Aaron Volpone

CPE 403 - 1001

Midterm 1 Project

Goal

The purpose of this midterm project is to apply the knowledge we've acquired in class and through lab work for the TivaC. This is done through a more open-ended assignment that allows us to start with a foundation of project code and build upon it with assigned tasks. For this midterm we were tasked with collecting data from an external light sensor and relaying it through I2C and to the cloud.

Detailed Implementation

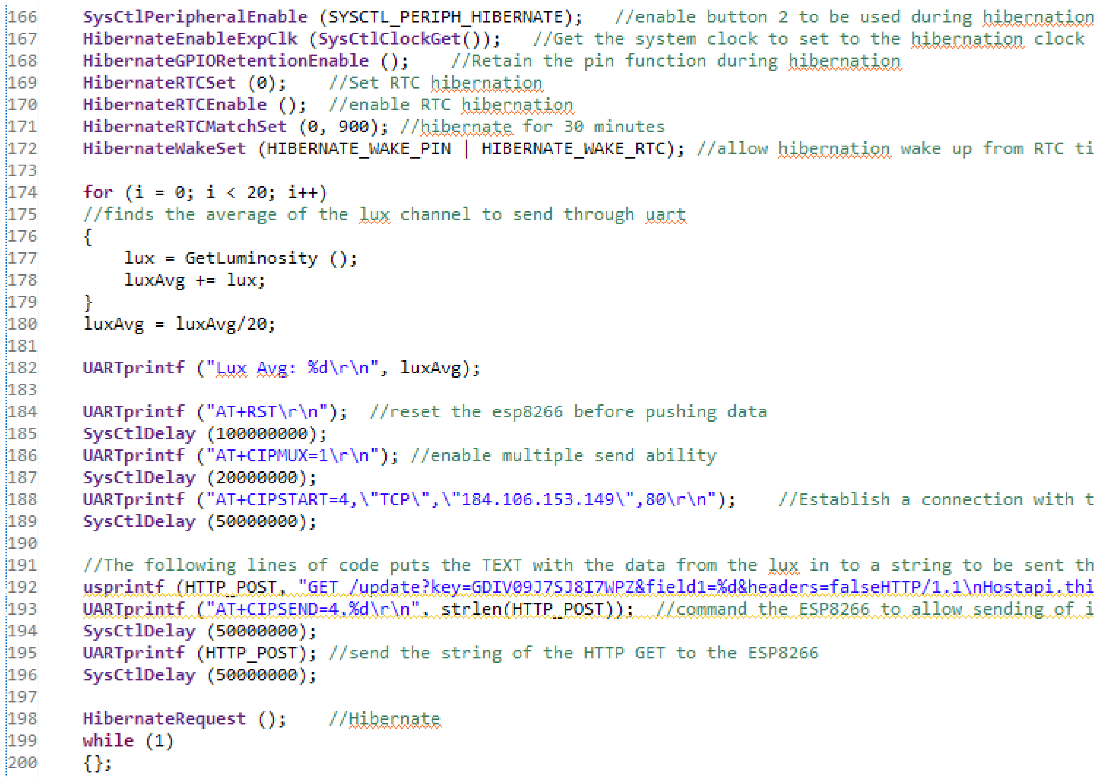
The midterm project uses a multitude of header files and library essential for the operation of I2C, UART, and the lux sensor. While many are used in previous labs, we introduce the I2C and hibernate header files for the purposes of our lower power usage demonstration with the sensor. There were some changes needed to be done to the TSL sensor because it was originally designed to be

Initialization of UART1_BASE serves two purposes: to function as console debugging for the FTDI to USB adapter and to relay information to the cloud with the ESP8266. Before initializing the UART, I2C, and TSL objects, we define the clock to run at 40MHz. Once these items are initialized, the ESP module is needed for data transmission. Unlike the other modules, the ESP requires AT commands to operate through UART. Thus, we perform more UART print statements that push these AT commands to the ESP. Since the ESP is connected through both RX and TX, we can see responses from the device after pushing and AT command to it.

Through the hibernation functions, the TivaC can save power between lux calculations. This is done by setting RTC type hibernation options that set the TivaC to operate only when the RTC match set parameter is met. For this assignment, the TivaC was set to wake up every 15 minutes to conduct lux operations and go back to hibernating.

Since the hibernation wake and requesting operations are like timer-based applications, we can have a loop that infinitely runs in the main function. This serves the purpose of keeping the TivaC in a stand-by state after the lux calculations are done. More AT commands are set to send the collected data up to ThingSpeak after data collection.

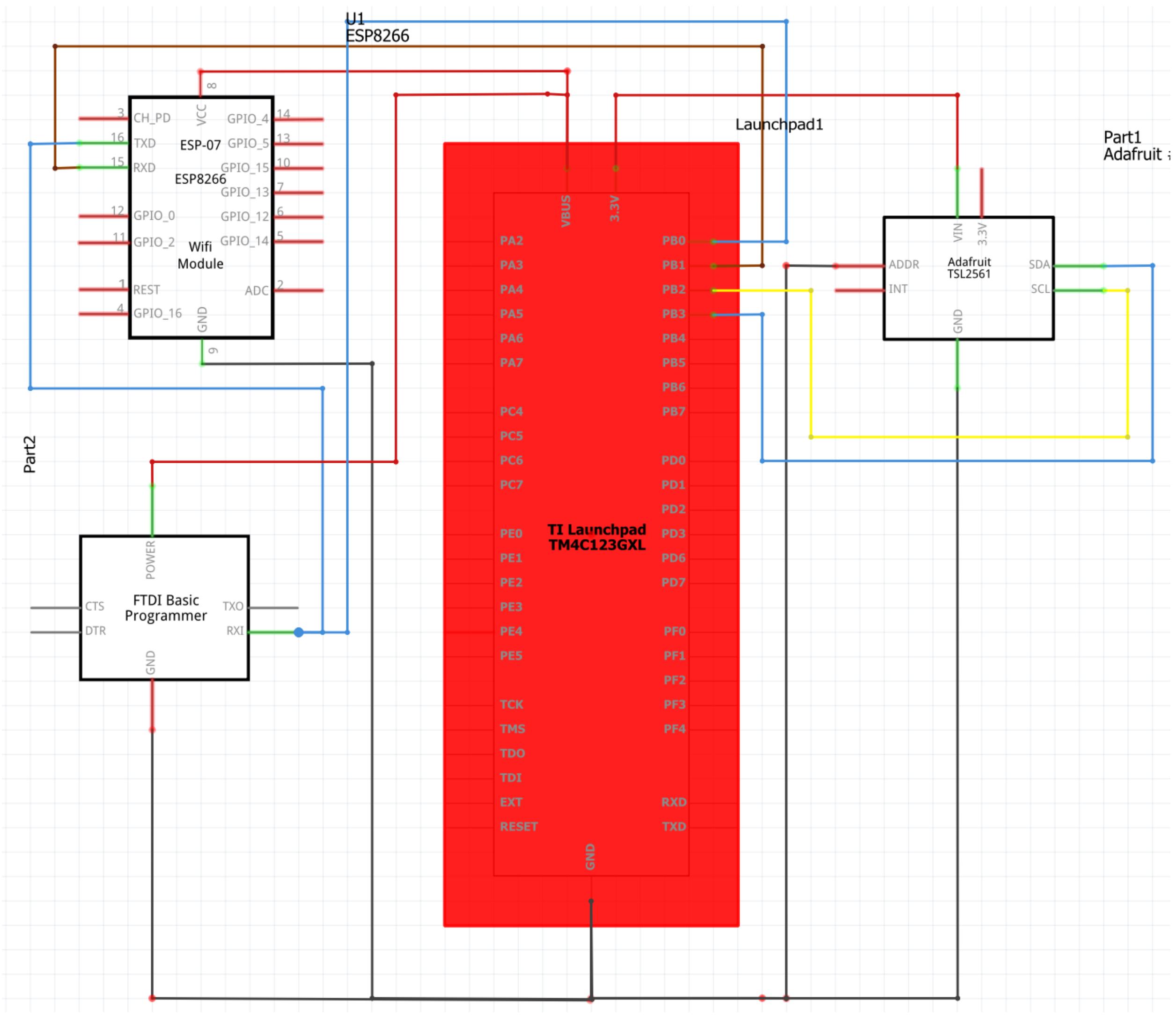
- Channel location – https://thingspeak.com/channels/614326



```
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 hibernation
HibernateRTCSet (0);    //Set RTC hibernation
HibernateRTCEnable ();  //enable RTC hibernation
HibernateRTCMatchSet (0, 900); //hibernate for 30 minutes
HibernateWakeSet (HIBERNATE_WAKE_PIN | HIBERNATE_WAKE_RTC); //allow hibernation wake up from RTC ti

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 ("Lux Avg: %d\r\n", luxAvg);

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 t
SysCtlDelay (50000000);

//The following lines of code puts the TEXT with the data from the lux in to a string to be sent th
usprintf (HTTP_POST, "GET /update?key=GDIV09J7SJ8I7WPZ&field1=%d&headers=falseHTTP/1.1\nHostapi.thi
UARTprintf ("AT+CIPSEND=4,%d\r\n", strlen(HTTP_POST));  //command the ESP8266 to allow sending of i
SysCtlDelay (50000000);
UARTprintf (HTTP_POST); //send the string of the HTTP GET to the ESP8266
SysCtlDelay (50000000);

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

Schematics

The ESP8266 is the standard variant in Fritzing. In actual use, the NodeMCU ESP8266 was used. For debugging, the FTDI to USB adapter was used to read UART printed strings through PuTTY.

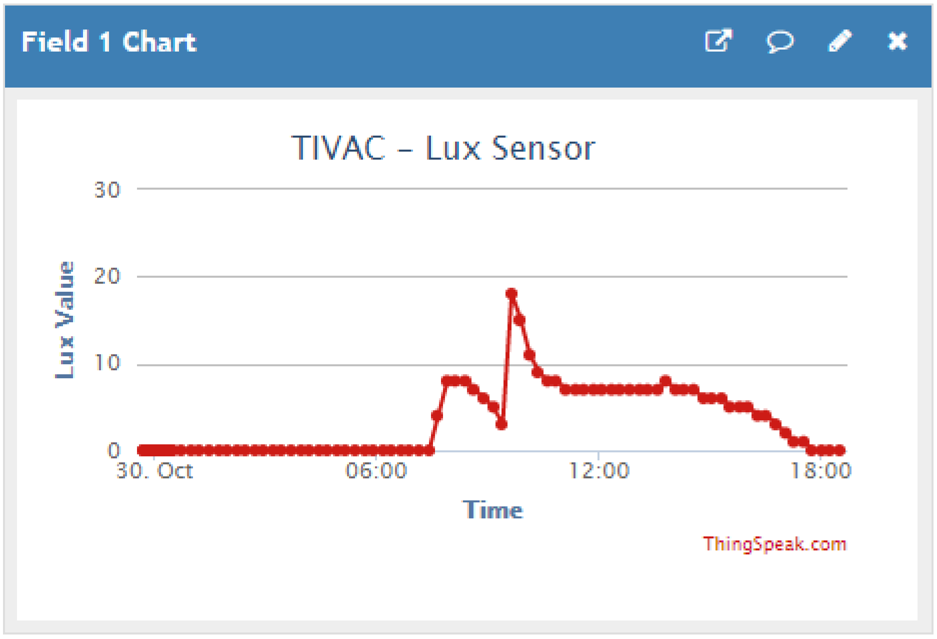


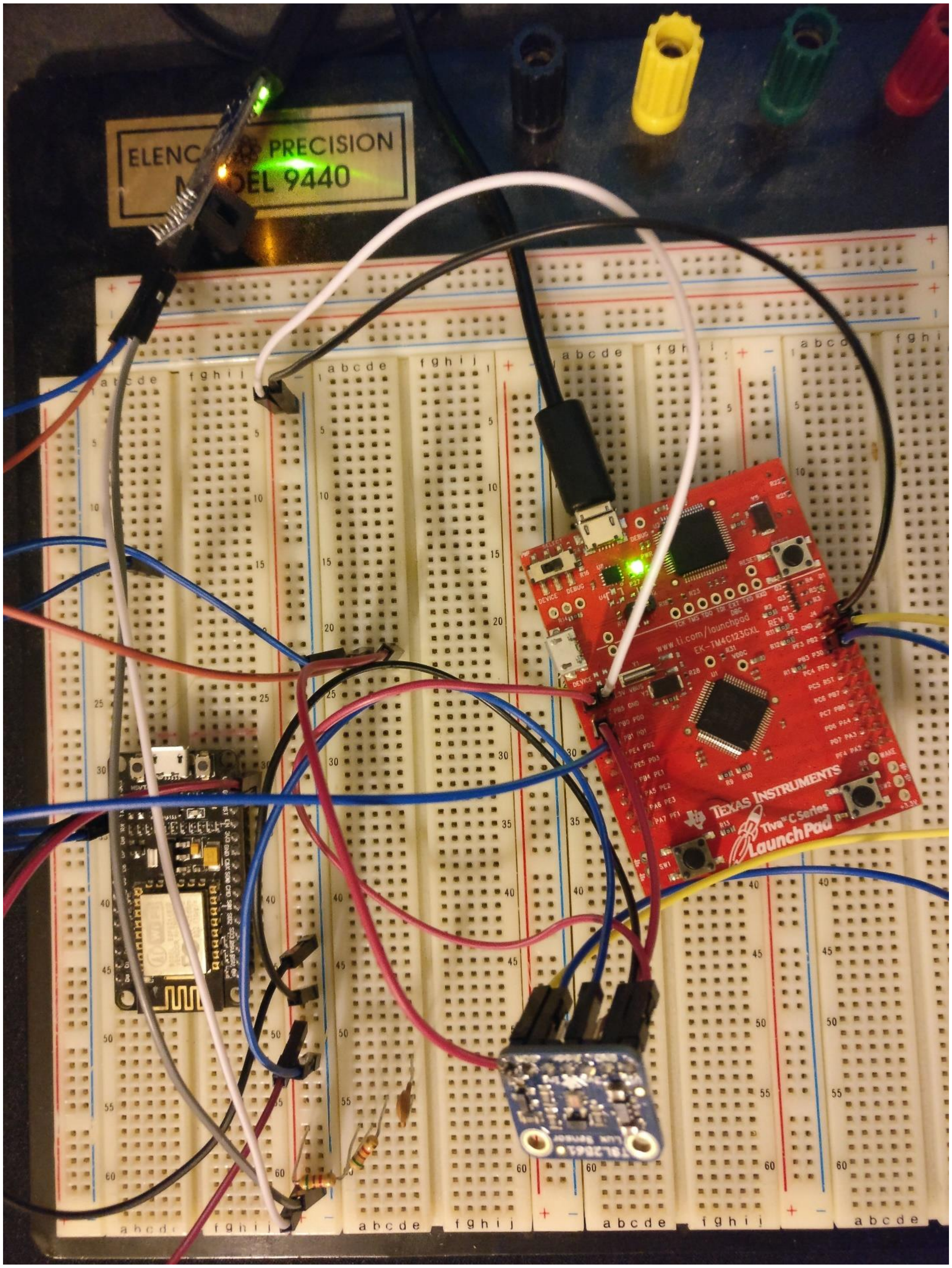
Video Link

Youtube link: https://youtu.be/Qi-nYEl_Gqg

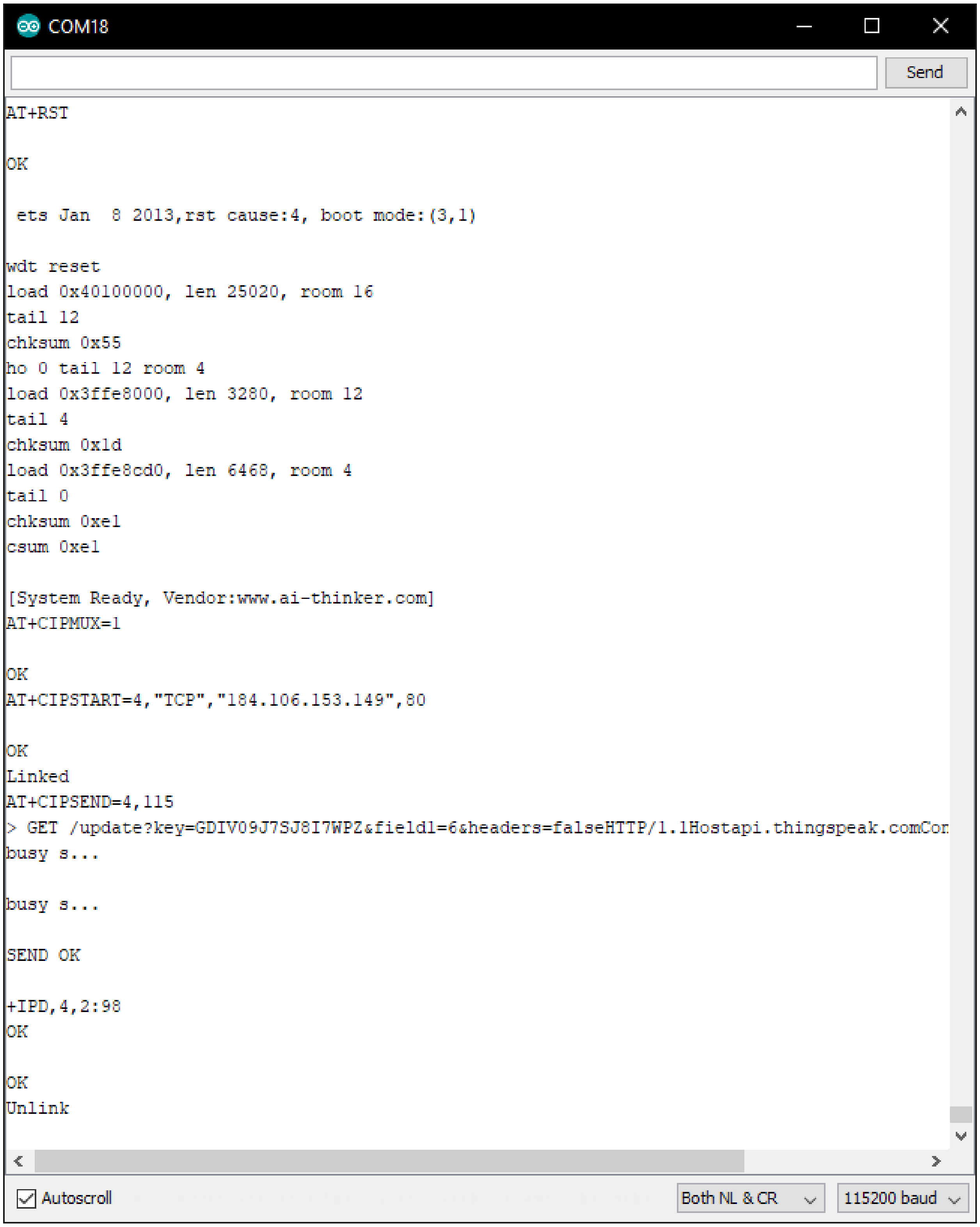
Screenshots

The first screenshot is a graph of the Lux sensor data through ThingSpeak. Submission interval for the sensor is one minute.




ELENCO PRECISION
MODEL 9440
TEXAS INSTRUMENTS
Tiva C Series
LaunchPad
EK-TM4C123GXL
TSL2561
Lux Sensor

Here is the terminal strings provided by the TivaC and the ESP module.



```
AT+RST

OK


 ets Jan  8 2013,rst cause:4, boot mode:(3,1)

wdt reset
load 0x40100000, len 25020, room 16
tail 12
chksum 0x55
ho 0 tail 12 room 4
load 0x3ffe8000, len 3280, room 12
tail 4
chksum 0x1d
load 0x3ffe8cd0, len 6468, room 4
tail 0
chksum 0xe1
csum 0xe1

[System Ready, Vendor:www.ai-thinker.com]
AT+CIPMUX=1

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

OK
Linked
AT+CIPSEND=4,115
> GET /update?key=GDIV09J7SJ8I7WPZ&field1=6&headers=falseHTTP/1.1Hostapi.thingspeak.comCor
busy s...

busy s...

SEND OK

+IPD,4,2:98
OK

OK
Unlink
```

Conclusion

The implementation of I2C with the lux sensor was relatively straightforward given the libraries and provided code. Most of the time committed to the project was a result of the debugging process when wiring the various components on a breadboard. It was important to understand how to manage the crosstalk from the ESP module and the TivaC when relaying it to the FTDI module. Overall, the midterm project help learn about uploading data to the cloud and also communication with an ESP module.