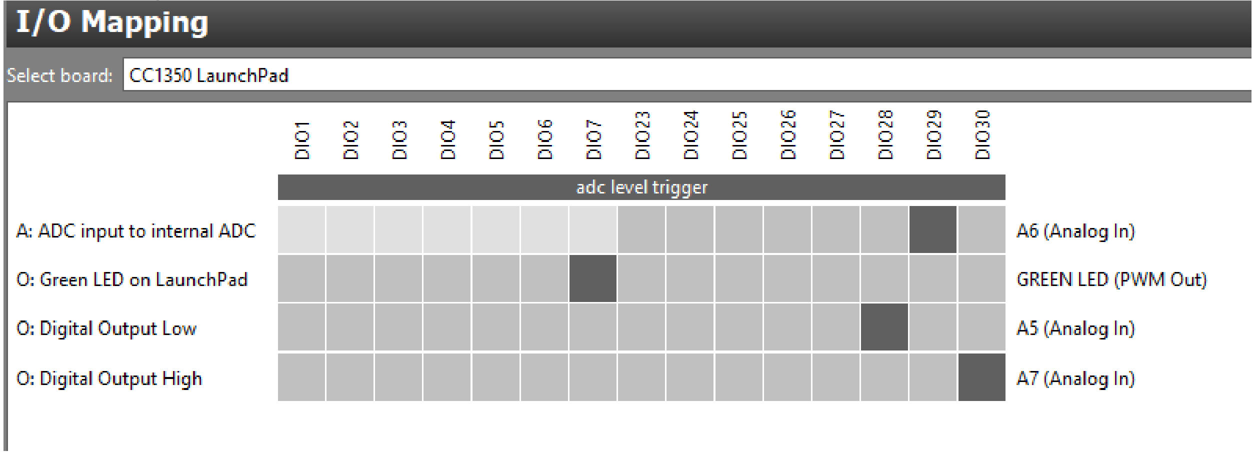
Serak Gebremedhin
Github root directory: https://github.com/Ber-geb/solid-octo-tribble.git

**Date Submitted:** 12/11/19

**Youtube Links:**

**Task 3:** https://youtu.be/pyQA2x4FFvw
**Task 4:** https://youtu.be/I1NA1IZTEp4
**Task 5:** https://youtu.be/xfUzoXcFzO0

**Images:**
**I/O Mapping from SCS**



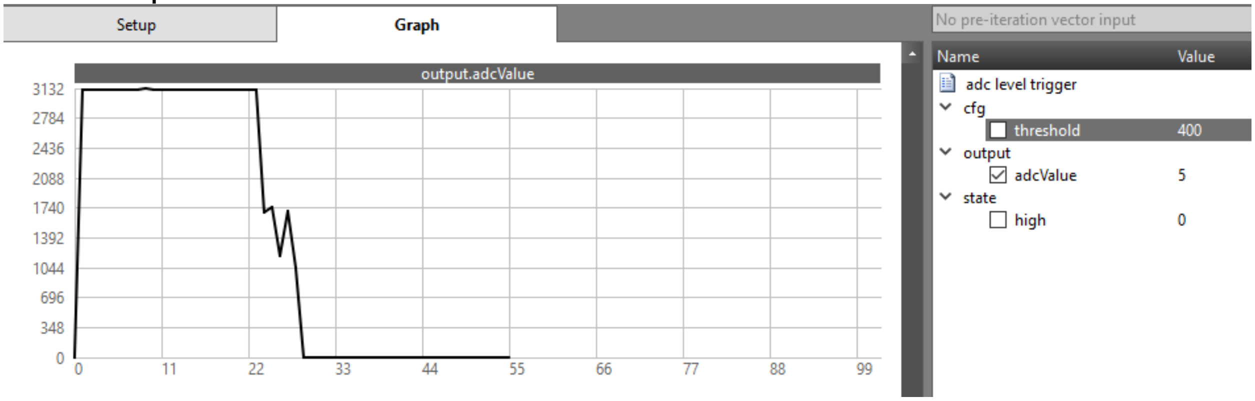
## I/O Mapping

Select board: CC1350 LaunchPad

adc level trigger

| | DIO1 | DIO2 | DIO3 | DIO4 | DIO5 | DIO6 | DIO7 | DIO23 | DIO24 | DIO25 | DIO26 | DIO27 | DIO28 | DIO29 | DIO30 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A: ADC input to internal ADC | | | | | | | | | | | | | | ■ | | A6 (Analog In) |
| O: Green LED on LaunchPad | | | | | | | ■ | | | | | | | | | GREEN LED (PWM Out) |
| O: Digital Output Low | | | | | | | | | | | | | ■ | | | A5 (Analog In) |
| O: Digital Output High | | | | | | | | | | | | | | | ■ | A7 (Analog In) |

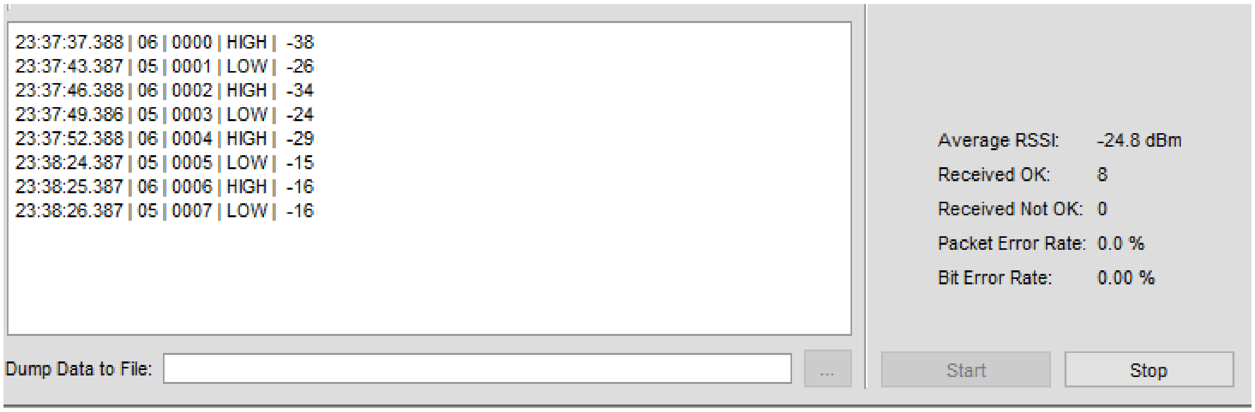
**Task 3 Graph**



No pre-iteration vector input

| Name | Value |
|---|---|
| adc level trigger | |
| cfg | |
| □ threshold | 400 |
| output | |
| ☑ adcValue | 5 |
| state | |
| □ high | 0 |

**Task 5 Packets received from transmitter**



```
23:37:37.388 | 06 | 0000 | HIGH |  -38
23:37:43.387 | 05 | 0001 | LOW |  -26
23:37:46.388 | 06 | 0002 | HIGH |  -34
23:37:49.386 | 05 | 0003 | LOW |  -24
23:37:52.388 | 06 | 0004 | HIGH |  -29
23:38:24.387 | 05 | 0005 | LOW |  -15
23:38:25.387 | 06 | 0006 | HIGH |  -16
23:38:26.387 | 05 | 0007 | LOW |  -16
```

Average RSSI: -24.8 dBm
Received OK: 8
Received Not OK: 0
Packet Error Rate: 0.0 %
Bit Error Rate: 0.00 %

Dump Data to File:

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

**Empty.c:**

```c
/*
 * Copyright (c) 2015-2019, 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.
 */

/*
 *  ======== empty.c ========
 */

/*Include Semaphores*/
#include <ti/sysbios/knl/Semaphore.h>
#include <ti/sysbios/BIOS.h>

// Main loop Semaphore
Semaphore_Struct semMainLoop;
Semaphore_Handle hSemMainLoop;


/*scif header and macro*/
#include "scif.h"
#define BV(x)   (1 << (x))

/* For usleep() */
#include <unistd.h>
#include <stdint.h>
```

```c
#include <stddef.h>

/* Driver Header files */
#include <ti/drivers/GPIO.h>
// #include <ti/drivers/I2C.h>
// #include <ti/drivers/SPI.h>
// #include <ti/drivers/UART.h>
// #include <ti/drivers/Watchdog.h>

/* Board Header file */
#include "Board.h"

/*SC Task Alert Handling*/
void processTaskAlert(void) {
    // Clear the ALERT interrupt source
    scifClearAlertIntSource();

    //Do SC Task processing here
    // Fetch 'state.high' variable from SC
    uint8_t high = scifTaskData.adcLevelTrigger.state.high;

    // Set Red LED state equal to the state.high variable
    GPIO_write(Board_GPIO_RLED, high);

    //Acknowledge the ALERT event
    scifAckAlertEvents();
}


/*SC callback functions*/
void scCtrlReadyCallback(void) {

} // scCtrlReadyCallback

void scTaskAlertCallback(void) {
    //Post to main loop semaphore
    Semaphore_post(hSemMainLoop);

} // scTaskAlertCallback


/*
 *  ======== mainThread ========
 */
void *tirtosScThread(void *arg0)
{
    // Semaphore initialization
    Semaphore_Params semParams;
    Semaphore_Params_init(&semParams);
    Semaphore_construct(&semMainLoop, 0, &semParams);
    hSemMainLoop = Semaphore_handle(&semMainLoop);

    /* 1 second delay */
    //uint32_t time = 1;
    /* Call driver init functions */
    GPIO_init();
    // I2C_init();
    // SPI_init();
    // UART_init();
    // Watchdog_init();

    /* Configure the LED pin */
    GPIO_setConfig(Board_GPIO_LED0, GPIO_CFG_OUT_STD | GPIO_CFG_OUT_LOW);

    /* Turn on user LED */
    GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_ON);

    /*SC Driver Initialization*/
    // Initialize the Sensor Controller
    scifOsalInit(); //init OSAL of the scif framework
    scifOsalRegisterCtrlReadyCallback(scCtrlReadyCallback); //init CTRL ready
callback
    scifOsalRegisterTaskAlertCallback(scTaskAlertCallback); //init Task Alert
callback
    scifInit(&scifDriverSetup); //init SC task driver

    // Set the Sensor Controller task tick interval to 1 second
    uint32_t rtc_Hz = 1; // 1Hz RTC
    scifStartRtcTicksNow(0x00010000 / rtc_Hz);
    //bits 31:16 represent the seconds, bits 15:0 represent 1/65536 of a second


    // Configure Sensor Controller tasks
    scifTaskData.adcLevelTrigger.cfg.threshold = 600; //set threshold value

    //Start Sensor Controller task
    scifStartTasksNbl(BV(SCIF_ADC_LEVEL_TRIGGER_TASK_ID)); //execute task


    while (1) {
        // Wait on sem indefinitely
        Semaphore_pend(hSemMainLoop, BIOS_WAIT_FOREVER);

        // Call process function
        processTaskAlert();
    }
}


-----------------------------------------------------------------------------------
```

## rfPacketTx.c:

```
/*
 * Copyright (c) 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.
 */

/***** Includes *****/

/*String and Semaphore Libraries*/
#include <string.h>   // strlen() and memcpy()
#include <ti/sysbios/knl/Semaphore.h>
#include <ti/sysbios/BIOS.h>
#include "scif.h"

#define BV(x)    (1 << (x))

// Semaphore variables - struct and handle
Semaphore_Struct semMainLoop;
Semaphore_Handle hSemMainLoop;


/* Standard C Libraries */
#include <stdlib.h>
#include <unistd.h>

/* TI Drivers */
#include <ti/drivers/rf/RF.h>
#include <ti/drivers/PIN.h>
#include <ti/drivers/pin/PINCC26XX.h>
#include <ti/drivers/GPIO.h>

/* Driverlib Header files */
#include DeviceFamily_constructPath(driverlib/rf_prop_mailbox.h)

/* Board Header files */
```

```
#include "Board.h"
#include "smartrf_settings/smartrf_settings.h"

/***** Defines *****/

/* Do power measurement */
//#define POWER_MEASUREMENT

/* Packet TX Configuration */
#define PAYLOAD_LENGTH      30
#ifdef POWER_MEASUREMENT
#define PACKET_INTERVAL     5  /* For power measurement set packet interval to 5s */
#else
#define PACKET_INTERVAL     500000  /* Set packet interval to 500000us or 500ms */
#endif

/***** Prototypes *****/

/***** Variable declarations *****/
static RF_Object rfObject;
static RF_Handle rfHandle;

/* Pin driver handle */
static PIN_Handle ledPinHandle;
static PIN_State ledPinState;

static uint8_t packet[PAYLOAD_LENGTH];
static uint16_t seqNumber;

/*
 * Application LED pin configuration table:
 *   - All LEDs board LEDs are off.
 */

//Board_LED1 is controlled by SC
PIN_Config pinTable[] =
{
    Board_PIN_LED0 | PIN_GPIO_OUTPUT_EN | PIN_GPIO_LOW | PIN_PUSHPULL |
PIN_DRVSTR_MAX,
#ifdef POWER_MEASUREMENT
#if defined(Board_CC1350_LAUNCHXL)
    Board_DIO30_SWPWR | PIN_GPIO_OUTPUT_EN | PIN_GPIO_HIGH | PIN_PUSHPULL |
PIN_DRVSTR_MAX,
#endif
#endif
    PIN_TERMINATE
};

/***** Function definitions *****/

/*SC Callbacks*/
void scCtrlReadyCallback(void) {
    // Do nothing
} // scCtrlReadyCallback
```

```
void scTaskAlertCallback(void) {
    // Signal main loop
    Semaphore_post(hSemMainLoop);
} // scTaskAlertCallback



void *txTaskFunction(void *arg0)
{
    RF_Params rfParams;
    RF_Params_init(&rfParams);

    /* Open LED pins */
    ledPinHandle = PIN_open(&ledPinState, pinTable);
    if (ledPinHandle == NULL)
    {
        while(1);
    }

#ifdef POWER_MEASUREMENT
#if defined(Board_CC1350_LAUNCHXL)
    /* Route out PA active pin to Board_DIO30_SWPWR */
    PINCC26XX_setMux(ledPinHandle, Board_DIO30_SWPWR, PINCC26XX_MUX_RFC_GPO1);
#endif
#endif

    RF_cmdPropTx.pktLen = PAYLOAD_LENGTH;
    RF_cmdPropTx.pPkt = packet;
    RF_cmdPropTx.startTrigger.triggerType = TRIG_NOW;

    /* Request access to the radio */
#if defined(DeviceFamily_CC26X0R2)
    rfHandle = RF_open(&rfObject, &RF_prop, (RF_RadioSetup*)&RF_cmdPropRadioSetup,
&rfParams);
#else
    rfHandle = RF_open(&rfObject, &RF_prop, (RF_RadioSetup*)&RF_cmdPropRadioDivSetup,
&rfParams);
#endif// DeviceFamily_CC26X0R2

    /* Set the frequency */
    RF_postCmd(rfHandle, (RF_Op*)&RF_cmdFs, RF_PriorityNormal, NULL, 0);



    // Main Loop Semaphore initialization
    Semaphore_Params semParams;
    Semaphore_Params_init(&semParams);
    semParams.mode = Semaphore_Mode_BINARY;
    Semaphore_construct(&semMainLoop, 0, &semParams);
    hSemMainLoop = Semaphore_handle(&semMainLoop);

    GPIO_init();

    // Initialize the Sensor Controller
    scifOsalInit();
    scifOsalRegisterCtrlReadyCallback(scCtrlReadyCallback);
    scifOsalRegisterTaskAlertCallback(scTaskAlertCallback);
    scifInit(&scifDriverSetup);

    // Set the Sensor Controller task tick interval to 1 second
    uint32_t rtc_Hz = 1;  // 1Hz RTC
    scifStartRtcTicksNow(0x00010000 / rtc_Hz);

    // Configure Sensor Controller tasks
    scifTaskData.adcLevelTrigger.cfg.threshold = 600;

    // Start Sensor Controller task
    scifStartTasksNbl(BV(SCIF_ADC_LEVEL_TRIGGER_TASK_ID));


    // Main loop
    while(1) {
      // Wait for signal
        Semaphore_pend(hSemMainLoop, BIOS_WAIT_FOREVER);

      // Clear the ALERT interrupt source
        scifClearAlertIntSource();

      // Get 'state.high', and set highStr to appropriate string
        uint16_t high = scifTaskData.adcLevelTrigger.state.high;
        const char *highStr = (high != 0) ? "HIGH" : "LOW";
        uint16_t highStrLen = strlen(highStr);

      // Populate packet, and set pktlen
        packet[0] = (uint8_t)(seqNumber >> 8);
        packet[1] = (uint8_t)(seqNumber++);
        memcpy(packet + 2, highStr, highStrLen);
        RF_cmdPropTx.pktLen = 2 + highStrLen;

      // Send packet Tx
        RF_runCmd(rfHandle, (RF_Op*)&RF_cmdPropTx, RF_PriorityNormal, NULL, 0);

      // Toggle pin
        PIN_setOutputValue(ledPinHandle, Board_PIN_LED0, high);

      // Acknowledge the ALERT event
        scifAckAlertEvents();

    }
}

------------------------------------------------------------------------------------
```

**<u>Grading scheme:</u>** 30% Coding, 30% Documentation, 40% Execution/Video.

Serak Gebremedhin
Github root directory: https://github.com/Ber-geb/solid-octo-tribble.git

## Task 03:

Youtube Link:

**Modified Schematic (if applicable):**

**Modified Code:**

```
// Insert code here
---------------------------------------------------------------------------------
```

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