TITLE: Lux Sensor (TSL2591) Data Collector

GOAL

- Build a schematic involving the TIVAC TM4C123G launchpad, TSL2591 lux sensor, and ESP8266 wifi module
- Read data from the lux sensor at intervals of 15 - 60 seconds over the course of an hour to 24 hours
- Upload the data to the cloud, thingspeak was used for this project, using the ESP2866 wifi module
- Finally use the UART interface for the ESP2866 module and the I2C interface for the TSL sensor

DELIVERABLES:

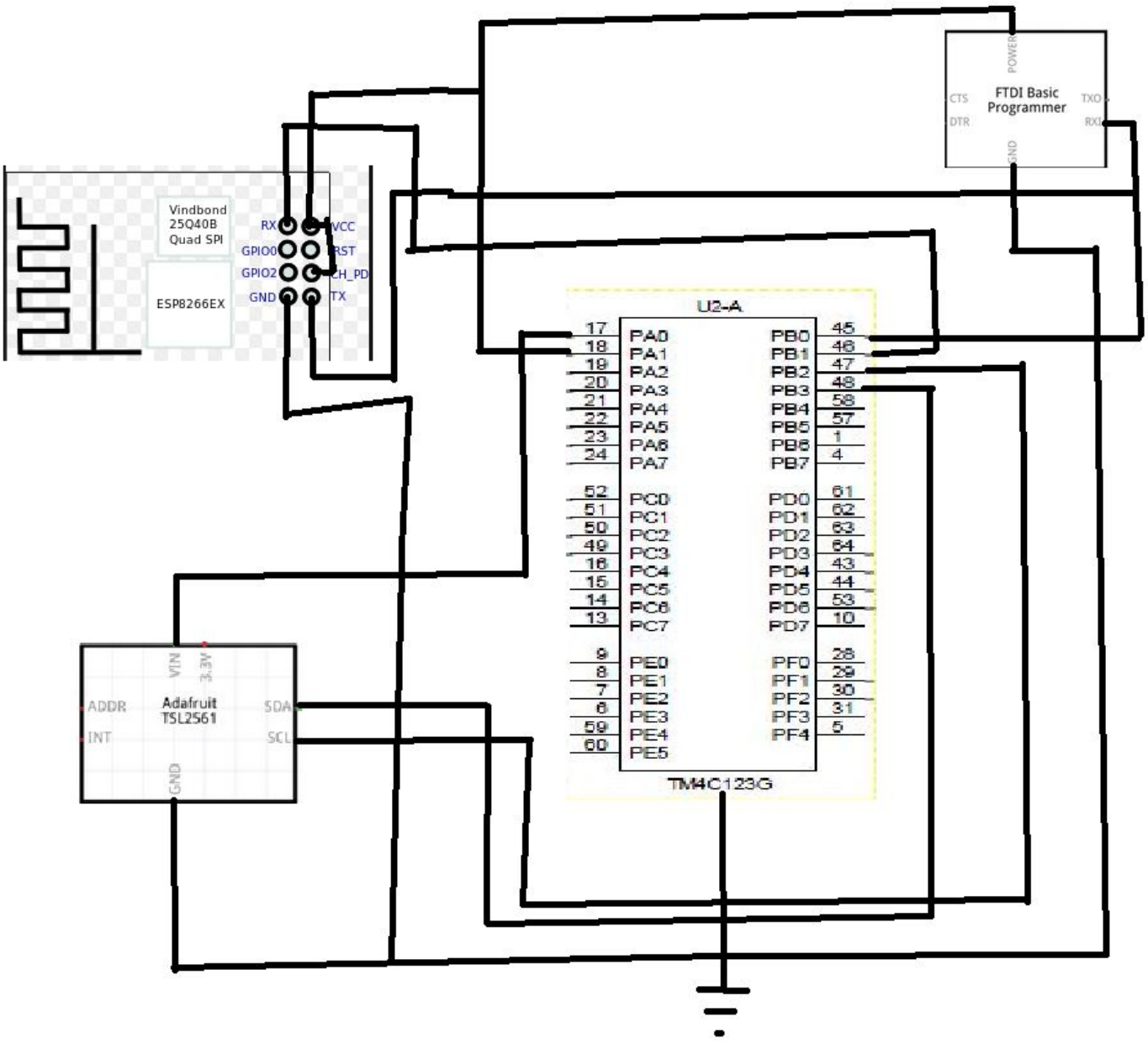
The intended project deliverable was to send lux data to a thingspeak server where it could be interpreted. We did this by using the ESP2866 wifi module to connect to the thingspeak server. The data was also sent over the UART interface where could be read right away and was used for debugging purposes.

COMPONENTS:

The main components of this project were the TIVAC TM4C123G launchpad which was connected to the TSL2591 lux sensor and the ESP2866 wifi module. The lux sensor utilized the I2C interface while the ESP2866 utilized the UART interface. The FTDI was used for debugging purposes. The mini IoT project sent the data to the cloud to a thingspeak server where a graph was formed. In the beginning of the program, I have a function that configures and initialization the UART interface, which enables UART module 1, enables GPIO port b, configures PB1 for TX and PB0 for RX, finally it will set the UART pin type, clock source, and enable the baud rate that UART uses. After that the program will initialize the I2C interface. Here I enable the I2C0 and PORTB, set the I2C PB3 as SDA and PB2 as SDA. I set the clock of the I2C which ensures a proper connection, finally there is a while loop to wait while the master SDA is busy. Last component to be initialized is the TSL2591 sensor, here it reads the device's ID and configures for medium gain and integration time of 100 ms.

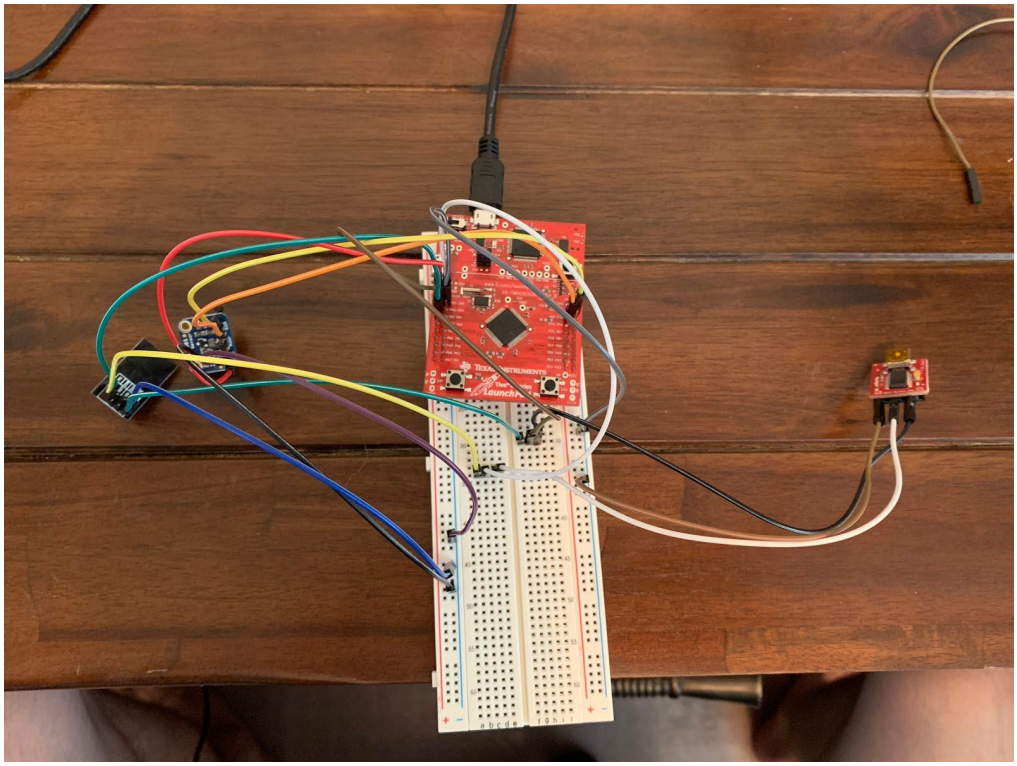
Video Link: https://youtu.be/CryLPSix9Gg

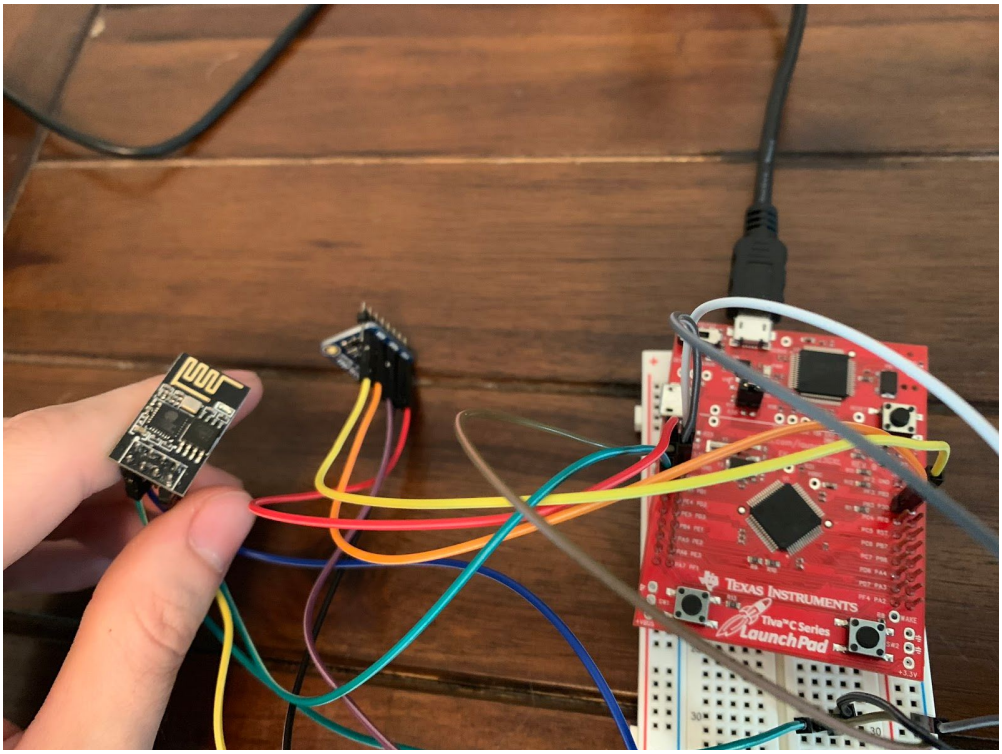
SCHEMATICS:



Video Link: https://youtu.be/CryLPSix9Gg

SCREENSHOTS:





Video Link: https://youtu.be/CryLPSix9Gg

```
AT+RST

OK

WIFI CONNECTED

WIFI GOT IF

AT+CIPMUX=1

OK

AT+CIPSTART=4,"TCP","184.106.153.149",80

OK

AT+CIPSEND=4,115

> GET /update?key=VKJVWJLUNBFOK6CG&field1=6&headers=falseHTTP/1.1Hostapi.thingspeak.comC

busy s...

+IPD,4,2:345
CLOSED
```

IMPLEMENTATION:

1. Configure UART for TIVAC, initialize I2C0 and the TSL2591
2. Enable button 2 for hibernation, setup hibernate clock, enable retention during hibernation, set and enable the RTC
3. Hibernation for 30 mins, wake up if button 2 pressed.
4. Get luminosity for 20 cycles then calculate the average
5. Display the average to the UART interface
6. Reset the ESP2866, enable multiple sends
7. Establish a connection with the thingspeak server
8. Send the data to the server through UART
9. Allow the ESP2866 to send the information to the given HTTP_POST
10. Hibernate

Video Link: https://youtu.be/CryLPSix9Gg

CODE:

```
#include <stdarg.h>
#include <stdbool.h>
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_i2c.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "driverlib/i2c.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/uart.h"
#include "utils/uartstdio.h"
#include "driverlib/interrupt.h"
#include "driverlib/hibernate.h"
#include "TSL2591_def.h"
#include "utils/ustdlib.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
    UARTStdioConfig(1, 19200, 16000000);    //enables UARTstdio baud rate,
clock, and which UART to use
}


void I2C0_Init ()
//Configure/initialize the I2C0
{
```

Video Link: https://youtu.be/CryLPSix9Gg

```
    SysCtlPeripheralEnable (SYSCTL_PERIPH_I2C0);    //enables I2C0
    SysCtlPeripheralEnable (SYSCTL_PERIPH_GPIOB);    //enable PORTB as
peripheral
    GPIOPinTypeI2C (GPIO_PORTB_BASE, GPIO_PIN_3);    //set I2C PB3 as SDA
    GPIOPinConfigure (GPIO_PB3_I2C0SDA);

    GPIOPinTypeI2CSCL (GPIO_PORTB_BASE, GPIO_PIN_2);    //set I2C PB2 as
SCLK
    GPIOPinConfigure (GPIO_PB2_I2C0SCL);

    I2CMasterInitExpClk (I2C0_BASE, SysCtlClockGet(), false);    //Set the
clock of the I2C to ensure proper connection
    while (I2CMasterBusy (I2C0_BASE));    //wait while the master SDA is
busy
}

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
{
    I2CMasterSlaveAddrSet (I2C0_BASE, addr, false);    //Find the device
based on the address given
    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

    I2CMasterDataPut (I2C0_BASE, va_arg(vargs, uint8_t));    //put the
first argument in the list in to the I2C bus
    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
    //if more than 1, loop through all the commands until they are all sent
```

Video Link: https://youtu.be/CryLPSix9Gg

```
    {
       I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);
       while (I2CMasterBusy (I2C0_BASE));
       uint8_t i;
       for (i = 1; i < N - 1; i++)
       {
             I2CMasterDataPut (I2C0_BASE, va_arg(vargs, uint8_t));
//send the next register address to the bus
             while (I2CMasterBusy (I2C0_BASE));

             I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_CONT);
//burst send, keeps receiving until the stop signal is received
             while (I2CMasterBusy (I2C0_BASE));
       }

       I2CMasterDataPut (I2C0_BASE, va_arg(vargs, uint8_t));    //puts the
last argument on the SDA bus
       while (I2CMasterBusy (I2C0_BASE));

       I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);
//send the finish signal to stop transmission
       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
{
    I2CMasterSlaveAddrSet (I2C0_BASE, addr, false);    //find the device
based on the address given
    while (I2CMasterBusy (I2C0_BASE));

    I2CMasterDataPut (I2C0_BASE, reg);    //send the register to be read on
to the I2C bus
    while (I2CMasterBusy (I2C0_BASE));

    I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND);    //send the
send signal to send the register value
```

Video Link: https://youtu.be/CryLPSix9Gg

```
    while (I2CMasterBusy (I2C0_BASE));

    I2CMasterSlaveAddrSet (I2C0_BASE, addr, true);    //set the master to
read from the device
    while (I2CMasterBusy (I2C0_BASE));

    I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE);    //send
the receive signal to the device
    while (I2CMasterBusy (I2C0_BASE));

    return I2CMasterDataGet (I2C0_BASE);    //return the data read from the
bus
}

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)
    {
       //UARTprintf ("GOT IT! %i\n", x);    //used during debuging to make
sure correct ID is received
    }
    else
    {
       while (1){};      //loop here if the dev ID is not correct
    }

    I2C0_Write (TSL2591_ADDR, 2, (TSL2591_COMMAND_BIT | TSL2591_CONFIG),
0x10);    //configures the TSL2591 to have medium gain adn integration time
of 100ms
    I2C0_Write (TSL2591_ADDR, 2, (TSL2591_COMMAND_BIT | TSL2591_ENABLE),
(TSL2591_ENABLE_POWERON | TSL2591_ENABLE_AEN | TSL2591_ENABLE_AIEN |
TSL2591_ENABLE_NPIEN));    //enables proper interrupts and power to work
with TSL2591
}

uint32_t GetLuminosity ()
//This function will read the channels of the TSL and returns the
calculated value to the caller
```

Video Link: https://youtu.be/CryLPSix9Gg

```
{
    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));
    x <<= 16;
    x |= I2C0_Read (TSL2591_ADDR, (TSL2591_COMMAND_BIT | TSL2591_C0DATAL));

    ch1 = x>>16;
    ch0 = x & 0xFFFF;

    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 main (void)
{
    char HTTP_POST[300];    //string buffer to hold the HTTP command

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

    ConfigureUART ();    //configure the UART of Tiva C
    I2C0_Init ();        //initialize the I2C0 of Tiva C
    TSL2591_init ();    //initialize the TSL2591

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

Video Link: https://youtu.be/CryLPSix9Gg

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

    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 ("AT+RST\r\n");    //reset the esp8266 before pushing data
    SysCtlDelay (100000000);
    UARTprintf ("AT+CIPMUX=1\r\n");    //enable multiple send ability
    SysCtlDelay (20000000);
    UARTprintf ("AT+CIPSTART=4,\"TCP\",\"184.106.153.149\",80\r\n");
//Establish a connection with the thingspeak servers
    SysCtlDelay (50000000);

    //The following lines of code puts the TEXT with the data from the lux
in to a string to be sent through UART
    usprintf (HTTP_POST, "GET
/update?key=VKJVWJLUNBFOK6CG&field1=%d&headers=falseHTTP/1.1\nHostapi.thing
speak.com\nConnection:close\Accept*\*\r\n\r\n", luxAvg);
    UARTprintf ("AT+CIPSEND=4,%d\r\n", strlen(HTTP_POST));    //command the
ESP8266 to allow sending of information
    SysCtlDelay (50000000);
    UARTprintf (HTTP_POST);    //send the string of the HTTP GET to the
ESP8266
    SysCtlDelay (50000000);

    HibernateRequest ();    //Hibernate
    while (1)
    {};
}
```

Video Link: https://youtu.be/CryLPSix9Gg

CONCLUSION:

In conclusion the coding aspect of the project wasn't too hard as it was mainly provided to us. Using the I2C interface with the lux sensor was relatively easy to follow. The hardest part was having all the components talk to each other in one big IoT design. I had a lot of hiccups with trying to get my data to my thingspeak server but I also had a lot trouble when I had to do that functionality in CPE 300 also. All together I learned how to read data and transfer it to a server.

Video Link: https://youtu.be/CryLPSix9Gg