**Date Submitted: 12/4/19**

Youtube Links:

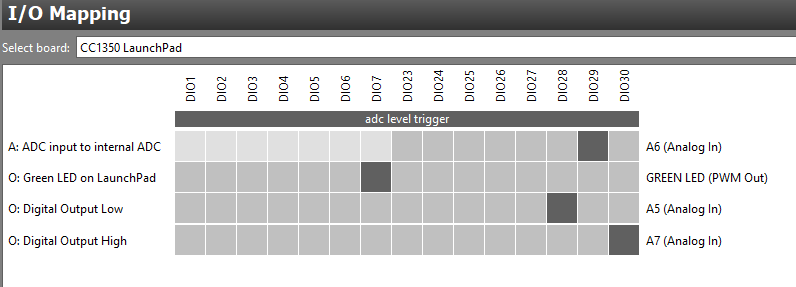
Task 3 - https://www.youtube.com/watch?v=3Q-srxBYgbQ

Task 4 - https://www.youtube.com/watch?v=6095Jg\_yvqM

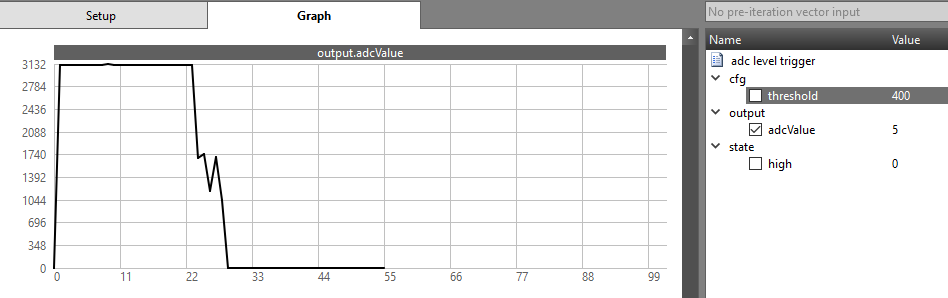
Task 5 - https://www.youtube.com/watch?v=HHGP1VU0s3I

Schematics/Images

Map of Assigned Pins from SCS

****

Task 3 – Graph of adcValue above/below the threshold



Task 5 – Packets received from transmitter to receiver



Code(s):

**Empty.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>

**#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();

}

}

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