Clinton Bess

Dr. Venkatesan Muthukumar

TivaC Lab 10 – I2C

CPE 403

Checklist for Lab 10

- ☑ *A text/word document of the initial code with comments*
- ☑ *In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also include the comments.*
- ☑ *Provide a permanent link to all main and dependent source code files only (name them as LabXX-TYY, XX-Lab# and YY-task#)Screenshots of debugging process along with pictures of actual circuit*
- ☑ *Video link of demonstration.*

Code for Experiment

Task 1:

Accelerometer:

```
#define ACCEL_W 0x3A     // Addresses for the accelerometer
#define ACCEL_R 0x3B
#define ACCEL_ADDR 0x1D

#ifndef PART_TM4C123GH6PM
#define PART_TM4C123GH6PM
#endif

#include <stdbool.h>
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_i2c.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/i2c.h"

void Accel_int();        // Function prototype to initialize the Accelerometer
signed int Accel_read(); // Function prototype to read the Accelerometer

void main(void) {

        signed short int LED_value = 1;

        SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);
        SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);      // Enable I2C
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);     // Enable GPIO

        GPIOPinConfigure(GPIO_PB3_I2C0SDA);              // Configure GPIO pin for I2C Data line
        GPIOPinConfigure(GPIO_PB2_I2C0SCL);              // Configure GPIO Pin for I2C clock line

        GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_2 | GPIO_PIN_3);  // Set Pin Type

        // Enable Peripheral ports for output
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);             // PORTC
        GPIOPinTypeGPIOOutput(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7);    // LED 1 LED 2

        GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_5);              // LED 4

        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);                     // PORT D
        GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_6);              // LED 3


// Setup the I2C
        GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD);
        GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_OD);
(I2C0_BASE, SysCtlClockGet(), false);    // The False sets the controller to 100kHz communication

        Accel_int();            // Function to initialize the Accelerometer
```

```
while(1){
                LED_value = LED_value + Accel_read();

                if(LED_value <= 1){
                        // Cycle through the LEDs on the Orbit board
                        GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x40); // LED 1 on LED 2 Off
                        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x00);          // LED 3 off
                        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x00);          // LED 4 off
                        LED_value = 1;   // reset value to maintain range
                }

                else if(LED_value == 2){
                        // Cycle through the LEDs on the Orbit board
                        GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x80); // LED 1 off LED 2 on
                        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x00);          // LED 3 off
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x00);                          // LED 4 on
                }
                else if(LED_value == 3){
                        // Cycle through the LEDs on the Orbit board
                        GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x00);// LED 1 off LED 2 off
                        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x40);          // LED 3 on
                        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x00);          // LED 4 0ff
                }
                else if(LED_value >= 4){
                        // Cycle through the LEDs on the Orbit board
                        GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x00);// LED 1 off LED 2 Off
                        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x00);          // LED 3 off
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x20);                          // LED 4 on
                        LED_value = 4;
                }
        }
}

void Accel_int(){

        I2CMasterSlaveAddrSet(I2C0_BASE, ACCEL_ADDR, false);             // false means transmit

        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);     // Send Start condition

        I2CMasterDataPut(I2C0_BASE, 0x2D);                                // Writing to the Accel control
reg
        SysCtlDelay(20000);
                // Delay for first transmission
        I2CMasterDataPut(I2C0_BASE, 0x08);                                // Send Value to control
Register

        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);    // Send Stop condition

        while(I2CMasterBusBusy(I2C0_BASE)){}; // Wait for I2C controller to finish operations

}

signed int Accel_read() {

        signed int data;
        signed short value = 0;
        unsigned char MSB;
        unsigned char LSB;

        I2CMasterSlaveAddrSet(I2C0_BASE, ACCEL_ADDR, false);      // false means transmit
```

```
        I2CMasterDataPut(I2C0_BASE, 0x32);
        SysCtlDelay(20000);
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND); // Request LSB of X Axis
        SysCtlDelay(2000000);


        I2CMasterSlaveAddrSet(I2C0_BASE, ACCEL_ADDR, true);       // false means transmit

        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE); //Request LSB of X Axis
        SysCtlDelay(20000);

        LSB = I2CMasterDataGet(I2C0_BASE);
        SysCtlDelay(20000);

        I2CMasterSlaveAddrSet(I2C0_BASE, ACCEL_ADDR, false);      // false means transmit
        I2CMasterDataPut(I2C0_BASE, 0x33);

        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND); // Request LSB of X Axis
        SysCtlDelay(2000000);

        I2CMasterSlaveAddrSet(I2C0_BASE, ACCEL_ADDR, true);       // false means transmit

        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE); //Request LSB of X Axis
        SysCtlDelay(20000);

        MSB = I2CMasterDataGet(I2C0_BASE);

        value = (MSB << 8 | LSB);

        if(value < -250 ){                                        // testing axis for value
                data = -1;
        }
        else if (value > 250){
                data = 1;
        }

        else{
                data = 0;
        }

        //SysCtlDelay(200);
        //SysCtlDelay(20000);

        return data;     // return value
}
```

Task 1:

Temperature Sensor:

```
#define TEMP_ADDR  0x4F          // Address for Temp Sensor

// Define needed for pin_map.h
#define PART_TM4C123GH6PM

#include <stdbool.h>
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
```

```
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "inc/hw_i2c.h"
#include "driverlib/i2c.h"

unsigned char start_screen[29] = "\n\n\r ATE Lab 8 Temp Sensor \n\n\r";
unsigned char log[18] = "\n\n\r Temp reading: ";

void Print_header();                                    // Prints Header
void Read_temp(unsigned char *data);     // Read Temperature sensor

void main(void) {

        unsigned char temp_data[10] = "00.0 C \n\n\r";          // Temp format to be edited by read
        unsigned short int i = 0;
        // Setup the I2C see lab 7
*****************************************************************************
        SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);  //setup
clock

        SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);             // Enable I2C
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);     // Enable GPIO

        GPIOPinConfigure(GPIO_PB3_I2C0SDA);                             // Configure GPIO pin for I2C
Data line
        GPIOPinConfigure(GPIO_PB2_I2C0SCL);                             // Configure GPIO Pin for I2C
clock line

        GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_2 | GPIO_PIN_3);  // Set Pin Type

        GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD);
        GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_OD);
(I2C0_BASE, SysCtlClockGet(), false); // False sets controller to 100kHz communication
        I2CMasterSlaveAddrSet(I2C0_BASE, TEMP_ADDR, true);              // false means transmit
        //*************************************************************************************************
**********

        // Setup the UART see lab 6
*****************************************************************************

        SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);            // Enable UART hardware
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);            // Enable Pin hardware

        GPIOPinConfigure(GPIO_PA0_U0RX);        // Configure GPIO pin for UART RX line
        GPIOPinConfigure(GPIO_PA1_U0TX);        // Configure GPIO Pin for UART TX line
        GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);      // Set Pins for UART

        UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,       // Configure UART to 8N1 at
115200bps
                        (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));
        //*************************************************************************************************
**********
        Print_header();                 // Print Header

        while(1){

                Read_temp(temp_data);    // Read Data from Temp Sensor
                SysCtlDelay(6000000);    // Delay
```

```
                for(i=0;i<10;i++){       // Loop to print out data string
                        UARTCharPut(UART0_BASE, temp_data[i]);
                }

        }

}

void Print_header(){                    // Print Header at start of program

        int i = 0;

        for(i=0;i<29;i++){       // Print Header at start of program
                UARTCharPut(UART0_BASE, start_screen[i]);
        }
}


void Read_temp(unsigned char *data){     // Read Temperature sensor

        unsigned char temp[2];           //  storage for data

        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_START);          // Start condition
        SysCtlDelay(20000);

        temp[0] = I2CMasterDataGet(I2C0_BASE);   // Read first char
        SysCtlDelay(20000);
                                         // Delay
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_CONT);
        SysCtlDelay(20000);

        temp[1] = I2CMasterDataGet(I2C0_BASE);   // Read second char
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_FINISH); // Stop Condition

        data[0] = (temp[0] / 10) + 0x30;         // convert 10 place to ASCII
        data[1] = (temp[0] - ((temp[0] / 10)*10)) + 0x30; // Convert 1's place to ASCII
        if(temp[1] == 0x80){                              // Test for .5 accuracy
                data[3] = 0x35;
        }
        else{
                data[3] = 0x30;
        }

}
```

Task 2:

```
// Addresses for the accelerometer
#define ACCEL_W 0x3A            // Write
#define ACCEL_R 0x3B            // read
#define ACCEL_X 0x32            // LSB x-axis reg
#define ACCEL_Y 0x34            // LSB y-axis reg
#define ACCEL_Z 0x36            // LSB z-axis reg
#define ACCEL_ADDR 0x1D

// Define needed for pin_map.h
#ifndef PART_TM4C123GH6PM
#define PART_TM4C123GH6PM
#endif
```

```
#include <stdbool.h>
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_i2c.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/i2c.h"
#include "driverlib/uart.h" // UART APIs


void Accel_int();                       // Function prototype to initialize the Accelerometer
signed int Accel_read(unsigned char);   // Function prototype to read the Accelerometer
void Print_header();                    // Print Header at start of program
void print_shortInt(signed short int);  // Print short int
void print_axis_header(char);           // Print label for axis being printed

void main(void) {
        signed short int value;

        SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);  //setup
clock

        SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);      // Enable I2C hardware
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);     // Enable Pin hardware

        GPIOPinConfigure(GPIO_PB3_I2C0SDA);              // Configure GPIO pin for I2C Data line
        GPIOPinConfigure(GPIO_PB2_I2C0SCL);              // Configure GPIO Pin for I2C clock line

        GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_2 | GPIO_PIN_3);        // Set Pin Type

        // Enable UART
        SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);     // Enable UART hardware
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);     // Enable Pin hardware

        GPIOPinConfigure(GPIO_PA0_U0RX);         // Configure GPIO pin for UART RX line
        GPIOPinConfigure(GPIO_PA1_U0TX);         // Configure GPIO Pin for UART TX line
        GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);     // Set Pins for UART

        UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,        // Configure UART to 8N1 at
115200bps
                        (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));

        // Setup the I2C
        GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD);// SDA MUST BE
STD
        GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_OD);// SCL MUST BE
OPEN DRAIN
        I2CMasterInitExpClk(I2C0_BASE, SysCtlClockGet(), false);
        // The False sets the controller to 100kHz communication

        Accel_int();            // Function to initialize the Accelerometer
        Print_header();
        while(1){
                value = Accel_read(ACCEL_X);
                print_axis_header('x');
                print_shortInt(value);
```

```
                value = Accel_read(ACCEL_Y);
                print_axis_header('y');
                print_shortInt(value);

                value = Accel_read(ACCEL_Z);
                print_axis_header('z');
                print_shortInt(value);
        }
}

void print_axis_header(char axis)        // Print header for axis
{
        unsigned char *label = "-axis = ";
        int i = 0;                       // general counter
        UARTCharPut(UART0_BASE, '\r');
        UARTCharPut(UART0_BASE, axis);

        while(label[i] != '\0'){ // Print Header at start of program
                UARTCharPut(UART0_BASE, label[i]);
                i++;
        }
}

void print_shortInt(signed short int value){     // Print Header at start of program
        char buffer[10];
        char sign = '\0';
        int i = 0; // iterator
        int temp = value;

        if (value == 0)
        {
                UARTCharPut(UART0_BASE, '0');
                UARTCharPut(UART0_BASE, '\n');UARTCharPut(UART0_BASE, '\r');
                return;
        }

        if (value < 0)
        {
                sign  = '-';
                value *= -1;
        }

        // Convert to string
        while(temp != 0) // count the number of digits
        {
                i++;
                temp /= 10;
        }
        buffer[i] = '\0';
        i--;
        for( i; i >= 0; i--) // convert digits to chars, and store in buffer
        {
                buffer[i] = value % 10 + '0';
                value /= 10;
        }
        UARTCharPut(UART0_BASE, sign);
        for(i = 0; i < sizeof(buffer); i++)  // Loop to print out data string
        {
                if (buffer[i] == '\0') break;
                UARTCharPut(UART0_BASE, buffer[i]);
        }
```

```
        UARTCharPut(UART0_BASE, '\n');UARTCharPut(UART0_BASE, '\r');

}

void Print_header(){                    // Print Header at start of program
        unsigned char *start_screen     = "\n\n\rLab 10 Accelerometer Sensor Read\n\r";
        int i = 0; // general counter

        while(start_screen[i] != '\0'){  // Print Header at start of program
                UARTCharPut(UART0_BASE, start_screen[i]);
                i++;
        }
}

void Accel_int(){
                // Function to initialize the Accelerometer

        I2CMasterSlaveAddrSet(I2C0_BASE, ACCEL_ADDR, false);            // false means transmit

        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);    // Send Start condition

        I2CMasterDataPut(I2C0_BASE, 0x2D);                               // Writing to the Accel control
reg
        SysCtlDelay(20000);
                // Delay for first transmission
        I2CMasterDataPut(I2C0_BASE, 0x08);                               // Send Value to control
Register

        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);   // Send Stop condition

        while(I2CMasterBusBusy(I2C0_BASE)){};    // Wait for I2C controller to finish operations

}

signed int Accel_read(unsigned char axis_addr) { // Function to read the Accelerometer

        //signed int data;
        signed short value = 0;                 // value of x

        unsigned char MSB;
        unsigned char LSB;

        I2CMasterSlaveAddrSet(I2C0_BASE, ACCEL_ADDR, false);            // false means transmit

        I2CMasterDataPut(I2C0_BASE, axis_addr);
        SysCtlDelay(20000);
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND);        // Request LSB of X Axis
        SysCtlDelay(2000000);                                           // Delay for first transmission

        I2CMasterSlaveAddrSet(I2C0_BASE, ACCEL_ADDR, true);             // false means transmit

        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE);     //Request LSB of X Axis
        SysCtlDelay(20000);

        LSB = I2CMasterDataGet(I2C0_BASE);
        SysCtlDelay(20000);

        I2CMasterSlaveAddrSet(I2C0_BASE, ACCEL_ADDR, false);            // false means transmit
        I2CMasterDataPut(I2C0_BASE, axis_addr + 1);

        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND);        // Request LSB of X Axis
```

```
        SysCtlDelay(2000000);                                           // Delay for first transmission

        I2CMasterSlaveAddrSet(I2C0_BASE, ACCEL_ADDR, true);             // false means transmit

        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE);     //Request LSB of X Axis
        SysCtlDelay(20000);

        MSB = I2CMasterDataGet(I2C0_BASE);

        value = (MSB << 8 | LSB);
        SysCtlDelay(2000);
        return value;
}
```

Task 3:

```
#define TEMP_ADDR  0x4F          // Address for Temp Sensor
// Define needed for pin_map.h
#define PART_TM4C123GH6PM
#include <stdbool.h>
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "inc/hw_i2c.h"
#include "driverlib/i2c.h"

void Print_header();     // Prints Header
float Read_temp();       // Read Temperature sensor

void main(void) {
        float value;

        SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);  //setup
clock

        SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);      // Enable I2C hardware
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);     // Enable Pin hardware

        GPIOPinConfigure(GPIO_PB3_I2C0SDA);              // Configure GPIO pin for I2C Data line
        GPIOPinConfigure(GPIO_PB2_I2C0SCL);              // Configure GPIO Pin for I2C clock line

        GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_2 | GPIO_PIN_3);  // Set Pin Type

        GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD);// SDA MUST BE
STD
        GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_OD);// SCL MUST BE
OPEN DRAIN
        I2CMasterInitExpClk(I2C0_BASE, SysCtlClockGet(), false); // The False sets the controller to
100kHz communication
        I2CMasterSlaveAddrSet(I2C0_BASE, TEMP_ADDR, true);       // false means transmit
        // Set up GPIO output for LEDs
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);             // PORTF
        GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_3);  // red and green LEDs
        while(1){
                value = Read_temp();     // Read Data from Temp Sensor
                // Chose 27 C since cur readings of room temp were 27.
```

```
                if (value > 27) // If temp > room temp, light red.
                        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_3, 2);
                else                    // else, light green.
                        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_3, 8);
                SysCtlDelay(6000000);    // Delay
        }
}
float Read_temp(){      // Read Temperature sensor
        unsigned char temp[2];   //  storage for data
        float value;

        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_START);  // Start condition
        SysCtlDelay(20000);                                               // Delay
        temp[0] = I2CMasterDataGet(I2C0_BASE);                            // Read first char
        SysCtlDelay(20000);                                               // Delay
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_CONT);   // Push second Char
        SysCtlDelay(20000);                                               // Delay
        temp[1] = I2CMasterDataGet(I2C0_BASE);                            // Read second char
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_FINISH); // Stop Condition

        value = temp[0];

        if (temp[1] != 128)
                value += 0.5;
        return value;
}
```

Video Link to Demo

Task 1: https://www.youtube.com/watch?v=0NcLEllnfHY

Task 2: https://www.youtube.com/watch?v=rwwS_c4vFi4

Task 3: https://www.youtube.com/watch?v=Qn-b4FWTSug