**10/27/2018:**

1. Goal

The goal of this project is to use the Tiva C board to collect data from the TSL2591 lux sensor and then push that data wirelessly from a nodeMCU to a thingspeak server as a demonstration of a simple Iot application.

2) Detailed Implimentation

The following is the code I used to both collect and send the data. This code was pushed into the Tiva C board using code composer.

**------------------------------------------------------------------------------------**

**Modified 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 "driverlib/timer.h"**

**#include "uartstdio.h"**

**#include "driverlib/interrupt.h"**

**#include "driverlib/hibernate.h"**

**#include "TSL2591\_def.h"**

**#include "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); // enable GPIO port b**

**GPIOPinConfigure(GPIO\_PB0\_U1RX); // configures PB0 as RX**

**GPIOPinConfigure(GPIO\_PB1\_U1TX); // configures PB1 as TX**

**GPIOPinTypeUART(GPIO\_PORTB\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1); // sets UART pin type**

**// sets uart clock, boudrate, and which uart to use**

**UARTConfigSetExpClk(UART1\_BASE, SysCtlClockGet(), 115200,**

**(UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE | UART\_CONFIG\_PAR\_NONE));**

**}**

**// prints uart command string**

**void UARTCommand(unsigned char \*str) {**

**char i=0;**

**while(str[i] != '\0') {**

**UARTCharPut(UART1\_BASE, str[i]);**

**i++;**

**}**

**}**

**// prints integer to uart**

**void UARTInt(int n) {**

**unsigned char uartChar[32];**

**int count=0;**

**while(n > 0) {**

**uartChar[count] = n%10;**

**n/=10;**

**uartChar[count] += '0';**

**count++;**

**}**

**count--;**

**while(count>=0) {**

**UARTCharPut(UART1\_BASE, uartChar[count]);**

**count--;**

**}**

**}**

**// calculates the number of chars to print int**

**int IntLen(int n) {**

**int count=0;**

**while(n > 0) {**

**n/=10;**

**count++;**

**}**

**return count;**

**}**

**void I2C0\_Init ()**

**//Configure/initialize the I2C0**

**{**

**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);**

**//Set the clock of the I2C to ensure proper connection**

**I2CMasterInitExpClk (I2C0\_BASE, SysCtlClockGet(), false);**

**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**

**{**

**//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**

**//if more than 1, loop through all the commands until they are all sent**

**{**

**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));**

**}**

**//puts the last argument on the SDA bus**

**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 send 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", x);**

**}**

**else**

**{**

**UARTprintf ( "Device not Found! %i", 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));**

**}**

**void MCU\_init ()**

**//Initializes the MCU**

**{**

**UARTCommand("AT+RST\r\n"); // resets the MCU**

**SysCtlDelay(200000);**

**UARTCommand("AT+CWMODE=1\r\n"); // sets MCU to station mode**

**SysCtlDelay(20000);**

**UARTCommand("AT+CWJAP=\"Zcolo123-2Ghz\",\"DataCenter123#\"\r\n"); // connects MCU to wifi router**

**SysCtlDelay(200000);**

**UARTCommand("AT+CIPMUX=0\r\n"); // sets MCU to single connection**

**SysCtlDelay(20000);**

**}**

**void timer0\_init ()**

**//Initializes the timer0 to 30 seconds**

**{**

**uint32\_t ui32Period;**

**// enable timer0 and port F**

**SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);**

**GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3);**

**SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER0);**

**TimerConfigure(TIMER0\_BASE, TIMER\_CFG\_PERIODIC);**

**// calculate timer period for 30 seconds**

**ui32Period = (SysCtlClockGet() \* 30);**

**TimerLoadSet(TIMER0\_BASE, TIMER\_A, ui32Period -1);**

**// enable timer0 interrupt**

**IntEnable(INT\_TIMER0A);**

**TimerIntEnable(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);**

**IntMasterEnable();**

**// start timer0**

**TimerEnable(TIMER0\_BASE, TIMER\_A);**

**}**

**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));**

**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)**

**{**

**//set the main clock to run at 40MHz**

**SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_XTAL\_16MHZ|SYSCTL\_OSC\_MAIN);**

**ConfigureUART (); //configure the UART of Tiva C**

**I2C0\_Init (); //initialize the I2C0 of Tiva C**

**TSL2591\_init (); //initialize the TSL2591**

**MCU\_init(); //initialize the nodeMCU**

**timer0\_init(); //initialize the timer0**

**while (1)**

**{};**

**}**

**void Timer0IntHandler(void)**

**{**

**// Clear the timer interrupt**

**TimerIntClear(TIMER0\_BASE, TIMER\_TIMA\_TIMEOUT);**

**uint32\_t lux = 0, i;**

**uint32\_t luxAvg = 0;**

**for (i = 0; i < 20; i++)**

**//finds the average of the lux channel to send through uart**

**{**

**lux = GetLuminosity ();**

**luxAvg += lux;**

**}**

**luxAvg = luxAvg/20;**

**// connect to thingspeak using TCP connection**

**UARTCommand("AT+CIPSTART=\"TCP\",\"184.106.153.149\",80\r\n");**

**SysCtlDelay(200000);**

**// calculate and send how many characters are necessary**

**UARTCommand("AT+CIPSEND=");**

**SysCtlDelay(20);**

**UARTInt(72+IntLen(luxAvg));**

**SysCtlDelay(20);**

**UARTCommand("\r\n");**

**SysCtlDelay(20000);**

**// send data to thingspeak**

**UARTCommand("GET https://api.thingspeak.com/update?api\_key=0XLLUY55C4GFKPH7&field1=");**

**SysCtlDelay(20);**

**UARTInt(luxAvg);**

**SysCtlDelay(20);**

**UARTCommand("\r\n");**

**SysCtlDelay(20000);**

**// blink LED**

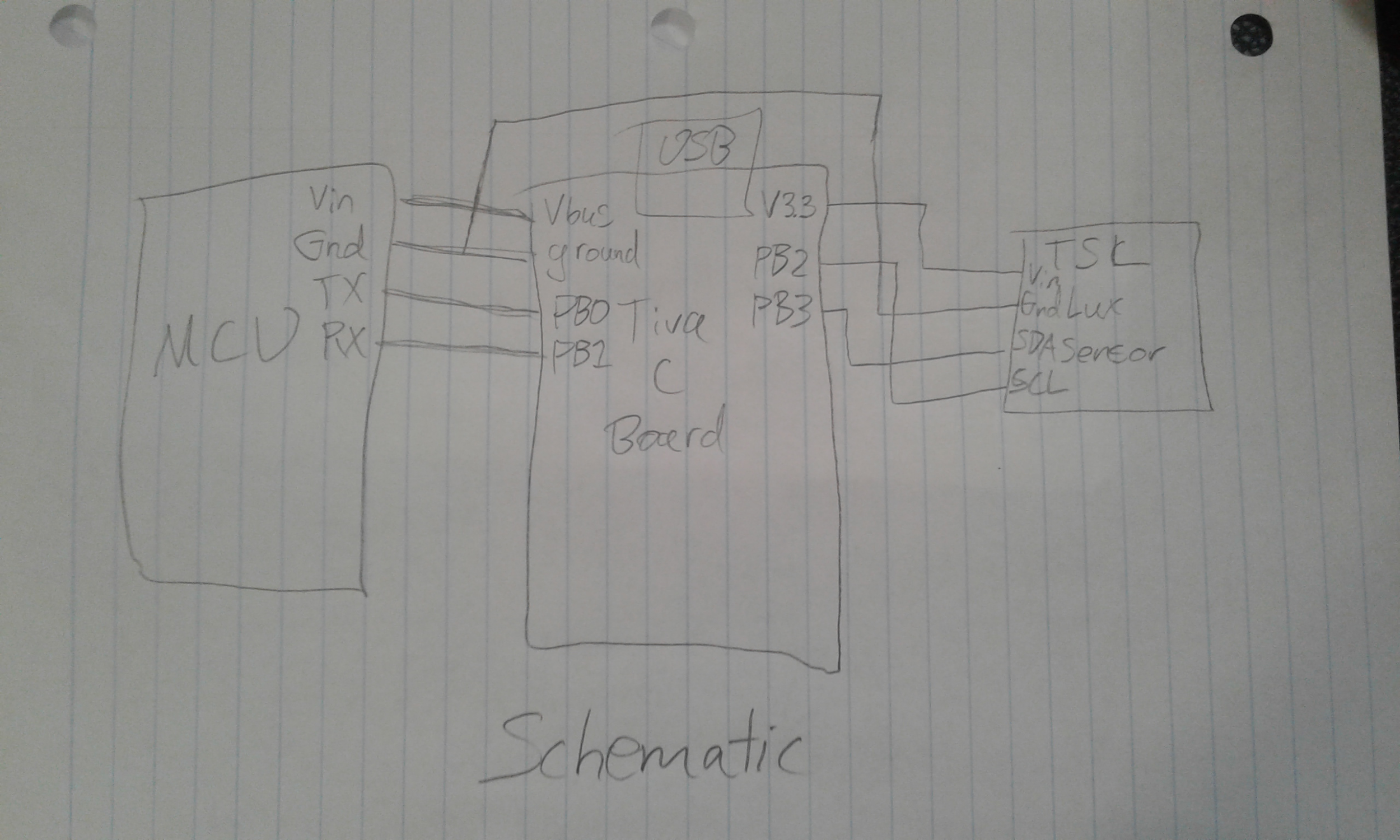
**GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 4);**

**SysCtlDelay(4000000);**

**GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0);**

**}**

**------------------------------------------------------------------------------------**

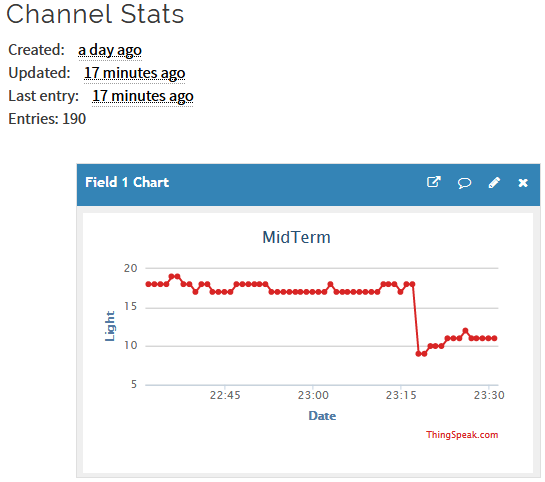
**3) Schematics  
  
 The following is a schematic of the connections needed to use these electronics together and perform the necessary operations.  
  
  
  
 Hand drawn schematic for the project**

**4) Video Links**

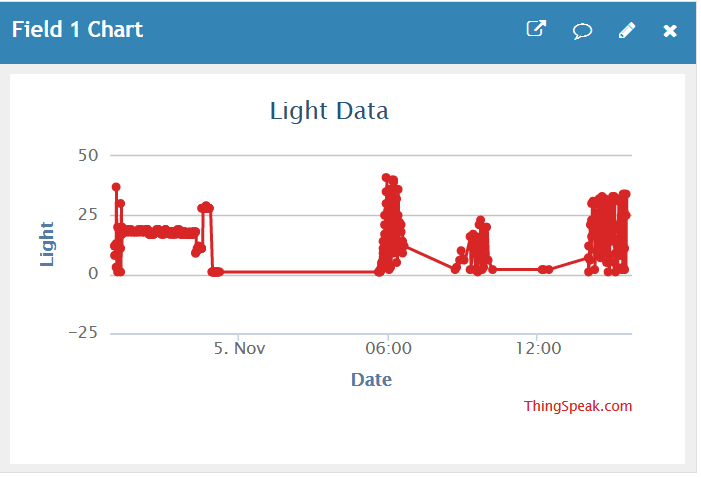
**Video Link:** [**https://www.youtube.com/watch?v=tHmhrQhCHYM**](https://www.youtube.com/watch?v=tHmhrQhCHYM)

**5) Screenshots**

**Here is some data it collecting for a couple hours in my house while I was shining lights on it and casting shadows on it.**

****

**Chart for data while in my house.**

****

**Chart with approximately 18 hours of data while on a window sill**

**The first part of the last chart was from indoor light, but when it drops down late at night on november 4 is when it was put on the window. There is also a minor error in the code which causes it to not send data if the reading is zero, which is why there are large gaps especially late at night.**

**6) Conclusions**

**Overall, this project was far more difficult than I had anticipated. Getting a few different devices to talk to each other especially using different communication interfaces was a challenge. In the end it seems I pulled through and got it working. Of all the work I have done in this class so far, this seems to be, by far, the most useful application. Sensors are always needed to gather more data, and transmitting it to a website is a pretty realistic end goal for a piece of technology like this.**