Abenezer Namaga

CPE 403

Github root directory: https://github.com/Ayertena/TIVAC

TIVAC the Light

GOAL:

* Configure I2c interface between TM4C123GXL board and TSL2951 sensor
* Configure UART communication between TM4C123GXL and ESP8266 module
* Upload TSL2951 data readings to cloud server hosted on thingspeak.com

DELIVERABLES:

The final project was supposed to record luminosity from the TSL2951 sensor and send that data via I2C to the TM4C123G; that light data was then processed and transmitted to the ESP8266 Wi-Fi module which would connect to the cloud and graph the data. All of this was accomplished with the most difficult part being configuring the TSL2951 sensor. I was not properly initializing the ALS portion of the sensor, and not leaving a delay for the values to be gathered so I would only get 0. After fixing this I was able to get values as low as 0 (pitch black) or as high as 1138.83.

COMPONENTS:

* **Tiva TM4C123GH6PM –** MCU to control project. Device is initialized by setting the system clock, enabling GPIO module, enabling the I2C module, enabling and setting the clock for I2C0 master module, enabling UART1, and configuring port pins.
* **TSL2591 –** A high dynamic digital light sensor which can measure the Lux up to ranges of 188u – 88,000 Lux. Its interfaced with the TivaC using I2C.
* **ESP8266 -** A Wi-Fi module interfaced with the TivaC using UART. It is initialized and will send information to server.
* **TSL2591 –** A high dynamic digital light sensor which can measure the Lux up to ranges of 188u – 88,000 Lux. Its interfaced with the TivaC using I2C. It is initialized by reading the Device ID register setting the gain, the timing, and the Power On Register to 1. It will read the channel data from Channel0 and Channel1; moreover, these two values are calculated to find the Lux value of the sensor and the value is returned to the caller.

SCREENSHOTS & PHOTO:

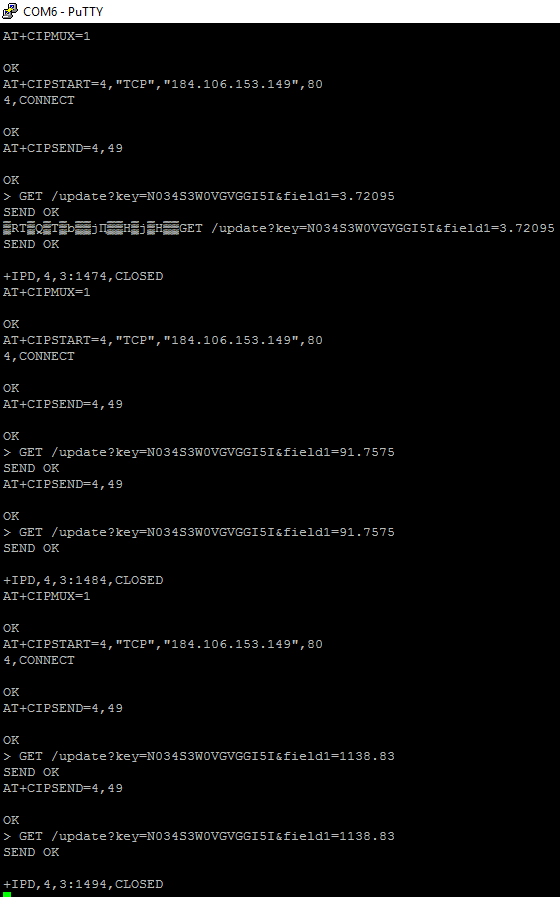


Figure - Putty terminal showing strings being transmitted from TM4C to ESP8266 shows 3 posts being made with low, medium, and high lighting on senso

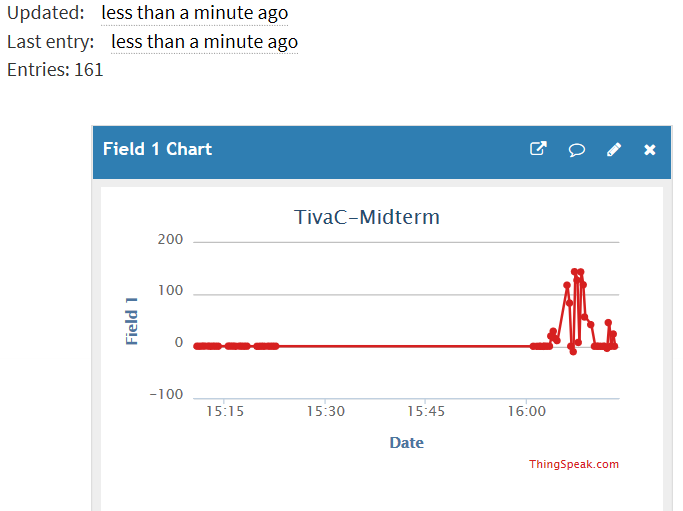


Figure 2- Thingspeak plot of data from TSL2591 (Peaks represent bright light contact with sensor)

ESP8266

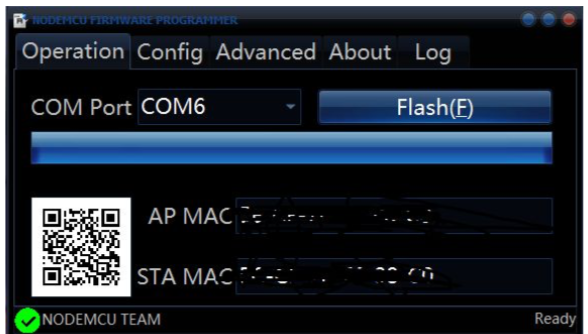


Figure 3- ESP8266 flash success

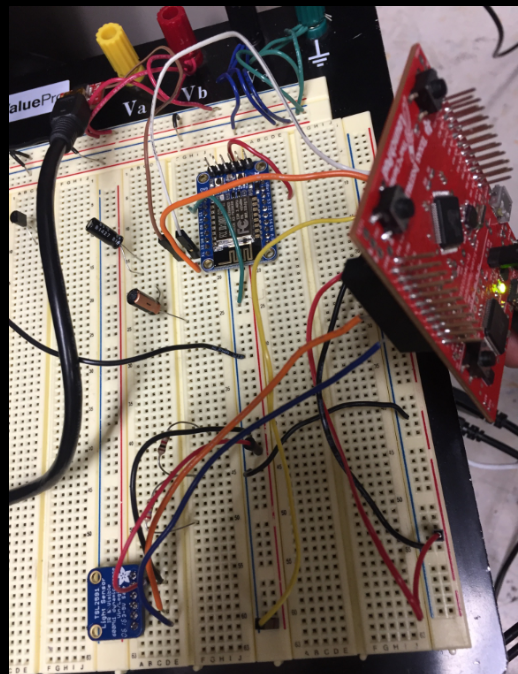
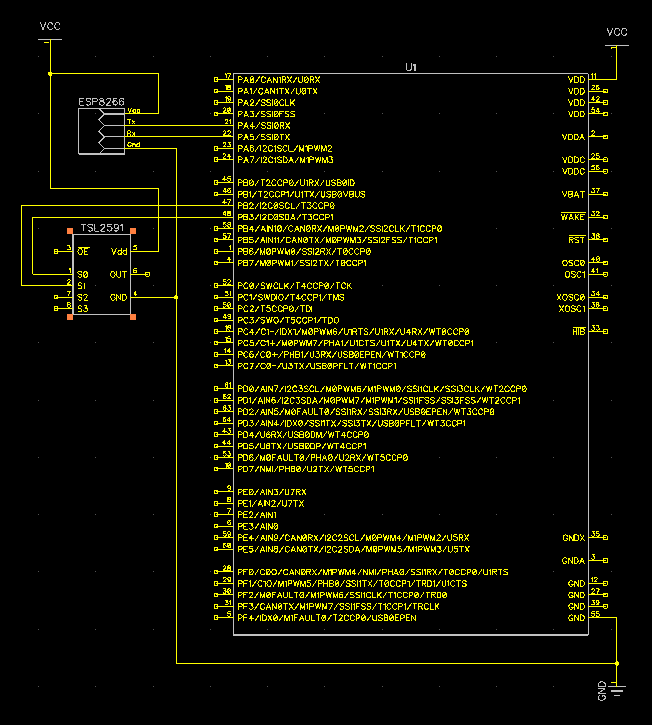


Figure 4- Photo showing interconnection of TM4C, TSL2951, and ESP8266

SCHEMATICS:



CODE:

**#include** <stdbool.h>

**#include** <stdint.h>

**#include** "inc/hw\_i2c.h"

**#include** "inc/hw\_ints.h"

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_types.h"

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

**#include** "driverlib/i2c.h"

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

**#include** "driverlib/pin\_map.h"

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

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

**#include** "utils/uartstdio.h"

**#include** "driverlib/rom\_map.h"

**#include** "inc/hw\_types.h"

**#include** "inc/hw\_gpio.h"

uint32\_t ui32SysClock;

**const** uint8\_t TSL2591address = 0x29;

//UART1

**void** **InitConsole**(**void**)

{

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

**GPIOPinConfigure**(GPIO\_PB0\_U1RX);

**GPIOPinConfigure**(GPIO\_PB1\_U1TX);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART1);

UARTClockSourceSet(UART1\_BASE, UART\_CLOCK\_PIOSC);

**GPIOPinTypeUART**(GPIO\_PORTB\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

UARTStdioConfig(1, 115200, 16000000);

}

**void** **ftoa**(**float** f,**char** \*buf)

{

/\*Function acquired from forum:

\* http://e2e.ti.com/support/microcontrollers/stellaris\_arm/f/471/p/44193/156824.aspx

\*/

/\*Parses through float # and stores the value as a character array\*/

**int** pos=0,ix,dp,num;

**if** (f<0)

{

buf[pos++]='-';

f = -f;

}

dp=0;

**while** (f>=10.0)

{

f=f/10.0;

dp++;

}

**for** (ix=1;ix<8;ix++)

{

num = (**int**)f;

f=f-num;

**if** (num>9)

buf[pos++]='#';

**else**

buf[pos++]='0'+num;

**if** (dp==0) buf[pos++]='.';

f=f\*10.0;

dp--;

}

}

// USING I2C0 - CHECK FOR TIVAC LAUNCHPAD PINS

**void** **i2c0\_init**()

{

//enable i2c0

MAP\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_I2C0);

//reset i2c0

MAP\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOB);

//Set pin type for PB3, configure as data pin

MAP\_GPIOPinTypeI2C(GPIO\_PORTB\_BASE, GPIO\_PIN\_3);

MAP\_GPIOPinConfigure(GPIO\_PB3\_I2C0SDA);

//Set pin type for PB2, configure as SCL

MAP\_GPIOPinTypeI2CSCL(GPIO\_PORTB\_BASE, GPIO\_PIN\_2);

MAP\_GPIOPinConfigure(GPIO\_PB2\_I2C0SCL);

//Enable and initialize i2c0 master module. Use System Clock. Set i2C data rate at 100kbps (try true for 400)

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

**while** (I2CMasterBusy(I2C0\_BASE));

}

**void** **i2c0\_write**(uint8\_t dev\_addr, uint8\_t dev\_reg, uint16\_t dev\_data)

{

//Tell device we want to write to bus

I2CMasterSlaveAddrSet(I2C0\_BASE, dev\_addr, false);

//Bitwise or command bit to tell device we want to write to dev\_reg

I2CMasterDataPut(I2C0\_BASE, TSL2591\_COMMAND\_BIT |dev\_reg);

//Send and wait

I2CMasterControl(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_SEND\_START);

**while** (I2CMasterBusy(I2C0\_BASE));

//Now write the data to the register

I2CMasterSlaveAddrSet(I2C0\_BASE, dev\_addr, true);

//place the data in Master and send

I2CMasterDataPut(I2C0\_BASE, dev\_data);

I2CMasterControl(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_RECEIVE\_FINISH);

**while** (I2CMasterBusy(I2C0\_BASE));

}

uint8\_t **i2c0\_read**(uint32\_t slave\_addr, uint8\_t reg)

{

//specify that we are writing (a register address) to the

//slave device

I2CMasterSlaveAddrSet(I2C0\_BASE, slave\_addr, false);

**while**(I2CMasterBusy(I2C0\_BASE));

//specify register to be read

I2CMasterDataPut(I2C0\_BASE, TSL2591\_COMMAND\_BIT | reg);

**while**(I2CMasterBusy(I2C0\_BASE));

//send control byte and register address byte to slave device

I2CMasterControl(I2C0\_BASE, I2C\_MASTER\_CMD\_SINGLE\_SEND);

//wait for MCU to finish transaction

**while**(I2CMasterBusy(I2C0\_BASE));

//specify that we are going to read from slave device

I2CMasterSlaveAddrSet(I2C0\_BASE, slave\_addr, true);

**while**(I2CMasterBusy(I2C0\_BASE));

//send control byte and read from the register we

//specified

I2CMasterControl(I2C0\_BASE, I2C\_MASTER\_CMD\_SINGLE\_RECEIVE);

//wait for MCU to finish transaction

**while**(I2CMasterBusy(I2C0\_BASE));

//return data pulled from the specified register

**return** I2CMasterDataGet(I2C0\_BASE);

}

**void** **TSL2591\_init**()

{

uint8\_t DevID;

i2c0\_write(TSL2591address,TSL2591address,1); //literally no idea what this does

DevID = i2c0\_read(TSL2591address,TSL2591\_REGISTER\_DEVICE\_ID); //read device ID

//Alert user if errors reading ID

// Set Gain and Timing

i2c0\_write(TSL2591address,0x01, 0xB0 );//CONFIG medium gain and reset the device (self-clearing)

i2c0\_write(TSL2591address, 0x00, 0x03);//ENABLE Power, Oscillator,

}

**void** **TSL2591\_disable**()

{

i2c0\_write(TSL2591address,0xA0, 0x00); //turn off TSL2591

}

**float** **getLuminosity** ()

{

**float** atime = 100.0F, again=25.0F; //For 100ms integration time and med gain

**float** cpl, lux1, lux2, lux;

uint16\_t ch0 ;

uint16\_t ch1 ;

// Get full luminosity

//read channel 0 into x0

uint16\_t x0;

uint16\_t y0;

y0 = i2c0\_read(TSL2591address,0x14);

**while**(I2CMasterBusy(I2C0\_BASE));

// UARTprintf("\n\nChannel 0L: %d", y0);

x0 = i2c0\_read(TSL2591address,0x15); //C0DataHigh

**while**(I2CMasterBusy(I2C0\_BASE));

// UARTprintf("\nChannel 0H: %d", x0);

x0 <<= 8;

uint16\_t x1;

uint16\_t y1;

//read channel 1 into x1

y1 = i2c0\_read(TSL2591address, 0x16);

**while**(I2CMasterBusy(I2C0\_BASE));

// UARTprintf("\nChannel 1L: %d", y1);

x1 = i2c0\_read(TSL2591address, 0x17);

**while**(I2CMasterBusy(I2C0\_BASE));

// UARTprintf("\nChannel 1H: %d", x1);

x1 <<= 8;

// Disable Sensor

ch0 = x0+y0;

ch1 = x1+y1;

//Calculate Lux value from sensor

cpl = (atime \* again) / TSL2591\_LUX\_DF;

lux1 = ( (**float**)ch0 - (TSL2591\_LUX\_COEFB \* (**float**)ch1) ) / cpl;

lux2 = ( ( TSL2591\_LUX\_COEFC \* (**float**)ch0 ) - ( TSL2591\_LUX\_COEFD \* (**float**)ch1 ) ) / cpl;

lux = lux1 > lux2 ? lux1 : lux2;

**return** lux;

}

**void** **ESPSENDPREP**()

{

**SysCtlDelay**(2000000);

UARTprintf("AT+CIPMUX=1\n\r");

**SysCtlDelay**(2000000);

UARTprintf("AT+CIPSTART=4,\"TCP\",\"184.106.153.149\",80\n\r");

**SysCtlDelay**(2000000);

// SysCtlDelay(20000000);

// UARTprintf("GET /update?key=N034S3W0VGVGGI5I&field1=50\n\r");

// SysCtlDelay(20000000);

// UARTprintf("GET /update?key=N034S3W0VGVGGI5I&field1=\n\r");

}

**int** **main**(**void**)

{

**char** myLux[48] = "GET /update?key=N034S3W0VGVGGI5I&field1=";//52 bit string, 42 instr, 6 are lux. (append \n\r at end)

**float** lux\_read;

ui32SysClock = MAP\_SysCtlClockFreqSet((SYSCTL\_XTAL\_25MHZ |

SYSCTL\_OSC\_MAIN | SYSCTL\_USE\_PLL |

SYSCTL\_CFG\_VCO\_480), 120000000);

InitConsole();

i2c0\_init();

**while**(1)

{

// UARTprintf("AT\n");

TSL2591\_init();

**SysCtlDelay**(2000000);

lux\_read = getLuminosity();

ftoa(lux\_read, &myLux[40]);

myLux[47]='\0';

TSL2591\_disable();

ESPSENDPREP();

UARTprintf("AT+CIPSEND=4,49\n\r");

**SysCtlDelay**(20000000);

UARTprintf("%s\n\r",&myLux);

**SysCtlDelay**(20000000);

UARTprintf("AT+CIPSEND=4,49\n\r");

**SysCtlDelay**(20000000);

UARTprintf("%s\n\r",&myLux);

**SysCtlDelay**(5000000\*20); //slightly more than 15 seconds per so

}

}

Name: Abenezer Namaga