CPE 325: Intro to Embedded Computer System

FINAL LAB PROJECT.

UART/ADC with ADXL335 Accelerometer.

Submitted by: Dan Otieno

Date of Experiment: 04/14/2022

Report Deadline: 04/20/2022

Demonstration Deadline: 04/21/2022

Introduction

The purpose of this project is to build on 2 of the past labs to enhance the various methods of outputs between the ADXL335 accelerometer and the MSP430 board. The code included below determines accelerations on a 3-dimensional axes relative to the gravity of earth. We also calculate the angle of deviation in reference to the y-axis. Depending on whether the angle is high, low or right in the resting z axes (greater than 15°, less than -15° or within -15° and 15°), there is a message displayed in the HyperTerminal, we use the Watchdog Timer ISR to interface the buzzer and LEDs.

Theory

Topic 1: **Serial Communication and UART.**

This is the means through which an MSP430-based platform can communicate with another system such as a personal computer, using either the synchronous or asynchronous communication mode. Two devices establish synchronous communication by sharing a common clock. The internal divider (determined by dividing the clock by the baud rate) and modulation register in the MSP430 are initialed to configure the MSP430 in UART mode. The modulation register accounts for the fraction part of that division. As stated, an MSP430 can be connected to a PC using a HyperTerminal application (e.g. MobaXterm) in Windows. This has previously been covered on a Lab.

Topic 2: **ADXL335 ACCELEROMETER:**

This device is a 3-axis with outputs controlled by signals. We connect the accelerometer to the MSP430 board via Header 8 pins, the ground and VCC pins. When programmed, we can read the axis output signals, in this case using the UAH Serial App and observe the changes as we change the accelerometer's orientation. Parts 1 and 2 of this lab required interfacing the accelerometer via ADC signals and configuring the LEDs on the MSP430 to logics high or low depending on the angular deviation relative to the x-axis and comparing that deviation to the values 15 and -15.

Topic 3: **ADC and DAC:**

Analog-to-Digital Conversion is achieved by converters that enable us to interact with analog signals, convert them to digital values, store, interpret and observe them for further studying. For this lab, we utilized the ADC12 converter of the MSP430 to achieve our outputs. This lab involves utilizing ADC12 to convert signals using the accelerometer into our microcontroller.

Results & Observation

Program 1:

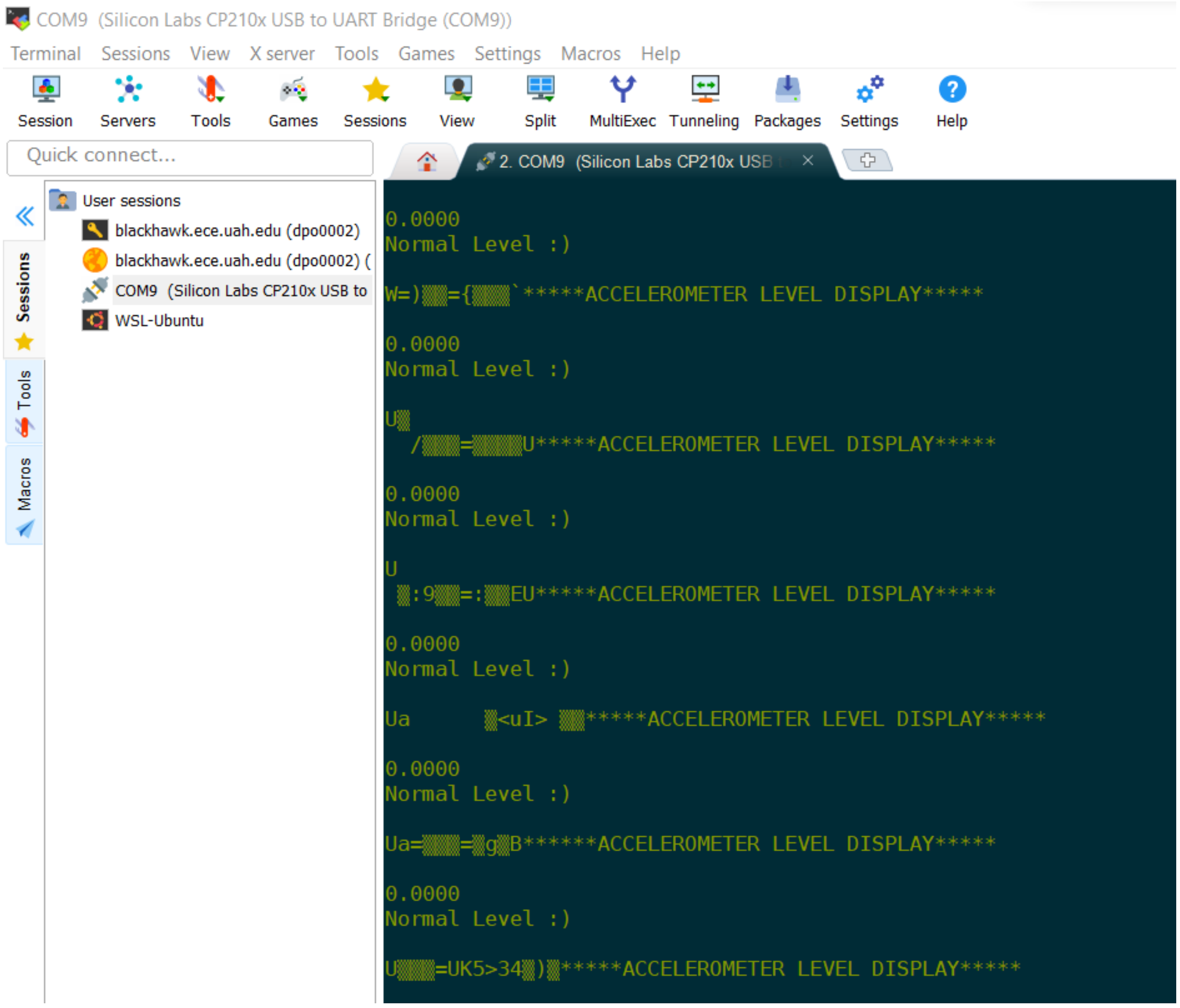
Program Description:

As stated, the approach here was to build on my previous codes from the past labs, incorporating UART and ADC to get different types of outputs by interfacing the accelerometer. The LED pins for the 4618 are from Port 2, with bits 1 and 2 as outputs. We also have Port 3 as a special port to run the buzzer. The LEDs are set to blink at 1Khz with the SMCLK as clock source for UART. The baud rate to transmit data to the terminal is set at 115200. Unfortunately, I was unable to determine why my terminal outputs had gibberish due to time constraints, I was unable to resolve this even with changing baud rates. I also discovered I could not send float values into the HyperTerminal, and did not get enough time to resolve this, despite using the sprintf function in the code.

The display in Mobaxterm is also set to output different font colors depending on whether the accelerometer is too high, too low or at the defined normal angle range. When the program starts, both LED lights are on as long as the accelerometer is not lifted or pulled down, when we move it up, the green LED blinks and the buzzer sounds, along with the words "too high" in the terminal, and when the accelerometer is lowered, the words "too low" are displayed in the terminal, the buzzer sounds, and the yellow LED blinks. When in normal range, both LEDs are on and the words "Normal Level" and a smiley are displayed in the terminal.

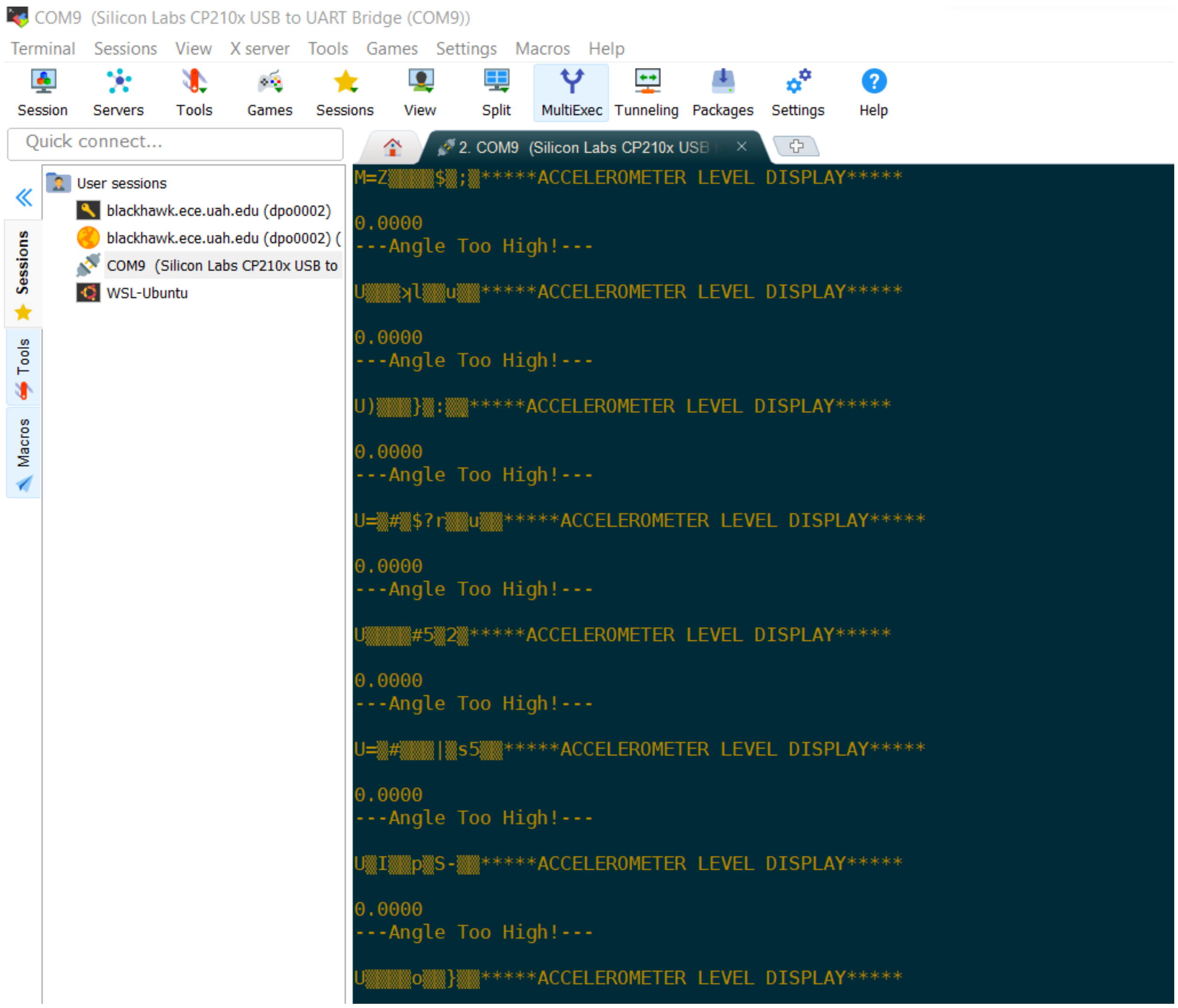
Program Output:

Normal Level: The accelerometer is not moved (Unable to resolve the gibberish).


COM9 (Silicon Labs CP210x USB to UART Bridge (COM9))
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
2. COM9 (Silicon Labs CP210x USB
User sessions
blackhawk.ece.uah.edu (dpo0002)
blackhawk.ece.uah.edu (dpo0002) (
COM9 (Silicon Labs CP210x USB to
WSL-Ubuntu
Sessions
Tools
Macros

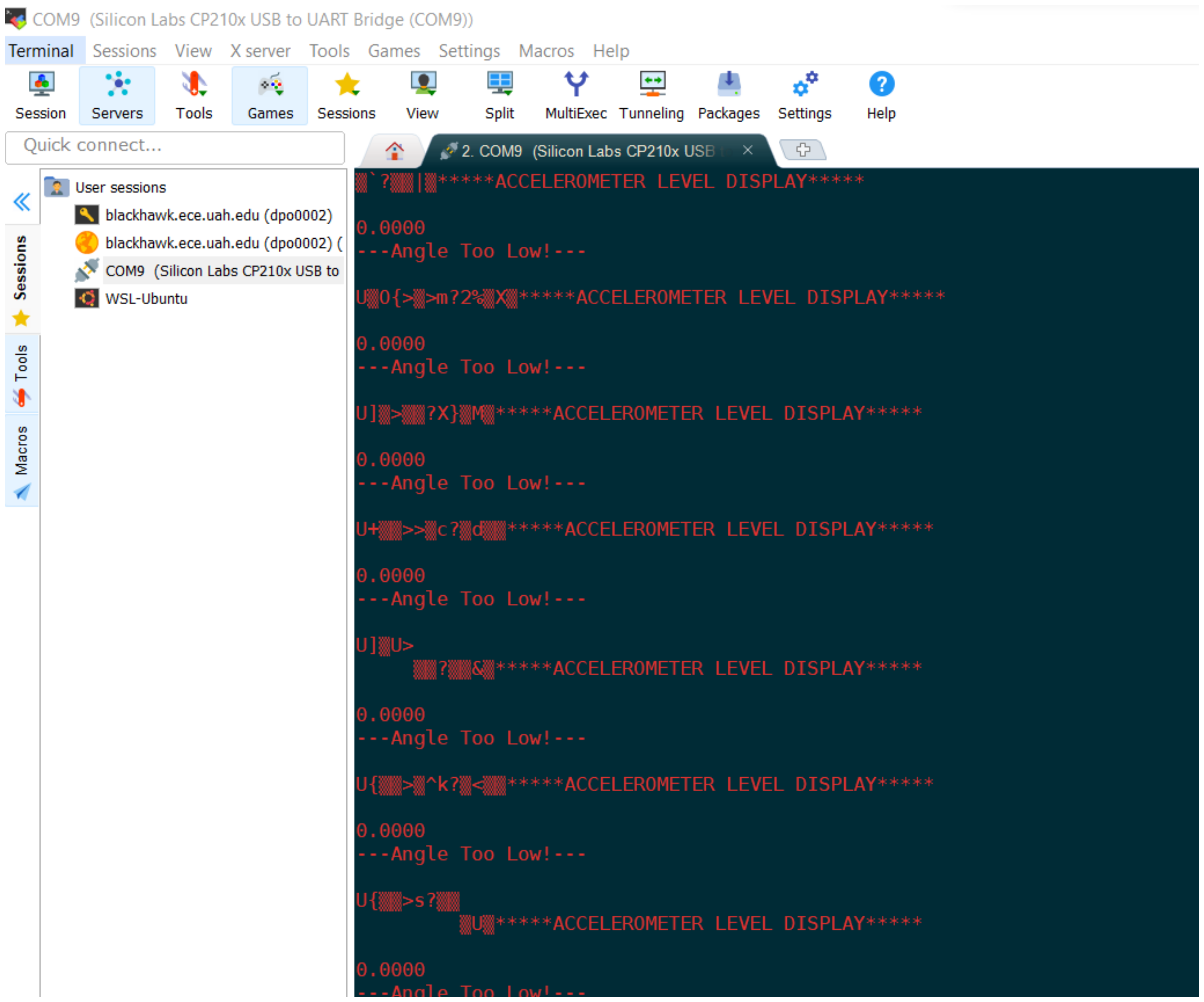
```
0.0000
Normal Level :)

W=)▒▒=(▒▒▒`*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
Normal Level :)

U▒
  /▒▒▒=▒▒▒▒U*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
Normal Level :)

U
 ▒:9▒▒=:▒▒EU*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
Normal Level :)

Ua      ▒<uI> ▒▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
Normal Level :)

Ua=▒▒▒=▒g▒B******ACCELEROMETER LEVEL DISPLAY*****

0.0000
Normal Level :)

U▒▒▒=UK5>34▒)▒*****ACCELEROMETER LEVEL DISPLAY*****
```

Accelerometer is hoisted high along y-axis.


COM9 (Silicon Labs CP210x USB to UART Bridge (COM9))
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
2. COM9 (Silicon Labs CP210x USB
User sessions
blackhawk.ece.uah.edu (dpo0002)
blackhawk.ece.uah.edu (dpo0002) (
COM9 (Silicon Labs CP210x USB to
WSL-Ubuntu
Sessions
Tools
Macros

```
M=Z▒▒▒▒$▒;▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too High!---

U▒▒▒>l▒▒u▒▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too High!---

U)▒▒▒}▒:▒▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too High!---

U=▒#▒$?r▒▒u▒▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too High!---

U▒▒▒▒#5▒2▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too High!---

U=▒#▒▒▒|▒s5▒▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too High!---

U▒I▒▒p▒S-▒▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too High!---

U▒▒▒▒o▒▒}▒▒*****ACCELEROMETER LEVEL DISPLAY*****
```

Accelerometer is lowered along the y-axis.


COM9 (Silicon Labs CP210x USB to UART Bridge (COM9))
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
2. COM9 (Silicon Labs CP210x USB
User sessions
blackhawk.ece.uah.edu (dpo0002)
blackhawk.ece.uah.edu (dpo0002) (
COM9 (Silicon Labs CP210x USB to
WSL-Ubuntu
Sessions
Tools
Macros

```
▒`?▒▒|▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too Low!---

U▒O{>▒>m?2%▒X▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too Low!---

U]▒>▒▒?X}▒M▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too Low!---

U+▒▒>>▒c?▒d▒▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too Low!---

U]▒U>
     ▒▒?▒▒&▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too Low!---

U{▒▒>▒^k?▒<▒▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too Low!---

U{▒▒>s?▒▒
         ▒U▒*****ACCELEROMETER LEVEL DISPLAY*****

0.0000
---Angle Too Low!---
```

Conclusion

Most of the outputs for this program will be demonstrated in the lab, aside from the gibberish that was included with the output displays in the HyperTerminal. However, this can be helpful with acting as a sensor in the real world to detect unusual deviation for flying vehicles or even modern day cars. The fact that there are several channels of output, means a warning can be passed through visual and audio means, along those lines, we can also use this to interface devices that may help with, say an elevator to using display and audio, where accessibility is considered for persons with disabilities of hearing or seeing.

Appendix

Table 01: Source Code

```
/*--------------------------------------------------------------------------------------------
----------*/
//*File:        FinalProject.c
//*Function:    This C code will interface the ADXL335 accelerometer using the UART and
ADC components of the MSP430.
//* Description: This program detects angle of deviation relative to the y-axis, we get a
buzzer sound, and LED blinks
//*              depending on whether the angle is too high, too low, LEDs stay on if
normal, and buzzer stays off.
//*              We also get messages displayed into the HypTerminal via UART on
deviation status.
//* Input:       None.
//* Output:      LEDs on when normal, LEDs blink when deviation is detected,
HyperTerminal messages, Buzzer tone.
//* Author(s):   Dan Otieno, dpo0002@uah.edu
//* Date:        April 16th, 2022.
/*--------------------------------------------------------------------------------------------
--------*/
#include <msp430.h>
#include <math.h>
#include <stdio.h>
#include <string.h>

char openMSG[] = "*****ACCELEROMETER LEVEL DISPLAY*****\n\r";
char steady[] = "\033[0;32mNormal Level :)\n\r";
char highdisp[] = "\033[0;33m---Angle Too High!---\n\r";
char lowdisp [] = "\033[0;31m---Angle Too Low!---\n\r";
char dvAngle[10];
volatile long int ADCXval, ADCYval, ADCZval;
volatile float Xper, Yper, Zper, aDev;
void UART_setup();
void sendData();
void UART_Send_Character(char);
void UART_send_String(char*, int);

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;       // stop watchdog timer;
/*****************************************************************************************
******
****WDT ISR
CONFIGURATIONS*******************************************************************/
    WDTCTL = WDT_ADLY_1000;         // Set interval to 1000ms/1s interval timer;
    IE1   |= WDTIE;                     // Enable WDT interrupts;
/*****************************************************************************************
******
****INPUT/OUTPUT CONFIGURATIONS FOR LEDs AND
BUZZER*******************************************/
    P3DIR |= BIT5;                      // 3.5 for buzzer;
    P3SEL |= BIT5;                      // Special function for P2.2 (TB4 Output);
    TBCTL |= TBSSEL_2 + MC_1;           // Timer B Clock Source: SMCLK;
    TBCCTL4 |= OUTMOD_4;                // Set Timer B4 to output;
    TBCCR0 = 2000;                      // Sets buzzer frequency to 1 KHz;
    P2DIR |= BIT1 + BIT2;               // P2.1 and P2.2 set up as output;
    P2OUT &= ~(BIT1 + BIT2);            // Ensure LED1 and LED2 are off;
    P3SEL &= ~BIT5;                     // Buzzer is off in the beginning;
/*****************************************************************************************
*******/
/*******************************ADC & UART SETUP IN
MAIN**************************************/
    ADC_setup();                    // Setup ADC;
    UART_setup();                   // Setup UART for RS-232;
    _EINT();                        // Enable Global interrupts;

    while (1)
    {
        ADC12CTL0 |= ADC12SC;                   // Start conversions
        __bis_SR_register(LPM0_bits + GIE); // Enter LPM0
    }
}/*-----------------------------------------------------------------------------------------
---*/
/*-----------------------UART CONFIGURATIONS--------------------------------------------
---*/
void UART_setup(void)
{
    UCA0CTL1 |= UCSWRST;                    // Software reset during initialization;
    P2SEL |= BIT4 + BIT5;                   // Set Rx and Tx bits (Transmit&Receive);
    UCA0CTL0 = 0;                           // Set up default RS-232 protocol;
    UCA0CTL1 |= UCSSEL_2;                   // Set clock source: SMCLK;
    UCA0BR0 = 0x09;                         // 1048576Hz/115200 - Baud Rate;
    UCA0BR1 = 0;
    UCA0MCTL = 0x02;                        //Modulation(CBRS0=0x02, UCOS16=0);
    UCA0CTL1 &= ~UCSWRST;                   // Clear software reset;
}/*-----------------------------------------------------------------------------------------
--*/
void sendData(void)
{
    int i;
    Xper = ((ADCXval*3.0/4095-1.5)/0.3);    // Calculate percentage outputs;
    Yper = ((ADCYval*3.0/4095-1.5)/0.3);
    Zper = ((ADCZval*3.0/4095-1.5)/0.3);
    aDev = atan(Yper/Zper)*180/3.14;

    // Use character pointers to send one byte at a time;
    char *xpointer=(char *)&Xper;
    char *ypointer=(char *)&Yper;
    char *zpointer=(char *)&Zper;
    char *dvpointer=(char *)&aDev;

    UART_Send_Character(0x55);            // Send header;
    for(i = 0; i < 4; i++)
    {             // Send x percentage - one byte at a time;
        UART_Send_Character(xpointer[i]);
    }
    for(i = 0; i < 4; i++)
    {             // Send y percentage - one byte at a time;
        UART_Send_Character(ypointer[i]);
    }
    for(i = 0; i < 4; i++)
    {             // Send z percentage - one byte at a time;
        UART_Send_Character(zpointer[i]);
    }

    for(i = 0; i < 4; i++)
    {             // Send deviation - one byte at a time;
        UART_Send_Character(dvpointer[i]);
    }
}/*-----------------------------------------------------------------------------------------
-*/
void UART_Send_Character(char my_char)
{
    while(!(IFG2&UCA0TXIFG));              // Wait until can transmit.
    UCA0TXBUF = my_char;                   // Tx Buffer gets my_char variable.
}/*-----------------------------------------------------------------------------------------
*/
void UART_send_String(char* string, int wrdlen)
{
    int i;

    for(i = 0; i<wrdlen; i++)          // Send character by character;
    {
        UART_Send_Character(string[i]);
    }
    UART_Send_Character('\n');
    UART_Send_Character('\r');
```

```
}/*-----------------------------------------------------------------------------------
*/
/*--------------------ADC CONFIGURATION-------------------------------------------------
*/
void ADC_setup(void)
{
    int i =0;

    P6DIR &= ~BIT3 + ~BIT5 + ~BIT7;     // Configure P6.4, P6.6 and P6.8 as input pins;
    P6SEL |= BIT3 + BIT5 + BIT7;        // Configure P6.4, P6.6 and P6.8 as analog pins;

    ADC12CTL0 = ADC12ON + SHT0_6 + MSC; // configure ADC converter;
    ADC12CTL1 = SHP + CONSEQ_1;         // Use sample timer, single sequence;
    ADC12MCTL0 = INCH_3;                // ADC A3 pin - Stick X-axis;
    ADC12MCTL1 = INCH_7;                // ADC A4 pin - Stick Y-axis;
    ADC12MCTL2 = INCH_5 + EOS;          // ADC A5 pin - Stick Z-axis;
                                        // EOS - End of Sequence for Conversions;
    ADC12IE |= 0x02;                    // Enable ADC12IFG.1;
    for (i = 0; i < 0x3600; i++);       // Delay for reference start-up;
    ADC12CTL0 |= ENC;                   // Enable conversions;
}/*-----------------------------------------------------------------------------------
*/
/*--------------------ADC12 ISR---------------------------------------------------------
*/
#pragma vector = ADC12_VECTOR
__interrupt void ADC12ISR(void)
{
    ADCXval = ADC12MEM0;                  // Move results, IFG is cleared;
    ADCYval = ADC12MEM1;
    ADCZval = ADC12MEM2;
    __bic_SR_register_on_exit(LPM0_bits); // Exit LPM0;
}/*------------------------------------------------------------------------------------
*/
/*-------------------------WDT ISR------------------------------------------------------
*/
#pragma vector=WDT_VECTOR
__interrupt void watchdog_timer(void)
{
    sendData();
    UART_send_String(openMSG, sizeof(openMSG));

    if(aDev > 15)
    {
        sprintf(dvAngle, "%d.%04u",round(aDev)); // Prints angle of deviation to a
string;
        P2OUT ^= BIT2;                           // Toggle LED1;
        P3SEL ^= BIT5;                          // Buzzer at P3.5;
        P2OUT &= ~BIT1;
        UART_send_String(dvAngle, sizeof(dvAngle));
        UART_send_String(highdisp, sizeof(highdisp));
    }
    else if(aDev < (-1*15))
    {
        sprintf(dvAngle, "%d.%04u",round(aDev));
        P2OUT ^= BIT1;                          // Toggle LED2;
        P3SEL ^= BIT5;                          // Buzzer at P3.5;
        P2OUT &= ~BIT2;
        UART_send_String(dvAngle, sizeof(dvAngle));
        UART_send_String(lowdisp, sizeof(lowdisp));
    }
    else if(aDev > (-1*15) && aDev < 15)
    {
        //LED1 OFF; LED2 OFF.
        sprintf(dvAngle, "%d.%04u",round(aDev));
        P2OUT |= (BIT1 + BIT2);
        P3SEL &= ~BIT5;                         // Buzzer is OFF;
        UART_send_String(dvAngle, sizeof(dvAngle));
        UART_send_String(steady, sizeof(steady));
    }
    __bic_SR_register_on_exit(LPM0_bits);       // Exit LPM0;
}/*-------------------------------------------------------------------------------------
*/
```