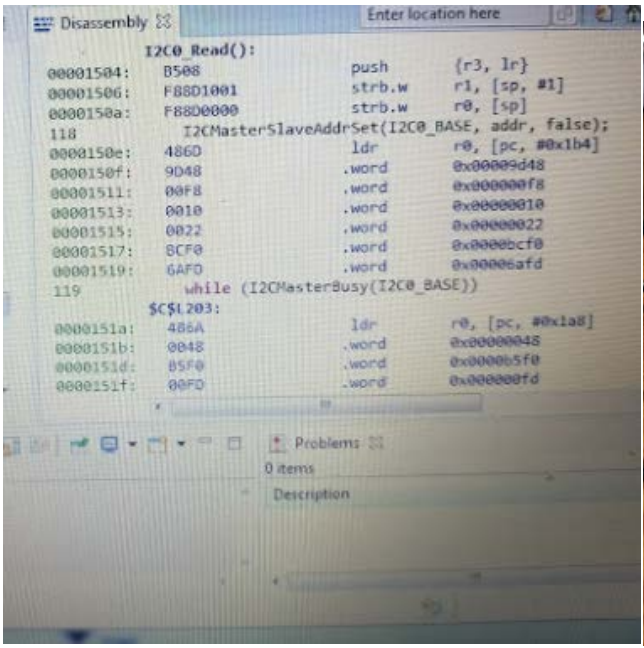
Estellar Raganit
CPE403
Midterm Project

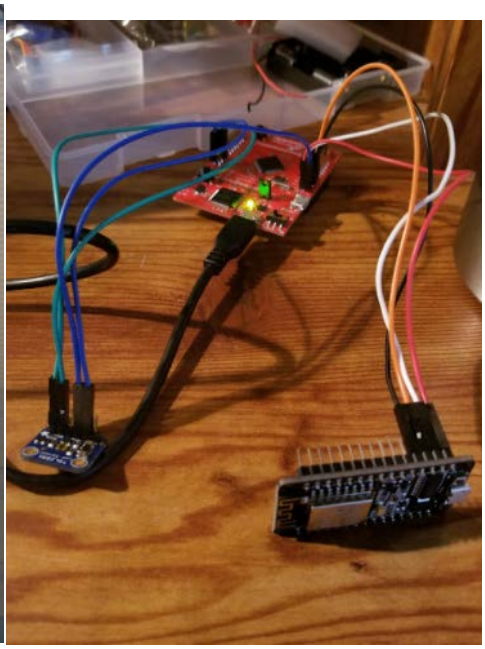
Purpose: We will be using the lux sensor to collect data and upload the data obtained to the the ESP8266 module into IOT cloud called thingspeak. I will also be capturing light data and sending it to the TIVA C. This information will then be sent to a wi-fi module and will be shown on thingspeak.

Implementation: First we will need to configure uart. Than with Tiva C we will need to integrate 12C which included modules, simultaneous master and slave operation and so on. I2C will have the ability to communicate to the light sensor to the SCL and SDA.

Video/Picture:

https://youtu.be/vT4-vVetmCI





CONNECTIONS:
TIVAC
Bp2 to scl
Pb3 to sda
Tiva tx to esprx
Esp tx to ftdi rx /

CODE:

```
uint32_t ui32SysClock;
const uint8_t TSL2591address = 0x29;
const bool DebuggingMode = true;

#define TSL2591_VISIBLE           (2)       // channel 0 - channel 1
#define TSL2591_INFRARED          (1)       // channel 1
#define TSL2591_FULLSPECTRUM      (0)       // channel 0

#define TSL2591_ADDR              (0x29)
#define TSL2591_READBIT           (0x01)

#define TSL2591_COMMAND_BIT       (0xA0)    // 1010 0000: bits 7 and 5 for 'command normal'
#define TSL2591_CLEAR_INT         (0xE7)
#define TSL2591_TEST_INT          (0xE4)
#define TSL2591_WORD_BIT          (0x20)    // 1 = read/write word (rather than byte)
#define TSL2591_BLOCK_BIT         (0x10)    // 1 = using block read/write

#define TSL2591_ENABLE_POWEROFF   (0x00)
#define TSL2591_ENABLE_POWERON    (0x01)
#define TSL2591_ENABLE_AEN        (0x02)    // ALS Enable. This field activates ALS function.
Writing a one activates the ALS. Writing a zero disables the ALS.
#define TSL2591_ENABLE_AIEN       (0x10)    // ALS Interrupt Enable. When asserted permits ALS
interrupts to be generated, subject to the persist filter.
#define TSL2591_ENABLE_NPIEN      (0x80)    // No Persist Interrupt Enable. When asserted NP
Threshold conditions will generate an interrupt, bypassing the persist filter

#define TSL2591_LUX_DF            (408.0F)
#define TSL2591_LUX_COEFB         (1.64F)  // CH0 coefficient
#define TSL2591_LUX_COEFC         (0.59F)  // CH1 coefficient A
#define TSL2591_LUX_COEFD         (0.86F)  // CH2 coefficient B

#define TSL2591_ENABLE 0x00
#define TSL2591_CONFIG 0x01

#define TSL2591_ID 0x12
#define TSL2591_REGISTER_DEVICE_STATUS 0x13
#define TSL2591_C0DATAL 0x14
#define TSL2591_C0DATAH 0x15
#define TSL2591_C1DATAL 0x16
#define TSL2591_C1DATAH 0x17

#define TSL2591_INTEGRATIONTIME_100MS 0x00
#define TSL2591_GAIN_MED 0x10  // medium gain (25x)
```

```c
#include <stdarg.h>
#include <stdbool.h> //boolean definition for the C99 standard
#include <stdint.h>  //variable definitions for the C99 standard
#include "inc/tm4c123gh6pm.h" //macros for the lunchpad
#include "inc/hw_i2c.h"      //i2c macros
#include "inc/hw_memmap.h" //macros defining the memory map of the TivaC series device
#include "inc/hw_types.h" //defines common types and definitions
#include "inc/hw_gpio.h"  //gpio macros
#include "driverlib/i2c.h"  //defines i2c
#include "driverlib/sysctl.h" //defines and macros for system control API of DriverLib
#include "driverlib/gpio.h"   //defines gpio
#include "utils/uartstdio.h"  //defines uart
#include "driverlib/pin_map.h"  //for pin map
#include "driverlib/uart.h"     //defines uart
#include "driverlib/interrupt.h" //defines interrrupts
#include "driverlib/hibernate.h" //defines hibernate
#include "TSL2591_def.h"     //defines sensor
#include "ustdlib.h"         //defines libraries
#include "driverlib/uart.h"   //defines uart
#include "inc/hw_uart.h"

void ConfigureUART(void)
//Configures the UART to run at 19200 baud rate
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART1);    //enables UART module 1
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);    //enables GPIO port b

    GPIOPinConfigure(GPIO_PB1_U1TX);    //configures PB1 as TX pin
    GPIOPinConfigure(GPIO_PB0_U1RX);    //configures PB0 as RX pin
    GPIOPinTypeUART(GPIO_PORTB_BASE, GPIO_PIN_0 | GPIO_PIN_1);  //sets the UART pin type

    UARTClockSourceSet(UART1_BASE, UART_CLOCK_PIOSC);   //sets the clock source
    //enables UARTstdio baud rate, clock, and which UART to use
    UARTStdioConfig(1, 115200, 16000000);

}
```

```c
void I2C0_Write(uint8_t addr, uint8_t N, ...) {
    // Writes data from master to slave
    // Takes the address of the device, the number of arguments, and a variable
    // amount of register addresses to write to
    // Find the device based on the address given
    I2CMasterSlaveAddrSet(I2C0_BASE, addr, false);
    while (I2CMasterBusy(I2C0_BASE));
    va_list vargs; // variable list to hold the register addresses passed
    va_start(vargs,N); // initialize the variable list with the number of arguments
    // put the first argument in the list in to the I2C bus
    I2CMasterDataPut(I2C0_BASE, va_arg(vargs, uint8_t));
    while (I2CMasterBusy(I2C0_BASE));
    if (N == 1) // if only 1 argument is passed, send that register command then stop
    {
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND);
        while (I2CMasterBusy(I2C0_BASE));
        va_end(vargs);
    } else
    {
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);
        while (I2CMasterBusy(I2C0_BASE));
        uint8_t i;
        for (i = 1; i < N - 1; i++) {
            // send the next register address to the bus
            I2CMasterDataPut(I2C0_BASE, va_arg(vargs, uint8_t));
            while (I2CMasterBusy(I2C0_BASE));
            // burst send, keeps receiving until the stop signal is received
            I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_CONT);
            while (I2CMasterBusy(I2C0_BASE)) ;
        }
        I2CMasterDataPut(I2C0_BASE, va_arg(vargs, uint8_t));
        while (I2CMasterBusy(I2C0_BASE));
        // send the finish signal to stop transmission
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);
        while (I2CMasterBusy(I2C0_BASE));
        va_end(vargs);
    }
}
```

```
uint32_t I2C0_Read (uint8_t addr, uint8_t reg)
//Read data from slave to master
//Takes in the address of the device and the register to read from
{
//find the device based on the address given
    I2CMasterSlaveAddrSet (I2C0 BASE, addr, false);
    while (I2CMasterBusy (I2C0_BASE));
//send the register to be read on to the I2C bus
    I2CMasterDataPut (I2C0_BASE, reg);
    while (I2CMasterBusy (I2C0_BASE));
//send the signal to send the register value
    I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND);
    while (I2CMasterBusy (I2C0_BASE));
//set the master to read from the device
    I2CMasterSlaveAddrSet (I2C0 BASE, addr, true);
    while (I2CMasterBusy (I2C0_BASE));
//send the receive signal to the device
    I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE);
    while (I2CMasterBusy (I2C0_BASE));
//return the data read from the bus
    return I2CMasterDataGet (I2C0_BASE);
}

void TSL2591_init ()
//Initializes the TSL2591 to have a medium gain,
{
    uint32_t x;
    x = I2C0_Read (TSL2591_ADDR, (TSL2591_COMMAND_BIT | TSL2591_ID));//read the device ID
    if (x == 0x50)
    {
    //used during debuging to make sure correct ID is received
     UARTprintf ("----------------->    Found Device! %i\n\r", x);

     SysCtlDelay(5000000);
    }



    else
    {
     UARTprintf ( "Device not Found! %i\n", x);
      while (1){

      };      //loop here if the dev ID is not correct
    }
//configures the TSL2591 to have medium gain adn integration time of 100ms
    I2C0_Write (TSL2591_ADDR, 2, (TSL2591_COMMAND_BIT | TSL2591_CONFIG), 0x10);
    //enables proper interrupts and power to work with TSL2591

    I2C0_Write (TSL2591_ADDR, 2, (TSL2591_COMMAND_BIT | TSL2591_ENABLE),
    (TSL2591_ENABLE_POWERON | TSL2591_ENABLE_AEN | TSL2591_ENABLE_AIEN |
                               TSL2591_ENABLE_NPIEN));
}

uint32_t GetLuminosity ()
//This function will read the channels of the TSL and returns the calculated value to the caller
{
    float atime = 100.0f, again = 25.0f;    //the variables to be used to calculate proper lux
value
    uint16_t ch0, ch1;  //variable to hold the channels of the TSL2591
    uint32_t cp1, lux1, lux2, lux;
    uint32_t x = 1;

    x = I2C0 Read (TSL2591 ADDR, (TSL2591 COMMAND BIT | TSL2591 C0DATAH));   //read register
channel 0 higher bits
    x <<= 16;  //shift 16 bits
    x |= I2C0_Read (TSL2591_ADDR, (TSL2591_COMMAND_BIT | TSL2591_C0DATAL));  //read register
channel 0 lower bits

    ch1 = x>>16;  // for channel 1
    ch0 = x & 0xFFFF;  //for channel 0
    cp1 =  (uint32_t) (atime * again) / TSL2591_LUX_DF;
    lux1 = (uint32_t) ((float) ch0 - (TSL2591_LUX_COEFB * (float) ch1)) / cp1;
    lux2 = (uint32_t) ((TSL2591_LUX_COEFC * (float) ch0) - (TSL2591_LUX_COEFD * (float) ch1)) /
cp1;
    lux = (lux1 > lux2) ? lux1: lux2;

    return lux;
}

void WIFI_Init ()
{
//    UARTprintf("AT+CWJAP=\"025C4E\",\"4CE15BWV00B1D\"\n"); //to connect to wifi
//    SysCtlDelay(10000000);
    UARTprintf("AT+CWMODE=3\n\r"); //set to BOTh -station mode STA and AP
    SysCtlDelay(100000);
}
```

```
HibernateGPIORetentionEnable ();
//Set RTC hibernation
HibernateRTCSet (0);
//enable RTC hibernation
HibernateRTCEnable ();
//hibernate for 30 minutes
HibernateRTCMatchSet (0, 1800);
//allow hibernation wake up from RTC time or button 2
HibernateWakeSet (HIBERNATE_WAKE_PIN | HIBERNATE_WAKE_RTC);

UARTprintf("AT+CIPMUX=0\n\r"); //creates single connection, if 1 is for multiple connections
SysCtlDelay(20000000);
  UARTprintf("AT+CIOBAUD?\n\r");  //get baud rates
  SysCtlDelay(3000);
using AT+CIPSTART="TCP","184.106.153.149",80 to connect to thingspeak server
UARTprintf("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\n\r" );
SysCtlDelay(50000000);
UARTprintf("AT+CIPSTATUS\n\r");  ///get TCP/IP connection status
SysCtlDelay(5000000);
for (i = 0; i < 20; i++)
//finds the average of the lux channel to send through uart
{
    lux = GetLuminosity ();
    luxAvg += lux;
}
luxAvg = luxAvg/20;
```

```
void main (void)
{
    //set the main clock to runat 40MHz
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
    uint32_t lux = 0, i;
    uint32_t luxAvg = 0;

    char HTTP_POST[60];
    ConfigureUART ();   //configure the UART of Tiva C
    UARTprintf("AT+RST\n\r");  //restart esp
    SysCtlDelay(50000000);
    I2C0_Init ();       //initialize the I2C0 of Tiva C
    TSL2591_init ();    //initialize the TSL2591
    WIFI Init ();       // //initialize the wifi

    //enable button SW 2 to be used during hibernation
    SysCtlPeripheralEnable (SYSCTL_PERIPH_HIBERNATE);
    //Get the system clock to set to the hibernation clock
    HibernateEnableExpClk (SysCtlClockGet());
    //Retain the pin function during hibernation
```

```
SysCtlDelay(10*SysCtlClockGet());
for (i = 0; i < 20; i++)
    // finds the average of the lux channel to send through uart
{
    lux = GetLuminosity();
    luxAvg += lux;
}
luxAvg = luxAvg / 20;
// UARTprintf("\nLux Value: %d\r\n", luxAvg);

//SEND TO THINGSPEAK
UARTprintf("sk=net.createConnection(net.TCP, 0)\r\n");
SysCtlDelay(2000000);
UARTprintf("sk:on(\"receive\", function(sck, c) print(c) end )\n\r");
SysCtlDelay(2000000);
UARTprintf("sk:connect(80,\"api.thingspeak.com\")\n\r");
SysCtlDelay(5000000);
UARTprintf("sk:send(\"GET /update?api_key=JK5PXB1P8JDMDO4Z&field1=%i HTTP/1.1\\r\\nHost: api.thingspeak.com\\r\\nConnection: keep-alive\\r\\nAccept:
// OLD AT COMMAND: GET /update?key=52D8QM61WCD47BDK&field1=9.99&headers=false HTTP/1.1{CrLf}Host: api.thingspeak.com{CrLf}Connection: close{CrLf}Acc
SysCtlDelay(2000000);
```

Schematic:

For the schematic I used what was given to us in the lecture slides.