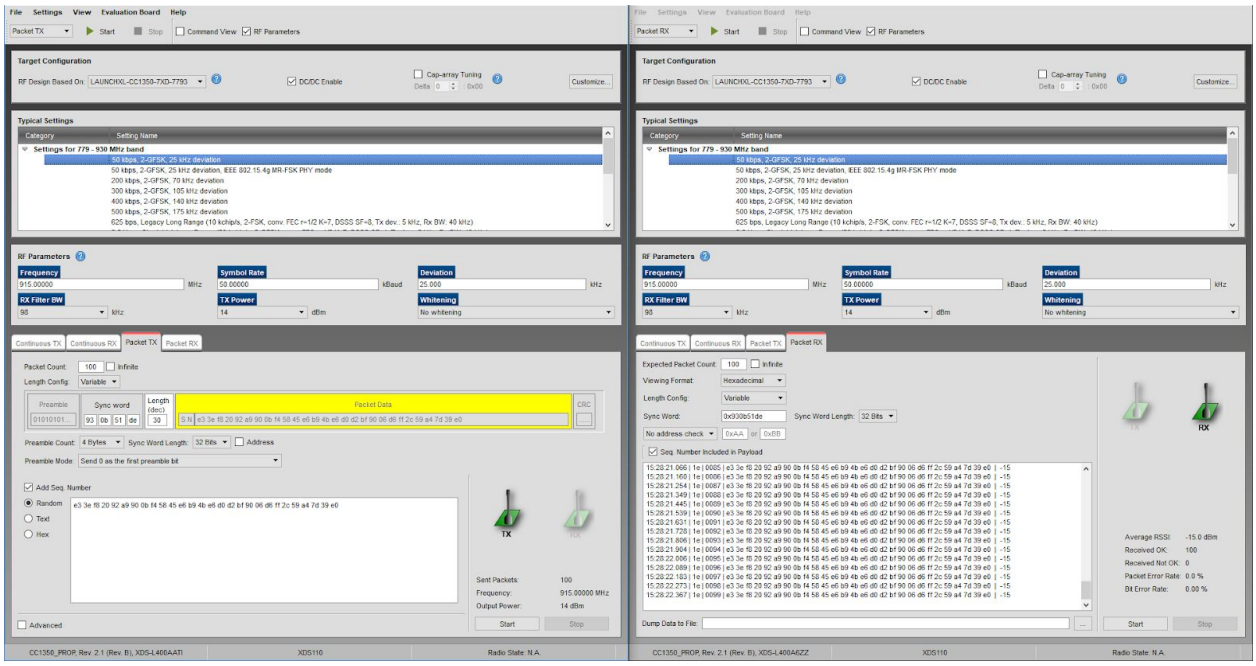
Joseph Sharp Halpin
CpE 403 Section 1001
Date Submitted: 11/16/2018

Youtube Link Task 1: https://youtu.be/Qz2ATVCz3cA
Youtube Link Task 2: https://youtu.be/k5fEy6Eaot4
Youtube Link Task 4: https://youtu.be/3gi4ATBNbFQ
Youtube Link Task 7: https://youtu.be/aieR4Kpqmjc

Task 1:
SmartRF Studio:



Task 2:
Code:

```
/* TI Drivers */
#include <ti/drivers/rf/RF.h>
#include <ti/drivers/PIN.h>
#include <ti/drivers/pin/PINCC26XX.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.
 */
PIN_Config pinTable[] =
```

```
{
    Board_PIN_LED1 | PIN_GPIO_OUTPUT_EN | PIN_GPIO_LOW |
PIN_PUSHPULL | PIN_DRVSTR_MAX,
#if defined Board_CC1352R1_LAUNCHXL
    Board_DIO30_RFSW | PIN_GPIO_OUTPUT_EN | PIN_GPIO_HIGH |
PIN_PUSHPULL | PIN_DRVSTR_MAX,
#endif
#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 *****/

void *mainThread(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 */
    rfHandle = RF_open(&rfObject, &RF_prop,
```

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

    while(1)
    {
        /* Create packet with incrementing sequence number and
random payload */
        packet[0] = (uint8_t)(seqNumber >> 8);
        packet[1] = (uint8_t)(seqNumber++);
        uint8_t i;
        for (i = 2; i < PAYLOAD_LENGTH; i++)
        {
            packet[i] = rand();
        }

        /* Send packet */
        RF_EventMask terminationReason = RF_runCmd(rfHandle,
(RF_Op*)&RF_cmdPropTx,

RF_PriorityNormal, NULL, 0);

        switch(terminationReason)
        {
            case RF_EventLastCmdDone:
                // A stand-alone radio operation command or the
last radio
                // operation command in a chain finished.
                break;
            case RF_EventCmdCancelled:
                // Command cancelled before it was started; it
can be caused
            // by RF_cancelCmd() or RF_flushCmd().
                break;
            case RF_EventCmdAborted:
                // Abrupt command termination caused by
RF_cancelCmd() or
                // RF_flushCmd().
                break;
            case RF_EventCmdStopped:
                // Graceful command termination caused by
RF_cancelCmd() or
                // RF_flushCmd().
                break;
            default:
                // Uncaught error event
                while(1);
        }
```

```
            while(1);
        }

        uint32_t cmdStatus = ((volatile RF_Op*)&RF_cmdPropTx)->
status;
        switch(cmdStatus)
        {
            case PROP_DONE_OK:
                // Packet transmitted successfully
                break;
            case PROP_DONE_STOPPED:
                // received CMD_STOP while transmitting packet
and finished
                // transmitting packet
                break;
            case PROP_DONE_ABORT:
                // Received CMD_ABORT while transmitting packet
                break;
            case PROP_ERROR_PAR:
                // Observed illegal parameter
                break;
            case PROP_ERROR_NO_SETUP:
                // Command sent without setting up the radio in
a supported
                // mode using CMD_PROP_RADIO_SETUP or
CMD_RADIO_SETUP
                break;
            case PROP_ERROR_NO_FS:
                // Command sent without the synthesizer being
programmed
                break;
            case PROP_ERROR_TXUNF:
                // TX underflow observed during operation
                break;
            default:
                // Uncaught error event - these could come from
the
                // pool of states defined in rf_mailbox.h
                while(1);
        }

#ifndef POWER_MEASUREMENT
        PIN_setOutputValue(ledPinHandle, Board_PIN_LED1,!
PIN_getOutputValue(Board_PIN_LED1));
#endif
        /* Power down the radio */
        RF_yield(rfHandle);
```

```
#ifndef POWER_MEASUREMENT
        PIN_setOutputValue(ledPinHandle, Board_PIN_LED1,!
PIN_getOutputValue(Board_PIN_LED1));
#endif
        /* Power down the radio */
        RF_yield(rfHandle);

#ifdef POWER_MEASUREMENT
        /* Sleep for PACKET_INTERVAL s */
        sleep(PACKET_INTERVAL);
#else
        /* Sleep for PACKET_INTERVAL us */
        usleep(PACKET_INTERVAL);
#endif

    }
}
```

Task 4: