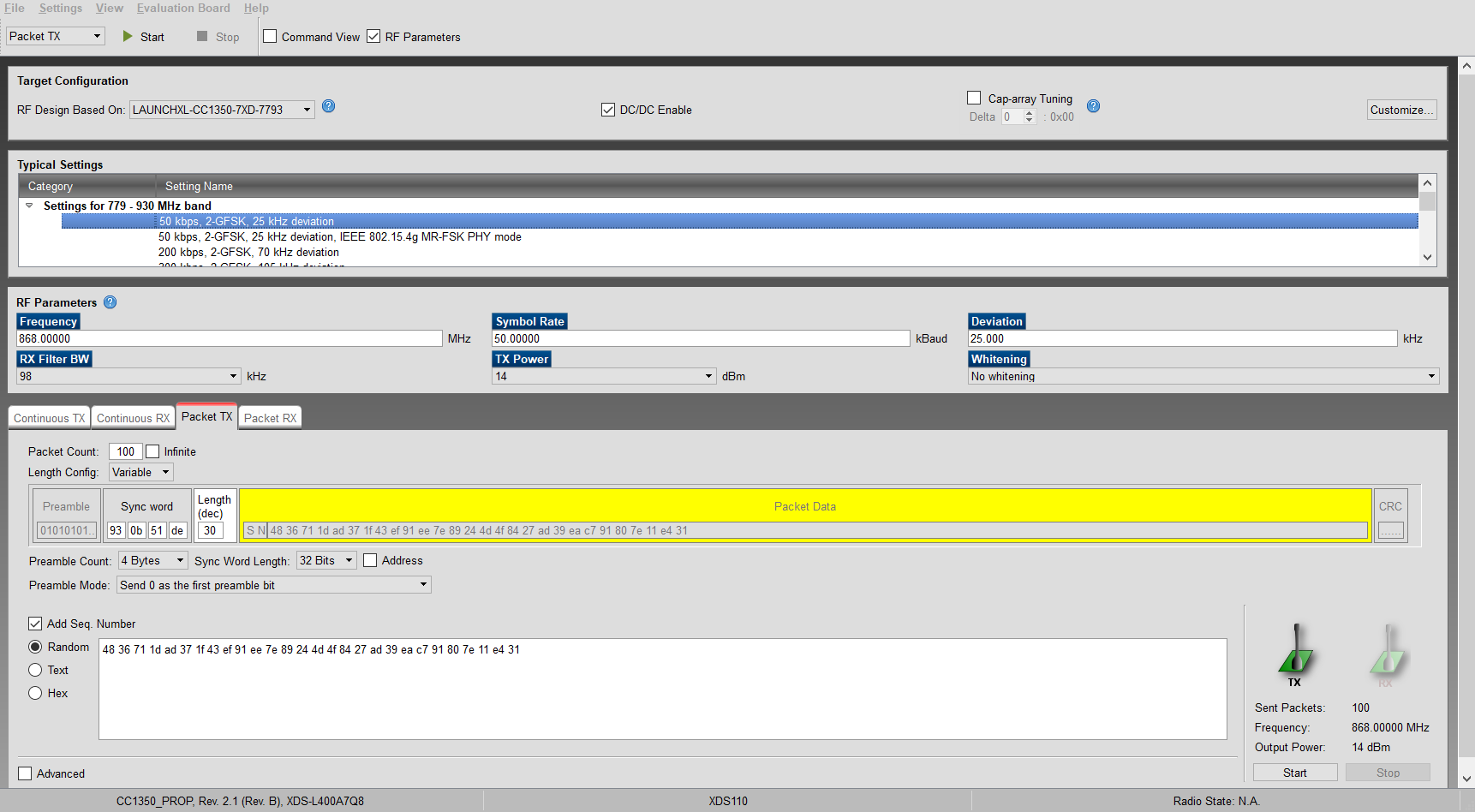
**Date Submitted: 11-20-2018**

**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)

Youtube Link: No video only screenshots of packets.

**Modified Code:** No coding necessary for this part.

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

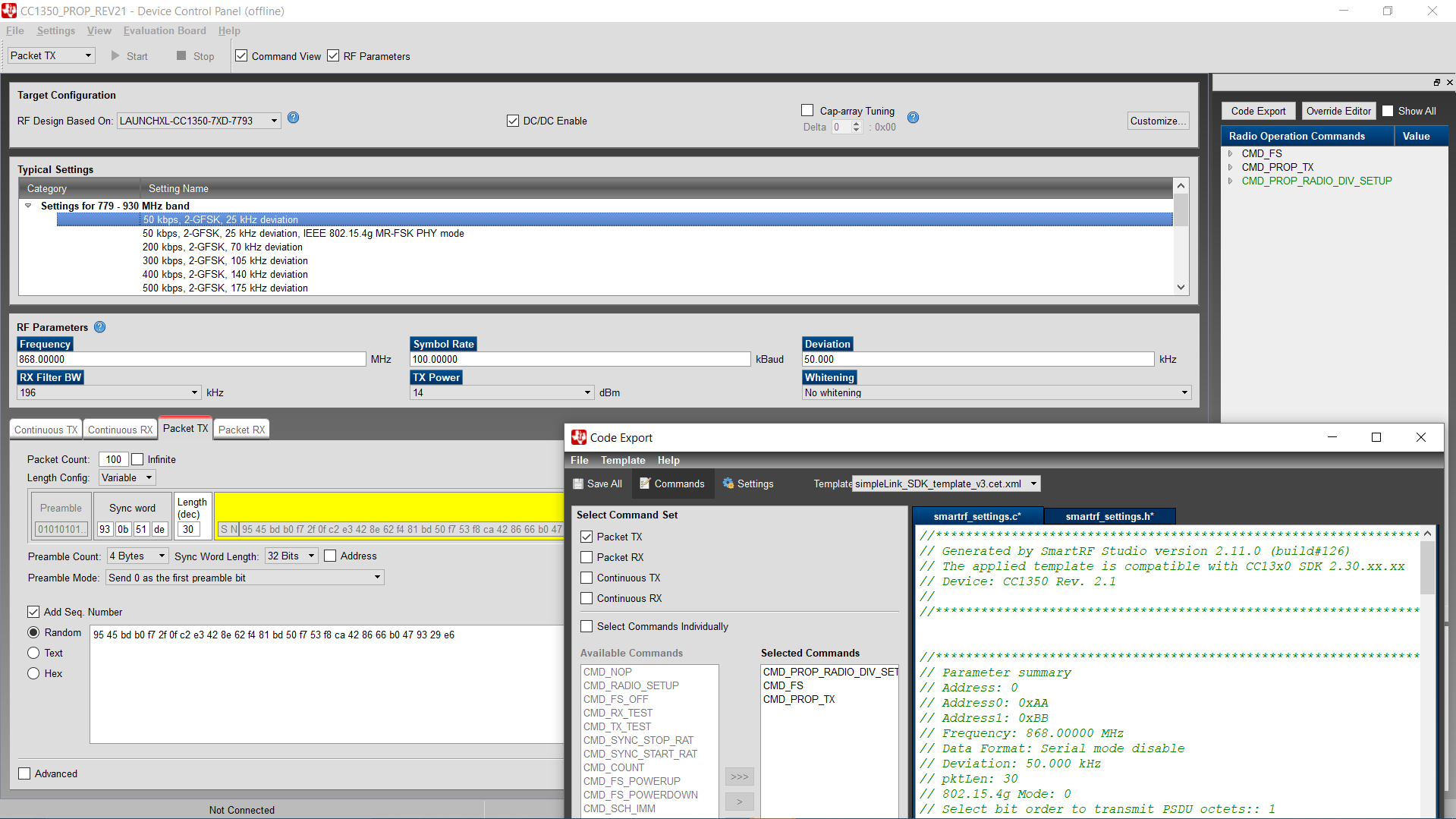
**Task 02: Import and run rfPacketTx example in Code Composer Studio.**

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

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

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

**Task 03: Exporting and using RF configuration.**



Settings used for modified RF configuration.

Youtube Link: <https://youtu.be/qsKIvCllREA>

**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

// 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,

// 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,

.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:

**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>

**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.