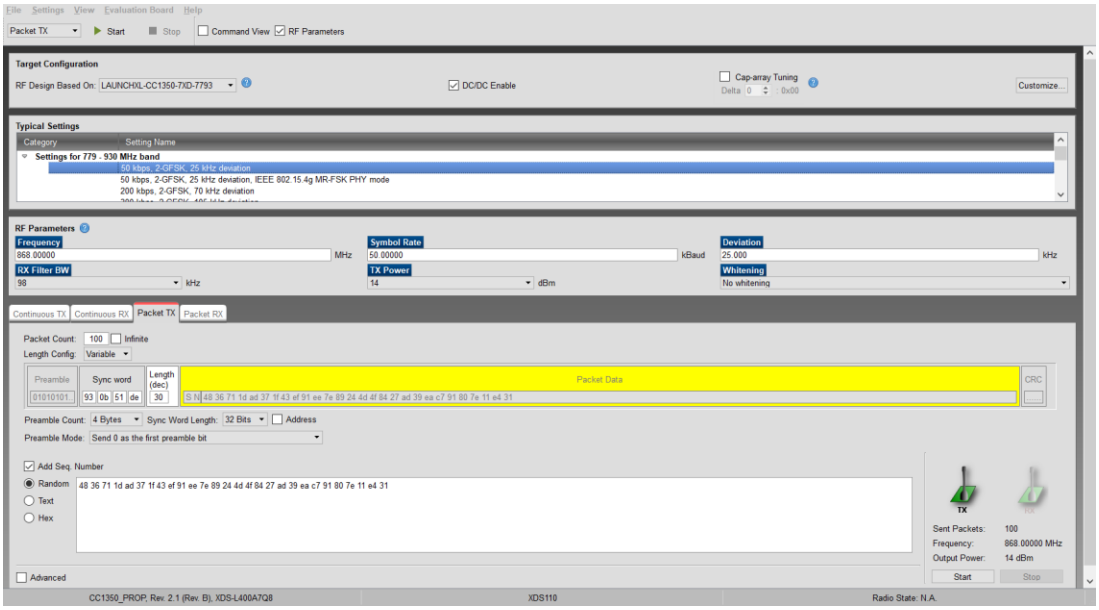
Name: Enrique Saldana
Partner: Damian Cisneros
Github root directory: https://github.com/enri10

Date Submitted: **11-20-2018**

Task 01: SmartRF Studio ↔ SmartRF Studio. Configure both launchpads in RFStudio one to transmit and the other to receive.

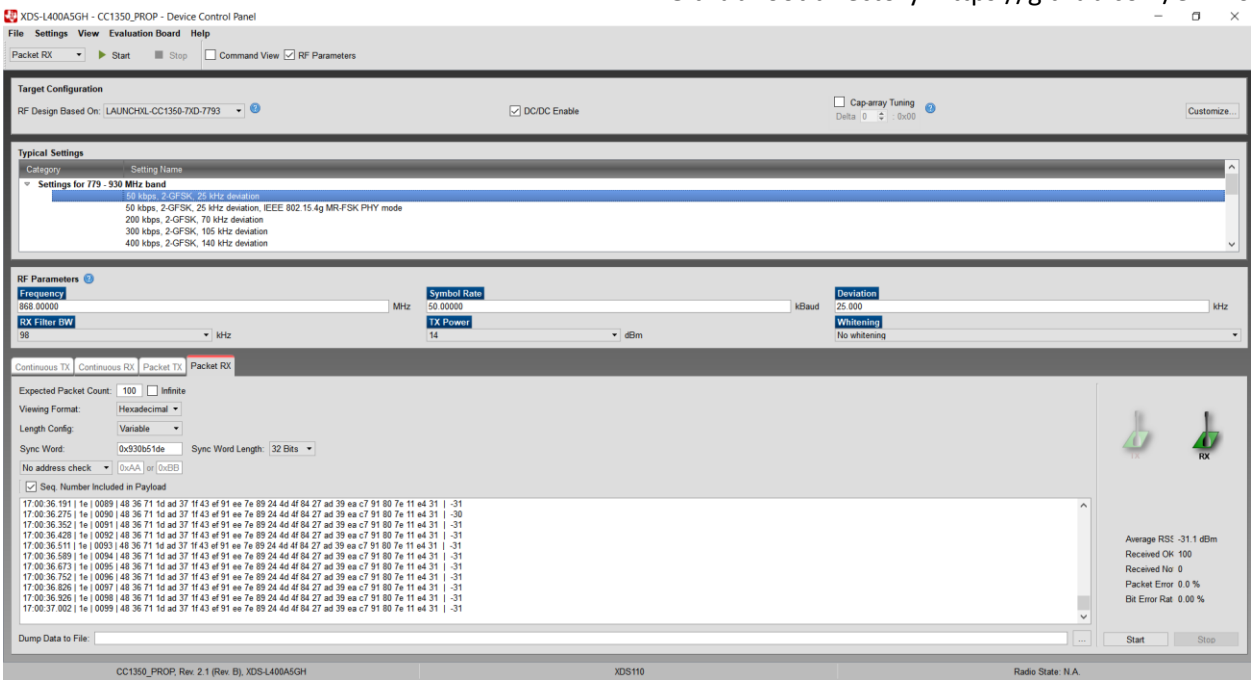
Transmitter side.



Transmitter packets sent. (100 of them)

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

Name: Enrique Saldana
Partner: Damian Cisneros
Github root directory: https://github.com/enri10



Receiver packets received (100)

Youtube Link: No video only screenshots of packets.

Modified Code: No coding necessary for this part.

Task 02: Import and run rfPacketTx example in Code Composer Studio.

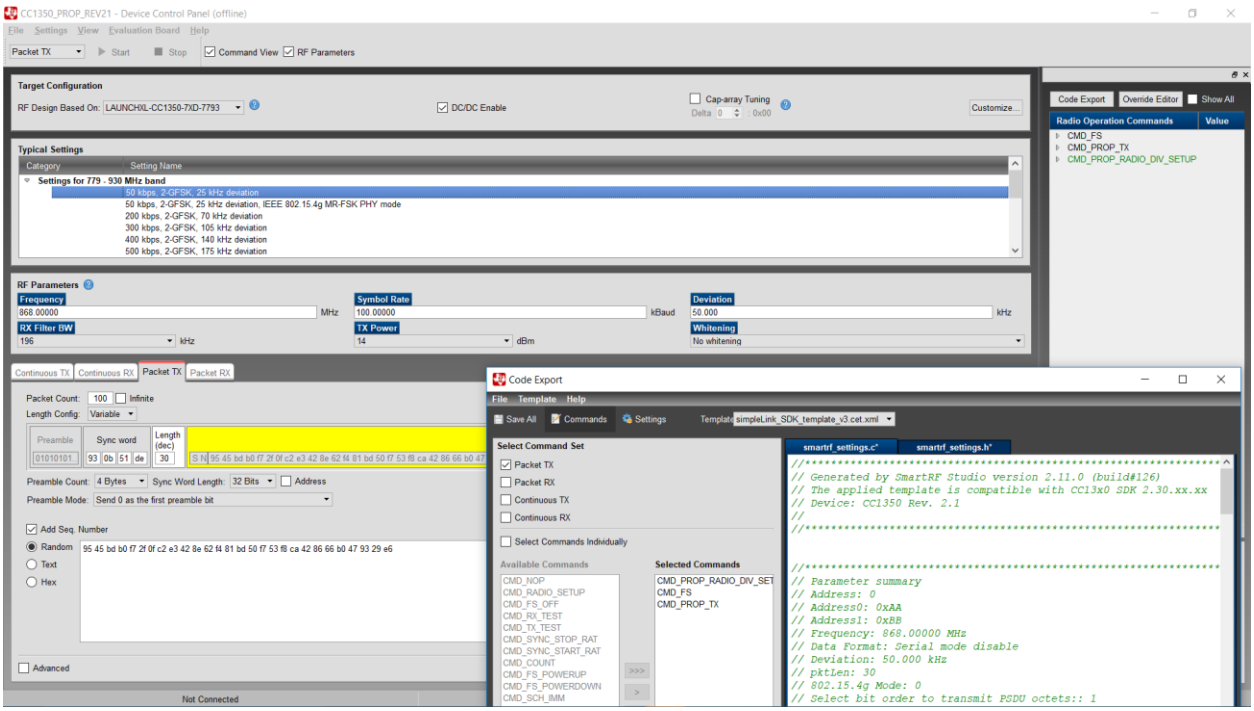
Youtube Link: https://youtu.be/ZMaipoOAAMs

Modified Code: rfPacketTx and rfPacketRx example codes were used with no changes.

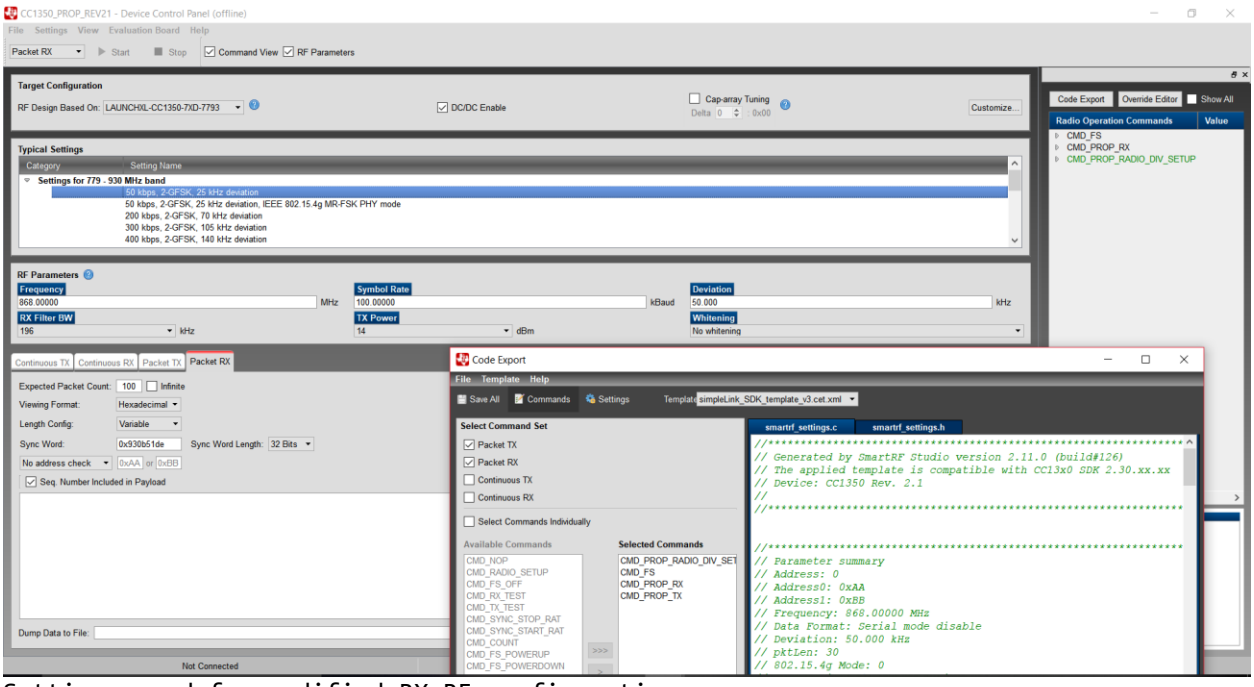
Grading scheme: 30% Coding, 30% Documentation, 40% Execution/Video.

Name: Enrique Saldana
Partner: Damian Cisneros
Github root directory: https://github.com/enri10

Task 03: Exporting and using RF configuration.



Settings used for modified TX RF configuration.



Settings used for modified RX RF configuration.

Youtube Link (TX): https://youtu.be/qsKIvCllREA

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

Name: Enrique Saldana
Partner: Damian Cisneros
Github root directory: https://github.com/enri10

Modified Code: rfPacketTx and rfPacketRx example codes were used with no changes. Newly Sensor Controller Studio generated smartrf_settings.c/.h were used for this.

smartrf_settings.h

```
#ifndef _SMARTRF_SETTINGS_H_
#define _SMARTRF_SETTINGS_H_

//*************************************************************************
******
// Generated by SmartRF Studio version 2.11.0 (build#126)
// The applied template is compatible with CC13x0 SDK 2.30.xx.xx
// Device: CC1350 Rev. 2.1
//
//*************************************************************************
******
#include <ti/devices/DeviceFamily.h>
#include DeviceFamily_constructPath(driverlib/rf_mailbox.h)
#include DeviceFamily_constructPath(driverlib/rf_common_cmd.h)
#include DeviceFamily_constructPath(driverlib/rf_prop_cmd.h)
#include <ti/drivers/rf/RF.h>


// TI-RTOS RF Mode Object
extern RF_Mode RF_prop;

// RF Core API commands
extern rfc_CMD_PROP_RADIO_DIV_SETUP_t RF_cmdPropRadioDivSetup;
extern rfc_CMD_FS_t RF_cmdFs;
extern rfc_CMD_PROP_TX_t RF_cmdPropTx;

// RF Core API Overrides
extern uint32_t pOverrides[];

#endif // _SMARTRF_SETTINGS_H_
```

smartrf_settings.c

```
//*************************************************************************
******
// Generated by SmartRF Studio version 2.11.0 (build#126)
// The applied template is compatible with CC13x0 SDK 2.30.xx.xx
// Device: CC1350 Rev. 2.1
//
//*************************************************************************
******


//*************************************************************************
******
// Parameter summary
// Address: 0
// Address0: 0xAA
// Address1: 0xBB
// Frequency: 868.00000 MHz
// Data Format: Serial mode disable
// Deviation: 50.000 kHz
// pktLen: 30
// 802.15.4g Mode: 0
```

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

Name: Enrique Saldana
Partner: Damian Cisneros
Github root directory: https://github.com/enri10

```
// Select bit order to transmit PSDU octets:: 1
// Packet Length Config: Variable
// Max Packet Length: 255
// Packet Length: 20
// Packet Data: 255
// RX Filter BW: 196 kHz
// Symbol Rate: 100.00000 kBaud
// Sync Word Length: 32 Bits
// TX Power: 14 dBm (requires define CCFG_FORCE_VDDR_HH = 1 in ccfg.c, see
CC13xx/CC26xx Technical Reference Manual)
// Whitening: No whitening

#include <ti/devices/DeviceFamily.h>
#include DeviceFamily_constructPath(driverlib/rf_mailbox.h)
#include DeviceFamily_constructPath(driverlib/rf_common_cmd.h)
#include DeviceFamily_constructPath(driverlib/rf_prop_cmd.h)
#include <ti/drivers/rf/RF.h>
#include DeviceFamily_constructPath(rf_patches/rf_patch_cpe_genfsk.h)
#include DeviceFamily_constructPath(rf_patches/rf_patch_rfe_genfsk.h)
#include "smartrf_settings.h"


// TI-RTOS RF Mode Object
RF_Mode RF_prop =
{
    .rfMode = RF_MODE_PROPRIETARY_SUB_1,
    .cpePatchFxn = &rf_patch_cpe_genfsk,
    .mcePatchFxn = 0,
    .rfePatchFxn = &rf_patch_rfe_genfsk,
};


// Overrides for CMD_PROP_RADIO_DIV_SETUP
uint32_t pOverrides[] =
{
    // override_use_patch_prop_genfsk.xml
    // PHY: Use MCE ROM bank 4, RFE RAM patch
    MCE_RFE_OVERRIDE(0,4,0,1,0,0),
    // override_synth_prop_863_930_div5.xml
    // Synth: Set recommended RTRIM to 7
    HW_REG_OVERRIDE(0x4038,0x0037),
    // Synth: Set Fref to 4 MHz
    (uint32_t)0x000684A3,
    // Synth: Configure fine calibration setting
    HW_REG_OVERRIDE(0x4020,0x7F00),
    // Synth: Configure fine calibration setting
    HW_REG_OVERRIDE(0x4064,0x0040),
    // Synth: Configure fine calibration setting
    (uint32_t)0xB1070503,
    // Synth: Configure fine calibration setting
    (uint32_t)0x05330523,
    // Synth: Set loop bandwidth after lock to 20 kHz
    (uint32_t)0x0A480583,
    // Synth: Set loop bandwidth after lock to 20 kHz
    (uint32_t)0x7AB80603,
```

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

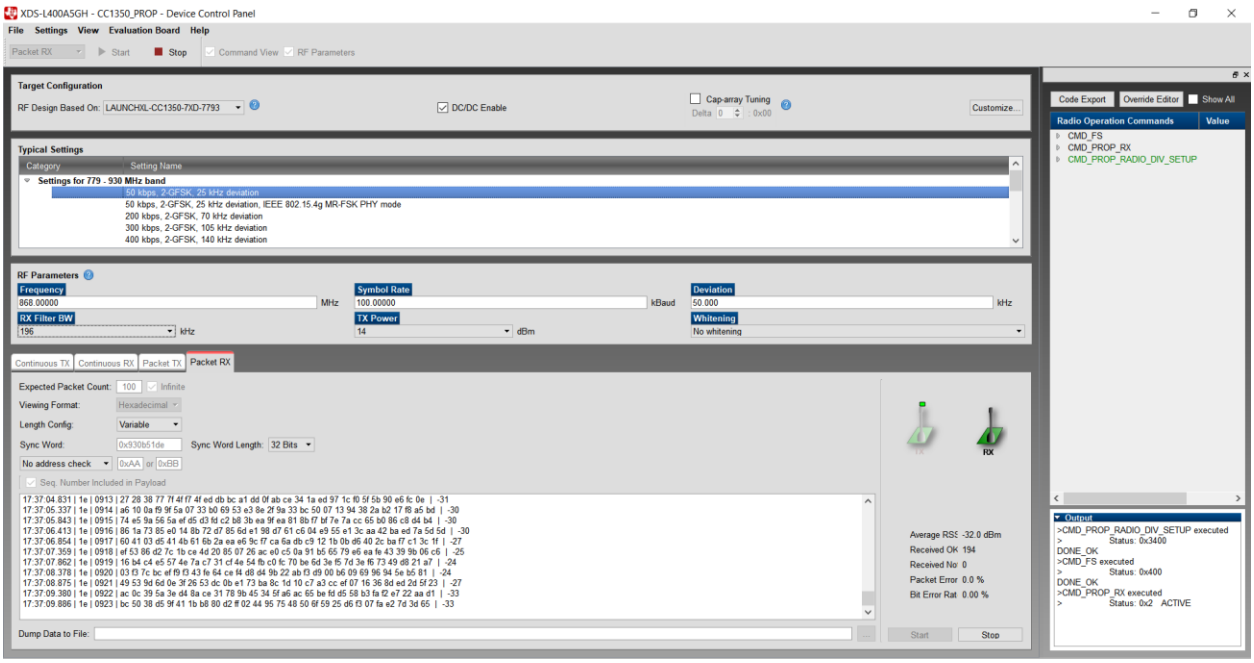
```
    // Synth: Configure VCO LDO (in ADI1, set VCOLDOCFG=0x9F to use voltage
input reference)
    ADI_REG_OVERRIDE(1,4,0x9F),
    // Synth: Configure synth LDO (in ADI1, set SLDOCTL0.COMP_CAP=1)
    ADI_HALFREG_OVERRIDE(1,7,0x4,0x4),
    // Synth: Use 24 MHz XOSC as synth clock, enable extra PLL filtering
    (uint32_t)0x02010403,
    // Synth: Configure extra PLL filtering
    (uint32_t)0x00108463,
    // Synth: Increase synth programming timeout (0x04B0 RAT ticks = 300 us)
    (uint32_t)0x04B00243,
    // override_phy_rx_aaf_bw_0xd.xml
    // Rx: Set anti-aliasing filter bandwidth to 0xD (in ADI0, set
IFAMPCTL3[7:4]=0xD)
    ADI_HALFREG_OVERRIDE(0,61,0xF,0xD),
    // override_phy_gfsk_rx.xml
    // Rx: Set LNA bias current trim offset to 3
    (uint32_t)0x00038883,
    // Rx: Freeze RSSI on sync found event
    HW_REG_OVERRIDE(0x6084,0x35F1),
    // override_phy_gfsk_pa_ramp_agc_reflevel_0x1a.xml
    // Tx: Configure PA ramping setting (0x41). Rx: Set AGC reference level
to 0x1A.
    HW_REG_OVERRIDE(0x6088,0x411A),
    // Tx: Configure PA ramping setting
    HW_REG_OVERRIDE(0x608C,0x8213),
    // override_phy_rx_rssi_offset_5db.xml
    // Rx: Set RSSI offset to adjust reported RSSI by +5 dB (default: 0),
trimmed for external bias and differential configuration
    (uint32_t)0x00FB88A3,
    // TX power override
    // Tx: Set PA trim to max (in ADI0, set PACTL0=0xF8)
    ADI_REG_OVERRIDE(0,12,0xF8),
    (uint32_t)0xFFFFFFFF
};


// CMD_PROP_RADIO_DIV_SETUP
// Proprietary Mode Radio Setup Command for All Frequency Bands
rfc_CMD_PROP_RADIO_DIV_SETUP_t RF_cmdPropRadioDivSetup =
{
    .commandNo = 0x3807,
    .status = 0x0000,
    .pNextOp = 0, // INSERT APPLICABLE POINTER: (uint8_t*)&xxx
    .startTime = 0x00000000,
    .startTrigger.triggerType = 0x0,
    .startTrigger.bEnaCmd = 0x0,
    .startTrigger.triggerNo = 0x0,
    .startTrigger.pastTrig = 0x0,
    .condition.rule = 0x1,
    .condition.nSkip = 0x0,
    .modulation.modType = 0x1,
    .modulation.deviation = 0xC8,
    .symbolRate.preScale = 0xF,
    .symbolRate.rateWord = 0x10000,
    .symbolRate.decimMode = 0x0,
```

```
    .rxBw = 0x27,
    .preamConf.nPreamBytes = 0x4,
    .preamConf.preamMode = 0x0,
    .formatConf.nSwBits = 0x20,
    .formatConf.bBitReversal = 0x0,
    .formatConf.bMsbFirst = 0x1,
    .formatConf.fecMode = 0x0,
    .formatConf.whitenMode = 0x0,
    .config.frontEndMode = 0x0,
    .config.biasMode = 0x1,
    .config.analogCfgMode = 0x0,
    .config.bNoFsPowerUp = 0x0,
    .txPower = 0xAB3F,
    .pRegOverride = pOverrides,
    .centerFreq = 0x0364,
    .intFreq = 0x8000,
    .loDivider = 0x05
};


// CMD_FS
// Frequency Synthesizer Programming Command
rfc_CMD_FS_t RF_cmdFs =
{
    .commandNo = 0x0803,
    .status = 0x0000,
    .pNextOp = 0, // INSERT APPLICABLE POINTER: (uint8_t*)&xxx
    .startTime = 0x00000000,
    .startTrigger.triggerType = 0x0,
    .startTrigger.bEnaCmd = 0x0,
    .startTrigger.triggerNo = 0x0,
    .startTrigger.pastTrig = 0x0,
    .condition.rule = 0x1,
    .condition.nSkip = 0x0,
    .frequency = 0x0364,
    .fractFreq = 0x0000,
    .synthConf.bTxMode = 0x0,
    .synthConf.refFreq = 0x0,
    .__dummy0 = 0x00,
    .__dummy1 = 0x00,
    .__dummy2 = 0x00,
    .__dummy3 = 0x0000
};


// CMD_PROP_TX
// Proprietary Mode Transmit Command
rfc_CMD_PROP_TX_t RF_cmdPropTx =
{
    .commandNo = 0x3801,
    .status = 0x0000,
    .pNextOp = 0, // INSERT APPLICABLE POINTER: (uint8_t*)&xxx
    .startTime = 0x00000000,
    .startTrigger.triggerType = 0x0,
    .startTrigger.bEnaCmd = 0x0,
    .startTrigger.triggerNo = 0x0,
```

Name: Enrique Saldana
Partner: Damian Cisneros
Github root directory: https://github.com/enri10

```
    .startTrigger.pastTrig = 0x0,
    .condition.rule = 0x1,
    .condition.nSkip = 0x0,
    .pktConf.bFsOff = 0x0,
    .pktConf.bUseCrc = 0x1,
    .pktConf.bVarLen = 0x1,
    .pktLen = 0x14, // SET APPLICATION PAYLOAD LENGTH
    .syncWord = 0x930B51DE,
    .pPkt = 0 // INSERT APPLICABLE POINTER: (uint8_t*)&xxx
};
```

Task 04: Firmware TX → SmartRF Studio RX



Youtube Link: https://youtu.be/B7pe81sgOtE

Modified Code: Example codes up to this point are unchanged besides the smartrf_settings.c/.h files.

Task 05: Importing and Modifying rfPacketRx

Youtube Link: https://youtu.be/xWNO9J8MCho

Modified Code: Original rfPacketRx example code was kept. Only the smartrf_settings.c/.h files have been modified.

Task 06: Firmware TX → Firmware RX

Youtube Link: https://youtu.be/yjUa38ov1pw

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

Name: Enrique Saldana
Partner: Damian Cisneros
Github root directory: https://github.com/enri10

Modified Code: Both TX and RX codes are the same as original packetTx/Rx example. Only the smartrf_settings.c/.h files have been modified.

Task 07: **SmartRF Studio TX → Firmware RX**

Rx board toggle red LED at the same rate that packets are sending from the SmartRF Studio Tx.

Youtube Link: https://youtu.be/ubLH6U5XBlA

Modified Code: Code for Firmware Rx up to this point has remain unchanged.