/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Program Operation - The program listens for incoming packets using the LoRa settings in the 'Settings.h'

file. The pins to access the lora device need to be defined in the 'Settings.h' file also.

There is a printout of the valid packets received, the packet is assumed to be in ASCII printable text,

if its not ASCII text characters from 0x20 to 0x7F, expect weird things to happen on the Serial Monitor.

The LED will flash for each packet received and the buzzer will sound, if fitted.

Sample serial monitor output;

1109s Hello World 1234567890\*,CRC,DAAB,RSSI,-61dBm,SNR,9dB,Length,23,Packets,1026,Errors,0,IRQreg,50

If there is a packet error it might look like this, which is showing a CRC error,

1189s PacketError,RSSI,-111dBm,SNR,-12dB,Length,0,Packets,1126,Errors,1,IRQreg,70,IRQ\_HEADER\_VALID,IRQ\_CRC\_ERROR,IRQ\_RX\_DONE

Serial monitor baud rate is set at 9600.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define Program\_Version "V1.0"

#include <SPI.h> //the lora device is SPI based so load the SPI library

#include <SX128XLT.h> //include the appropriate library

#include "Settings.h" //include the settings file, frequencies, LoRa settings etc

#include <Wire.h> //Library for using I2C

#include <LiquidCrystal\_I2C.h> //Library for using I2C type LCD display

LiquidCrystal\_I2C lcd(0x27, 16, 2); //initialize object lcd for class LiquidCrystal\_I2C with I2C address of 0x27 and 16x2 type LCD display

SX128XLT LT; //create a library class instance called LT

uint32\_t RXpacketCount;

uint32\_t errors;

uint8\_t RXBUFFER[RXBUFFER\_SIZE]; //create the buffer that received packets are copied into

uint8\_t RXPacketL; //stores length of packet received

int16\_t PacketRSSI; //stores RSSI of received packet

int8\_t PacketSNR; //stores signal to noise ratio of received packet

void loop()

{

RXPacketL = LT.receive(RXBUFFER, RXBUFFER\_SIZE, 60000, WAIT\_RX); //wait for a packet to arrive with 60seconds (60000mS) timeout

digitalWrite(LED1, HIGH); //something has happened

if (BUZZER > 0) //turn buzzer on

{

digitalWrite(BUZZER, HIGH);

}

PacketRSSI = LT.readPacketRSSI(); //read the received RSSI value

PacketSNR = LT.readPacketSNR(); //read the received SNR value

if (RXPacketL == 0) //if the LT.receive() function detects an error, RXpacketL == 0

{

packet\_is\_Error();

}

else

{

packet\_is\_OK();

}

if (BUZZER > 0)

{

digitalWrite(BUZZER, LOW); //buzzer off

}

digitalWrite(LED1, LOW); //LED off

Serial.println();

}

void packet\_is\_OK()

{

uint16\_t IRQStatus, localCRC;

IRQStatus = LT.readIrqStatus(); //read the LoRa device IRQ status register

RXpacketCount++;

printElapsedTime(); //print elapsed time to Serial Monitor

Serial.print(F(" "));

LT.printASCIIPacket(RXBUFFER, RXPacketL); //print the packet as ASCII characters

localCRC = LT.CRCCCITT(RXBUFFER, RXPacketL, 0xFFFF); //calculate the CRC, this is the external CRC calculation of the RXBUFFER

Serial.print(F(",CRC,")); //contents, not the LoRa device internal CRC

Serial.print(localCRC, HEX);

Serial.print(F(",RSSI,"));

Serial.print(PacketRSSI);

Serial.print(F("dBm,SNR,"));

Serial.print(PacketSNR);

Serial.print(F("dB,Length,"));

Serial.print(RXPacketL);

Serial.print(F(",Packets,"));

Serial.print(RXpacketCount);

Serial.print(F(",Errors,"));

Serial.print(errors);

Serial.print(F(",IRQreg,"));

Serial.print(IRQStatus, HEX);

}

void packet\_is\_Error()

{

uint16\_t IRQStatus;

IRQStatus = LT.readIrqStatus(); //read the LoRa device IRQ status register

printElapsedTime(); //print elapsed time to Serial Monitor

if (IRQStatus & IRQ\_RX\_TIMEOUT) //check for an RX timeout

{

Serial.print(F(" RXTimeout"));

}

else

{

errors++;

Serial.print(F(" PacketError"));

Serial.print(F(",RSSI,"));

Serial.print(PacketRSSI);

Serial.print(F("dBm,SNR,"));

Serial.print(PacketSNR);

Serial.print(F("dB,Length,"));

Serial.print(LT.readRXPacketL()); //get the real packet length

Serial.print(F(",Packets,"));

Serial.print(RXpacketCount);

Serial.print(F(",Errors,"));

Serial.print(errors);

Serial.print(F(",IRQreg,"));

Serial.print(IRQStatus, HEX);

LT.printIrqStatus(); //print the names of the IRQ registers set

}

lcd.setCursor(0,0);

lcd.print("RSSI: ");

lcd.setCursor(6,0);

lcd.print(PacketRSSI);

delay(250); //gives a longer buzzer and LED flash for error

}

void printElapsedTime()

{

float seconds;

seconds = millis() / 1000;

Serial.print(seconds, 0);

Serial.print(F("s"));

}

void led\_Flash(uint16\_t flashes, uint16\_t delaymS)

{

uint16\_t index;

for (index = 1; index <= flashes; index++)

{

digitalWrite(LED1, HIGH);

delay(delaymS);

digitalWrite(LED1, LOW);

delay(delaymS);

}

}

void setup()

{

pinMode(LED1, OUTPUT); //setup pin as output for indicator LED

led\_Flash(2, 125); //two quick LED flashes to indicate program start

Serial.begin(9600);

Serial.println();

Serial.print(F(\_\_TIME\_\_));

Serial.print(F(" "));

Serial.println(F(\_\_DATE\_\_));

Serial.println();

Serial.println(F("LoRa\_Receiver Starting"));

Serial.println();

if (BUZZER > 0)

{

pinMode(BUZZER, OUTPUT);

digitalWrite(BUZZER, HIGH);

delay(50);

digitalWrite(BUZZER, LOW);

}

SPI.begin();

//SPI beginTranscation is normally part of library routines, but if it is disabled in the library

//a single instance is needed here, so uncomment the program line below

//SPI.beginTransaction(SPISettings(8000000, MSBFIRST, SPI\_MODE0));

//setup hardware pins used by device, then check if device is found

if (LT.begin(NSS, NRESET, RFBUSY, DIO1, DIO2, DIO3, RX\_EN, TX\_EN, LORA\_DEVICE))

{

Serial.println(F("LoRa Device found"));

led\_Flash(2, 125);

delay(1000);

}

else

{

Serial.println(F("No device responding"));

while (1)

{

led\_Flash(50, 50); //long fast speed LED flash indicates device error

}

}

//The function call list below shows the complete setup for the LoRa device using the information defined in the

//Settings.h file.

//The 'Setup LoRa device' list below can be replaced with a single function call;

//LT.setupLoRa(Frequency, Offset, SpreadingFactor, Bandwidth, CodeRate);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Setup LoRa device

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LT.setMode(MODE\_STDBY\_RC);

LT.setRegulatorMode(USE\_LDO);

LT.setPacketType(PACKET\_TYPE\_LORA);

LT.setRfFrequency(Frequency, Offset);

LT.setBufferBaseAddress(0, 0);

LT.setModulationParams(SpreadingFactor, Bandwidth, CodeRate);

LT.setPacketParams(12, LORA\_PACKET\_VARIABLE\_LENGTH, 255, LORA\_CRC\_ON, LORA\_IQ\_NORMAL, 0, 0);

LT.setDioIrqParams(IRQ\_RADIO\_ALL, (IRQ\_TX\_DONE + IRQ\_RX\_TX\_TIMEOUT), 0, 0);

LT.setHighSensitivity();

//LT.setLowPowerRX();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Serial.println();

LT.printModemSettings(); //reads and prints the configured LoRa settings, useful check

Serial.println();

LT.printOperatingSettings(); //reads and prints the configured operating settings, useful check

Serial.println();

//LT.printRegisters(0x900, 0x9FF); //print contents of device registers

Serial.println();

Serial.print(F("Receiver ready - RXBUFFER\_SIZE "));

Serial.println(RXBUFFER\_SIZE);

Serial.println();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Setup LCD

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// initialize the LCD

lcd.init();

lcd.backlight(); // Turn on the blacklight and print a welcome message.

lcd.setCursor(0,0);

lcd.print("Hi!!!");

lcd.setCursor(0,1);

lcd.print("Please wait …");

//delay(1000);

//lcd.clear();

}