**LoRa – Internal project**

Task documentation

Introduction - LoRa:

LoRa is a wireless modulation technique derived from Chirp Spread Spectrum (CSS) technology. It encodes information on radio waves using chirp pulses - similar to the way dolphins and bats communicate. LoRa modulated transmission is robust against disturbances and can be received across great distances.

LoRa module: SX1262 (Transceiver)

Datasheet: <https://drive.google.com/drive/folders/12BDdpUGZjJlHjT2dPzhDq6xYcmyNpv5C>

Task 1: To achieve basic communication through two LoRa modules

Task 2: To integrate a sensor in one LoRa module and receive its output in another and to check the Transceiver functionality by sending back an acknowledgement of output reception.

**Task 1: To achieve basic communication through two LoRa modules**

The basic communication is achieved by configuring one LoRa module as transmitter and another as Receiver.

Controller for LoRa module integration: ESP32 Lilygo(Dev board)

Communication protocol: SPI

Power supply to the module: 5V (through ESP32)

Transmitter session pin connections are as follows.

|  |  |
| --- | --- |
| **SX1262** | **ESP32 Lilygo** |
| CS | 5 |
| DI01 | 34 |
| NRST | 32 |
| BUSY | 33 |
| MISO | 19 |
| MOSI | 23 |
| CLK | 18 |
| GND | GND |
| 5V | 5V-OUT |

The same applicable for receiver session as well.

Basic communication test is carried out by sending a string from transmitter and checking for its reception in receiver, the firmware code covering the configuration and application are as follows:

Firmware

IDE of testing: Arduino IDE

Library used: RadioLib.h // <https://www.arduino.cc/reference/en/libraries/radiolib/>

Transmitter:

// include the library

#include <RadioLib.h>

    // SX1262 has the following connections:

    // NSS pin:   5

    // DIO1 pin:  34

    // NRST pin:  32

    // BUSY pin:  33

SX1262 radio = new Module(5, 34, 32, 35);

void setup() {

//intiliazing serial communication

  Serial.begin(9600);

  // initialize SX1262 with default settings

  Serial.print(F("[SX1262] Initializing ... "));

// select appropriate frequency as per the country

  int state = radio.begin(868.0, 250.0, 7, 5, 0x34, 20, 10, 0, false);

  if (state == RADIOLIB\_ERR\_NONE) {

    Serial.println(F("success!"));

  } else {

    Serial.print(F("failed, code "));

    Serial.println(state);

    while (true);

  }

}

void loop() {

  Serial.print(F("[SX1262] Transmitting packet ... "));

// Transmitting a string to the receiver in same band of frequency

int state = radio.transmit("Hello World!");

 if (state == RADIOLIB\_ERR\_NONE) {

    // the packet was successfully transmitted

    Serial.println(F("success!"));

// print measured data rate

    Serial.print(F("[SX1262] Datarate:\t"));

    Serial.print(radio.getDataRate());

    Serial.println(F(" bps"));

} else if (state == RADIOLIB\_ERR\_PACKET\_TOO\_LONG) {

    // the supplied packet was longer than 256 bytes

    Serial.println(F("too long!"));

  } else if (state == RADIOLIB\_ERR\_TX\_TIMEOUT) {

    // timeout occured while transmitting packet

    Serial.println(F("timeout!"));

  } else {

    // some other error occurred

    Serial.print(F("failed, code "));

    Serial.println(state);

  }

  // wait for a second before transmitting again

  delay(1000);

}

Receiver:

// include the library

#include <RadioLib.h>

/ SX1262 has the following connections:

// NSS pin:   5

// DIO1 pin:  34

// NRST pin:  32

// BUSY pin:  33

SX1262 radio = new Module(5, 34, 32, 35);

void setup() {

  Serial.begin(9600);

  Serial.print(F("[SX1262] Initializing ... "));

// select appropriate frequency as per the country

  int state = radio.begin(868.0, 250.0, 7, 5, 0x34, 20, 10, 0, false);

  if (state == RADIOLIB\_ERR\_NONE) {

    Serial.println(F("success!"));

  } else {

    Serial.print(F("failed, code "));

    Serial.println(state);

    while (true);

  }

}

void loop() {

  Serial.print(F("[SX1262] Waiting for incoming transmission ... "));

  String str;

  int state = radio.receive(str);

  if (state == RADIOLIB\_ERR\_NONE) {

    // packet was successfully received

    Serial.println(F("success!"));

    // print the data of the packet

    Serial.print(F("[SX1262] Data:\t\t"));

    Serial.println(str);

    Serial.print(F("[SX1262] RSSI:\t\t"));

    Serial.print(radio.getRSSI());

    Serial.println(F(" dBm"));

    Serial.print(F("[SX1262] SNR:\t\t"));

    Serial.print(radio.getSNR());

    Serial.println(F(" dB"));

  } else if (state == RADIOLIB\_ERR\_RX\_TIMEOUT) {

    // timeout occurred while waiting for a packet

    Serial.println(F("timeout!"));

  } else if (state == RADIOLIB\_ERR\_CRC\_MISMATCH) {

    // packet was received, but is malformed

    Serial.println(F("CRC error!"));

  } else {

    // some other error occurred

    Serial.print(F("failed, code "));

    Serial.println(state);

  }

}

**Serial monitor outputs:**

Transmitter:

Receiver:

**Task 2:**

**To integrate a sensor in one LoRa module and receive its output in another.**

**To test the transceiver functionality by sending back an acknowledgement of output reception.**

Controller for LoRa module integration: ESP32

Communication protocol: SPI, I2C

Power supply to the module: 5V (through ESP32)

Sensor + ESP32 + LoRa module pin connections are as follows, (Transmitter)

|  |  |
| --- | --- |
| **SX1262** | **ESP32 Lilygo** |
| CS | 5 |
| DI01 | 34 |
| NRST | 32 |
| BUSY | 33 |
| MISO | 19 |
| MOSI | 23 |
| CLK | 18 |
| GND | GND |
| 5V | 5V-OUT |
| **LiDAR Sensor** | **ESP32 Lilygo** |
| Vin | 3.3V |
| GND | GND |
| SCL | 22 |
| SDA | 21 |

ESP32+LoRa module(Receiver) pin connections as follows,

|  |  |
| --- | --- |
| **SX1262** | **ESP32 Lilygo** |
| CS | 5 |
| DI01 | 34 |
| NRST | 32 |
| BUSY | 33 |
| MISO | 19 |
| MOSI | 23 |
| CLK | 18 |
| GND | GND |
| 5V | 5V-OUT |

Firmware

IDE of testing: Arduino IDE

Library used: RadioLib.h // <https://www.arduino.cc/reference/en/libraries/radiolib/>

**Sensor+Esp32+LoRa(transmitter ) firmware code :**

TMF882X liDAR sensor is integrated with ESP 32 using I2C protocol and the acquired value is sent as a string through LoRa module .Then command is initiated to wait for reception of acknowledgement from the receiver. Once the acknowledgement is received. Next set of values are transmitted to the receiver.

// Peer to Peer: Monitor Sensor Data using LoRa - Transmitter

// Include necessary libraries

#include <RadioLib.h>                  // Transmit & receive - https://github.com/jgromes/RadioLib/archive/refs/heads/master.zip

#include"SparkFun\_TMF882X\_Library.h"  //http://librarymanager/All#SparkFun\_Qwiic\_TMPF882X

    // SX1262 has the following connections:

    // NSS pin:   5

    // DIO1 pin:  34

    // NRST pin:  32

    // BUSY pin:  33

SX1262 radio = new Module(5, 34, 32, 35);

SparkFun\_TMF882X myTMF882X;

// Structure to hold the measurement results - this is defined by the TMF882X SDK.

static struct tmf882x\_msg\_meas\_results myResults;

void setup() {

  delay(1000);

  Serial.begin(115200);

  Serial.println("");

  Serial.println("In setup");

  Serial.println("==============================");

  // Initialize the TMF882X device

  if (!myTMF882X.begin()) {

    Serial.println("Error - The TMF882X failed to initialize - is the board connected?");

    while (1)

      ;

  } else

    Serial.println("TMF882X started.");

  // The device is now ready for operations

  // Initialize SX1262 with default settings

  Serial.print(F("[SX1262] Initializing ... "));

  int state = radio.begin(868.0, 250.0, 7, 5, 0x34, 20, 10, 0, false);

  if (state == RADIOLIB\_ERR\_NONE) {

    Serial.println(F("success!"));

  } else {

    Serial.print(F("failed, code "));

    Serial.println(state);

    while (true) ;

  }

}

void loop() {

  Serial.println(F("[SX1262] Transmitting packet ... "));

  if (myTMF882X.startMeasuring(myResults)) {

    // print out results

    Serial.println("Measurement:");

    Serial.print("     Result Number: ");

    Serial.print(myResults.result\_num);

    Serial.print("  Number of Results: ");

    Serial.println(myResults.num\_results);

    Serial.print("       conf: ");

    Serial.print(myResults.results[4].confidence);

    Serial.print(" distance mm: ");

    Serial.print(myResults.results[4].distance\_mm);

    Serial.print(" channel: ");

    Serial.print(myResults.results[4].channel);

    Serial.print(" sub\_capture: ");

    Serial.println(myResults.results[4].sub\_capture);

    Serial.print("     photon: ");

    Serial.print(myResults.photon\_count);

    Serial.print(" ref photon: ");

    Serial.print(myResults.ref\_photon\_count);

    Serial.print(" ALS: ");

    Serial.println(myResults.ambient\_light);

    Serial.println();

  }

  // Create a String to send (data must be sent as a string, so we'll tell our computer friend that we want our integers to be set as strings)

  String myData =  String(myResults.results[4].distance\_mm)+ "mm " + "Channel :" + String(myResults.results[4].channel) ;  // I added some units for our data here as well

  // Transmit data and units

  int state = radio.transmit(myData);

  if (state == RADIOLIB\_ERR\_NONE) {

    // The packet was successfully transmitted

    Serial.println(F("Success!"));

    // Print measured data rate

    Serial.print(F("[SX1262] Datarate:\t"));

    Serial.print(radio.getDataRate());

    Serial.println(F(" bps"));

    //wait for receiver ack

     String str;                      // Declare a data type to receive (this must be a string)

  int ackstate = radio.receive(str);

  while(ackstate != RADIOLIB\_ERR\_NONE){

    ackstate = radio.receive(str);

       }

  // Print the data of the packet to the serial monitor

    Serial.print(F("ack from receiver:\t\t"));

    Serial.println(str);

  } else if (state == RADIOLIB\_ERR\_PACKET\_TOO\_LONG) {

    // The supplied packet was longer than 256 bytes

    Serial.println(F("Packet size too long!"));

  } else if (state == RADIOLIB\_ERR\_TX\_TIMEOUT) {

    // Timeout occurred while transmitting packet

    Serial.println(F("Timeout!"));

  } else {

    // Some other error occurred

    Serial.print(F("Error"));

    Serial.println(state);

  }

  // Wait for a second before transmitting again

  delay(6000);}

**Esp32+LoRa(receiver) firmware code :**

Receiver LoRa module is configured to receive the LiDAR sensor values as a string and send back an acknowledgement to transmitter.

// Peer to Peer: Monitor Sensor Data using LoRa - Receiver

// SparkFun Electronics, Mariah Kelly, November 2022

// Original receive file can be found here: https://cdn.sparkfun.com/assets/learn\_tutorials/1/4/9/4/Receive-v3.ino

// Include necessary libraries

#include <RadioLib.h>

// SX1262 has the following connections:

// NSS pin:   5

// DIO1 pin:  34

// NRST pin:  32

// BUSY pin:  33

SX1262 radio = new Module(5, 34, 32, 35);

void setup() {

  Serial.begin(115200);

  // initialize SX1262 with default settings

  Serial.print(F("[SX1262] Initializing ... "));

  //int state = radio.begin();

  int state = radio.begin(868.0, 250.0, 7, 5, 0x34, 20, 10, 0, false);

  if (state == RADIOLIB\_ERR\_NONE) {

    Serial.println(F("success!"));

  } else {

    Serial.print(F("failed, code "));

    Serial.println(state);

    while (true)

      ;

  }

}

void loop() {

  Serial.print(F("[SX1262] Waiting for incoming transmission ... "));

  String str;                      // Declare a data type to receive (this must be a string)

  int state = radio.receive(str);  // Receive data from transmitter

while(state != RADIOLIB\_ERR\_NONE)

{

  state = radio.receive(str);

}

  if (state == RADIOLIB\_ERR\_NONE) {

    // The packet was successfully received

    Serial.println(F("Success!"));

    // Print the data of the packet to the serial monitor

    Serial.print(F("[SX1262] Data:\t\t"));

    Serial.println(str);

    // Print the RSSI (Received Signal Strength Indicator) of the last received packet

    Serial.print(F("[SX1262] RSSI:\t\t"));

    Serial.print(radio.getRSSI());

    Serial.println(F(" dBm"));

    // Print the SNR (Signal-to-Noise Ratio) of the last received packet

    Serial.print(F("[SX1262] SNR:\t\t"));

    Serial.print(radio.getSNR());

    Serial.println(F(" dB"));

    Serial.println("sending ack");

    int ackstate =radio.transmit("data received");

    while(ackstate != RADIOLIB\_ERR\_NONE){

    ackstate = radio.transmit("data received");

    }

    Serial.print("state :");

    Serial.print(ackstate);

  }

  else if (state == RADIOLIB\_ERR\_RX\_TIMEOUT) {

    // Timeout occurred while waiting for a packet

    Serial.println(F("Timeout!"));  // We won't print this to the screen since the Success/data print will freeze on its own if a timeout occurs

  } else if (state == RADIOLIB\_ERR\_CRC\_MISMATCH) {

    // The packet was received, but is malformed

    Serial.println(F("CRC error!"));

  } else {

    // Some other error occurred

    Serial.print(F("Error"));

    Serial.println(state);

  }

}

Note : Serial monitor values are in the test record.  
  
**Way forward in LoRa Project:**

**1.Test for more than 1 km**  
2. To create a LoRaWan (LoRa Hub).  
3. To achieve communication from all loRa devices to one Common LoRahub  
4. To build Network server using thingsnetwork and Aws LoRawan and monitor data at the server.  
5. Cross check server data from another device.  
6. File transfer/image transfer using LoRa and OTA.  
  
Supporting teams:  
1. Electrical hardware team – LoRa hardware development   
2. Software team – Server network in AwsLorawan or Thingsnetwork  
3. Firmware team – built sustainable Firmware for the LoRa hardware