Argenis Jimenez Aguirre

CPE 403 FALL 2018

TIVAC MIDTERM

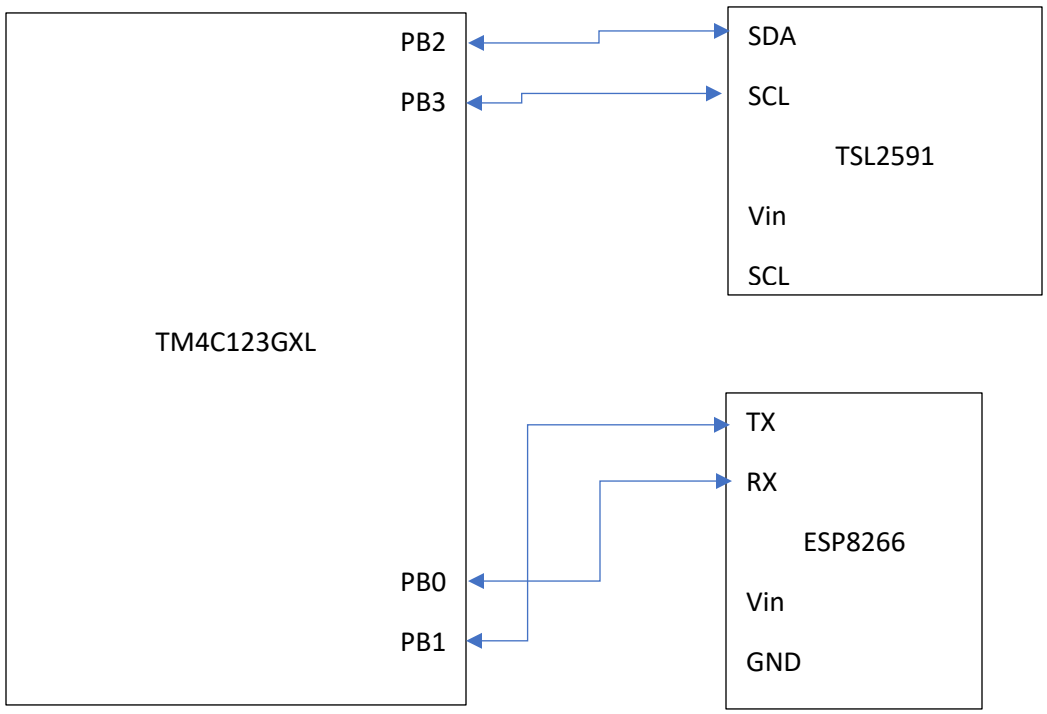
1) Goal

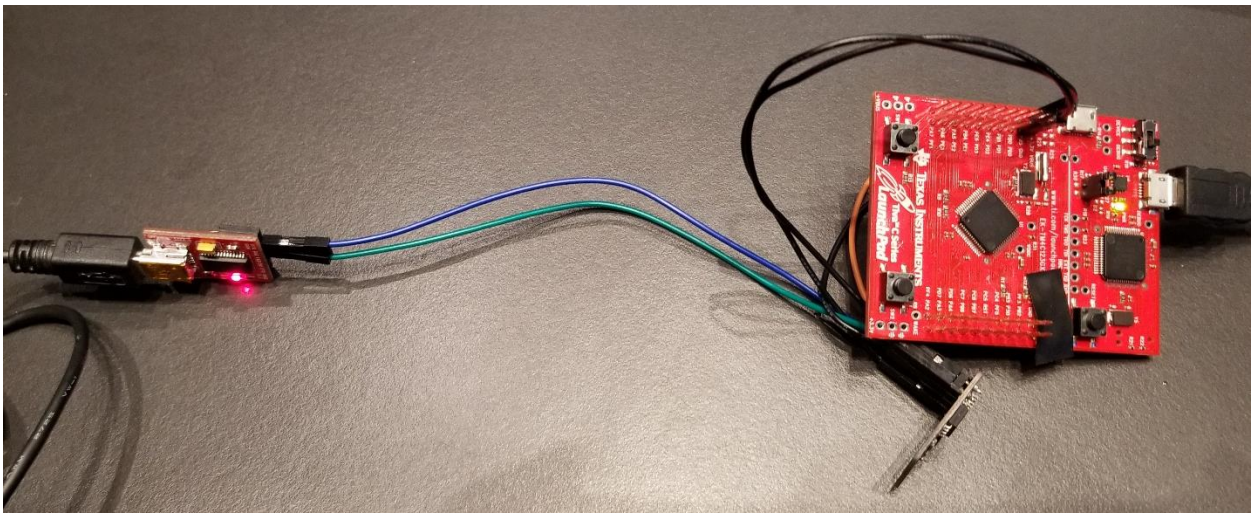
The TIVAC Midterm project is conformed of the supplied TSL2561 lux sensor, ESP-01S, TM4C123GXL and FTDI Basic. This project requires the use of the I2C interface for the TSL sensor and UART interface for the ESP8266 WiFi module. The goal for this project will be to be able to display the data collected by the lux sensor.

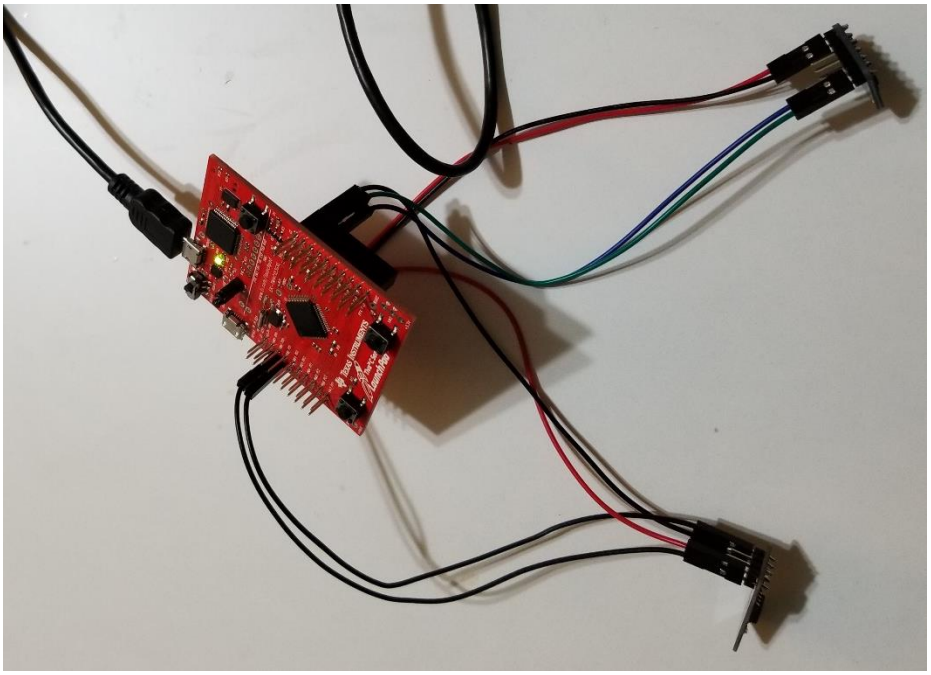
2) Detailed Implementation

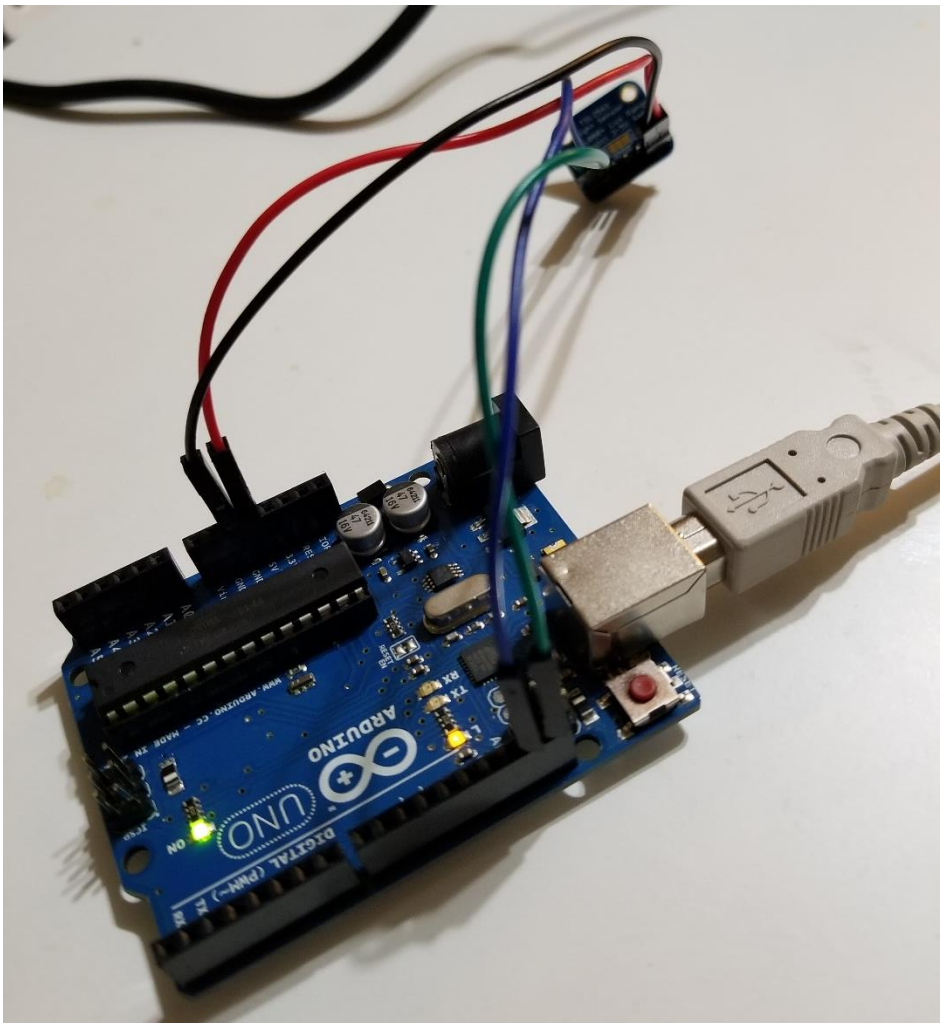
The TM4C123GXL Launchpad is used to transmit data to the ESP module from the TSL2961 sensor. The use of the I2C helps with the communication of launchpad and the ESP module. Ports PB0 and PB1 function as Rx and Tx to connect to their respective Rx and Tx on the ESP module. Ports PB2 and PB3 are connected to the SCL and SDA ports of the lux sensor. In order to kickstart the project the ESP module was flashed. Then, modifications were made to the main code and the TSL2591 header file to adjust for the given TSL2561 lux sensor. Lastly, the use of thingSpeak allows for the display of the given data by the TSL2561 sensor. This is possible with AT commands that connect to the wifi and are able to send data to the ESP module.

3) Schematics









4) Video links

https://www.youtube.com/watch?v=pNSygQrSUX4

5) Screenshots

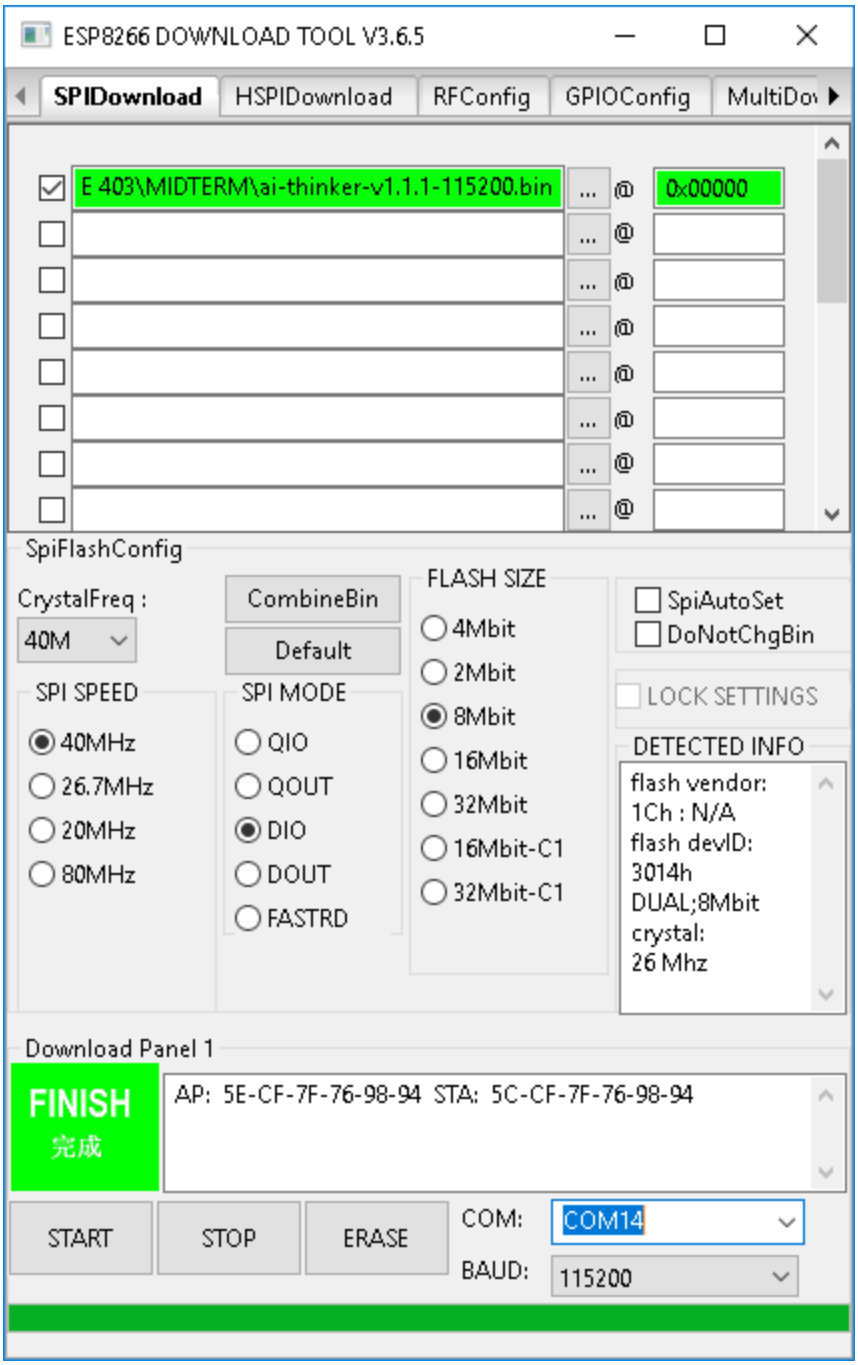
```c
#include <stdarg.h>
#include <stdbool.h>
#include <stdint.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_i2c.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "driverlib/i2c.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "utils/ustdlib.h"
#include "driverlib/uart.h"
#include "utils/uartstdio.h"
#include "driverlib/interrupt.h"
#include "driverlib/hibernate.h"
#include "TSL2591_def.h"

void ConfigureUART(void)
//Configures the UART to run at 115200 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);
}


void I2C0_Init ()
//Configure/initialize the I2C0
{
    SysCtlPeripheralEnable (SYSCTL_PERIPH_I2C0);    //enables I2C0
```

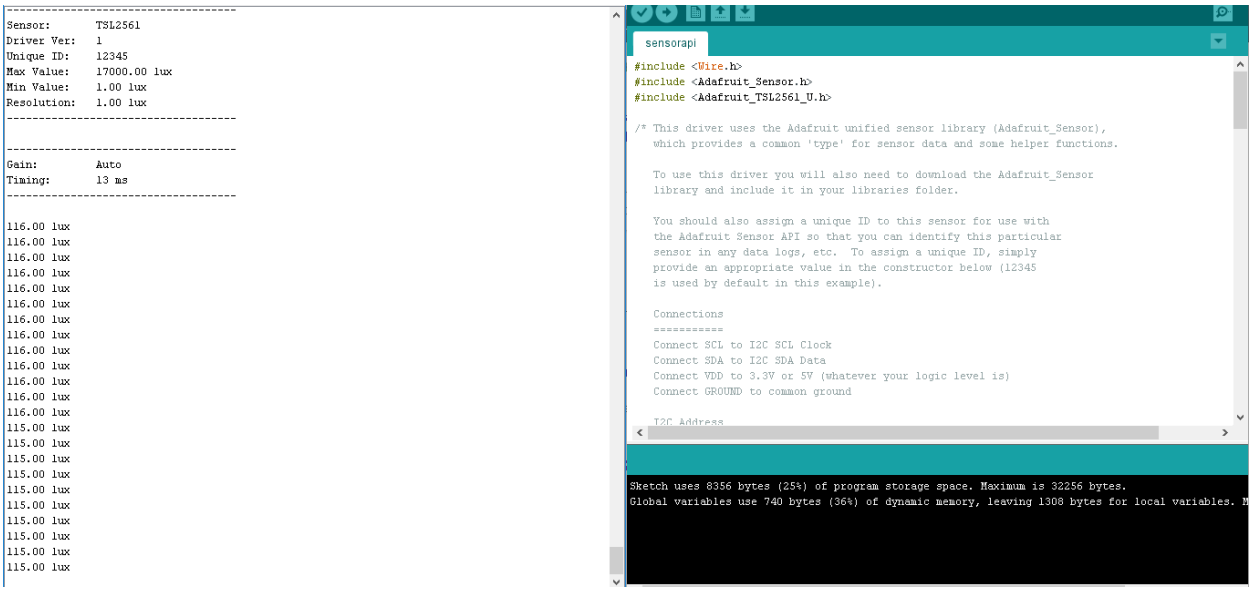
```
40      SysCtlPeripheralEnable (SYSCTL_PERIPH_GPIOB);   //enable PORTB as peripheral
41      GPIOPinTypeI2C (GPIO_PORTB_BASE, GPIO_PIN_3);   //set I2C PB3 as SDA
42      GPIOPinConfigure (GPIO_PB3_I2C0SDA);
43
44      GPIOPinTypeI2CSCL (GPIO_PORTB_BASE, GPIO_PIN_2);    //set I2C PB2 as SCLK
45      GPIOPinConfigure (GPIO_PB2_I2C0SCL);
46      //Set the clock of the I2C to ensure proper connection
47      I2CMasterInitExpClk (I2C0_BASE, SysCtlClockGet(), false);
48      while (I2CMasterBusy (I2C0_BASE));  //wait while the master SDA is busy
49  }
50
51  void I2C0_Write (uint8_t addr, uint8_t N, ...)
52  //Writes data from master to slave
53  //Takes the address of the device, the number of arguments, and a variable amount of register addresses to write t
54  {
55  //Find the device based on the address given
56      I2CMasterSlaveAddrSet (I2C0_BASE, addr, false);
57      while (I2CMasterBusy (I2C0_BASE));
58
59      va_list vargs;  //variable list to hold the register addresses passed
60
61      va_start (vargs, N); //initialize the variable list with the number of arguments
62  //put the first argument in the list in to the I2C bus
63      I2CMasterDataPut (I2C0_BASE, va_arg(vargs, uint8_t));
64      while (I2CMasterBusy (I2C0_BASE));
65      if (N == 1) //if only 1 argument is passed, send that register command then stop
66      {
67          I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND);
68          while (I2CMasterBusy (I2C0_BASE));
69          va_end (vargs);
70      }
71      else
72      //if more than 1, loop through all the commands until they are all sent
73      {
74          I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);
75          while (I2CMasterBusy (I2C0_BASE));
76          uint8_t i;
77          for (i = 1; i < N - 1; i++)
78          {
```

```
79          //send the next register address to the bus
80              I2CMasterDataPut (I2C0_BASE, va_arg(vargs, uint8_t));
81              while (I2CMasterBusy (I2C0_BASE));
82          //burst send, keeps receiving until the stop signal is received
83              I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_CONT);
84              while (I2CMasterBusy (I2C0_BASE));
85          }
86          //puts the last argument on the SDA bus
87          I2CMasterDataPut (I2C0_BASE, va_arg(vargs, uint8_t));
88          while (I2CMasterBusy (I2C0_BASE));
89          //send the finish signal to stop transmission
90          I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);
91          while (I2CMasterBusy (I2C0_BASE));
92
93          va_end (vargs);
94      }
95
96  }
97
98  uint32_t I2C0_Read (uint8_t addr, uint8_t reg)
99  //Read data from slave to master
100 //Takes in the address of the device and the register to read from
101 {
102 //find the device based on the address given
103     I2CMasterSlaveAddrSet (I2C0_BASE, addr, false);
104     while (I2CMasterBusy (I2C0_BASE));
105 //send the register to be read on to the I2C bus
106     I2CMasterDataPut (I2C0_BASE, reg);
107     while (I2CMasterBusy (I2C0_BASE));
108 //send the send signal to send the register value
109     I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND);
110     while (I2CMasterBusy (I2C0_BASE));
111 //set the master to read from the device
112     I2CMasterSlaveAddrSet (I2C0_BASE, addr, true);
113     while (I2CMasterBusy (I2C0_BASE));
114 //send the receive signal to the device
115     I2CMasterControl (I2C0_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE);
116     while (I2CMasterBusy (I2C0_BASE));
117 //return the data read from the bus
```

```
156     ch1 = x>>16;
157     ch0 = x & 0xFFFF;
158
159     cp1 =  (uint32_t) (atime * again) / TSL2591_LUX_DF;
160     lux1 = (uint32_t) ((float) ch0 - (TSL2591_LUX_COEFB * (float) ch1)) / cp1;
161     lux2 = (uint32_t) ((TSL2591_LUX_COEFC * (float) ch0) - (TSL2591_LUX_COEFD * (float) ch1)) / cp1;
162     lux = (lux1 > lux2) ? lux1: lux2;
163
164     return lux;
165 }
166
167 void main (void)
168 {
169
170     //set the main clock to runat 40MHz
171     SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
172     uint32_t lux = 0, i;
173     uint32_t luxAvg = 0;
174
175     ConfigureUART ();   //configure the UART of Tiva C
176     UARTprintf("AT+RST\n\r"); //AT command used to reset device
177     SysCtlDelay (10000000);
178     I2C0_Init ();       //initialize the I2C0 of Tiva C
179     TSL2591_init ();    //initialize the TSL2591
180
181     UARTprintf("AT+CIOBAUD=%d\n\r", 115200);
182     SysCtlDelay (5000000);
183     UARTprintf("AT+CWMODE=1\n\r"); //designating mode for ESP8266
184     SysCtlDelay (5000000);
185     UARTprintf("AT+CWJAP=\"SSID\",\"password\"\n\r"); //conenct to AP
186     SysCtlDelay (50000000);
187
188     //enable button 2 to be used during hibernation
189     SysCtlPeripheralEnable (SYSCTL_PERIPH_HIBERNATE);
190     //Get the system clock to set to the hibernation clock
191     HibernateEnableExpClk (SysCtlClockGet());
192     //Retain the pin function during hibernation
193     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"); //configure for muli ip connections
    SysCtlDelay (5000000);
    UARTprintf("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\n\r"); //AT command that starts TCP connection
    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;

    UARTprintf ("AT+CIPSEND=%d\n", 46);
    SysCtlDelay (50000000);
    UARTprintf("GET /update?key=8WBWPJI4IVZ5OTW9&field1=%d\n\r", luxAvg);
    SysCtlDelay (50000000);
    HibernateRequest ();    //Hibernate
    while (1)
    {
    };
}

```


ESP8266 DOWNLOAD TOOL V3.6.5
SPIDownload
HSPIDownload
RFConfig
GPIOConfig
MultiDo
E 403\MIDTERM\ai-thinker-v1.1.1-115200.bin
0x00000
SpiFlashConfig
CrystalFreq :
40M
CombineBin
Default
SPI SPEED
40MHz
26.7MHz
20MHz
80MHz
SPI MODE
QIO
QOUT
DIO
DOUT
FASTRD
FLASH SIZE
4Mbit
2Mbit
8Mbit
16Mbit
32Mbit
16Mbit-C1
32Mbit-C1
SpiAutoSet
DoNotChgBin
LOCK SETTINGS
DETECTED INFO
flash vendor:
1Ch : N/A
flash devID:
3014h
DUAL;8Mbit
crystal:
26 Mhz
Download Panel 1
FINISH
完成
AP: 5E-CF-7F-76-98-94 STA: 5C-CF-7F-76-98-94
START
STOP
ERASE
COM:
COM14
BAUD:
115200



6) Conclusions

In conclusion, I have accomplished the goal to display the data given by the TSL2561 Lux sensor. This was possible due to the Arduino Uno. I ran into problems transmitting to thingSpeak. The ESP module flashed after much trial and error. The modification of the main file and the TSL2561 header file did not result in any positive results. I managed to display the luminosity with an Arduino Uno and the use f the given libraries by Adafruit.