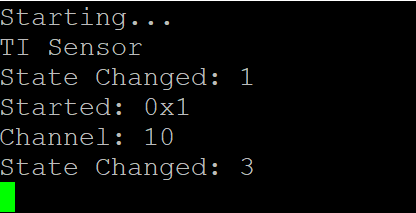
**Date Submitted: 12/11/2019**

For tasks 4+, I only took screenshots of things I changed since the code files are too big to paste into doc.

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

**Task 01:**

Youtube Link: No link required since we are just setting up project.

A screenshot of a social media post

Description automatically generated

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

**Task 02:**

Youtube Link: No link required since we are just setting up project.

Project was able to build:A screenshot of a social media post

Description automatically generated

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

**Task 03:**

Youtube Link: <https://youtu.be/3SvWIaa7oHg>

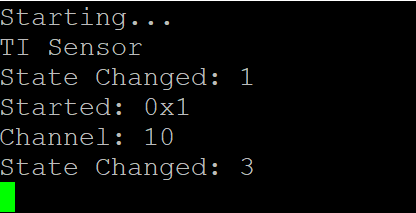
A screenshot of a social media post

Description automatically generated

**A close up of a logo

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

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

**Task 04:**

Youtube Link: https://youtu.be/nFhr\_M2Zb1w

**A screenshot of a cell phone

Description automatically generated**

****

**Sensor code successfully compiled:**

**A screenshot of a social media post

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**A black sign with white text

Description automatically generated**

**Sensor terminal**

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

**Task 01 Part 2:**

Youtube Link: https://youtu.be/F0ucW7RY0Z8

/\*

\* ======== temperature.c ========

\*/

**#include** <stdint.h>

**#include** <stddef.h>

**#include** <unistd.h>

**#include** <ti/display/Display.h>

/\* POSIX Header files \*/

**#include** <pthread.h>

**#include** <semaphore.h>

**#include** <signal.h>

**#include** <time.h>

/\* Driver Header files \*/

**#include** <ti/drivers/GPIO.h>

**#include** <ti/drivers/I2C.h>

/\* Example/Board Header files \*/

**#include** "Board.h"

**#define** Si7021\_TMP\_REG 0xE3

**#define** Si7021\_HUM\_REG 0xE5

**#define** Si7021\_ADDR 0x40

/\*

\* ======== HIGH\_TEMP ========

\* Send alert when this temperature (in Celsius) is exceeded

\*/

**#define** HIGH\_TEMP 30

/\*

\* ======== TMP Registers ========

\*/

**#define** TMP006\_REG 0x0001 /\* Die Temp Result Register for TMP006 \*/

**#define** TMP116\_REG 0x0000 /\* Die Temp Result Register for TMP116 \*/

/\*

\* The CC32XX LaunchPads come with an on-board TMP006 or TMP116 temperature

\* sensor depending on the revision. Newer revisions come with the TMP116.

\* The Build Automation Sensors (BOOSTXL-BASSENSORS) BoosterPack

\* contains a TMP116.

\*

\* We are using the DIE temperature because it's cool!

\*

\* Additionally: no calibration is being done on the TMPxxx device to simplify

\* the example code.

\*/

**#define** TMP006\_ADDR 0x41;

**#define** TMP116\_BP\_ADDR 0x48;

**#define** TMP116\_LP\_ADDR 0x49;

/\* Temperature written by the temperature thread and read by console thread \*/

**volatile** **float** temperatureC;

**volatile** **float** temperatureF;

**volatile** **float** temperaturef;

**volatile** **float** temperature;

**volatile** **float** temp;

**volatile** **float** sample;

Display\_Handle display;

/\* Mutex to protect the reading/writing of the temperature variables \*/

**extern** pthread\_mutex\_t temperatureMutex;

/\*

\* ======== postSem ========

\* Function called when the timer (created in setupTimer) expires.

\*/

**static** **void** **postSem**(**union** sigval val)

{

sem\_t \*sem = (sem\_t\*)(val.sival\_ptr);

**sem\_post**(sem);

}

/\*

\* ======== setupTimer ========

\* Create a timer that will expire at the period specified by the

\* time arguments. When the timer expires, the passed in semaphore

\* will be posted by the postSem function.

\*

\* A non-zero return indicates a failure.

\*/

**int** **setupTimer**(sem\_t \*sem, timer\_t \*timerid, time\_t sec, **long** nsec)

{

**struct** sigevent sev;

**struct** itimerspec its;

**int** retc;

retc = **sem\_init**(sem, 0, 0);

**if** (retc != 0) {

**return**(retc);

}

/\* Create the timer that wakes up the thread that will pend on the sem. \*/

sev.sigev\_notify = SIGEV\_SIGNAL;

sev.sigev\_value.sival\_ptr = sem;

sev.sigev\_notify\_function = &postSem;

sev.sigev\_notify\_attributes = NULL;

retc = **timer\_create**(CLOCK\_MONOTONIC, &sev, timerid);

**if** (retc != 0) {

**return**(retc);

}

/\* Set the timer to go off at the specified period \*/

its.it\_interval.tv\_sec = sec;

its.it\_interval.tv\_nsec = nsec;

its.it\_value.tv\_sec = sec;

its.it\_value.tv\_nsec = nsec;

retc = **timer\_settime**(\*timerid, 0, &its, NULL);

**if** (retc != 0) {

**timer\_delete**(\*timerid);

**return**(retc);

}

**return**(0);

}

/\*

\* ======== temperatureThread ========

\* This thread reads the temperature every second via I2C and sends an

\* alert if it goes above HIGH\_TEMP.

\*/

**void** \***temperatureThread**(**void** \*arg0)

{

uint8\_t txBuffer[1];

uint8\_t rxBuffer[2];

I2C\_Handle i2c;

I2C\_Params i2cParams;

I2C\_Transaction i2cTransaction;

sem\_t semTimer;

/\* Configure the LED and if applicable, the TMP116\_EN pin \*/

**GPIO\_setConfig**(Board\_GPIO\_LED0, GPIO\_CFG\_OUT\_STD | GPIO\_CFG\_OUT\_LOW);

**#ifdef** Board\_GPIO\_TMP116\_EN

**GPIO\_setConfig**(Board\_GPIO\_TMP116\_EN, GPIO\_CFG\_OUT\_STD | GPIO\_CFG\_OUT\_HIGH);

/\* 1.5 ms reset time for the TMP116 \*/

**sleep**(1);

**#endif**

/\*

\* Create/Open the I2C that talks to the TMP sensor

\*/

**I2C\_init**();

Display\_init();

**I2C\_Params\_init**(&i2cParams);

i2cParams.bitRate = *I2C\_400kHz*;

i2c = **I2C\_open**(Board\_I2C\_TMP, &i2cParams);

**if** (i2c == NULL) {

**while** (1);

}

/\* Common I2C transaction setup \*/

i2cTransaction.writeBuf = txBuffer;

i2cTransaction.writeCount = 1;

i2cTransaction.readBuf = rxBuffer;

i2cTransaction.readCount = 2;

// Try Si7021

txBuffer[0] = Si7021\_TMP\_REG;

i2cTransaction.slaveAddress = Si7021\_ADDR;

**if** (!**I2C\_transfer**(i2c, &i2cTransaction))

{

// Could not resolve a sensor, error

Display\_printf(display, 0, 0, "Error. No TMP sensor found!");

**while**(1);

}

**else**

{

Display\_printf(display, 0, 0, "Detected Si7021 sensor.");

}

// Take 20 samples and print them out onto the console

**for** (sample = 0; sample < 100; sample++)

{

**if** (**I2C\_transfer**(i2c, &i2cTransaction))

{

//

// Extract degrees C from the received data;

// see Si7021 datasheet

//

temp = (rxBuffer[0] << 8) | (rxBuffer[1]);

temperature = (((175.72 \* temp)/ 65536) - 46.85); // celsius

temperaturef = (temperature \* (1.8)) + 32; //farenheit

Display\_printf(display, 0, 0, "Sample %u: %d (C)", sample, temperaturef);

}

**else**

{

Display\_printf(display, 0, 0, "I2C Bus fault.");

}

}

}

For console.c:

I just changed this portion: to a lowercase temperaturef

Help commands pressing h current temperature pressing t

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

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

**Task 02 Part 2:**

Youtube Video: https://youtu.be/9v2Z12C3djA

Terminal output:A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

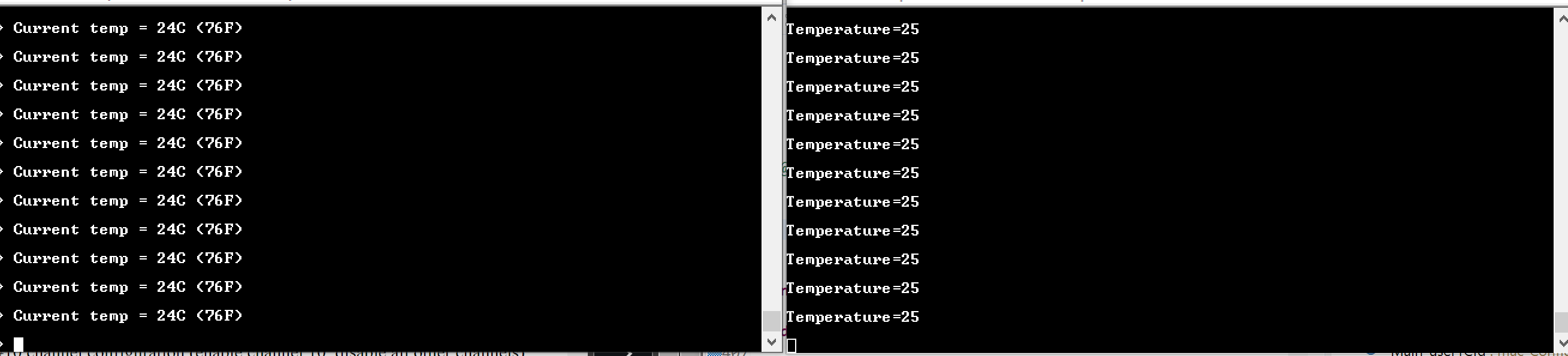
Description automatically generated

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

**Task 03 Part 2:**

Youtube link: https://youtu.be/jo0EB3dm71s

SENSOR COLLECTOR



In the video, the sensor screen is on the left and the collector screen is on the right. When I press ‘t’, the collector will receive the temperature. I added the changes the lab document told us about so I don’t think I need to reiterate the changes here.