Gravity: LoRaWAN Node Module (US915) - DFRobot

SKU:DFR1115-915 (https://www.dfrobot.com/product-2927.html)

(https://www.dfrobot.com/product-2927.html)

1. Introduction

This is an easy-to-use, dual-mode long-distance wireless communication module designed for outdoor IoT applications, providing stable, reliable, and low-power long-range communication solutions. Based on LoRa modulation technology, the module is designed for the US915 frequency band, suitable for use in North America and other regions using this frequency. The main features are as follows:

Kilometer-Range Long-Distance Transmission

The module provides effective transmission distances of 1.5 kilometers in urban areas and 4.5 kilometers in open areas, making it suitable for outdoor IoT applications.

Note: Actual transmission distance may be affected by environmental factors such as weather, traffic, and building density.

Dual-Mode Flexible Configuration (LoRa/LoRaWAN)

Supports two communication modes: LoRa point-to-point (P2P) direct connection and LoRaWAN networking, providing flexible configuration to meet various

providing flexible configuration to meet various application needs.

l In LoRa mode, the module supports one-to-one, one-to-many, many-to-one, and bridge communication;
l In LoRaWAN mode, the module supports both Class A and Class C operation modes, acting as a data acquisition node that connects to a gateway and forwards data to IoT cloud platforms such as TTN, ChirpStack, etc.

Simple to Use, Quick Deployment

With an integrated LoRa/LoRaWAN protocol stack, no lower-level development is required. The module supports Arduino IDE, Mind+, and MakeCode graphical programming, reducing development complexity. Additionally, it provides standard I2C and UART communication interfaces, compatible with popular microcontrollers such as Micro:bit, Arduino UNO, ESP32, and other development boards. This enables quick integration into existing IoT projects and greatly improves project deployment efficiency.

Outdoor IoT communication solutions

Outdoor IoT Scene Pain Points	Traditional Solutions	This Product's Solution
High Deployment Cost	Requires wiring/ intermediate devices, complex and high cost	Fully wireless deployment, low cost
Short Transmission Range	WiFi < 100m, unstable network	Stable kilometer-range transmission
Short Device Battery Life	4G/WiFi communication consumes a lot of power	Low power consumption with LoRa/ LoRaWAN

		communication

Suitable for long-distance, low-power IoT communication scenarios, such as farm environmental monitoring,

weather station data collection, industrial monitoring, garden planting monitoring, and more.

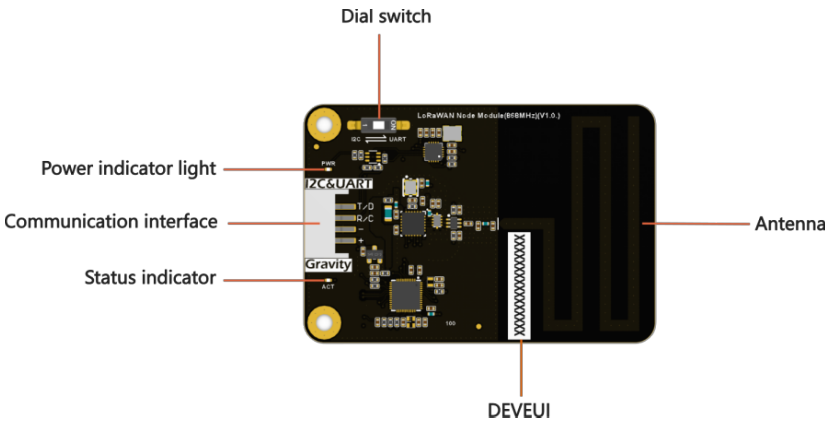
2. Features

- Compatible with 3.3V and 5V logic levels
- Supports both UART and I2C communication methods
- Onboard PCB antenna, integrated module design
- Effective coverage: 1.5km in urban areas / 4.5km in open areas
- Suitable for North America and other regions using the US915 frequency band
- Supports LoRa one-to-one, one-to-many, many-to-one, and bridge communication
- Supports ABP and OTAA LoRaWAN activation modes
- Wide compatibility with microcontrollers: Micro:bit, Arduino UNO, ESP32, and other development boards
- Simple to use, supports Arduino IDE, Mind+, and MakeCode graphical programming

3. Applications

- Crop growth monitoring
- Greenhouse temperature and humidity monitoring
- Ecological area environmental monitoring
- Beehive automatic monitoring
- LoRaWAN weather stations
- Garden planting projects
- Outdoor IoT education

4. Function indication



Name	Full Name	Function
T/D	UART_TX/ I2C_SDA	UART transmit pin (TX) / I2C data line (SDA)
R/C	UART_RX/ I2C_SCL	UART receive pin (RX) / I2C clock line (SCL)
-	DGND	Digital ground, connects to the GND of the host controller
+	VCC	PPower supply input, DC 3.3V~5V (must match the host system voltage level)
PWR	Power	Red power indicator, remains on when power is supplied
ACT	Active	Green status indicator:
		1. Blinks for 1 second when transmitting a network join request.
		2. Remains on for 5 seconds when the network join is successful.
		3. Blinks for 300ms during

		3. Blinks for 500ms during data transmission or reception.

Note: The DIP switch defaults to I2C mode and can switch between I2C and UART. Restart the module after switching.

5. Specification

Basic Parameters

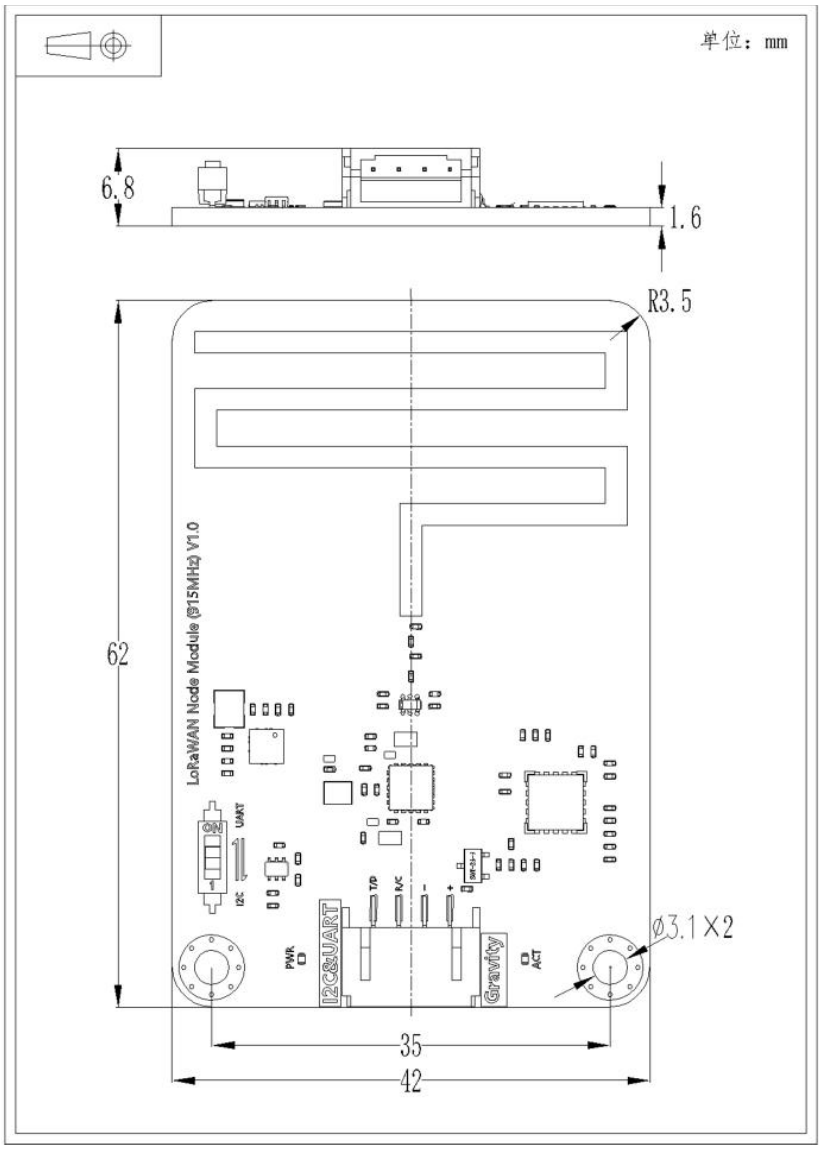
Power Supply Voltage	**DC 3.3V~5V**
Communication Mode	I2C/UART
Power/Communication Interface	PH2.0-4P
Mounting Hole Diameter	3.0mm
Mounting Hole Spacing	35mm
Product Dimensions	42x62mm
Net Weight	10g

LoRa Parameters

RF Chip	**SX1262**
Operating Frequency	915MHz
Supported Regions	North America and other regions using the US915 band
Modulation Method	LoRa modulation
Spreading Factor	7~12
Maximum Transmit Power	+22dBm
Receiver Sensitivity	-137dBm (SF=12/BW=125kHz)

Sensitivity	

6. Dimensions



7.1.1 Hardware Preparation

- FireBeetle 2 ESP32-E (https://www.dfrobot.com.cn/goods-3009.html) (SKU: DFR0654) ×3
- Gravity: LoRaWAN Node Module （US915） (https://www.dfrobot.com/product-2927.html)(DFR1115-915) ×3
- Gravity: DHT11 Temperature and Humidity Sensor (https://www.dfrobot.com.cn/goods-109.html) (SKU: DFR0067) ×1
- Gravity: UV Sensor (https://www.dfrobot.com.cn/

goods-3651.html) (SKU: SEN0540) ×1

- PH2.0-4P Cable ×3
- USB Data Cable ×3

7.1.2 Software Preparation

- Download Arduino IDE: Click to Download Arduino IDE (https://www.arduino.cc/en/Main/Software)
- Install SDK: Visit the FireBeetle 2 ESP32-E WIKI page (https://wiki.dfrobot.com.cn/_SKU_DFR0654_FireBeetle_Board_ESP32_E#target_6) for SDK installation instructions
- Download Arduino Library: Click to download DFRobot_LWNode Library (https://github.com/cdjq/DFRobot_LWNode) and refer to the guide: How to Install a Library? (https://www.arduino.cc/en/guide/libraries)

7.1.3 Transmit-Receive Application Example

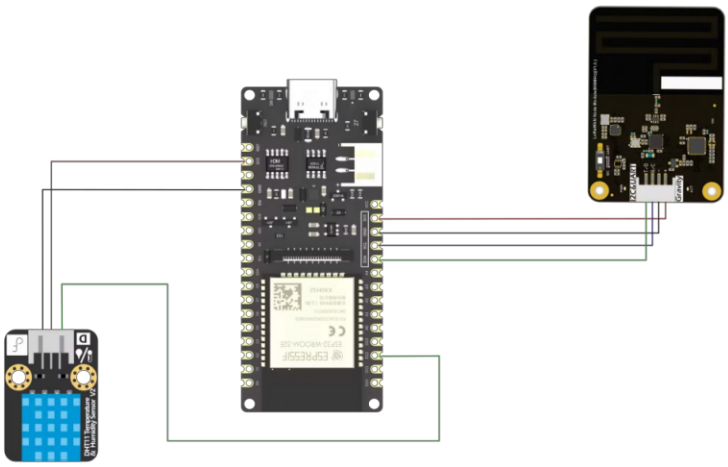
In LoRa communication mode, each node device needs to set a custom address (range 1~255):

Address	Description
0	Invalid address (not usable)
1~244	Valid reusable addresses, e.g., set two nodes to address 3
255	Broadcast address (when sending to 255, all devices in the network can receive)

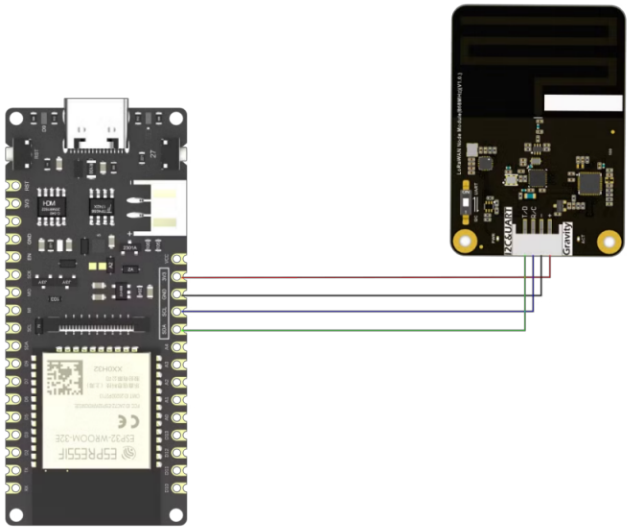
Therefore, this example demonstrates: two FireBeetle 2 ESP32-E main controllers, each expanding one node module (addresses 1 and 2 respectively), achieving long-distance, directional data transmission from the temperature and humidity sensor via differentiated address configurations.

Hardware Connection:

- Transmitter:



- Receiver:



Example Code:

Transmitter Program: Set node address to 1, send data to node with address 2.

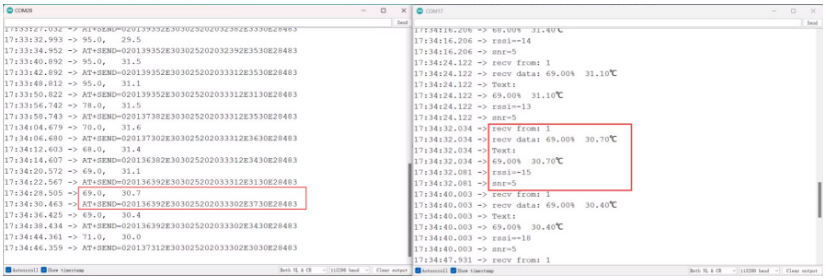
```
#include <DFRobot_LWNode.h>
#include <dht11.h>
dht11 DHT;
#define DHT11_PIN 4
#define FREQ  914900000
DFRobot_LWNode_IIC node(1); // Set node addre

void setup( void ) {
    Serial.begin(115200);
    delay(5000);
    node.begin(/*communication IIC*/&Wire,/*c
    const uint32_t loraConfig[] = {FREQ, DBM2
    while(!node.setFreq(loraConfig[0])  ||
          !node.setEIRP(loraConfig[1])  ||
          !node.setBW(loraConfig[2])    ||
          !node.setSF(loraConfig[3])    ||
          !node.start()) {
        Serial.println("LoRa init failed");
        delay(2000);
    }
}

void loop( void ){
    DHT.read(DHT11_PIN);   // Get data from D
    Serial.print(DHT.humidity,1);
    Serial.print(",\t");
    Serial.println(DHT.temperature,1);
    String DHT11_DATE=String(DHT.humidity)+"%
    delay(2000);
    node.sendPacket(2, DHT11_DATE);  // Send
    node.sleep(5000);
}
```

Receiver Program: Node with address 2 should burn the following code.

```
#include <DFRobot_LWNode.h>
#define FREQ  914900000
DFRobot_LWNode_IIC node(2);  // Set node addr

void rxCBFunc(uint8_t from, void *buffer, uir
    char *p = (char *)buffer;
    Serial.print("recv from: ");
    Serial.println(from, HEX);
    Serial.print("recv data: ");
    for(uint8_t i = 0; i < size; i++){
        Serial.print(p[i]);
    }
    Serial.println();
    Serial.println("Text:");
    Serial.println((char *)buffer);
    Serial.print("rssi=");Serial.println(rssi
    Serial.print("snr=");Serial.println(snr);
}

void setup( void ){
    Serial.begin(115200);
    delay(5000);
    node.begin(/*communication IIC*/&Wire,/*c
    const uint32_t config[] = {FREQ, DBM22, 1
    while(!node.setFreq(config[0])    ||
          !node.setEIRP(config[1])   ||
          !node.setBW(config[2])     ||
          !node.setSF(config[3])     ||
          !node.start()) {
        Serial.println("LoRa init failed, ret
        delay(2000);
    }
    node.setRxCB(rxCBFunc);
}
```

```
void loop( void ){
    node.sleep(5000);
}
```

Results: Communication successful, the serial output from both the transmitter and receiver is as follows:



7.1.4 One-to-Many Communication Example

There are two types of one-to-many communication modes for nodes:

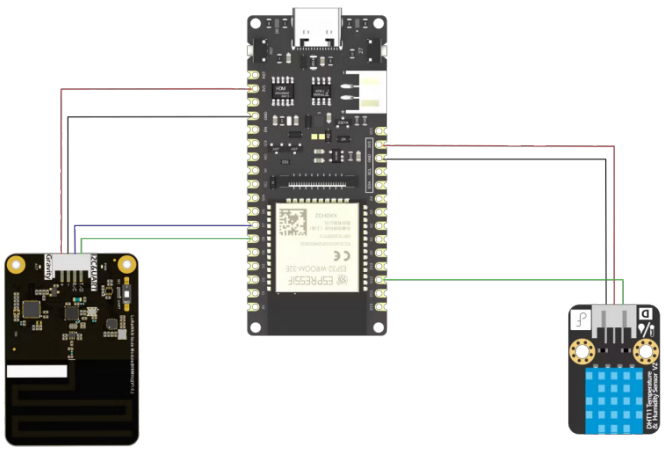
1. **Single Transmit, Multiple Receive Mode**: One node sends data, and multiple nodes receive simultaneously.
2. **Single Receive, Multiple Transmit Mode**: One node receives data, and multiple nodes transmit simultaneously.

This example demonstrates the first type, **Single Transmit, Multiple Receive Mode**: Node with address 1 is set as the transmitter, and nodes with address 2 and 3 are set as receivers, realizing the "one-to-many" communication scenario.

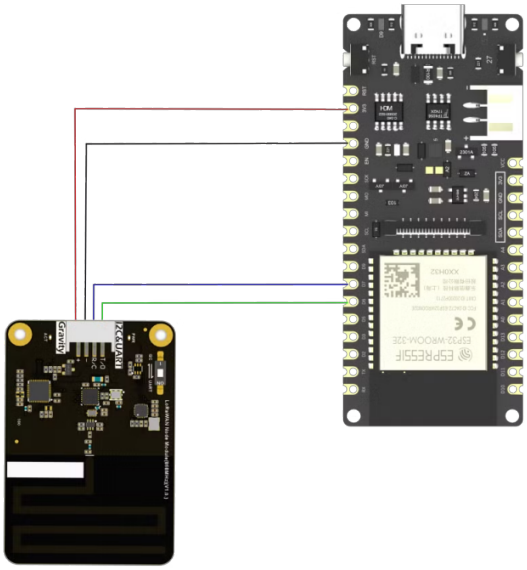
Hardware Connection:

Here, UART communication is used.

- Transmitter:



- Receiver (The connection method for both the first and second groups is as follows):



Example Code:

Transmitter Program: Set node address to 1, send temperature and humidity sensor data to node with

address 2.

```
#include <DFRobot_LWNode.h>
#include <dht11.h>
dht11 DHT;
#define DHT11_PIN 4
#define FREQ  914900000
DFRobot_LWNode_UART node(1); // Set node addr

void setup( void ) {
    Serial.begin(115200);
    Serial1.begin(9600, SERIAL_8N1, /*rx =*/D
    delay(5000);
    node.begin(/*communication UART*/&Serial1
     const uint32_t config[] = {FREQ, DBM22,
    while(!node.setFreq(config[0])    ||
          !node.setEIRP(config[1])   ||
          !node.setBW(config[2])     ||
          !node.setSF(config[3])     ||
          !node.start()) {
        Serial.println("LoRa init failed, ret
        delay(2000);
    }
}

void loop( void ){
    DHT.read(DHT11_PIN);  // Get data from DH
    Serial.print(DHT.humidity,1);
    Serial.print(",\t");
    Serial.println(DHT.temperature,1);
    String DHT11_DATE=String(DHT.humidity)+"%
    delay(2000);
    node.sendPacket(2, DHT11_DATE);  // Send
    node.sleep(5000);
}
```

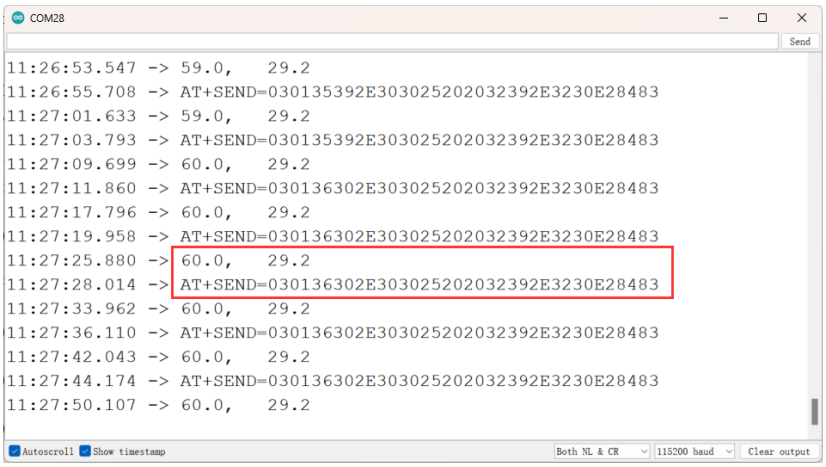
The first and second groups of receivers both use the following program (note: set different serial ports): set the node address to 2 to receive data from the node with address 1.

```
#include <DFRobot_LWNode.h>
#define FREQ  914900000
DFRobot_LWNode_UART node(2);  // Set node add

void rxCBFunc(uint8_t from, void *buffer, uir
    char *p = (char *)buffer;
    Serial.print("recv from: ");
    Serial.println(from, HEX);
    Serial.print("recv data: ");
    for(uint8_t i = 0; i < size; i++){
        Serial.print(p[i]);
    }
    Serial.println();
    Serial.print("rssi=");Serial.println(rssi
    Serial.print("snr=");Serial.println(snr);
}

void setup( void ){
    Serial.begin(115200);
    Serial1.begin(9600, SERIAL_8N1, /*rx =*/D
    delay(5000);
    node.begin(/*communication UART*/&Serial1
    const uint32_t config[] = {FREQ, DBM22, 1
    while(!node.setFreq(config[0])    ||
          !node.setEIRP(config[1])   ||
          !node.setBW(config[2])     ||
          !node.setSF(config[3])     ||
          !node.start()) {
        Serial.println("LoRa init failed, ret
        delay(2000);
    }
```

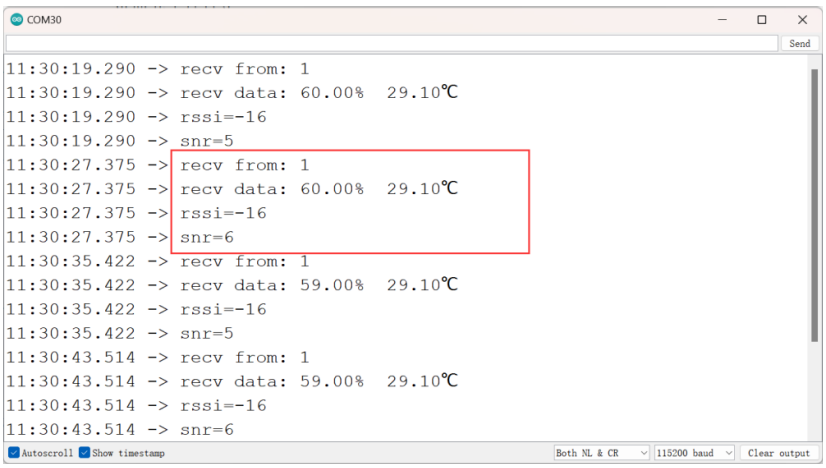
```
    }
    node.setRxCB(rxCBFunc);
}


void loop( void ){
    node.sleep(5000);
}
```

Result:

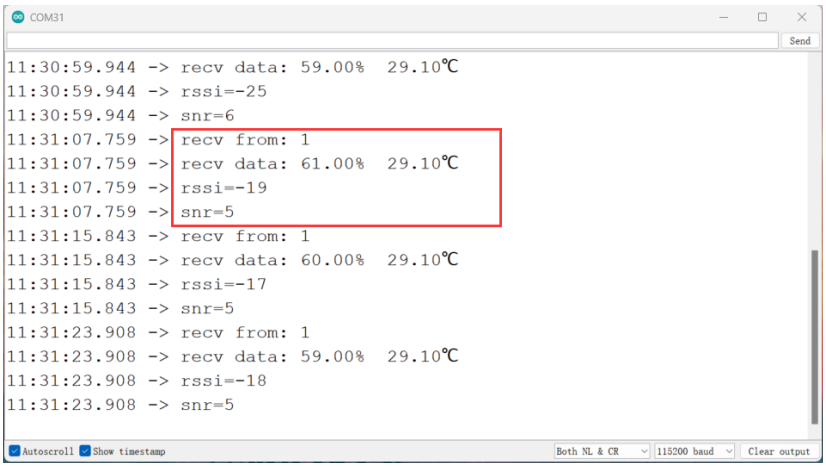
Serial output from the transmitter:



Serial output from the first group of receivers:



Serial output from the second group of receivers:

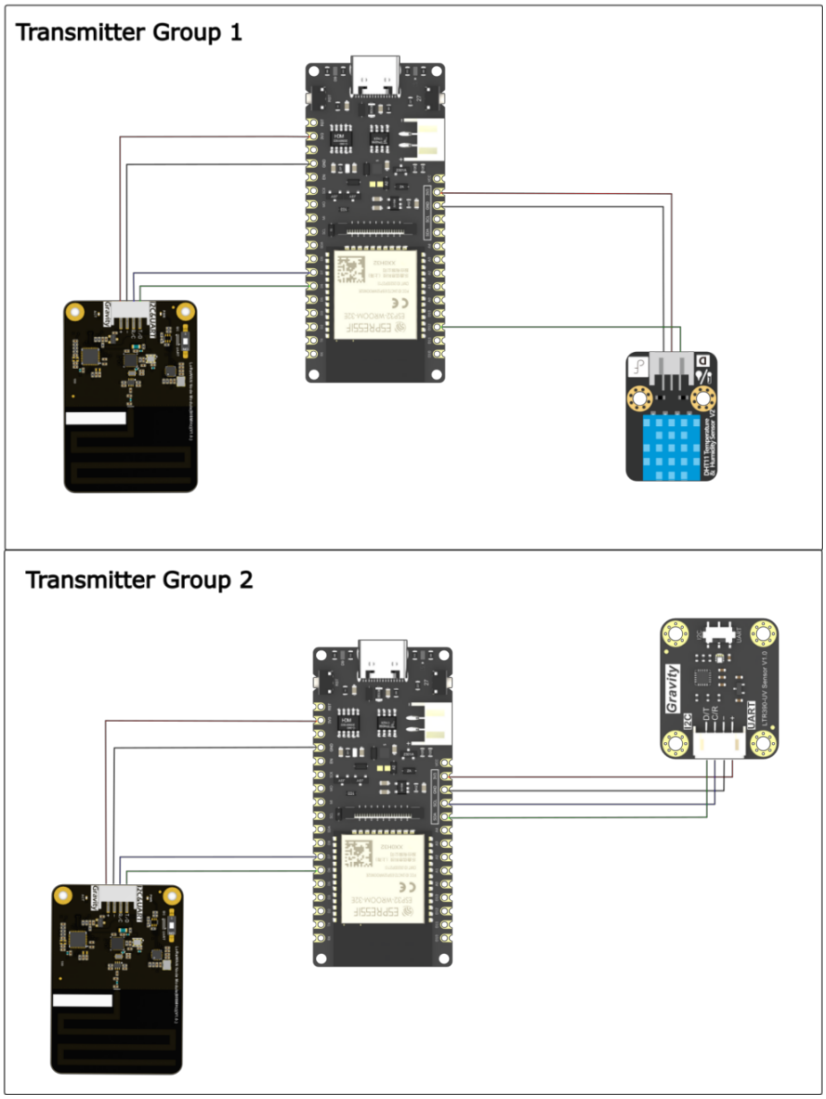


7.1.5 Two-to-One Communication Example

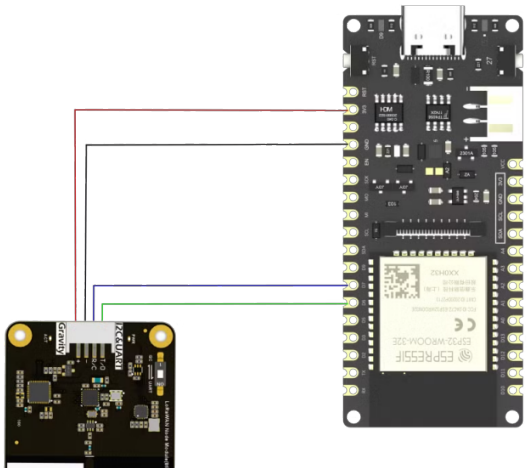
This example demonstrates the second type of one-to-many communication mode: setting nodes with addresses 1 and 2 as transmitters, and the node with address 3 as the receiver, achieving a "two-to-one" communication scenario.

Hardware Connections:

- Transmitter:



- Receiver:



Example Code:

Transmitter Group 1 Program: Sets the node address to 1 and sends temperature and humidity data to the node with address 3.

```
#include <DFRobot_LWNode.h>
#include <dht11.h>
dht11 DHT;
#define DHT11_PIN 4
#define FREQ  914900000
DFRobot_LWNode_UART node(1); // Set node addr

void setup( void ) {
    Serial.begin(115200);
    Serial1.begin(9600, SERIAL_8N1, /*rx =*/D
    delay(5000);
    node.begin(/*communication UART*/&Serial1
    const uint32_t loraConfig[] = {FREQ, DBM2
    while(!node.setFreq(loraConfig[0])  ||
          !node.setEIRP(loraConfig[1])  ||
          !node.setBW(loraConfig[2])    ||
          !node.setSF(loraConfig[3])    ||
          !node.start()) {
        Serial.println("LoRa init failed");
        delay(2000);
    }
}

void loop( void ){
    DHT.read(DHT11_PIN);   // Read data from
    Serial.print(DHT.humidity,1);
    Serial.print(",\t");
    Serial.println(DHT.temperature,1);
    String DHT11_DATE=String(DHT.humidity)+"%
    delay(2000);
```

```
    node.sendPacket(3, DHT11_DATE);  // Send
    node.sleep(5000);
}
```

Transmitter Group 2 Program: Sets the node address to 2 and sends UV sensor data to the node with address 3.

```
#include <DFRobot_LWNode.h>
#define FREQ  914900000

// Initialize LoRa and UV sensor
#include "DFRobot_LTR390UV.h"
DFRobot_LTR390UV ltr390(/*addr = */LTR390UV_D
DFRobot_LWNode_UART node(2);  // Set node add

void setup( void ) {
    Serial.begin(115200);
    Serial1.begin(9600, SERIAL_8N1, /*rx =*/D
    delay(5000);
    node.begin(/*communication UART*/&Serial1
    const uint32_t loraConfig[] = {FREQ, DBM2
    while(!node.setFreq(loraConfig[0])  ||
          !node.setEIRP(loraConfig[1])  ||
          !node.setBW(loraConfig[2])    ||
          !node.setSF(loraConfig[3])    ||
          !node.start()) {
        Serial.println("LoRa init failed");
        delay(2000);
    }

    // Initialize UV sensor
    while(ltr390.begin() != 0){
    Serial.println(" Sensor initialize failed
    delay(1000);
  }
    Serial.println(" Sensor  initialize succe

    // Configure UV sensor
    ltr390.setALSOrUVSMeasRate(ltr390.e18bit,
    ltr390.setALSOrUVSGain(ltr390.eGain3);
    ltr390.setMode(ltr390.eUVSMode);
}
```

```
void loop(void) {
    // Read UV sensor data
    uint32_t uv = ltr390.readOriginalData();
    Serial.print("UV data: ");
    Serial.println(uv);

    // Prepare and send UV data to node with
    String ltr390_DATE = String(uv);
    node.sendPacket(3, ltr390_DATE);

    // Sleep to conserve power (adjust if nee
    node.sleep(5000);
}
```

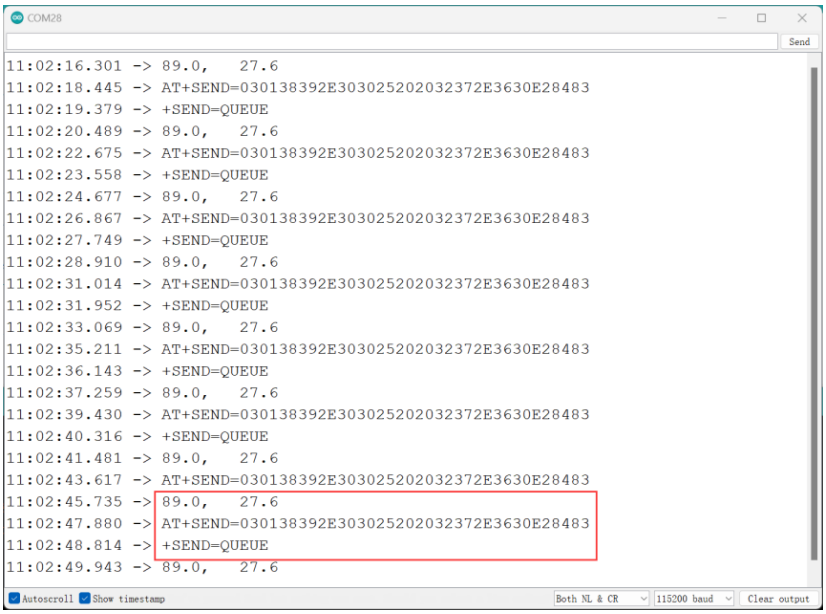
Receiver Program: Set the receiver node address to 3 and keep it in message receiving mode.

```
#include <DFRobot_LWNode.h>
#define FREQ  914900000
DFRobot_LWNode_UART node(3); // Set node addr

void rxCBFunc(uint8_t from, void *buffer, uir
    char *p = (char *)buffer;
    Serial.print("recv from: ");
    Serial.println(from, HEX);
    Serial.print("recv data: ");
    for(uint8_t i = 0; i < size; i++){
        Serial.print(p[i]);
    }
    Serial.println();
    Serial.println("Text:");
    Serial.println((char *)buffer);
    Serial.print("rssi=");Serial.println(rssi
    Serial.print("snr=");Serial.println(snr);
}

void setup( void ){
    Serial.begin(115200);
    Serial1.begin(9600, SERIAL_8N1, /*rx =*/D
    delay(5000);
    node.begin(/*communication UART*/&Serial1
    const uint32_t loraConfig[] = {FREQ, DBM2
    while(!node.setFreq(loraConfig[0])  ||
          !node.setEIRP(loraConfig[1])  ||
          !node.setBW(loraConfig[2])    ||
          !node.setSF(loraConfig[3])    ||
          !node.start()) {
        Serial.println("LoRa init failed");
        delay(2000);
    }
    node.setRxCB(rxCBFunc);
}
```

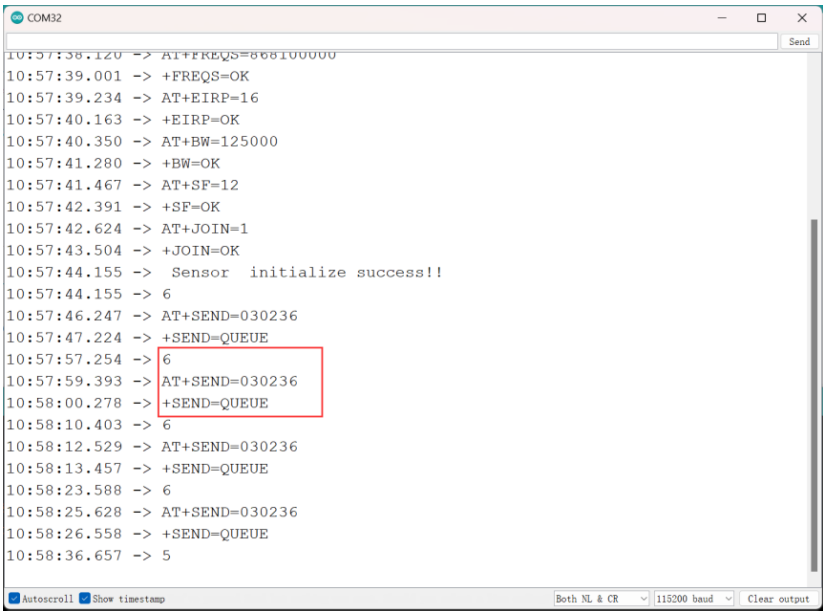
```
void loop( void ){
    node.sleep(5000);
}
```

Result:

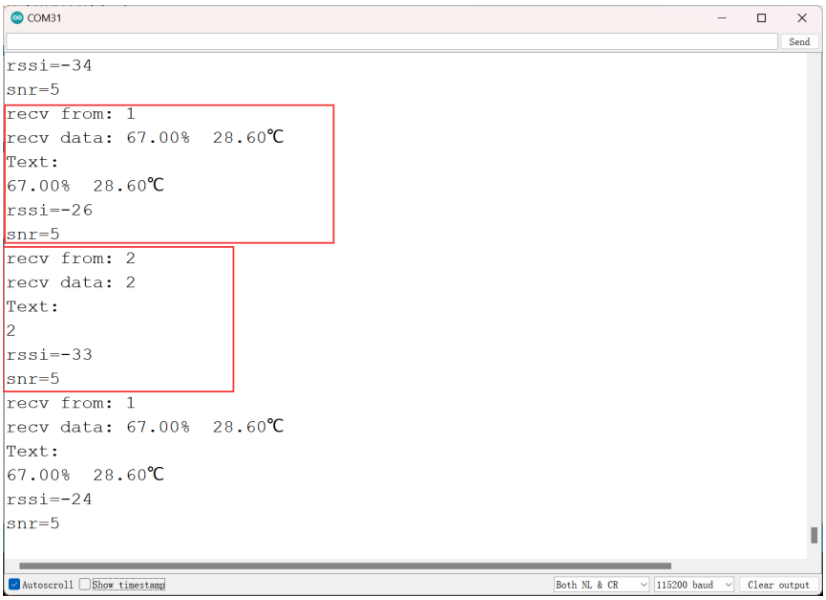
Serial output of transmitter Group 1:



Serial Output of Transmitter Group 2:



Serial Output of Receiver: Data received from Transmitter Group 1 and Group 2.



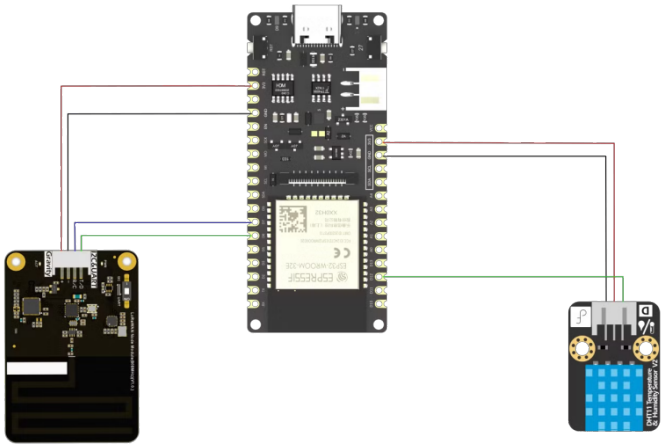
7.1.6 Data Relay Application Example

In addition to direct data transmission and reception, nodes also support data relay to extend communication range.

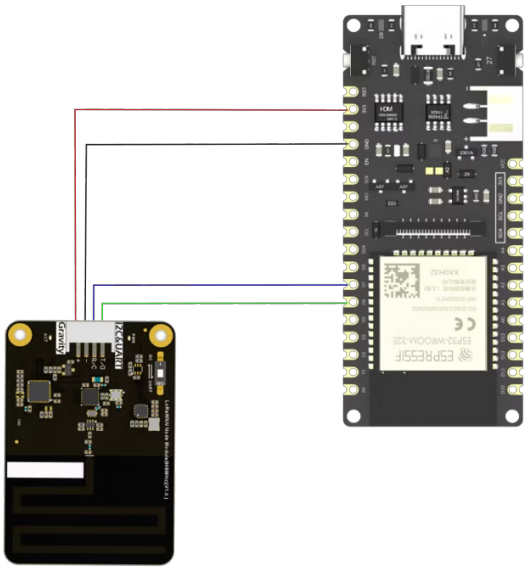
In this example, three nodes (A, B, and C) with addresses 1, 2, and 3 are used to demonstrate the A→B→C data relay process. Node A sends data to Node B, which then forwards it to Node C. This allows Node C, which is outside the direct communication range of Node A, to receive data from Node A.

Hardware Connection:

- Node A:



- Both Node B and Node C use the following connection method:



Example Code:

Node A Program: Set the node address to 1 and send temperature and humidity sensor data to Node B with address 2.

```
#include <DFRobot_LWNode.h>
#include <dht11.h>
dht11 DHT;
#define DHT11_PIN 4
#define FREQ  914900000
DFRobot_LWNode_UART node(1);  // Set node add

void setup( void ) {
    Serial.begin(115200);
    Serial1.begin(9600, SERIAL_8N1, /*rx =*/D
    delay(5000);
    node.begin(/*communication UART*/&Serial1
    const uint32_t loraConfig[] = {FREQ, DBM2
    while(!node.setFreq(loraConfig[0])  ||
          !node.setEIRP(loraConfig[1])  ||
          !node.setBW(loraConfig[2])    ||
          !node.setSF(loraConfig[3])    ||
          !node.start()) {
        Serial.println("LoRa init failed");
        delay(2000);
    }
}

void loop( void ){
    DHT.read(DHT11_PIN);  // Read temperature
    Serial.print(DHT.humidity,1);
    Serial.print(",\t");
    Serial.println(DHT.temperature,1);
    String DHT11_DATE=String(DHT.humidity)+"%
    delay(2000);
    node.sendPacket(2, DHT11_DATE);  // Send
    node.sleep(1000);
```

```
}
```

Node B Program: Set the node address to 2, receive data from Node A, and forward it to Node C with address 3.

```
#include <DFRobot_LWNode.h>
#define FREQ  914900000
DFRobot_LWNode_UART node(2);   // Set node ac
char p[36];

void rxCBFunc(uint8_t from, void *buffer, uir
    memcpy(p,buffer,size);  // Memory copy, c
    Serial.print("recv from: ");
    Serial.println(from, HEX);
    Serial.print("recv data: ");
    for(uint8_t i = 0; i < size; i++){
        Serial.print(p[i]);
    }
    Serial.println();
    Serial.print("rssi=");Serial.println(rssi
    Serial.print("snr=");Serial.println(snr);
}

void setup( void ){
    Serial.begin(115200);
    Serial1.begin(9600, SERIAL_8N1, /*rx =*/D
    delay(5000);
    node.begin(/*communication UART*/&Serial1
    const uint32_t loraConfig[] = {FREQ, DBM2
    while(!node.setFreq(loraConfig[0])  ||
          !node.setEIRP(loraConfig[1])  ||
          !node.setBW(loraConfig[2])    ||
          !node.setSF(loraConfig[3])    ||
          !node.start()) {
        Serial.println("LoRa init failed");
        delay(2000);
    }
```

```
    }
    node.setRxCB(rxCBFunc);
}
```

```
void loop( void ){
    node.sleep(5000);
    node.sendPacket(3, p); // Forward the rec
}
```

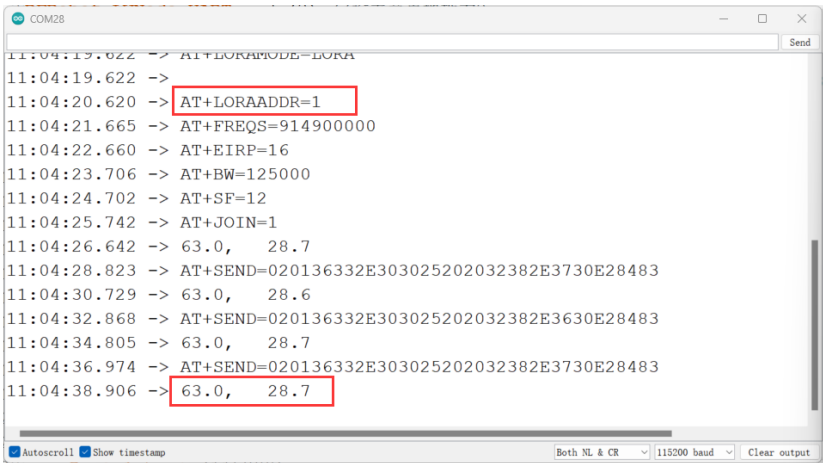
Node C Program: Set the node address to 3 and keep it in message receiving mode to receive data from Node B.

```
#include <DFRobot_LWNode.h>
#define FREQ  914900000
DFRobot_LWNode_UART node(3);// Set the node a

void rxCBFunc(uint8_t from, void *buffer, uir
    char *p = (char *)buffer;
    Serial.print("recv from: ");
    Serial.println(from, HEX);
    Serial.print("recv data: ");
    for(uint8_t i = 0; i < size; i++){
        Serial.print(p[i]);
    }
    Serial.println();
    Serial.print("rssi=");Serial.println(rssi
    Serial.print("snr=");Serial.println(snr);
}

void setup( void ){
    Serial.begin(115200);
    Serial1.begin(9600, SERIAL_8N1, /*rx =*/D
    delay(5000);
    node.begin(/*communication UART*/&Serial1
    const uint32_t loraConfig[] = {FREQ, DBM2
    while(!node.setFreq(loraConfig[0])  ||
          !node.setEIRP(loraConfig[1])  ||
          !node.setBW(loraConfig[2])    ||
          !node.setSF(loraConfig[3])    ||
          !node.start()) {
        Serial.println("LoRa init failed");
        delay(2000);
    }
    node.setRxCB(rxCBFunc);
```

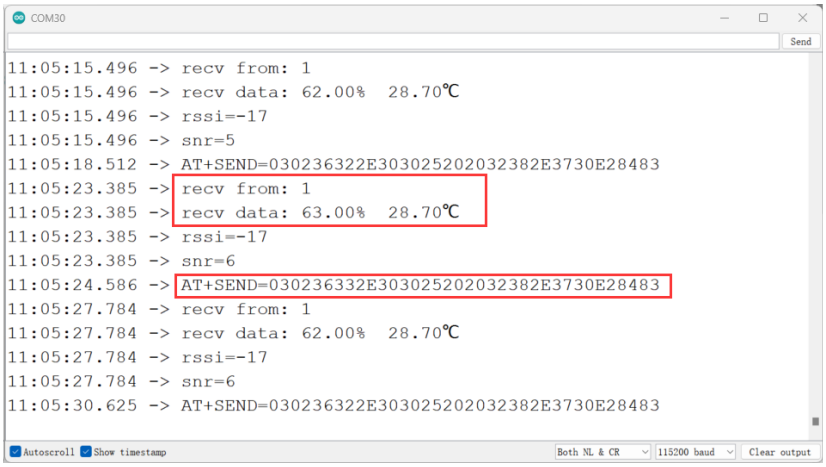
```
}

void loop( void ){

    node.sleep(5000);
}
```

Result:

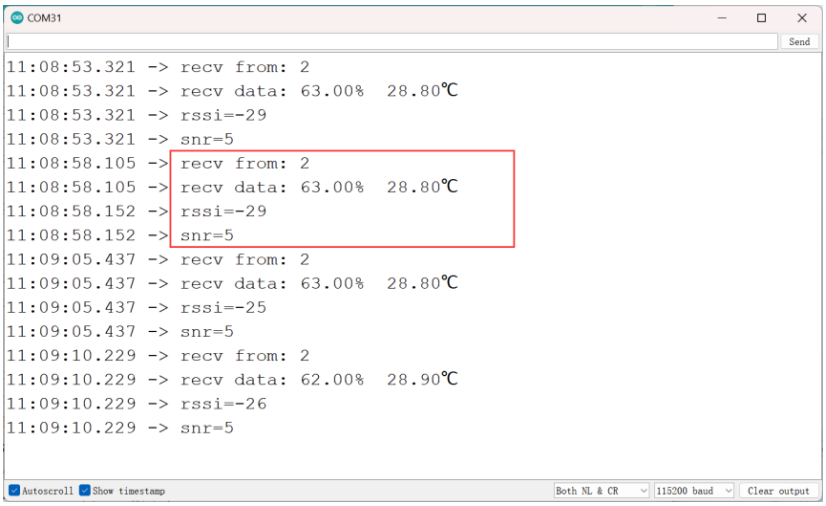
Serial Output of Node A:



Serial Output of Node B: Received data from Node A and forwarded it.



Serial Output of Node C: Received data from Node B.



8. LoRaWAN Tutorial

8.1 FireBeetle ESP32-E Tutorial

8.1.1 Hardware Preparation

- FireBeetle 2 ESP32-E (https://www.dfrobot.com.cn/goods-3009.html) (SKU: DFR0654) ×1
- Gravity: LoRaWAN Node Module（US915） (https://www.dfrobot.com/product-2927.html)(DFR1115-915) ×1
- PH2.0-4P Cable ×1
- USB Data Cable ×1

8.1.2 Software Preparation

- Download Arduino IDE: Click to Download Arduino IDE (https://www.arduino.cc/en/Main/Software)
- Install SDK: Visit the FireBeetle 2 ESP32-E WIKI Page (https://wiki.dfrobot.com.cn/

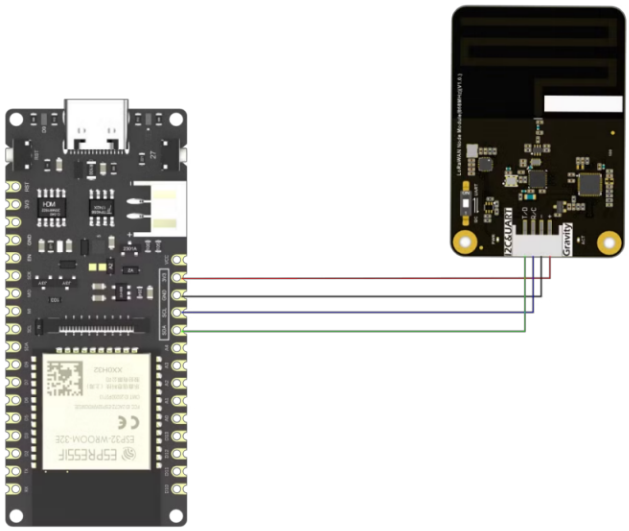
(https://wiki.dfrobot.com.cn/_SKU_DFR0654_FireBeetle_Board_ESP32_E#target_6) to find the SDK installation tutorial.

- Download Arduino Library: Click to download the

  DFRobot_LWNode Library (https://github.com/cdjq/DFRobot_LWNode) and refer to this link for guidance: How to Install a Library? (http://www.dfrobot.com.cn/community/forum.php?mod=viewthread&tid=1854&page=1&extra=#pid6955)

8.1.3 Hardware Connection

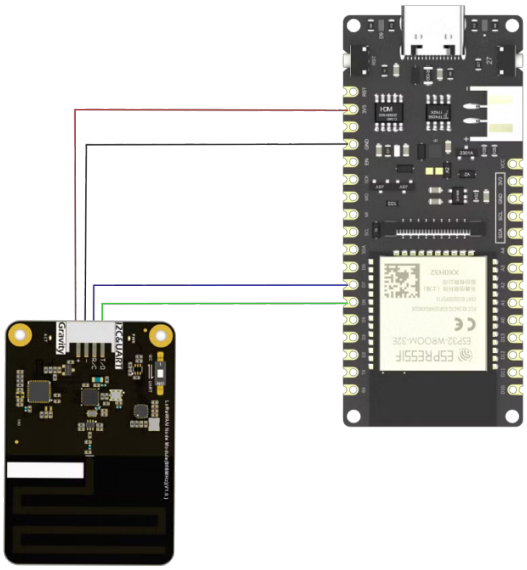
I2C Communication Example (This connection method is used in all the following examples)



Pin Connection Description:

- Node Module: SDA Pin --- (Connects to) --- Main Controller: 21/SDA
- Node Module: SCL Pin --- (Connects to) --- Main Controller: 22/SCL
- Node Module: - Pin --- (Connects to) --- Main Controller: GND
- Node Module: + Pin --- (Connects to) --- Main Controller: 3V3

UART Communication Example



Pin Connection Description:

- Node Module: TX Pin --- (Connects to) --- Main Controller: 14/D6
- Node Module: RX Pin --- (Connects to) --- Main Controller: 13/D7
- Node Module: - Pin --- (Connects to) --- Main Controller: GND
- Node Module: + Pin --- (Connects to) --- Main Controller: 3V3

8.1.3.1 OTAA Network Joining

Note: Before using this example, ensure that the gateway is set to **manual device addition** mode and configured to allow **OTAA** for device registration.

Example 1: OTAA Network Joining and Sending Data to the Gateway

Sample Code:

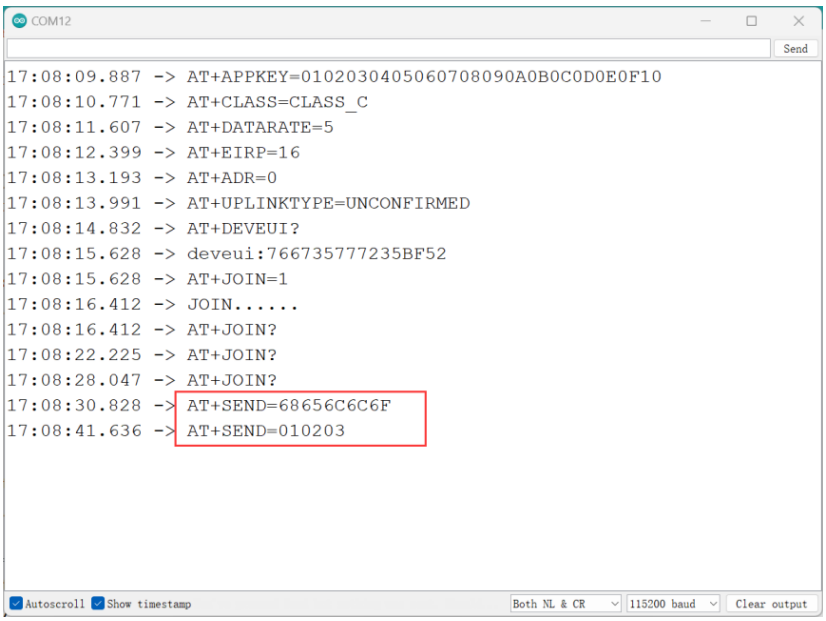
```
#include "DFRobot_LWNode.h"
#define REGION US915
#define DATARATE  DR3
#define SUBBAND   2
// OTAA credentials (replace these with your
const char _APPEUI[]={"DFDFDFDF00000000"} ;
const char _APPKEY[]={"0102030405060708090A0E

DFRobot_LWNode_IIC node(_APPEUI,_APPKEY);

void setup(void){
    Serial.begin(115200);
    node.begin(/*communication IIC*/&Wire,/*c
    while(!node.setRegion(REGION)){
        delay(2000);
        Serial.println("REGION set fail");
    }
    if(!node.setAppEUI(_APPEUI)){
        Serial.println("AppEUI set fail");
    }
    if(!node.setAppKEY(_APPKEY)){
        Serial.println("AppKEY set fail");
    }
    if(!node.setDevType(CLASS_C)){
        Serial.println("DevType set fail");
    }
    while (!node.setDataRate(DATARATE)) {
        delay(2000);
        Serial.println("DataRate set fail");
    }
    while (!node.setEIRP(DBM16)) {
        delay(2000);
        Serial.println("EIRP set fail");
    }
    while(!node.setSubBand(SUBBAND)){
```

```
    while(!node.setSubBand(SUBBAND)){
        Serial.println("SubBand set fail");
    }
    while(!node.enableADR(false)){
        delay(2000);
        Serial.println("ADR set fail");
    }
    while(!node.setPacketType(UNCONFIRMED_PAC
        delay(2000);
        Serial.println("Packet type set fail"
    }

    String deveui = node.getDevEUI();
    Serial.print("DEVEUI: ");
    Serial.println(deveui);

    Serial.print("DATARATE: ");
    Serial.println(node.getDataRate());

    Serial.print("EIRP: ");
    Serial.println(node.getEIRP());

  //Attempt to join LoRaWAN network
    if(node.join()){
        Serial.println("JOIN......");
    }
    while(!node.isJoined()){
      delay(5000);
    }
}

void loop(){
   node.sendPacket("hello");  // Send text me
   node.sleep(10 * 1000);

   uint8_t buf[3]={1,2,3};  // Send binary me
```

```
    node.sendPacket(buf,3);
    node.sleep(10 * 1000);
}
```

Result:

On the node side: The serial monitor prints a message indicating successful network joining, and the node sends data to the gateway every 10 seconds.



Example 2: Receiving Data from the Gateway via Polling after OTAA Network Joining

Sample Code:

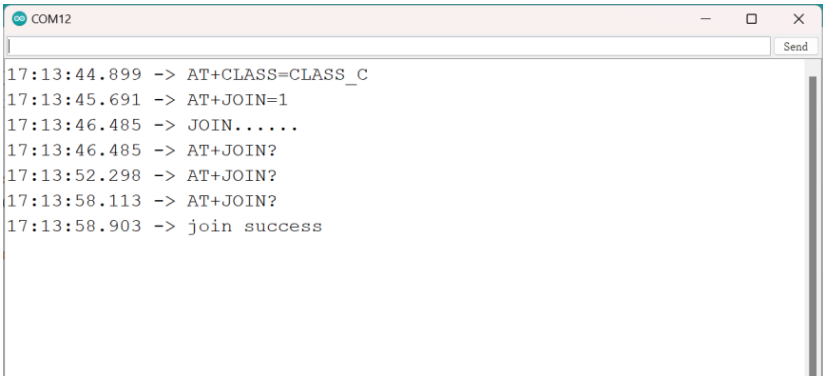
```
#include "DFRobot_LWNode.h"
#define REGION US915
#define DATARATE  DR3
#define SUBBAND   2
// OTAA credentials (replace these with your
const char _APPEUI[]={"DFDFDFDF00000000"} ;
const char _APPKEY[]={"0102030405060708090A0E
uint8_t buf[256]={0x0};  // Buffer to store r

DFRobot_LWNode_IIC node(_APPEUI,_APPKEY);

void setup(void){
    Serial.begin(115200);
    node.begin(/*communication IIC*/&Wire,/*c
    while(!node.setRegion(REGION)){
        delay(2000);
        Serial.println("REGION set fail");
    }
    while(!node.setDevType(CLASS_C)){
        delay(2000);
        Serial.println("DevType set fail");
    }
    while(!node.setSubBand(SUBBAND)){
        delay(2000);
        Serial.println("SubBand set fail");
    }
    String deveui = node.getDevEUI();
    Serial.print("DEVEUI: ");
    Serial.println(deveui);
   // Attempt to join the LoRaWAN network
    if(node.join()){
        Serial.println("JOIN.....");
```

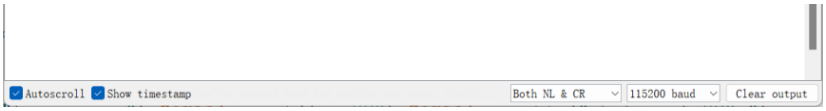
```
    }
    while(!node.isJoined()){
        delay(5000);
```

```
    }
    Serial.println("join success");
}

void loop(){
  uint8_t len = node.readData(buf);  // Read
  // If data is received, print it in both he
  if(len > 0){
    Serial.print("\nreceive ");
    Serial.print(len,HEX);
    Serial.println(" bytes  \nHEX:");

    for(uint8_t i = 0;i<len;i++){
       Serial.print(buf[i],HEX);
    }
    Serial.println();
    Serial.println("Text:");
    Serial.println((char *)buf);
  }
  delay(500);
}
```

Result:

On the node side: The serial monitor prints "**join success**", and the node enters a polling state to check whether data is received in the buffer.





8.1.3.2 ABP Network Joining

Note: Before using this example, ensure that the gateway is set to **manual device addition** mode and configured to allow **ABP** for device registration.

Example 1: ABP Network Joining and Sending Data to the Gateway

Sample Code:

```
#include "DFRobot_LWNode.h"
// ABP credentials (replace these with your a
const char NWKSKEY[]={"878888888888888888888
const char APPSKEY[]={"898888888888888888888
uint32_t devAddr = 0xDF000011;

#define REGION US915
#define DATARATE  DR3
#define SUBBAND   2
DFRobot_LWNode_IIC node(devAddr, NWKSKEY, APP

void setup(void) {
    Serial.begin(115200);
    node.begin(/*communication IIC*/&Wire,/*d
    while(!node.setRegion(REGION)){
        delay(2000);
        Serial.println("REGION set fail");
    }
    if (!node.setAppSKey(APPSKEY)) {
      Serial.println("APPSKEY set fail");
    }
    if (!node.setNwkSKey(NWKSKEY)) {
      Serial.println("NWKSKEY set fail");
    }
    if (!node.setDevAddr(devAddr)) {
      Serial.println("devAddr set fail");
    }
    while (!node.setDataRate(DATARATE)) {
        delay(2000);
        Serial.println("DataRate set fail");
    }
    while (!node.setEIRP(DBM16)) {
        delay(2000);
        Serial.println("EIRP set fail");
    }
```

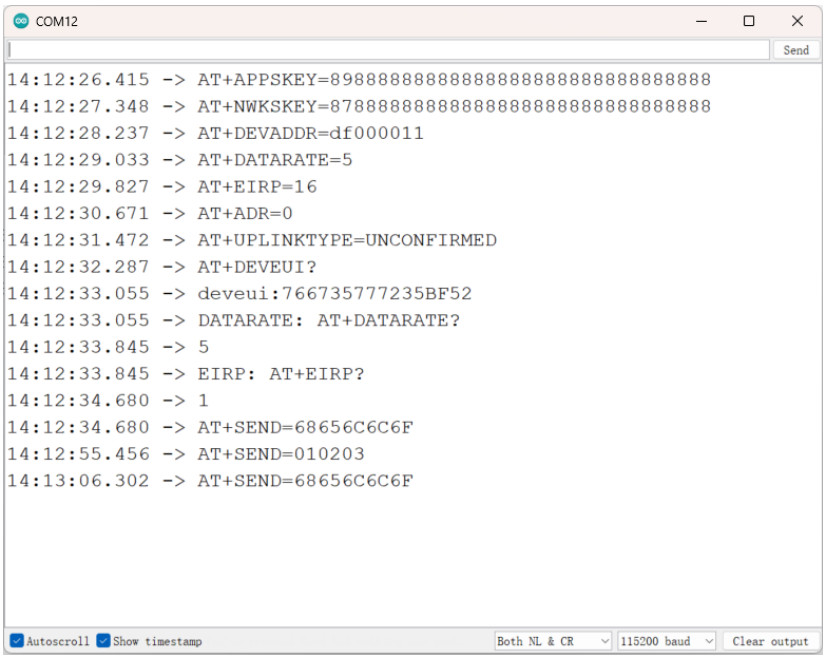
```
    }
    while(!node.setSubBand(SUBBAND)) {
        delay(2000);
        Serial.println("SubBand set fail");
```

```
    }
    while(!node.enableADR(false)) {
        delay(2000);
        Serial.println("ADR set fail");
    }
    while(!node.setPacketType(UNCONFIRMED_PAC
        delay(2000);
        Serial.println("Packet type set fail"
    }
    node.start();  // Start LoRaWAN communica
    String deveui = node.getDevEUI();
    Serial.print("DEVEUI: ");
    Serial.println(deveui);

    Serial.print("DATARATE: ");
    Serial.println(node.getDataRate());

    Serial.print("EIRP: ");
    Serial.println(node.getEIRP());
}

void loop() {
  node.sendPacket("hello");  // Send a text m
  node.sleep(10 * 1000);

  uint8_t buf[3] = {1, 2, 3};  // Send a bina
  node.sendPacket(buf, 3);
  node.sleep(10 * 1000);
}
```

Result:

On the node side: The serial monitor shows that the node

On the node side, the serial monitor shows that the node sends data to the gateway every 10 seconds.



Example 2: Receiving Data from the Gateway via Polling after ABP Network Joining

Sample Code:

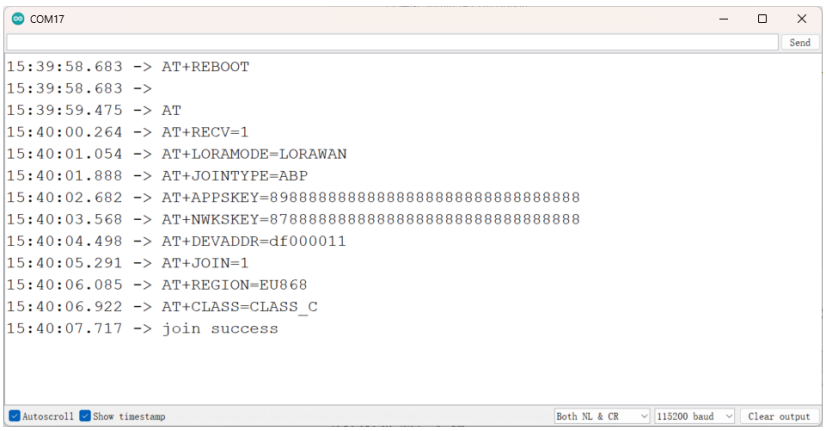
```
#include "DFRobot_LWNode.h"
// ABP credentials (replace these with your a
const char NWKSKEY[]={"878888888888888888888
const char APPSKEY[]={"898888888888888888888
uint32_t devAddr = 0xDF000011;
uint8_t buf[256];    // Buffer to store recei

#define REGION US915
#define DATARATE  DR3
#define SUBBAND   2
DFRobot_LWNode_IIC node(devAddr,NWKSKEY,APPSK

void setup(void){
    Serial.begin(115200);
    delay(5000);
    node.begin(/*communication IIC*/&Wire,/*c
    while(!node.setRegion(REGION)){
        delay(2000);
        Serial.println("REGION set fail");
    }
    while(!node.setDevType(CLASS_C)){
        delay(2000);
        Serial.println("DevType set fail");
    }
    String deveui = node.getDevEUI();
    Serial.print("DEVEUI: ");
    Serial.println(deveui);
     // Start LoRaWAN communication in ABP mo
    node.start();
    Serial.println("join success");
}
```

```
void loop(){
    uint8_t len = node.readData(buf);   // Ch
```

```
    if(len > 0){
        Serial.print("\nreceive ");
        Serial.print(len);
        Serial.println(" bytes  \nHEX:");
        for(uint8_t i = 0; i < len; i++){
            Serial.print(buf[i],HEX);    // P
        }
        Serial.println();
        Serial.println("Text:");    // Print
        Serial.println((char *)buf);
    }
    node.sleep(500);
}
```

Result:

On the node side: The serial monitor displays the message "**join success**", and the node enters a polling state to check whether data is received from the buffer.



10. API Library

```
/**
 * @fn setRegion
 * @brief Sets the LoRaWAN region.
 * @param region Region enum value
 * @return Returns true if successful, othe
 */
bool setRegion(eRegion_t region);

/**
 * @fn setFreq
 * @brief Sets the frequency.
 * @param freq Frequency value
 * @return Returns true if successful, othe
 */
bool setFreq(uint32_t freq);

/**
 * @fn setBW
 * @brief Sets the bandwidth.
 * @param bw Bandwidth value
 * @return Returns true if successful, othe
 */
bool setBW(uint32_t bw);

/**
 * @fn setSF
 * @brief Sets the spreading factor.
 * @param sf Spreading factor value
 * @return Returns true if successful, othe
 */
bool setSF(uint8_t sf);
```

```
/**
 * @fn setRxCB
 * @brief Sets the receive callback functic
 * @param callback Pointer to the callback
 */
void setRxCB(rxCB *callback);

/**
 * @fn setRxCB
 * @brief Sets the receive callback functic
 * @param callback Pointer to the callback
 */
void setRxCB(rxCB3 *callback);

/**
 * @fn setAppEUI
 * @brief Sets the Application EUI.
 * @param appeui Application EUI
 * @return Returns true if successful, othe
 */
bool setAppEUI(const char *appeui);

/**
 * @fn setAppKEY
 * @brief Sets the Application Key.
 * @param appkey Application Key
 * @return Returns true if successful, othe
 */
bool setAppKEY(const char *appkey);

/**
 * @fn setDevType
 * @brief Sets the device type.
 * @param classType Device class enum value
 * @return Returns true if successful, othe
 */
bool setDevType(eDeviceClass_t classType);
```

```
/**
 * @fn setDataRate
 * @brief Sets the data rate.
 * @param dataRate Data rate enum value
```

```
 * @return Returns true if successful, othe
 */
bool setDataRate(eDataRate_t dataRate);

/**
 * @fn setEIRP
 * @brief Sets the transmission power.
 * @param EIRP Transmission power value
 * @return Returns true if successful, othe
 */
bool setEIRP(uint8_t EIRP);

/**
 * @fn setSubBand
 * @brief Sets the sub-band.
 * @param subBand Sub-band value
 * @return Returns true if successful, othe
 */
bool setSubBand(uint8_t subBand);

/**
 * @fn enableADR
 * @brief Enables or disables Adaptive Data
 * @param adr If true, enables ADR; if fals
 * @return Returns true if successful, othe
 */
bool enableADR(bool adr);

/**
 * @fn setDevAddr
 * @brief Sets the device address.
 * @param devAddr Device address
```

```
 * @return Returns true if successful, othe
 */
bool setDevAddr(const uint32_t devAddr);
```

```
/**
 * @fn setAppSKey
 * @brief Sets the Application Session Key.
 * @param appSKey Application Session Key
 * @return Returns true if successful, othe
 */
bool setAppSKey(const char *appSKey);

/**
 * @fn setNwkSKey
 * @brief Sets the Network Session Key.
 * @param nwkSKey Network Session Key
 * @return Returns true if successful, othe
 */
bool setNwkSKey(const char *nwkSKey);

/**
 * @fn join
 * @brief Initiates the LoRaWAN join proced
 * @return Returns true if successfully ini
 */
bool join();

/**
 * @fn start
 * @brief Starts the device's operation.
 * @return Returns true if successful, othe
 */
bool start();

/**
 * @fn setLoRaAddr
 * @brief Sets the LoRa address.
 * @param addr LoRa address
 * @return Returns true if successful, othe
```

```
 */
bool setLoRaAddr(uint8_t addr);

/**
```

```
 * @fn isJoined
 * @brief Checks if the device is already j
 * @return Returns true if joined, otherwis
 */
bool isJoined();

/**
 * @fn sendPacket
 * @brief Sends a data packet.
 * @param v Value to be sent
 * @return Returns true if successful, othe
 */
bool sendPacket(double v);
bool sendPacket(int32_t v);
bool sendPacket(uint32_t v);
bool sendPacket(void *buffer, uint8_t size)

/**
 * @fn sendPacket
 * @brief Sends a data packet to a specific
 * @param addr Destination address
 * @param v Value to be sent
 * @return Returns true if successful, othe
 */
bool sendPacket(uint8_t addr, double v);
bool sendPacket(uint8_t addr, int32_t v);
bool sendPacket(uint8_t addr, uint32_t v);
bool sendPacket(uint8_t addr, void *buffer,

/**
 * @fn sendPacket
 * @brief Sends a string data packet.
```

```
 * @param data String data to be sent
 * @return Returns true if successful, othe
 */
bool sendPacket(String data);
```

```
/**
 * @fn sendPacket
 * @brief Sends a string data packet to a s
 * @param addr Destination address
 * @param data String data to be sent
 * @return Returns true if successful, othe
 */
bool sendPacket(uint8_t addr, String data);

/**
 * @fn sendATCmd
 * @brief Sends a generic AT command.
 * @param cmd Preformatted AT command withc
 * @return The response to the AT command
 */
String sendATCmd(String cmd);

/**
 * @fn sendATCmdTest
 * @brief Sends a test AT command.
 * @param cmd Test AT command
 * @return The response to the test AT comm
 */
String sendATCmdTest(char *cmd);

/**
 * @fn setPacketType
 * @brief Sets the packet type.
 * @param type Packet type (CONFIRMED_PACKE
 * @return Returns true if successful, othe
 */
bool setPacketType(ePacketType_t type = UNC

/**
```

```
 * @fn getDevEUI
 * @brief Retrieves the device EUI.
 * @return The device EUI as a string
 */
```

```
String getDevEUI();

/**
 * @fn getNetID
 * @brief Retrieves the network ID.
 * @return 3-byte network ID information
 */
uint32_t getNetID();

/**
 * @fn getDevAddr
 * @brief Retrieves the device address. In
 * @return 4-byte device address informatic
 */
uint32_t getDevAddr();

/**
 * @fn getDataRate
 * @brief Retrieves the current data rate.
 * @return The current data rate
 */
uint8_t getDataRate();

/**
 * @fn getEIRP
 * @brief Retrieves the current transmissic
 * @return The current transmission power
 */
uint8_t getEIRP();

/**
 * @fn getRSSI
```

```
     * @brief Retrieves the Received Signal Str
     * @return The RSSI value
     */
    int16_t getRSSI();
```

```
    /**
     * @fn getSNR
     * @brief Retrieves the Signal-to-Noise Rat
     * @return The SNR value
     */
    int8_t getSNR();

    /**
     * @fn atTest
     * @brief Executes an AT test command.
     * @return The result of the test command
     */
    bool atTest();
```

11. Product Compatibility

This product is theoretically compatible with all 3.3V and 5V Arduino mainboards. The table below lists the testing status of this product on various mainboards.

Mainboard Name	Functioning Normally	Function Abnormal	Not Verified
Arduino Uno	√		
Arduino MEGA2560	√		
Arduino Leonardo	√		
FireBeetle-ESP8266	√		
FireBeetle-			

FireBeetle-ESP32	√		

Mainboard Name	Functioning Normally	Function Abnormal	Not Verified
FireBeetle 2 ESP32-E	√		
micro:bit	√		

12. Downloads

- DFR1115-915_STP_file.zip (https://dfimg.dfrobot.com/5d57611a3416442fa39bffca/wiki/a4b920b71e78428f169432dfa4e312d2.zip)
- DFR1115-915_2D_file.pdf (https://dfimg.dfrobot.com/5d57611a3416442fa39bffca/wiki/476259524eca3c4d4986ae6d08e4b8ac.pdf)

13. FAQ

1. **Q: Why does switching the DIP switch to change the communication mode not work?**

   A: After switching the communication mode while the module is powered on, you need to power cycle (reboot) the module for the change to take effect.

2. **Q: What is the communication range of the module?**

	Stable Communication Range	Maximum Communication Range

EU868 Version	1.2 km (urban)/4 km (open area)	1.4 km (urban)/4.4 km (open area)

	Stable Communication Range	Maximum Communication Range
US915 Version	1.5 km (urban)/4.5 km (open area)	2 km (urban)/4.7 km (open area)

The above test data is for reference only. Communication range is significantly affected by environmental factors (such as traffic density, weather conditions, and base station interference). The actual range should be confirmed through real-world deployment.

For more questions and interesting applications, you can **visit the forum** (https://mc.dfrobot.com.cn/forum.html) for reference or posting.

Back to Top