LoRa Communication Shield for Mobile Phone

GE-107 Tinkering lab Project Write up

Team Members (Group 4)

1. Abhinav Barman	2021EEB1145
2. Gaurav Wani	2021EEB1171
3. Ranjeet Singh	2021EEB1203
4. Reetika	2021EEB1204
5. Shashwat Srivastava	2021EEB1210
6. Yash Agarwal	2021EEB1225

Introduction:

LoRa (Long Range) technology has emerged as an excellent choice for long-range, low-power IoT communications. It is a wireless communication protocol that is specifically designed to provide long-range, low-power communication for the Internet of Things (IoT). In this project, we will be discussing the development of a LoRa communication shield for mobile phones using the ESP-32 and IA-02 LoRa.

Components Required:

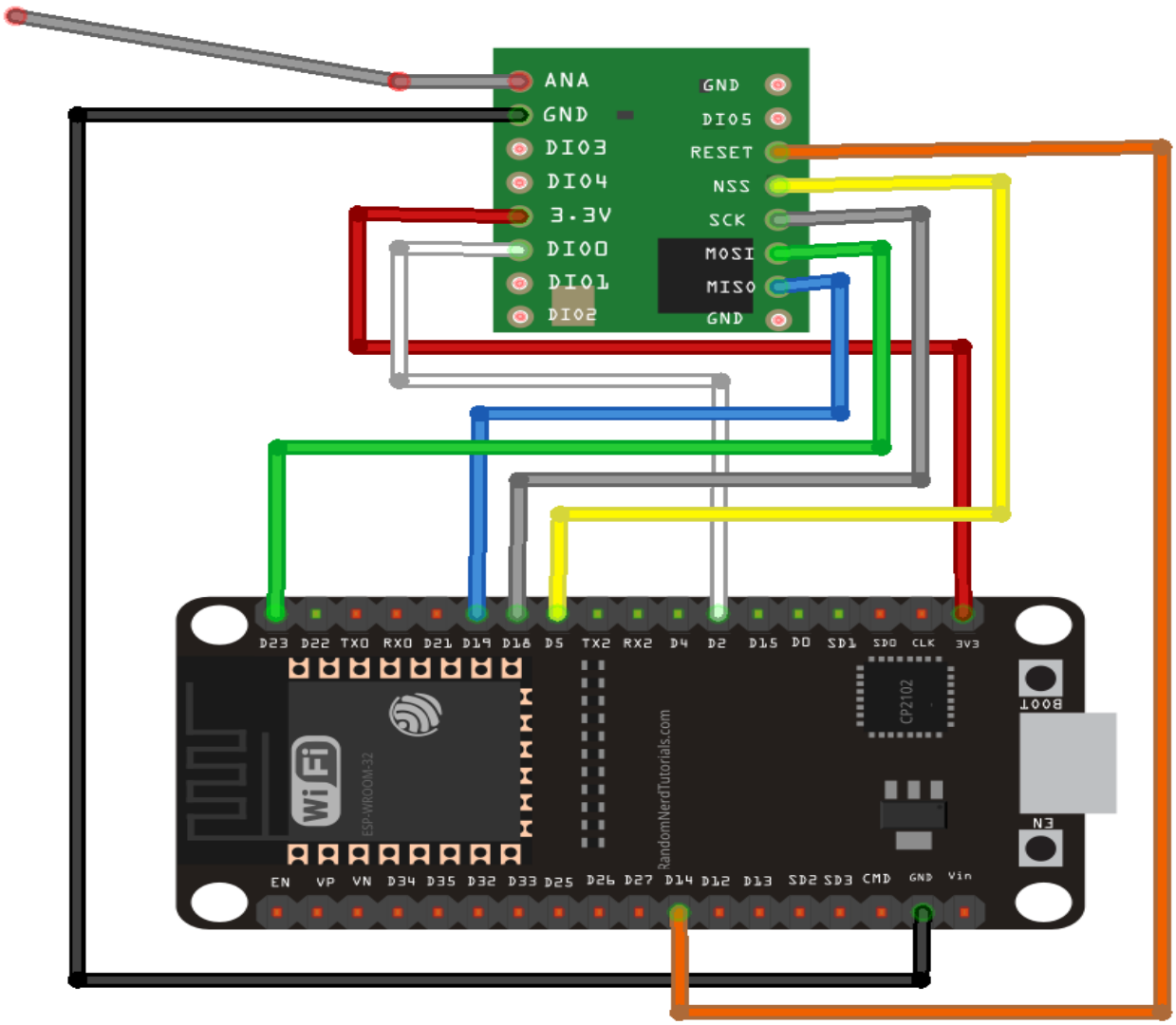
- ESP-32 Development Board
- IA-02 LoRa Module
- Breadboard
- Jumper Wires
- USB Cable
- Mobile Phone with USB OTG Support

Hardware Setup:

1. Connect the ESP-32 development board to a breadboard using jumper wires.
2. Connect the IA-02 LoRa module to the breadboard.
3. Connect the VCC pin of the IA-02 module to the 3.3V pin of the ESP-32 board.
4. Connect the GND pin of the AI-02 module to the GND pin of the ESP-32 board.
5. Connect the CS pin of the AI-02 module to GPIO 5 of the ESP-32 board.

6. Connect the MISO pin of the AI-02 module to GPIO 19 of the ESP-32 board.
7. Connect the MOSI pin of the AI-02 module to GPIO 27 of the ESP-32 board.
8. Connect the SCK pin of the AI-02 module to GPIO 18 of the ESP-32 board.
9. Connect the NSS pin of the AI-02 module to GPIO 5 of the ESP-32 board.

CIRCUIT DIAGRAM:



Software Setup:

1. Install the Arduino IDE and the ESP-32 board manager.
2. Install the LoRa library by Sandeep Mistry from the library manager.
3. Create a new project in the Arduino IDE.
4. Include the LoRa library by adding "#include <LoRa.h>" at the beginning of the sketch.
5. In the setup function, initialize the serial port using "Serial.begin(9600)".
6. In the setup function, initialize the LoRa module using "LoRa.begin(433E6)".
7. In the loop function, read incoming LoRa messages using "int packetSize = LoRa.parsePacket()" and process the message using the onReceive function.
8. In the sendMessage function, send outgoing LoRa messages using "LoRa.beginPacket()", "LoRa.print()", and "LoRa.endPacket()".

LoRa Receiver code (screenshots and explanation)

lorareceiver1

```
/*********
  Modified from the examples of the Arduino LoRa library
  More resources: https://randomnerdtutorials.com
*********/

#include <SPI.h>
#include <LoRa.h>
#include <BluetoothSerial.h>
//define the pins used by the transceiver module
#define SCK 18
#define MISO 19
#define MOSI 23
#define SS 5
#define RST 14
#define DIO0 2
BluetoothSerial SerialBT;
// int counter = 0;

void setup() {
  //initialize Serial Monitor
  Serial.begin(9600);
  while (!Serial);
```

lorareceiver1

```
  //replace the LoRa.begin(---E-) argument with your location's frequency
  //433E6 for Asia
  //866E6 for Europe
  //915E6 for North America
  while (!LoRa.begin(433E6)) {
    Serial.println(".");
    delay(500);
  }
   // Change sync word (0xF3) to match the receiver
  // The sync word assures you don't get LoRa messages from other LoRa transceivers
  // ranges from 0-0xFF
  LoRa.setSyncWord(0xF3);
  Serial.println("LoRa Initializing OK!");
}

void loop() {

// Recieveing the data
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packet
    Serial.print("Received packet '");

    // read packet
    while (LoRa.available()) {
      String LoRaData = LoRa.readString();
```

lorareceiver1

```
        SerialBT.print(LoRaData);
        SerialBT.println();
      }

      // print RSSI of packet
      //Serial.print("' with RSSI ");
      //Serial.println(LoRa.packetRssi());
   }

// Sending the data
   if(SerialBT.available()){
     String d = SerialBT.readString();
     Serial.print("Sending packet: ");
     Serial.print(d);
     Serial.print(" : ");
//     Serial.println(counter);
     Serial.println(d);
     //Send LoRa packet to receiver
     LoRa.beginPacket();
     LoRa.print(d);
//     LoRa.print(counter);
     LoRa.endPacket();

//     counter++;
```

This code is for a LoRa receiver module which receives data from a LoRa transmitter module. The received data is then sent via Bluetooth to a connected device such as a smartphone or tablet.

The code uses the LoRa library which is included in the Arduino IDE. It also uses the BluetoothSerial library to establish a Bluetooth connection with the device that will receive the data.

In the setup function, the serial monitor is initialized with a baud rate of 9600. The BluetoothSerial library is also initialized with the name "Receiver". The pins used by the LoRa transceiver module are defined as constants and assigned to their respective variables. The LoRa module is then initialized with the begin method, using a frequency of 433MHz and a sync word of 0xF3.

In the loop function, the code first checks if there is any data received by the LoRa module by calling the parsePacket method. If there is data, it is read and stored as a string variable LoRaData. The received data is then printed to the serial monitor and sent via Bluetooth to the connected device.

The code then checks if there is any data available from the BluetoothSerial library. If there is data, it is read and stored as a string variable d. The string is then printed to the serial monitor and sent as a LoRa packet using the beginPacket, print and endPacket methods. The delay function is used to avoid sending data too frequently.

LoRa Sender code (screenshots and explanation)

sender2

```
#include <SoftwareSerial.h>

#include <SPI.h>
#include <LoRa.h>

//define the pins used by the transceiver module
#define SCK 13
#define MISO 12
#define MOSI 11
#define SS 10
#define RST 9
#define DIO0 4

//int counter = 0;
//String c="Himasnhu";
//String d = "hji";

SoftwareSerial BTSerial(2,3);

void setup() {
  //initialize Serial Monitor
  Serial.begin(9600);
  BTSerial.begin(9600);

  while (!Serial);
  Serial.println("LoRa Sender");
```

sender2

```
  //setup LoRa transceiver module
  LoRa.setPins(SS, RST, DIO0);

  //replace the LoRa.begin(---E-) argument with your location's frequency
  //433E6 for Asia
  //866E6 for Europe
  //915E6 for North America
  while (!LoRa.begin(433E6)) {
    Serial.println(".");
    delay(500);
  }
   // Change sync word (0xF3) to match the receiver
  // The sync word assures you don't get LoRa messages from other LoRa transceivers
  // ranges from 0-0xFF
  LoRa.setSyncWord(0xF3);
  Serial.println("LoRa Initializing OK!");
}

void loop() {

// Recieving the data
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packet
    Serial.print("Received packet '");

```

sender2

```
    // received a packet
    Serial.print("Received packet '");

    // read packet
    while (LoRa.available()) {
      String LoRaData = LoRa.readString();
      Serial.println(LoRaData);
    }
  }

// Sending the data
  if (Serial.available()) {
    String d = Serial.readString();
    Serial.println(d);
    Serial.print("Sending packet: ");
    Serial.println(d);

    //Send LoRa packet to receiver
    LoRa.beginPacket();
    LoRa.print(d);
    //LoRa.print(counter);
    LoRa.endPacket();

    delay(2000);
  }
}
```

This code is for a LoRa wireless communication system that consists of a sender and a receiver. The code provided is for the sender.

The code starts by including necessary libraries such as SoftwareSerial, SPI, and LoRa. It then defines the pins used by the LoRa transceiver module, namely SCK, MISO, MOSI, SS, RST, and DIO0.

In the setup() function, the code initializes the Serial Monitor and a SoftwareSerial object called BTSerial with a baud rate of 9600. It also sets up the LoRa transceiver module using LoRa.setPins(SS, RST, DIO0) and LoRa.begin(433E6), which sets the frequency to 433 MHz for this example.

The sync word is then set to 0xF3 using LoRa.setSyncWord(0xF3) to ensure that the sender and receiver are using the same sync word to communicate. The sync word is a unique value that helps the receiver

identify the sender's message and reject messages from other transceivers that are using a different sync word.

In the loop() function, the code waits for incoming messages by calling LoRa.parsePacket(), which returns the size of the received packet if one exists. If a packet is received, the code reads the packet using LoRa.readString() and prints it to the Serial Monitor.

The code also waits for input from the Serial Monitor using Serial.available(). If there is input available, it reads it using Serial.readString() and sends the data using LoRa.beginPacket(), LoRa.print(), and LoRa.endPacket(). The delay(2000) at the end of the loop ensures that the sender does not send packets too quickly.

Overall, this code initializes and sets up the LoRa transceiver module for wireless communication, waits for incoming messages, and sends messages via LoRa when input is detected from the Serial Monitor.

Mobile Phone Integration:

1. Install the USB OTG (On-The-Go) support application on the mobile phone.
2. Connect the ESP-32 board to the mobile phone using a USB cable.
3. Open the USB OTG support application and allow the device to access the ESP-32 board.
4. Open a serial bluetooth communication application on the mobile phone, such as Serial USB Terminal or Bluetooth Serial Terminal.
5. Set the serial communication parameters to 9600 baud, 8 data bits, no parity, and 1 stop bit.
6. Send and receive LoRa messages between the mobile phone and the ESP-32 board using the serial communication application.

Conclusion:

In this project, we have developed a LoRa communication shield for mobile phones using the ESP-32 and IA-02 LoRa. We have also discussed the hardware and software setup required for the project, as well as the integration of the shield with a mobile phone using USB OTG support. We can also increase the range of communication by setting up different spreading factors. This project can be further expanded to include additional features, such as encryption and decryption of LoRa messages, and integration with mobile applications using Bluetooth or Wi-Fi.