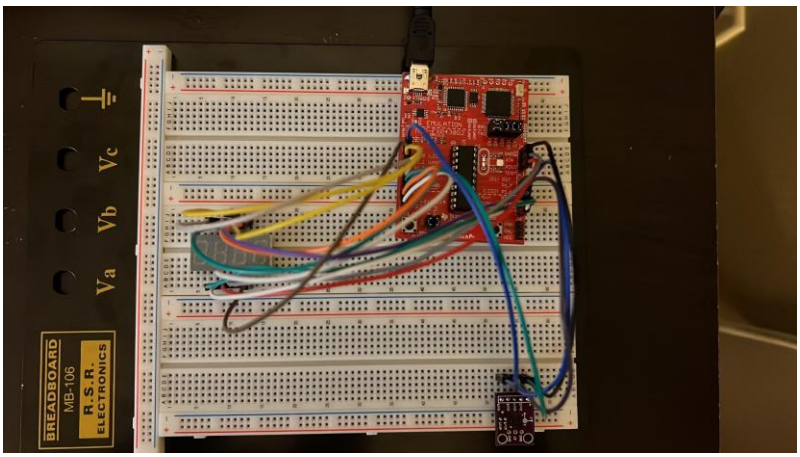

Figure 2 – Circuit Connection

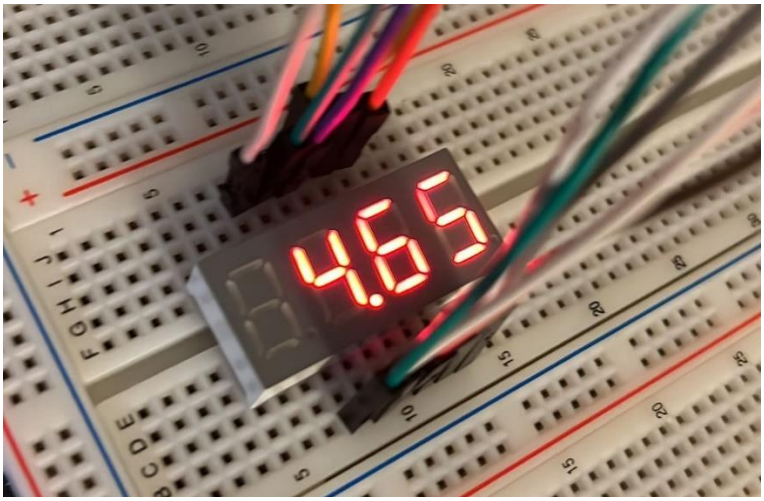

Figure 3 – Quad Digit 7 segment LED displaying raw y-axis values

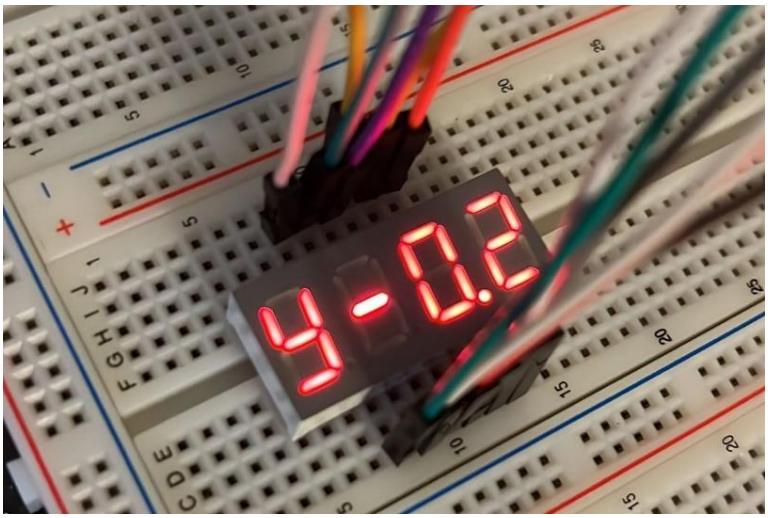

Figure 4 – Quad Digit 7 segment LED displaying G values for y-axis

Lab Code:

```
#include <msp430.h>


//Global Variables
unsigned int OFCount = 0;
unsigned int button_pressed = 0;
int step1=0, step2=0;
int step3=0, step4=0;
int flag1=0, flag2=0;
unsigned int adc[3];



//Prototyped Functions
void Port_setup();
void ADC_Setup();
void initTimer_A(void);
void delayMS(int millisec);
void DisplayLED(int n_SingleDigit);
void Control_Dx(int n, char xyz);
void correct_osciallations_raw(char xyz);
void Display_g(int adcValue, char xyz);
void correct_osciallations_g(char xyz);
void rawValues();
void gValues();



//Main Function
void main(void)
{
    //Stop WatchDog Timer
    WDTCTL = WDTPW + WDTHOLD;

    //Turn on GPIO for Xin and Xout
    P2SEL = 0;
    P2SEL2 = 0;

    //Set P1.5, P1.6 and P1.7
    //direction as input(Accelerometer)
    //Push Button P1.3 as input
    //D1, D2, D3, D4 are connected to
    //P1.0, P1.1, P1.2, P1.4 as outputs
    P1DIR |= 0x17;

    //Setup ports for button, interrupt
    //and timer
    Port_setup();

    //Initialize Timer
    initTimer_A();
```

```
    //Setup ADC for P1.0, P1.1
    //and P1.2
    ADC_Setup();

    //Enable interrupt
    _enable_interrupt();

    OFCount  = 0;
    TACCR0 = 1000-1;

    while(1){

        //Used to activate ADC
        //and record x, y and z
        //axis ADC values from the
        //accelerometer and
        //store them in an array
        ADC10CTL0 &= ~ENC;
        while(ADC10CTL1 & BUSY);
        ADC10CTL0 |= ENC + ADC10SC;
        ADC10SA = (unsigned int)adc;

        //Switch between raw and converted values when button is
        //pressed
        if(button_pressed == 0) rawValues();

        else if(button_pressed == 1) gValues();

    }
}



//*****************************************************************
//Name    : Port_setup()
//Input   : void
//Returns : void
//
//Setup ports for button, interrupt and timer.
//*****************************************************************
void Port_setup(){

    //Set P2.0-P1.7 direction as input
    //(Quad 7-segment LED)
    P2DIR |= 0xFF;

    //Enable Pull Up resistor for SW2
    P1REN |= 0x08;

    //P1.3 interrupt enabled
    P1IE |= 0x08;

    //P1.3 Hi/lo edge
    P1IES |= 0x08;
```

```
    //P1.3 IFG cleared
    P1IFG &= ~(0x08);

    //MCLK = SMCLK = 1MHZ
    BCSCTL1 = CALBC1_1MHZ;
    DCOCTL = CALDCO_1MHZ;

}

//*****************************************************************
//Name    : ADC_Setup()
//Input   : void
//Returns : void
//
//Function to Setup ADC
//*****************************************************************
void ADC_Setup(){

    //SREF     -> 000b = VR+ = VCC and VR- = VSS
    //ADC10SHT -> 10b = 16 ADC10CLK cycles
    //ADC10ON  -> ADC10 on
    //INCH     -> Input channel select

    ADC10CTL1 = INCH_7 + ADC10DIV_0 + CONSEQ_3 + SHS_0;
    ADC10CTL0 = SREF_0 + ADC10SHT_2 + MSC + ADC10ON;
    ADC10AE0 |= BIT7 + BIT6 + BIT5;
    ADC10DTC1 = 8;
}



//*****************************************************************
//Name    : initTimer_A(void)
//Input   : Void
//Returns : Void
//
//Function to Setup Timer COnfiguration
//*****************************************************************
void initTimer_A(void){

    //Initially, Stop the Timer
    TACCR0 = 0;

    //Enable interrupt for CCR0.
    TACCTL0 |= CCIE;

    //SMCLK, ID = 1, SMCLK/1 , Up Mode->TACCR0
    TACTL = TASSEL_2 + ID_0 + MC_1;
}



//*****************************************************************
//Name    : __interrupt void Port_1(void)
//Input   : void
```

```
//Returns : void
//
//Port 1 interrupt service routine
//*************************************************************
#pragma vector=PORT1_VECTOR
__interrupt void PORT1_ISR(void){

    //button pressed is either 1 or 0
    //0 -> display raw values of ADC
    //1 -> -3G to 3G scale ADC values
    if(P1IFG && BIT3){

        //P1OUT ^= 0x00;
        button_pressed = (button_pressed + 1) % 2;
    }

    __delay_cycles(100000);

    //P1.3 IFG cleared
    P1IFG &= ~(0x08);
}



//*************************************************************
//Name    : __interrupt void Timer_A_CCR0_ISR(void)
//Input   : void
//Returns : void
//
//Timer ISR. increments OFCount
//*************************************************************
#pragma vector = TIMER0_A0_VECTOR
__interrupt void Timer_A_CCR0_ISR(void){

    //Increment Over-Flow Counter
    OFCount++;
    if(OFCount >= 3000){

        flag1 = flag2 % 3;
        flag2++;
        OFCount = 0;
    }
}



//*************************************************************
//Name    : DisplayLED(char n_SingleDigit)
//Input   : char
//Returns : void
//
//Used to display a number.
//char 0-9 is given as an input and the corresponding
//light turns on using a switch statement
//Case 10 is used to turn off LED and turn off All the LED digits
```

```
//X,y,Z are for axises
//- is for negative symbol
//*************************************************************
void DisplayLED(int n_SingleDigit){

    switch(n_SingleDigit){

        //P2OUT = 0xhgfedcba
        //0x00111111
        case 0:
            P2OUT = 0xC0;
            break;

        //0x00000110
        case 1:
            P2OUT = 0xF9;
            break;

        //0x01011011
        case 2:
            P2OUT = 0xA4;
            break;

        //0x01001111
        case 3:
            P2OUT = 0xB0;
            break;

        //0x01100110
        case 4:
            P2OUT = 0x99;
            break;

        //0x01101101
        case 5:
            P2OUT = 0x92;
            break;

        //0x01111101
        case 6:
            P2OUT = 0x82;
            break;

        //0x00000111
        case 7:
            P2OUT = 0xF8;
            break;

        //0x01111111
        case 8:
            P2OUT = 0x80;
            break;

        //0x01110111
        case 9:
```

```
            P2OUT = 0x90;
            break;

        //0x01110110
        //Displays X
        case 10:
            P2OUT = 0x89;
            break;

        //0x01101110
        //Displays y
        case 11:
            P2OUT = 0x91;
            break;

        //0x01011011
        //Displays Z
        case 12:
            P2OUT = 0xA4;
            break;

        //0x01000000
        //Negative Sign
        case 13:
            P2OUT = 0xBF;
            break;

        //0x10000000
        //Decimal Point
        case 14:
            P2OUT = 0x7F;
            break;

        //case n is used to show the symbol '+'
        //which is turn off that particular LED
        //in our case
        //case 'n':
        //    P2OUT = 0xFF;
        //    break;

        //case E is used to turn off the LED
        //completely
        case 15:
            P1OUT &= 0x08;
            P2OUT = 0xFF;
            break;


    }

}


//*************************************************************
```

```
//Name    : Control_Dx(int n, char xyz)
//Input   : int, char
//Returns : void
//
//This function is responsible for displaying the numbers.
//It gets an input of the reading of the potentiometer.
//Depending on the number of digits required, the digits are turned on.
//Then the Display LED function is used to turn on the numbers, one
//number at a time.
//If the ADC number is 357. 7 is displayed on D1, 5 on D2 and 3 on D3.
//DisplayLED(10) is used to turn off everything
//*************************************************************
void Control_Dx(int n, char xyz){

    int SingleDigit;

    //if n = 271, 2 is displayed on D1
    //then D1 is turned off and D2 is turned on
    //and 7 is displayed. Then D3 is turned on
    //and 1 is displayed

    if(n <= 9){

        DisplayLED(15);

        if(xyz == 'x'){

            P1OUT ^= 0x10;
            DisplayLED(14);
        }

        else if(xyz == 'y'){

            P1OUT ^= 0x04;
            DisplayLED(14);
        }

        else if(xyz == 'z'){

            P1OUT ^= 0x02;
            DisplayLED(14);
        }
        __delay_cycles(1000);

        //D4 - 0x00100000

        SingleDigit = n;
        DisplayLED(15);
        P1OUT ^= 0x01;
        DisplayLED(SingleDigit);
        __delay_cycles(1000);

    }

    else if((n>=10) && (n<=99)){
```

```
    //D4 - 0x00100000
    //D3 - 0x00010000

    DisplayLED(15);

    if(xyz == 'x'){

        P1OUT ^= 0x10;
        DisplayLED(14);
    }

    else if(xyz == 'y'){

        P1OUT ^= 0x04;
        DisplayLED(14);
    }

    else if(xyz == 'z'){

        P1OUT ^= 0x02;
        DisplayLED(14);
    }
    __delay_cycles(1000);

    SingleDigit = n % 10;
    DisplayLED(15);
    P1OUT ^= 0x01;
    DisplayLED(SingleDigit);
    __delay_cycles(1000);

    SingleDigit = n / 10 % 10;
    DisplayLED(15);
    P1OUT ^= 0x02;
    DisplayLED(SingleDigit);
    __delay_cycles(1000);

    DisplayLED(15);
}

else if((n>=100) && (n<=999)){

    //D4 - 0x00100000
    //D3 - 0x00010000
    //D2 - 0x00000100

    DisplayLED(15);

    if(xyz == 'x'){

        P1OUT ^= 0x10;
        DisplayLED(14);
    }

    else if(xyz == 'y'){
```

```
            P1OUT ^= 0x04;
            DisplayLED(14);
        }

        else if(xyz == 'z'){

            P1OUT ^= 0x02;
            DisplayLED(14);
        }
        __delay_cycles(1000);

        DisplayLED(15);

        SingleDigit = n % 10;
        DisplayLED(15);
        P1OUT ^= 0x01;
        DisplayLED(SingleDigit);
        __delay_cycles(1000);

        SingleDigit = n / 10 % 10;
        DisplayLED(15);
        P1OUT ^= 0x02;
        DisplayLED(SingleDigit);
        __delay_cycles(1000);

        SingleDigit = n / 100 % 10;
        DisplayLED(15);
        P1OUT ^= 0x04;
        DisplayLED(SingleDigit);
        __delay_cycles(1000);

    }

    else if(n >= 1000){

        //D4 - 0x00100000
        //D3 - 0x00010000
        //D2 - 0x00000100
        //D1 - 0x00000010

        DisplayLED(15);

        if(xyz == 'x'){

            P1OUT ^= 0x10;
            DisplayLED(14);
        }

        else if(xyz == 'y'){

            P1OUT ^= 0x04;
            DisplayLED(14);
        }
```

```
        else if(xyz == 'z'){

            P1OUT ^= 0x02;
            DisplayLED(14);
        }
        __delay_cycles(1000);

        DisplayLED(15);

        SingleDigit = n % 10;
        DisplayLED(15);
        P1OUT ^= 0x01;
        DisplayLED(SingleDigit);
        __delay_cycles(1000);

        SingleDigit = n / 10 % 10;
        DisplayLED(15);
        P1OUT ^= 0x02;
        DisplayLED(SingleDigit);
        __delay_cycles(1000);

        SingleDigit = n / 100 % 10;
        DisplayLED(15);
        P1OUT ^= 0x04;
        DisplayLED(SingleDigit);
        __delay_cycles(1000);

        SingleDigit = n / 1000 % 10;
        DisplayLED(15);
        P1OUT ^= 0x10;
        DisplayLED(SingleDigit);
        __delay_cycles(1000);

        DisplayLED(15);
    }
}



//*****************************************************************
//Name    : correct_osciallations_raw()
//Input   : void
//Returns : void
//
//This function is responsible for fixing oscillations.
//If n=50 and it oscillates between n=49 and
//n=51, this code check to see if this kind of
//oscillation has occured and sets n=50.
//Also accounts for n=0 and n=1023. Responsible for preventing
//oscillations for only X-axis
//*****************************************************************
void correct_osciallations_raw(char xyz){

    if(step1 == 0){
```

```
        Control_Dx(0, xyz);
    }

    else if(step1 == 1023){

        Control_Dx(1023, xyz);
    }

    else if(step2 == 0){

        Control_Dx(step1, xyz);
    }

    else if((step1 == 1) && (step2 == 1)){

        Control_Dx(step1, xyz);
    }

    else if((step2 == (step1+1))){

        Control_Dx(step2, xyz);
        step1 = step2;
    }

    else if(step2 == (step1-1)){

        Control_Dx(step2, xyz);
        step1 = step2;
    }

    else if((step2 != (step1+1)) || (step2 != (step1-1))){

        Control_Dx(step1, xyz);
    }

    step2 = step1;
}



void Display_g(int adcValue, char xyz){

    int gVal, SingleDigit;

    DisplayLED(15);
    P1OUT ^= 0x02;
    DisplayLED(14);
    __delay_cycles(1000);
    DisplayLED(15);


    P1OUT ^= 0x10;
    if(xyz == 'x') DisplayLED(10);

    else if(xyz == 'y') DisplayLED(11);
```

```
else if(xyz == 'z') DisplayLED(12);

__delay_cycles(1000);
DisplayLED(15);

if(adcValue == 512){

    P1OUT ^= 0x02;
    DisplayLED(0);
    __delay_cycles(1000);

    DisplayLED(15);

    P1OUT ^= 0x01;
    DisplayLED(0);
    __delay_cycles(1000);

}

else if(adcValue < 512){

    P1OUT ^= 0x04;
    DisplayLED(13);
    __delay_cycles(1000);

    //gVal = (((6*(float)(adcVal-minVal))/(float)(maxVal-minVal)) - 3) * 10;
    gVal = (((6*((float)adcValue))/1023) - 3) * 10;
    gVal = abs(gVal);

    DisplayLED(15);

    SingleDigit = gVal % 10;
    P1OUT ^= 0x01;
    DisplayLED(SingleDigit);
    __delay_cycles(1000);

    DisplayLED(15);

    SingleDigit = gVal / 10 % 10;
    P1OUT ^= 0x02;
    DisplayLED(SingleDigit);
    __delay_cycles(1000);

}

else if(adcValue > 512){


    //gVal = (((6*(float)(adcVal-minVal))/(float)(maxVal-minVal)) - 3) * 10;
    gVal = (((6*((float)adcValue))/1023) - 3) * 10;
    gVal = abs(gVal);

    DisplayLED(15);
```

```
        SingleDigit = gVal % 10;
        P1OUT ^= 0x01;
        DisplayLED(SingleDigit);
        __delay_cycles(1000);

        DisplayLED(15);

        SingleDigit = gVal / 10 % 10;
        P1OUT ^= 0x02;
        DisplayLED(SingleDigit);
        __delay_cycles(1000);
    }


}



//*****************************************************************
//Name    : correct_osciallations_g(char xyz)
//Input   : char
//Returns : void
//
//Used to fix oscillations before displaying -3G to 3G values
//*****************************************************************
void correct_osciallations_g(char xyz){

    if(step3 == 0){

        Display_g(0, xyz);
    }

    else if(step3 == 1023){

        Display_g(1023, xyz);
    }

    else if(step4 == 0){

        Display_g(step3, xyz);
    }

    else if((step3 == 1) && (step4 == 1)){

        Display_g(step3, xyz);
    }

    else if((step4 == (step3+1))){

        Display_g(step4, xyz);
        step3 = step4;
    }

    else if(step4 == (step3-1)){
```

```
        Display_g(step4, xyz);
        step3 = step4;
    }

    else if((step4 != (step3+1)) || (step4 != (step3-1))){

        Display_g(step3, xyz);
    }


    step4 = step3;
}



//*****************************************************************
//Name    : rawValues()
//Input   : void
//Returns : void
//
//Used to display raw values of the accelerometer. Displays x, y
//and z axis raw values on the LED display with 3 second delay.
//*****************************************************************
void rawValues(){


    step1 = adc[0];
    while(flag1==0 && button_pressed==0){

    //X-axis
    //step1 = adc[0];
    correct_osciallations_raw('x');
    }

    step1 = adc[1];
    while(flag1 == 1 && button_pressed==0){

    //Y-axis
    //step1 = adc[1];
    correct_osciallations_raw('y');
    }

    step1 = adc[2];
    while(flag1 == 2 && button_pressed==0){

    //Z-axis
    //step1 = adc[2];
    correct_osciallations_raw('z');
    }

}



//*****************************************************************
```

```
//Name    : gValues()
//Input   : void
//Returns : void
//
//Used to display raw values of the accelerometer. Displays x, y
//and z axis G values on the LED display with 3 second delay.
//G scale is -3G to 3G.
//*************************************************************
void gValues(){


    step3 = adc[0];
    while(flag1==0 && button_pressed==1){

    //X-axis
    //step3 = adc[0];
    correct_osciallations_g('x');
    }

    step3 = adc[1];
    while(flag1==1 && button_pressed==1){

    //Y-axis
    //step3 = adc[1];
    correct_osciallations_g('y');
    }

    step3 = adc[2];
    while(flag1==2 && button_pressed==1){

    //Z-axis
    //step3 = adc[2];
    correct_osciallations_g('z');

    }

}
```