Adrian Ruiz

CPE 403

Midterm

# 1.Goal

* Read luminosity from a TSL2591 lux sensor and upload to the lux reading to Thinkspeak.

# 2. Detailed Implementation

* First, the luminosity data from the TSL2591 sensor with the launchpad. The Tiva-C is configured for I2C protocol. PB3 is set to SDA, and PB2 is set to SCL. The lux reading is then converted to a string using the UART. The string is sent to the NodeMCU by the RX pin it has. The string is then sent to ThingSpeak. Every 15 seconds a new reading is sent to the NodeMCU to be sent to ThingSpeak.

# 3.Code

Tiva:

#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 "Adafruit\_TSL2591.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

//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

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\_REGISTER\_DEVICE\_ID));//read the device ID

if (x == 0x50)

{

}

else

{

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);

I2C0\_Write (TSL2591\_ADDR, 2, (TSL2591\_COMMAND\_BIT | TSL2591\_REGISTER\_CONTROL), 0x10);

//enables proper interrupts and power to work with TSL2591

//I2C0\_Write (TSL2591\_ADDR, 2, (TSL2591\_COMMAND\_BIT | TSL2591\_ENABLE),

I2C0\_Write (TSL2591\_ADDR, 2, (TSL2591\_COMMAND\_BIT | TSL2591\_REGISTER\_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\_REGISTER\_CHAN0\_HIGH));

x = I2C0\_Read (TSL2591\_ADDR, (TSL2591\_COMMAND\_BIT | TSL2591\_C0DATAH));

x <<= 16;

//x |= I2C0\_Read (TSL2591\_ADDR, (TSL2591\_COMMAND\_BIT | TSL2591\_REGISTER\_CHAN0\_lOW));

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);

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

//enable button 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

HibernateGPIORetentionEnable ();

//Set RTC hibernation

HibernateRTCSet (0);

//enable RTC hibernation

HibernateRTCEnable ();

//hibernate for 30 minutes

//HibernateRTCMatchSet (0, 1800);

HibernateRTCMatchSet (0, 30);

//allow hibernation wake up from RTC time or button 2

HibernateWakeSet (HIBERNATE\_WAKE\_PIN | HIBERNATE\_WAKE\_RTC);

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 Value: %d \r\n",luxAvg);

UARTprintf("%d\r",luxAvg);

SysCtlDelay(100000);

HibernateRequest (); //Hibernate

while (1)

{};

}

Esp:

#include <ESP8266WiFi.h>

#define period 15 //15 secs for Thingspeak

//My SSID information

const char**\*** server **=** "api.thingspeak.com"**;**

String apikey **=** "S5WR2DBEPDZ2UB0L"**;**

const char**\*** ssid **=** "\*\*\*\*\*\*\*\*"**;**

const char**\*** password **=** "\*\*\*\*\*\*"**;**

//holds lux value

String luxValue**;**

//character read in

char readIn**;**

int sent **=** 0**;**

void sendValue**(**String lux**)**

**{**

//Creates a client that can connect to to a specified internet IP address and port

WiFiClient client**;**

**if** **(**client**.**connect**(**server**,** 80**))** **{** // use ip 184.106.153.149 or api.thingspeak.com

Serial**.**println**(**"WiFi Client connected "**);**

//String to be sent off to server

String postStr **=** apikey**;**

postStr **+=** "&field1="**;**

postStr **+=** lux**;**

postStr **+=** "\r\n\r\n"**;**

//Make an HTTP request

client**.**print**(**"POST /update HTTP/1.1\n"**);**

client**.**print**(**"Host: api.thingspeak.com\n"**);**

//Transmit Information once request has been approved

client**.**print**(**"Connection: close\n"**);**

client**.**print**(**"X-THINGSPEAKAPIKEY: " **+** apikey **+** "\n"**);**

client**.**print**(**"Content-Type: application/x-www-form-urlencoded\n"**);**

client**.**print**(**"Content-Length: "**);**

client**.**print**(**postStr**.**length**());**

client**.**print**(**"\n\n"**);**

client**.**print**(**postStr**);**

delay**(**1000**);**

}//end if

sent++;

client.stop();

}

void connectwifi()

{

Serial.print("Connecting to "+\*ssid);

//Initializes the WiFi library's network settings and provides the current status

WiFi.begin(ssid, password);

//Keep trying to successfuly connect

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.print(".");

}

//Successful connections

Serial.println("");

Serial.println("Connected");

Serial.println("");

}

void setup() {

// put your setup code here, to run once:

Serial.begin(115200);

connectwifi();

}

void loop() {

// put your main code here, to run repeatedly:

if(Serial.available())

{

//

}

Serial.print(String(sent)+" Luminosity Sensor: ");

Serial.println(luxValue);

//Send Lux value to client

sendValue(luxValue);

int count = period;

while(count--){

luxValue = "";

delay(1000);

}

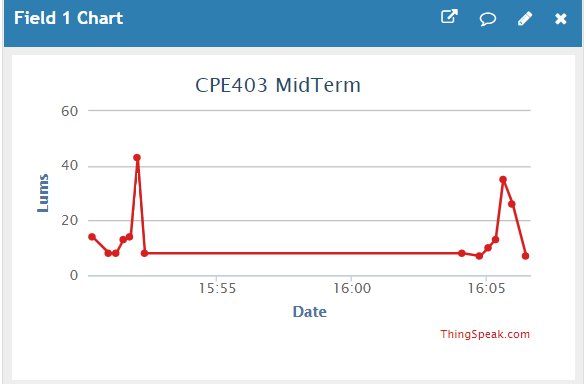
Serial.flush();

}

# 4. Screenshots



Terminal



ThingSpeak Chart

# 5.Conclusion

* I was able to get a reading from the TSL2591 using the launchpad. I was also able to send that data to ThingSpeak using the NodeMCU.