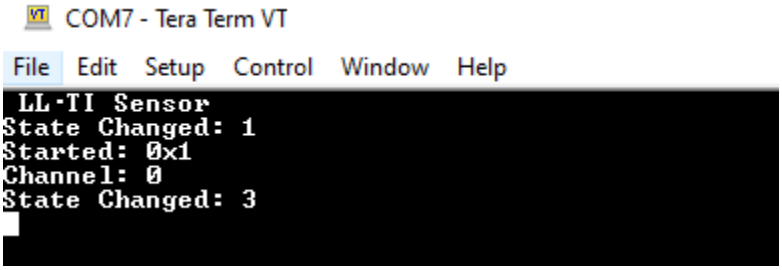
John Patrick Buen
Github root directory: https://github.com/buenj1/submission_labs

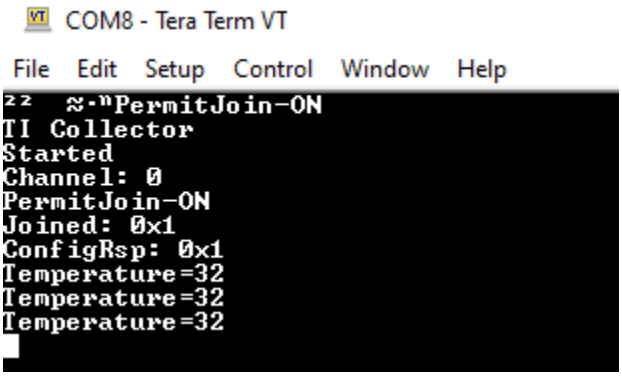
Date Submitted: **12/8/2019**

Youtube Link: https://www.youtube.com/watch?v=Bq_x91CdNGs

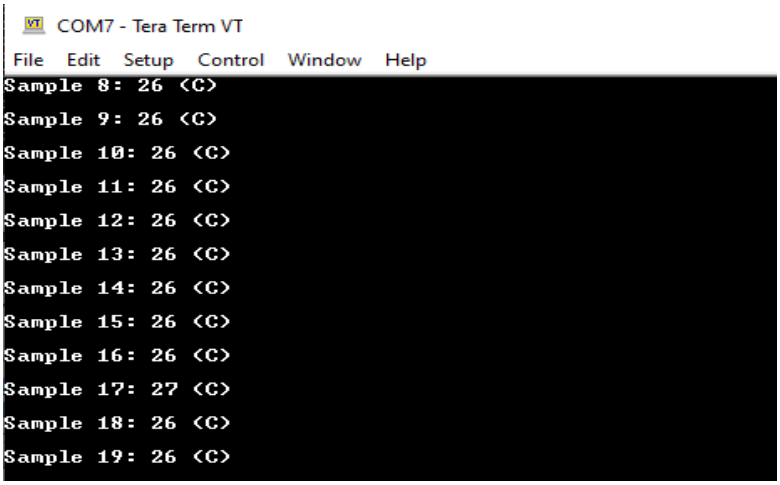
Schematics/Images:

- Sensor (COM 7) and Collector (COM 8) Initialization on Tera Term – Task 3 on **TI 15.4 Stack Project Zero**:





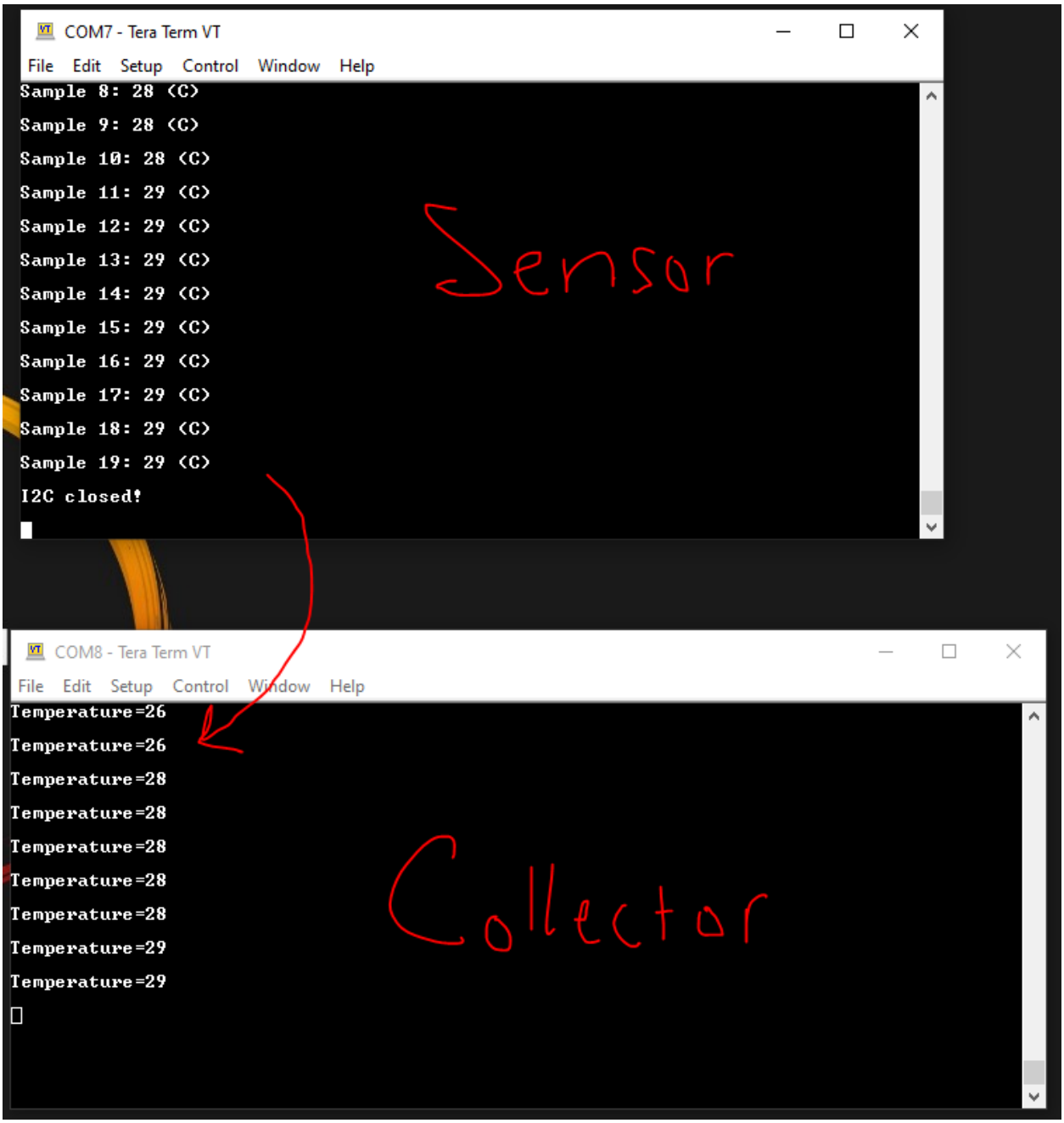
- Sensor Portable App Temperature Samples (x20) – Task 1 on **Using Stack and Portable App to Create a Remote Sensor** section:



Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

John Patrick Buen
Github root directory: https://github.com/buenj1/submission_labs

- Using Stack to Send Sampled Temperature Values from Sensor to Collector – Task 3 on **Using Stack and Portable App to Create a Remote Sensor** section:



Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

---

temperature.c (from sensor) :

```
/*
 * Copyright (c) 2018-2019, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/*
 *  ======== i2ctmp116.c ========
 */
#include <stdint.h>
#include <stddef.h>
#include <unistd.h>
#include "smsgs.h"
#include "mac_util.h"
#include "api_mac.h"
#include "sensor.h"
extern Smsgs_tempSensorField_t tempSensor;

/* Driver Header files */
#include <ti/drivers/GPIO.h>
#include <ti/drivers/I2C.h>
#include <ti/display/Display.h>

/* Example/Board Header files */
#include "Board.h"
```

```
#define TASKSTACKSIZE       640

/*
 *  ======== TMP Registers ========
 */
#define Si7021_TMP_REG          0xE3
#define Si7021_HUM_REG          0xE5

#define Si7021_ADDR         0x40;



static Display_Handle display;

/*
 *  ======== mainThread ========
 */
void *mainThread(void *arg0)
{
    uint16_t        sample;
    uint16_t        temperature,temperaturef;
    uint8_t         txBuffer[1];
    uint8_t         rxBuffer[2];
    I2C_Handle      i2c;
    I2C_Params      i2cParams;
    I2C_Transaction i2cTransaction;

    /* Call driver init functions */
    Display_init();
    GPIO_init();
    I2C_init();

    /* 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

    /* Open the HOST display for output */
    display = Display_open(Display_Type_UART, NULL);
    if (display == NULL) {
        while (1);
    }

    /* Turn on user LED */
    GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_ON);
    Display_printf(display, 0, 0, "Starting the i2ctmp example.");

    /* Create I2C for usage */
    I2C_Params_init(&i2cParams);
    i2cParams.bitRate = I2C_400kHz;
    i2c = I2C_open(Board_I2C_TMP, &i2cParams);
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
if (i2c == NULL) {
    Display_printf(display, 0, 0, "Error Initializing I2C\n");
    while (1);
}
else {
    Display_printf(display, 0, 0, "I2C Initialized!\n");
}

/* 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 < 20; sample++) {
    if (I2C_transfer(i2c, &i2cTransaction)) {
        /*
         * Extract degrees C from the received data;
         * see Si7021 datasheet
         */
        temperature = (rxBuffer[0] << 8) | (rxBuffer[1]);
        temperaturef = (((175.72 * temperature)/ 65536) - 46.85);

        Display_printf(display, 0, 0, "Sample %u: %d (C)",
                       sample, temperaturef);
    }
    else {
        Display_printf(display, 0, 0, "I2C Bus fault.");
    }

    tempSensor.objectTemp = temperaturef;
    tempSensor.ambienceTemp = temperaturef;
    Util_setEvent(&Sensor_events, EXT_SENSOR_READING_TIMEOUT_EVT);

    /* Sleep for 1 second */
    sleep(1);
}

I2C_close(i2c);
Display_printf(display, 0, 0, "I2C closed!");

return (NULL);
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

```
}
```

Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.