First Name	Last Name	Student ID	Overall Grade
Johnny	Nguyen	801119047	
Nathan	Jackson	800915532	

Lab Introduction:

Objective is to familiarize the student to takes input from the terminal to interact with the board using UART and allow communication between two different I2C modules using I2C communication.

Pre-Lab Questions:

- No pre-lab questions…

Developed Code:

```
//*****************************************************************************
// Lab 4: Sensor Protocols
// Johnny Nguyen & Nathan Jackson
// April 28th, 2022
// ECGR 3101
//*****************************************************************************
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_types.h"
#include "inc/hw_ints.h"
#include "inc/hw_i2c.h"
#include "inc/hw_memmap.h"
#include "inc/hw_uart.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/uart.h"
#include "utils/uartstdio.h"
#include "utils/uartstdio.c"
#include "driverlib/i2c.h"
//*****************************************************************************
// The error routine that is called if the driver library encounters an error.
//*****************************************************************************
#ifdef DEBUG
void
__error__(char *pcFilename, uint32_t ui32Line)
{
    while(1);
}
#endif
//*****************************************************************************
// Sensor Protocols
//*****************************************************************************
#define SLAVE_ADDRESS           0x3C
static uint32_t ui32DataRx;

void
```

First Name	Last Name	Student ID	Overall Grade
Johnny	Nguyen	801119047	
Nathan	Jackson	800915532	

```
UART0IntHandler(void)
{
    uint32_t ui32Status;

    // Obtain interrupt status.
    ui32Status = UARTIntStatus(UART0_BASE, true);

    // Clears the interrupt.
    UARTIntClear(UART0_BASE, ui32Status);

    // Loops through each characters typed.
    while(UARTCharsAvail(UART0_BASE))
    {
        UARTCharPutNonBlocking(UART0_BASE, UARTCharGetNonBlocking(UART0_BASE));
    }
};

void
I2C0SlaveIntHandler(void)
{
    // Clear I2C0 interrupt.
    I2CSlaveIntClear(I2C0_BASE);

    // Read data from slave.
    ui32DataRx = I2CSlaveDataGet(I2C0_BASE);

};

void
InitConsole(void)
{
    // Enable UART peripheral.
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    // Configure the pins.
    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    // Set Clock Source of UART for terminal.
    UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200, (UART_CONFIG_WLEN_8 |
UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));
    UARTStdioConfig(0, 115200, SysCtlClockGet());

    // Enable UART interrupt
    IntMasterEnable();
    IntEnable(INT_UART0);
    UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT); //only enable RX and TX
interrupts

    // Blink on Startup.
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, GPIO_PIN_2);
    SysCtlDelay(SysCtlClockGet() / 3);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
```

First Name	Last Name	Student ID	Overall Grade
Johnny	Nguyen	801119047	
Nathan	Jackson	800915532	

```
    UARTprintf("Enter a command: + or -\n");
};

void
I2C_Comm(unsigned char character)
{
    uint32_t ui32DataTx;

    HWREG(I2C0_BASE + I2C_O_MCR) |= 0x01;

    // Enables the slave interrupt.
    IntEnable(INT_I2C0);
    I2CSlaveIntEnableEx(I2C0_BASE, I2C_SLAVE_INT_DATA);

    // Sets the clocks speed and transfer speed of the master.
    I2CMasterInitExpClk(I2C0_BASE, SysCtlClockGet(), false);

    // Enable Slave module and set the address.
    I2CSlaveEnable(I2C0_BASE);
    I2CSlaveInit(I2C0_BASE, SLAVE_ADDRESS);

    I2CMasterSlaveAddrSet(I2C0_BASE, SLAVE_ADDRESS, false);

    // Enable master interrupt.
    IntMasterEnable();

    ui32DataTx = character;

    // Transfer the character data from the slave to master.
    I2CMasterDataPut(I2C0_BASE, ui32DataTx);
    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND);
};


int
main(void) {

    unsigned char data = 0;

    // Setting the clock speed.
    SysCtlClockSet(SYSCTL_SYSDIV_2_5 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN |
SYSCTL_XTAL_16MHZ);

    // Enables the LEDs.
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2);

    // Enable the I2C0 peripheral.
    SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);

    // Configure the Master and Slave pins
    GPIOPinConfigure(GPIO_PB2_I2C0SCL);
    GPIOPinConfigure(GPIO_PB3_I2C0SDA);
    GPIOPinTypeI2CSCL(GPIO_PORTB_BASE, GPIO_PIN_2);
```

First Name	Last Name	Student ID	Overall Grade
Johnny	Nguyen	801119047	
Nathan	Jackson	800915532	

```
    GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_3);

    // Enabling access to terminal.
    InitConsole();

    while (1)
    {
        // Obtains your character input.
        data = UARTCharGet(UART0_BASE);

        // Runs through I2C.
        I2C_Comm(data);

        // Comparison made with the received data from I2C.
        if (ui32DataRx == '+')
        {
            UARTprintf("sending LED on...\n");
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, GPIO_PIN_2);
        }
        else if (ui32DataRx == '-')
        {
            UARTprintf("sending LED off...\n");
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0x0);
        }
        else
        {
            UARTprintf("Invalid Command...\n");
        }
    }
}
```

Lessons Learned:

The student had learned to use I2C and familiarize with the master and slave relationship to communicate with other devices. They have also learned to set up UART and takes in terminal input to interact with the board.

Self-Assessment:

	Objective	Self-Review	TA/Instructor Review
P1	Code is formatted correctly.	15	
P2	Setting up a virtual terminal for communicating with board.	15	
P3	Typing + turns the LED on based on the requirement above,	8	
P4	Typing - turns the LED off based on the requirement above	8	
P5	Any input other that + and - says the command is unrecognized	8	
P6	Configuring Blue LED to blink once and waits for command	10	
P7	Configuring second l2c to be a slave	5	

First Name	Last Name	Student ID	Overall Grade
Johnny	**Nguyen**	**801119047**	
Nathan	**Jackson**	**800915532**	

P8	Having an ISR waiting for a message from l2c to turn the LED on or off	10	
P9	The slave l2c makes an interrupt and the ISR runs	10	

Comment:

The LEDs, UART communication, and using a I2C slave interrupt handler to send and receive data is all functioning properly. Unfortunately, there is a problem with the board taking inputs from the terminal as it would only register every second input that is typed in the terminal. Another problem that occurred was difficulty setting up the board to use two different modules for the I2C modules.