Ebyte LoRa E22 device for Arduino, esp32 or esp8266: specs and basic use – 1

BY [RENZO MISCHIANTI](https://mischianti.org/author/reef/) · PUBLISHED 25 SEPTEMBER 2020 · UPDATED 16 JUNE 2023

**Spread the love**



Ebyte LoRa E22 device for Arduino, esp32 or esp8266 specs basic usage

**LoRa or Long Range wireless data telemetry** is a technology pioneered by Semtech that operates at a lower frequency than NRF24L01 (433 MHz, 868 MHz, or 916 MHz against 2.4 GHz for the NRF24L01) but at thrice the distance (from 4000m to 10000m).

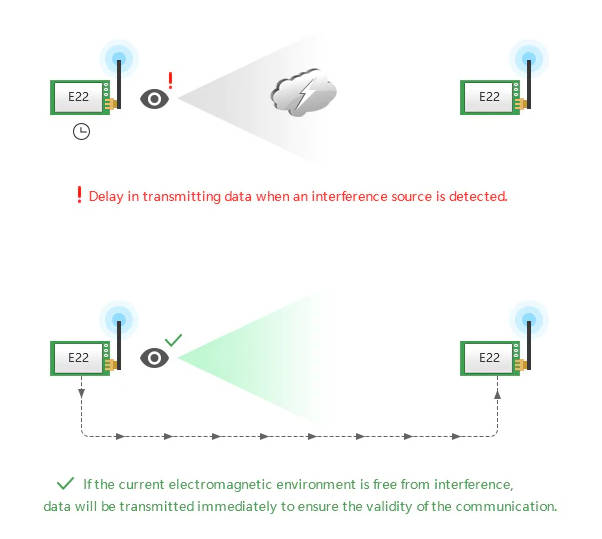
[Support Forum](https://mischianti.org/forums/forum/mischiantis-libraries/ebyte-lora-e22-uart-devices-sx1262-sx1268/)

Ebyte LoRa E22 device for Arduino, esp32 or esp8266 2 devices

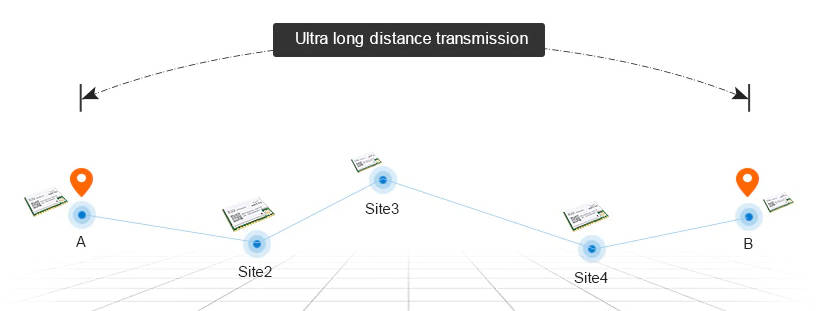
You can find here [AliExpress (433MHz 5.5Km)](https://s.click.aliexpress.com/e/_DlLGLbF) - [AliExpress (433MHz 10Km)](https://s.click.aliexpress.com/e/_DCGzJIt) - [AliExpress (868MHz 915Mhz 5.5Km)](https://s.click.aliexpress.com/e/_DczWHrb) - [AliExpress (868MHz 915Mhz 10Km)](https://s.click.aliexpress.com/e/_DeoQrNn)

**We are going to test E22-400T22D and E22-400T22S**. It is a wireless transceiver module, operates at 410 493 MHz based on original RFIC SX1268 from SEMTECH, transparent transmission is available, TTL level. The module adopts LORA spread spectrum technology.

Like the little brother e22 the module features **FEC Forward Error Correction algorithm**, which ensure its **high coding efficiency & good correction performance**. In the case of sudden interference, it can correct the interfered data packets automatically, so that the reliability and transmission range are improved correspondingly. **But without FEC, those da te packets can only be dropped**.  
And with the rigorous encryption & decryption, data interception becomes pointless. The function of **data compression can decrease the transmission time** & probability of being interference, while improving the reliability & transmission efficiency.

[](https://mischianti.org/wp-content/uploads/2020/07/Ebyte-LoRa-E22-device-for-Arduino-esp32-or-esp8266-carrier-sense.jpg)Ebyte LoRa E22 device for Arduino, esp32 or esp8266 carrier sense

But more than e22 this module have a **carrier sense** features, that wait transmission if there are some air interference, can be used like a **repeater**, have **watchdog** that if there is an error device restart and continue working, have **deep sleep** for better power saving, and first the sx1268 with **less power grant more power**.

[](https://mischianti.org/wp-content/uploads/2020/07/Ebyte-LoRa-E22-device-repeater-relay-function-ultra-long-distance.jpg)Ebyte LoRa E22 device repeater relay function ultra long distance

**Remotely configure** or read wireless module parameters by sending command packets wirelessly.

And finally have an important features the **RSSI**, you can check the signal strenght on all message sended.

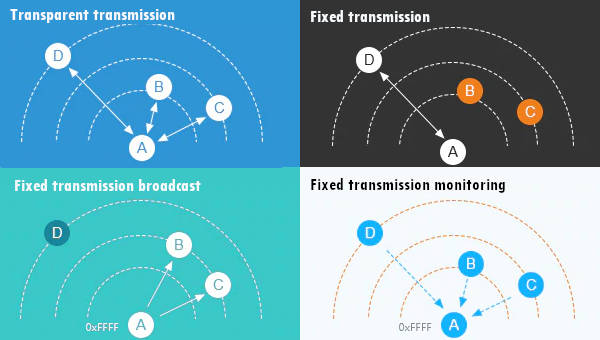
[](https://mischianti.org/wp-content/uploads/2020/07/Ebyte-LoRa-E22-device-for-Arduino-esp32-or-esp8266-RSSI-signal-strength.jpg)Ebyte LoRa E22 device for Arduino, esp32 or esp8266 RSSI signal strength

* [Operating and transmission type](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Operating_and_transmission_type)
  + [Transmission](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Transmission)
    - [Transparent transmission](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Transparent_transmission)
    - [Fixed transmission](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Fixed_transmission)
  + [Normal mode](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Normal_mode)
  + [Wake-up mode and power-saving mode](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Wake-up_mode_and_power-saving_mode)
  + [Program/sleep mode](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Programsleep_mode)
* [Specifications](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Specifications)
* [Pinout](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Pinout)
* [Normal mode connection](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Normal_mode_connection)
  + [Connecting Wemos D1 mini (esp8266) for a basic usage](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Connecting_Wemos_D1_mini_esp8266_for_a_basic_usage)
  + [Connecting esp32 for a basic usage](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Connecting_esp32_for_a_basic_usage)
  + [Connecting Arduino MKR for a basic usage](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Connecting_Arduino_MKR_for_a_basic_usage)
  + [Connecting Arduino for a basic usage](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Connecting_Arduino_for_a_basic_usage)
* [Simple communication sketch](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Simple_communication_sketch)
  + [Arduino sketch](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Arduino_sketch)
  + [Wemos D1 mini sketch](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Wemos_D1_mini_sketch)
  + [esp32 sketch](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#esp32_sketch)
  + [Arduino MKR sketch](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Arduino_MKR_sketch)
* [Library](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Library)
  + [Arduino sketch](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Arduino_sketch-2)
  + [Wemos D1 (esp8266) sketch](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Wemos_D1_esp8266_sketch)
  + [esp32 sketch](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#esp32_sketch-2)
  + [Arduino MKR sketch](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Arduino_MKR_sketch-2)
* [Thanks](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-specs-and-basic-usage-1/#Thanks)

Operating and transmission type

This device have some interesting function:

Transmission

LoRa E22 transmitting scenarios

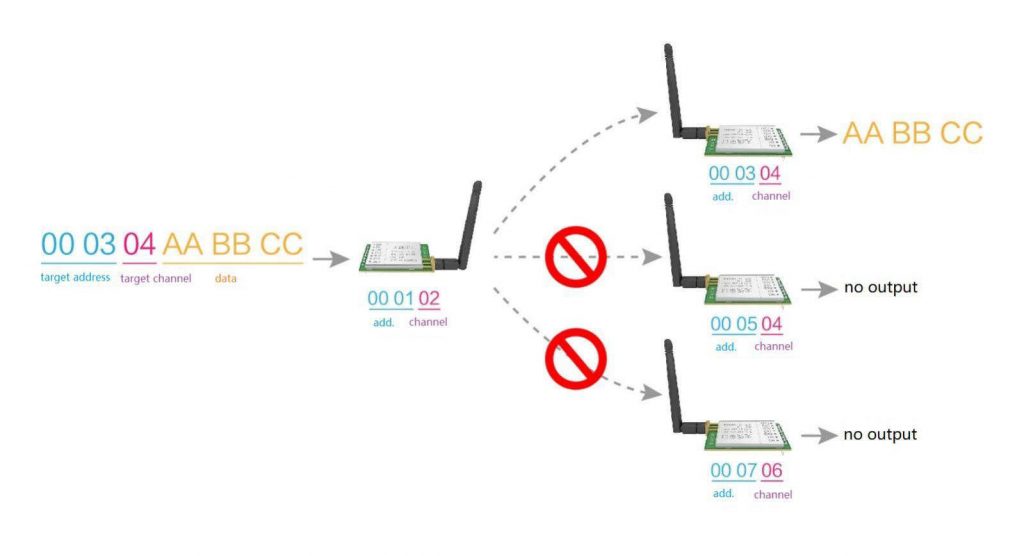
Transparent transmission

This can be considered like a “Demo mode”, by default you can send message to all device of same configured address and channel.

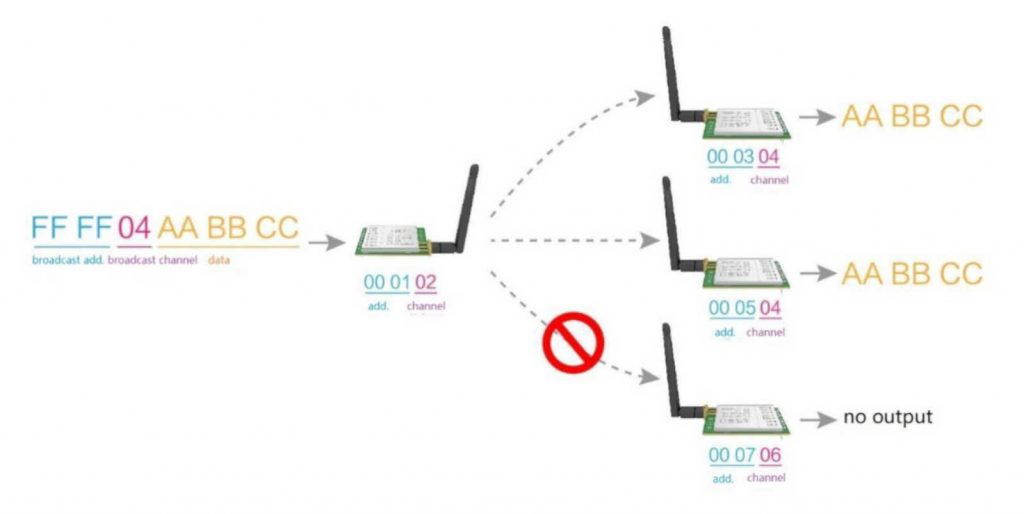
Fixed transmission

This type of transmission you can specify an address and a channel where where you want send the message.  
You can send message to a:

* Specified device with a predeterminated Address Low, Address High and Channel.

[](https://mischianti.org/wp-content/uploads/2019/10/fixedMessageToASpecifiedDevice.jpg)LoRa E22 Fixed message to a specified device

* Broadcast a message on predeterminated Channel.

[](https://mischianti.org/wp-content/uploads/2019/10/broadcastMessageToASpecifiedChannelDevice.jpg)Broadcast message to a set of channel devices

Normal mode

Simply send message.

[](https://www.pcbway.com/?from=mischianti723)

Wake-up mode and power-saving mode

As you can intend if a device is in Wake-up mode can “wake” one or more devices that are in power-saving mode with a preamble communication.

Program/sleep mode

With this configuration you can change configuration of your device.

Specifications

Here’s the specifications for the module:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Main Parameters | Value | | | Remarks |
| Min. | Typical | Max |
| Operating voltage（V） | 2.3 | 5.0 | 5.5 | ≥5.0 V ensures output power |
| Communication level（V） | – | 3.3 | – | For 5V TTL, it may be at risk of burning down |
| Operating temperature（℃） | -40 | – | 85 | Industrial design |
| Operating frequency（MHz） | 410.125 | 433.125 | 493.125 | Support ISM band |
| TX Current(mA) | – | 110 | – | Instant power consumption |
| RX current（mA） | – | 12 | – | – |
| Sleep current（μA） | – | 2 | – | Software is shut down |
| Max Tx power（dBm） | 21.5 | 22.0 | 22.5 | – |
| Receiving sensitivity（dBm） | -146 | -147 | -148 | Air data rate is 0.3 kbps |
| Air data rate（bps） | 0.3k | 2.4k | 62.5k | Controlled via user’s programming |

|  |  |  |
| --- | --- | --- |
| Main parameter | Description | Remarks |
| Distance for reference | 5km | Test condition：clear and open area, antenna gain: 5dBi，antenna height:2.5m，air data rate: 2.4kbps |
| TX length | 240 Byte | Can be configured via command as 32/64/128/240 bytes per packet totransmit |
| Buffer | 1000 Byte | – |
| Modulation | LoRa | New generation LoRa modulation technology |
| Communication interface | UART | TTL level |
| Package | DIP | – |
| Connector | 1\*7\*2.54mm | – |
| Size | 21\*36 mm | – |
| Antenna | SMA | 50 ohm impedance |

* Communication Interface: UART – 8N1, 8E1, 8O1, Eight kinds of UART baud Rate, from 1200 to 115200bps (Default: 9600)

| **Electronic parameter** | **Min.** | **Typ.** | **Max.** | **Unit** | **Condition** |
| --- | --- | --- | --- | --- | --- |
| Power supply | 3.3 | 3.6 | 5.5 | V |  |
| Communication level | 3.0 | 3.3 | 3.6 | V |  |
| Transmitting current | 95 | 100 | 105 | mA | 22dBm（160mW） |
| Receiving current | 6 | 6.5 | 7 | mA |  |
| Sleep current | 1 | 2 | 3 | nA |  |
| Operating temperature | -40 | 20 | +85 | ℃ |  |
| Operating humidity | 10 | 60 | 90 | % |  |
| Storage temperature | -40 | 20 | +125 | ℃ |  |

You must pay attention on communication level that differ from power supply, the second can receive voltage like 3.3v (esp8266 and esp32) and 5v (Arduino), but the first want a 3.3v, so to connecto to an Arduino you must use a Voltage divider ([Voltage divider: calculator and application](https://mischianti.org/voltage-divider-calculator-and-application/)) to prevent damage to the device.

Pinout

[](https://mischianti.org/wp-content/uploads/2019/09/ebyte-lora-exx-pinout-1.jpg)sx1278 sx1276 wireless lora uart module serial 3000m arduino 433 rf

| **Pin No.** | **Pin item** | **Pin direction** | **Pin application** |
| --- | --- | --- | --- |
| 1 | M0 | Input（weak pull-up） | Work with M1 & decide the four operating modes.Floating is not allowed, can be ground. |
| 2 | M1 | Input（weak pull-up） | Work with M0 & decide the four operating modes.Floating is not allowed, can be ground. |
| 3 | RXD | Input | TTL UART inputs, connects to external (MCU, PC) TXD outputpin. Can be configured as open-drain or pull-up input. |
| 4 | TXD | Output | TTL UART outputs, connects to external RXD (MCU, PC) input pin. Can be configured as open-drain or push-pull output |
| 5 | AUX | Output | To indicate module’s working status & wakes up the external MCU. During the procedure of self-check initialization, the pin outputs low level. Can be configured as open-drain output orpush-pull output (floating is allowed). |
| 6 | VCC |  | Power supply 2.3V~5.5V DC |
| 7 | GND |  | Ground |

As you can see you can set various modes via M0 and M1 pins.

[](https://www.sunfounder.com/?ref=fz59AUVD)

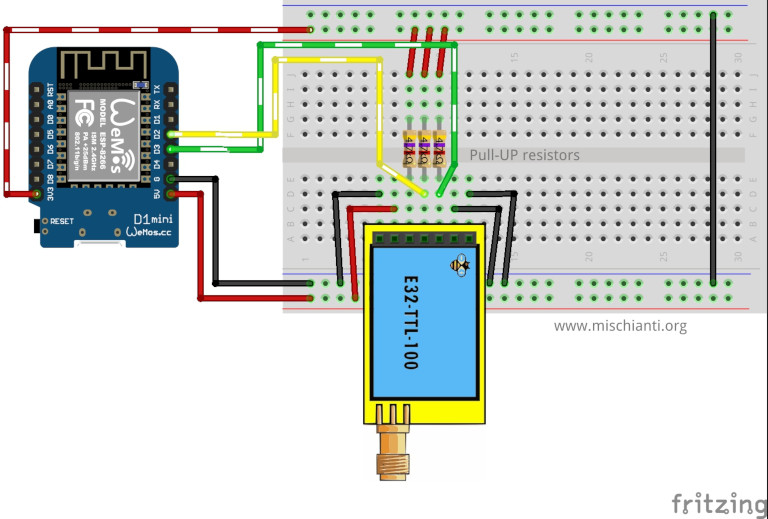
| **Mode** | **M1** | **M0** | **Explanation** |
| --- | --- | --- | --- |
| Normal | 0 | 0 | UART and wireless channel are open, transparent transmission is on |
| WOR mode | 0 | 1 | Can be defined as WOR transmitter and WOR receiver |
| Configuration mode | 1 | 0 | Users can access the register through the serial port to control the working state of the module |
| Deep sleep mode | 1 | 1 | Sleep mode. |

Normal mode connection

For the next simple test we are going to use Normal mode.

Connecting Wemos D1 mini (esp8266) for a basic usage

esp8266 have the advantage to have same voltage of communication interface so the connection schema is more simple than Arduino.

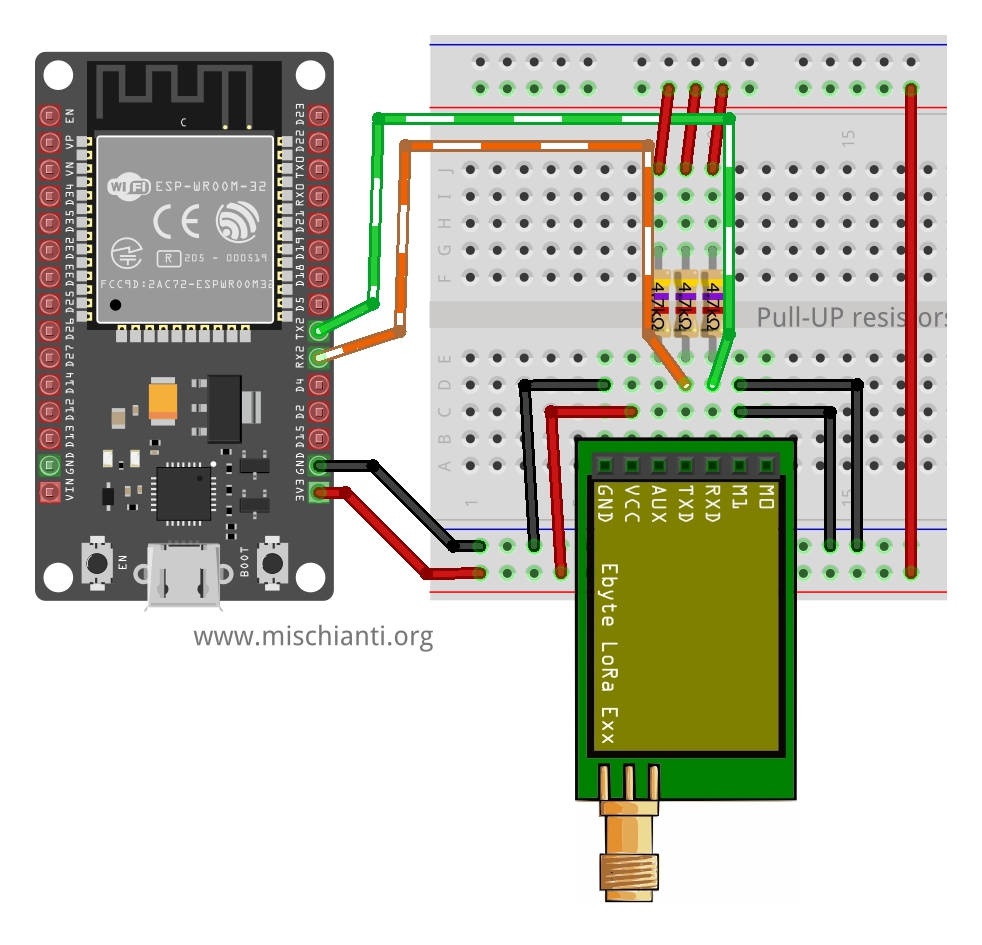
[](https://mischianti.org/wp-content/uploads/2019/09/LoRa_E32-TTL-100_WemosD1_VD_PU_bb-e1569566707815.jpg)LoRa E22-TTL-100 Wemos D1 breadboard

It’s important to add pull-up resistor (4,7Kohm) to get good stability.

| **E22** | **WeMos** |
| --- | --- |
| M0 | GND (Set normal mode) |
| M1 | GND (Set normal mode) |
| RX | PIN D2 (PullUP 4,7KΩ) |
| TX | PIN D3 (PullUP 4,7KΩ) |
| AUX | Not connected |
| VCC | 3.3v |
| GND | GND |

Connecting esp32 for a basic usage

As WeMos, esp32 logic works at 3.3v but It have 3 HardwareSerial, so we are going to use Serial2.

[](https://mischianti.org/wp-content/uploads/2020/07/Ebyte-LoRa-E22-device-esp32-dev-kit-v1-breadboard-transparent.jpg)Ebyte LoRa E22 device esp32 dev kit v1 breadboard transparent transmission (normal mode)

| **E22** | **esp32** |
| --- | --- |
| M0 | GND (Set normal mode) |
| M1 | GND (Set normal mode) |
| RX | TX2 (PullUP 4,7KΩ) |
| TX | RX2 (PullUP 4,7KΩ) |
| AUX | Not connected |
| VCC | 3.3v |
| GND | GND |

Connecting Arduino MKR for a basic usage

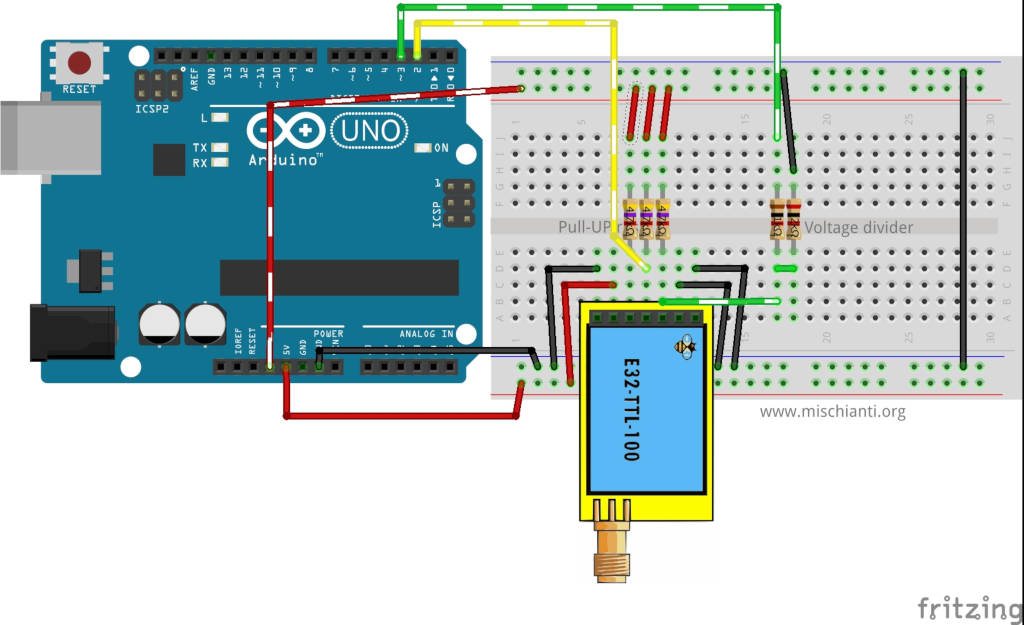
Ebyte LoRa Exx Arduino MKR WiFi 1010 normal mode connected breadboard

| **E22** | **Arduino MKR** |
| --- | --- |
| M0 | GND (Set normal mode) |
| M1 | GND (Set normal mode) |
| TX | PIN 14 Tx (PullUP 4,7KΩ) |
| RX | PIN 13 Rx (PullUP 4,7KΩ) |
| AUX | PIN 1 (PullUP 4,7KΩ) |
| VCC | 5V |
| GND | GND |

Connecting Arduino for a basic usage

Arduino working voltage is 5v, so we need to add a voltage divider on RX pin of LoRa module to prevent damage, you can get more information here [Voltage divider: calculator and application](https://mischianti.org/voltage-divider-calculator-and-application/).

You can use a 2Kohm resistor to GND and 1Kohm from signal than put together on RX.

[](https://mischianti.org/wp-content/uploads/2019/09/LoRa_E32-TTL-100_Arduino_VD_PU_bb-e1569566634311.jpg)LoRa E22-TTL-100 Arduino breadboard

| **E22** | **Arduino** |
| --- | --- |
| M0 | GND (Set normal mode) |
| M1 | GND (Set normal mode) |
| RX | PIN D2 (PullUP 4,7KΩ & Voltage divider) |
| TX | PIN D3 (PullUP 4,7KΩ) |
| AUX | Not connected |
| VCC | 3.3v |
| GND | GND |

Simple communication sketch

If you put to 0 M1 and M0 pin you enter in “Normal” mode, than you can receive and trasmit all the data from device A to B, this modality is defined “Trasparent transmission”.

**You can use 2 Arduinos or 2 Wemos D1 mini or one of kind.**

At start send a message and if you write on serial from one of device the text is transferred to the other device. You can use 2 Arduinos or 2 Wemos or one and one as you prefer.

Arduino sketch

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | /\*   \* LoRa E22   \* Start device or reset to send a message   \* [https://mischianti.org](https://mischianti.org/)   \*   \* E22        ----- Arduino UNO   \* M0         ----- GND   \* M1         ----- GND   \* TX         ----- PIN 2 (PullUP)   \* RX         ----- PIN 3 (PullUP & Voltage divider)   \* AUX        ----- Not connected   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/  #include "Arduino.h"    #include <SoftwareSerial.h>    SoftwareSerial mySerial(2, 3); // Arduino RX --> e22 TX - Arduino TX --> e22 RX    **void** setup() {    Serial.begin(9600);    delay(500);      Serial.println("Hi, I'm going to send message!");      mySerial.begin(9600);    mySerial.println("Hello, world?");  }    **void** loop() {  **if** (mySerial.available()) {      Serial.write(mySerial.read());    }  **if** (Serial.available()) {      mySerial.write(Serial.read());    }  } |

Wemos D1 mini sketch

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | /\*   \* LoRa E22   \* Start device or reset to send a message   \* [https://mischianti.org](https://mischianti.org/)   \*   \* E22         ----- Wemos D1 mini   \* M0         ----- GND   \* M1         ----- GND   \* TX         ----- PIN D2 (PullUP)   \* RX         ----- PIN D3 (PullUP)   \* AUX        ----- Not connected   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/  #include "Arduino.h"  #include <SoftwareSerial.h>    SoftwareSerial mySerial(D2, D3); // WeMos RX --> e22 TX - WeMos TX --> e22 RX    **void** setup() {    Serial.begin(9600);    delay(500);      Serial.println("Hi, I'm going to send message!");      mySerial.begin(9600);    mySerial.println("Hello, world?");  }    **void** loop() {  **if** (mySerial.available()) {      Serial.write(mySerial.read());    }  **if** (Serial.available()) {      mySerial.write(Serial.read());    }  } |

esp32 sketch

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | /\*   \* LoRa E22   \* Start device or reset to send a message   \* [https://mischianti.org](https://mischianti.org/)   \*   \* E22        ----- esp32   \* M0         ----- GND   \* M1         ----- GND   \* TX         ----- RX2 (PullUP)   \* RX         ----- TX2 (PullUP)   \* AUX        ----- Not connected   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/  #include "Arduino.h"    **void** setup() {    Serial.begin(9600);    delay(500);      Serial.println("Hi, I'm going to send message!");      Serial2.begin(9600);    Serial2.println("Hello, world?");  }    **void** loop() {  **if** (Serial2.available()) {      Serial.write(Serial2.read());    }  **if** (Serial.available()) {      Serial2.write(Serial.read());    }  } |

Arduino MKR sketch

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | /\*   \* LoRa E22   \* Start device or reset to send a message   \* [https://mischianti.org](https://mischianti.org/)   \*   \* E22        ----- Arduino MKR   \* M0         ----- 2 (or GND)   \* M1         ----- 3 (or GND)   \* RX         ----- 14 (PullUP)   \* TX         ----- 13 (PullUP)   \* AUX        ----- 1  (PullUP)   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/  #include "Arduino.h"    **void** **setup**() {    Serial.begin(9600);    delay(500);      Serial.println("Hi, I'm going to send message!");      Serial1.begin(9600);    Serial1.println("Hello, world?");  }    **void** **loop**() {  **if** (Serial1.available()) {      Serial.write(Serial1.read());    }  **if** (Serial.available()) {      Serial1.write(Serial.read());    }  } |

But this basic usage is quite unusefully, so in the next chapter we are going to use my library and go in deep of device features.

Library

EByte LoRa E22 E32 Arduino library manager

Here the last example with [my library](https://github.com/xreef/LoRa_E32_Series_Library):

Arduino sketch

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | /\*   \* LoRa E22   \* Write on serial to transfer a message to other device   \* [https://mischianti.org](https://mischianti.org/)   \*   \* E22        ----- Arduino UNO   \* M0         ----- GND   \* M1         ----- GND   \* TX         ----- PIN 2 (PullUP)   \* RX         ----- PIN 3 (PullUP & Voltage divider)   \* AUX        ----- Not connected   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/  #include "Arduino.h"  #include "LoRa\_E22.h"    LoRa\_E22 e22ttl(2, 3); // Arduino RX --> e22 TX - Arduino TX --> e22 RX    **void** setup() {    Serial.begin(9600);    delay(500);      Serial.println("Hi, I'm going to send message!");      // Startup all pins and UART    e22ttl.begin();      // Send message    ResponseStatus rs = e22ttl.sendMessage("Hello, world?");    // Check If there is some problem of successfully send    Serial.println(rs.getResponseDescription());  }    **void** loop() {      // If something available  **if** (e22ttl.available()>1) {        // read the String message      ResponseContainer rc = e22ttl.receiveMessage();      // Is something goes wrong print error  **if** (rc.status.code!=1){          rc.status.getResponseDescription();      }**else**{          // Print the data received          Serial.println(rc.data);      }    }  **if** (Serial.available()) {        String input = Serial.readString();        e22ttl.sendMessage(input);    }  } |

Wemos D1 (esp8266) sketch

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | /\*   \* LoRa E22   \* Start device or reset to send a message   \* [https://mischianti.org](https://mischianti.org/)   \*   \* E22        ----- Wemos D1 mini   \* M0         ----- GND   \* M1         ----- GND   \* TX         ----- PIN D2 (PullUP)   \* RX         ----- PIN D3 (PullUP)   \* AUX        ----- Not connected   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/  #include "Arduino.h"  #include "LoRa\_E22.h"    LoRa\_E22 e22ttl(D2, D3); // WeMos RX --> e22 TX - WeMos TX --> e22 RX    **void** setup() {    Serial.begin(9600);    delay(500);      Serial.println("Hi, I'm going to send message!");      // Startup all pins and UART    e22ttl.begin();      // Send message    ResponseStatus rs = e22ttl.sendMessage("Hello, world?");    // Check If there is some problem of successfully send    Serial.println(rs.getResponseDescription());  }    **void** loop() {      // If something available  **if** (e22ttl.available()>1) {        // read the String message      ResponseContainer rc = e22ttl.receiveMessage();      // Is something goes wrong print error  **if** (rc.status.code!=1){          rc.status.getResponseDescription();      }**else**{          // Print the data received          Serial.println(rc.data);      }    }  **if** (Serial.available()) {        String input = Serial.readString();        e22ttl.sendMessage(input);    }  } |

esp32 sketch

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | /\*   \* LoRa E22   \* Start device or reset to send a message   \* [https://mischianti.org](https://mischianti.org/)   \*   \* E22        ----- esp32   \* M0         ----- GND   \* M1         ----- GND   \* TX         ----- RX2 (PullUP)   \* RX         ----- TX2 (PullUP)   \* AUX        ----- Not connected   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/  #include "Arduino.h"  #include "LoRa\_E22.h"    LoRa\_E22 e22ttl(&Serial2); // WeMos RX --> e22 TX - WeMos TX --> e22 RX    **void** setup() {    Serial.begin(9600);    delay(500);      Serial.println("Hi, I'm going to send message!");      // Startup all pins and UART    e22ttl.begin();      // Send message    ResponseStatus rs = e22ttl.sendMessage("Hello, world?");    // Check If there is some problem of successfully send    Serial.println(rs.getResponseDescription());  }    **void** loop() {      // If something available  **if** (e22ttl.available()>1) {        // read the String message      ResponseContainer rc = e22ttl.receiveMessage();      // Is something goes wrong print error  **if** (rc.status.code!=1){          rc.status.getResponseDescription();      }**else**{          // Print the data received          Serial.println(rc.data);      }    }  **if** (Serial.available()) {        String input = Serial.readString();        e22ttl.sendMessage(input);    }  } |

Arduino MKR sketch

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | /\*   \* LoRa E22   \* Start device or reset to send a message   \* [https://mischianti.org](https://mischianti.org/)   \*   \* E22        ----- Arduino MKR   \* M0         ----- GND   \* M1         ----- GND   \* RX         ----- 14 (PullUP)   \* TX         ----- 13 (PullUP)   \* AUX        ----- 1  (PullUP)   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/  #include "Arduino.h"  #include "LoRa\_E22.h"    LoRa\_E22 e22ttl(&Serial1); // WeMos RX --> e22 TX - WeMos TX --> e22 RX    **void** setup() {    Serial.begin(9600);    delay(500);      Serial.println("Hi, I'm going to send message!");      // Startup all pins and UART    e22ttl.begin();      // Send message    ResponseStatus rs = e22ttl.sendMessage("Hello, world?");    // Check If there is some problem of successfully send    Serial.println(rs.getResponseDescription());  }    **void** loop() {      // If something available  **if** (e22ttl.available()>1) {        // read the String message      ResponseContainer rc = e22ttl.receiveMessage();      // Is something goes wrong print error  **if** (rc.status.code!=1){          rc.status.getResponseDescription();      }**else**{          // Print the data received          Serial.println(rc.data);      }    }  **if** (Serial.available()) {        String input = Serial.readString();        e22ttl.sendMessage(input);    }  } |

If you have already change configuration you must restore base parameter:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | //  If you have ever change configuration you must restore It      ResponseStructContainer c;      c = e22ttl.getConfiguration();      Configuration configuration = \*(Configuration\*) c.data;      Serial.println(c.status.getResponseDescription());      configuration.CHAN = 0x17;      configuration.OPTION.fixedTransmission = FT\_TRANSPARENT\_TRANSMISSION;      e22ttl.setConfiguration(configuration, WRITE\_CFG\_PWR\_DWN\_SAVE); |

but we are going to see It better in the next article.

Thanks

Ebyte LoRa E22 device for Arduino, esp32 or esp8266: library – 2

BY [RENZO MISCHIANTI](https://mischianti.org/author/reef/) · PUBLISHED 28 JANUARY 2021 · UPDATED 11 FEBRUARY 2024

**Spread the love**

**LoRa or Long Range wireless data telemetry** is a technology pioneered by Semtech that operates at a lower frequency than NRF24L01 (433 MHz, 868 MHz, or 916 MHz against 2.4 GHz for the NRF24L01) but at thrice the distance (from 4000m to 10000m).

[Support Forum](https://mischianti.org/forums/forum/mischiantis-libraries/ebyte-lora-e22-uart-devices-sx1262-sx1268/)



Ebyte LoRa E22 device for Arduino, esp32 or esp8266 Library

I created a library to manage EBYTE E22 based on the Semtech SX1268 series of LoRa devices, a very powerful, simple, and cheap device.

Ebyte LoRa E22 device for Arduino, esp32 or esp8266 3 devices module SMD

You can find here [AliExpress (433MHz 5.5Km)](https://s.click.aliexpress.com/e/_DlLGLbF) - [AliExpress (433MHz 10Km)](https://s.click.aliexpress.com/e/_DCGzJIt) - [AliExpress (868MHz 915Mhz 5.5Km)](https://s.click.aliexpress.com/e/_DczWHrb) - [AliExpress (868MHz 915Mhz 10Km)](https://s.click.aliexpress.com/e/_DeoQrNn)

* [Library](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Library)
* [Pinout](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Pinout)
* [Fully connected schema](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Fully_connected_schema)
  + [AUX pin](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#AUX_pin)
  + [esp8266](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#esp8266)
  + [esp32](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#esp32)
  + [Arduino](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Arduino)
  + [Arduino MKR WiFi 1010](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Arduino_MKR_WiFi_1010)
* [Constructor](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Constructor)
* [Begin](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Begin)
* [Configuration and information method](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Configuration_and_information_method)
  + [Response container](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Response_container)
  + [ResponseStatus](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#ResponseStatus)
  + [ResponseContainer](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#ResponseContainer)
  + [ResponseStructContainer](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#ResponseStructContainer)
* [getConfiguration and setConfiguration](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#getConfiguration_and_setConfiguration)
  + [Basic configuration option](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Basic_configuration_option)
    - [SPED detail](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#SPED_detail)
    - [SPED detail](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#SPED_detail-2)
    - [OPTION detail](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#OPTION_detail)
      * [Sub packet setting](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Sub_packet_setting)
      * [RSSI Ambient noise enable](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#RSSI_Ambient_noise_enable)
      * [Transmission power](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Transmission_power)
    - [TRANSMISSION\_MODE Detail](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#TRANSMISSION_MODE_Detail)
      * [Enable RSSI](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Enable_RSSI)
      * [Transmission type](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Transmission_type)
      * [Enable repeater function](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Enable_repeater_function)
      * [Monitor data before transmission](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Monitor_data_before_transmission)
      * [WOR](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#WOR)
      * [WOR cycle](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#WOR_cycle)
* [Check buffer](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Check_buffer)
* [Send receive messages](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Send_receive_messages)
  + [Normal transmission mode](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Normal_transmission_mode)
    - [Manage structure](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Manage_structure)
    - [Read partial structure](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Read_partial_structure)
  + [Fixed mode instead of normal mode](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Fixed_mode_instead_of_normal_mode)
    - [Fixed transmission](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Fixed_transmission)
* [Wireless configuration](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Wireless_configuration)
* [Thanks](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-library-part-2/#Thanks)

Library

You can find my library [here](https://github.com/xreef/EByte_LoRa_E22_Series_Library).

And It’s available on Arduino IDE library manager.

EByte LoRa E22 E32 Arduino library manager

To download.

Click the [DOWNLOADS](https://github.com/xreef/EByte_LoRa_E22_Series_Library/archive/master.zip)button in the top right corner, rename the uncompressed folder LoRa\_E22.

Check that the LoRa\_E22 folder contains LoRa\_E22.cpp and LoRa\_E22.h.

Place the LoRa\_E22 library folder your /libraries/ folder.

You may need to create the libraries subfolder if its your first library.

Restart the IDE.

Pinout

[](https://mischianti.org/wp-content/uploads/2019/09/ebyte-lora-exx-pinout-1.jpg)sx1278 sx1276 wireless lora uart module serial 3000m arduino 433 rf

| **Pin No.** | **Pin item** | **Pin direction** | **Pin application** |
| --- | --- | --- | --- |
| 1 | M0 | Input（weak pull-up） | Work with M1 & decide the four operating modes.Floating is not allowed, can be ground. |
| 2 | M1 | Input（weak pull-up） | Work with M0 & decide the four operating modes.Floating is not allowed, can be ground. |
| 3 | RXD | Input | TTL UART inputs, connects to external (MCU, PC) TXD outputpin. Can be configured as open-drain or pull-up input. |
| 4 | TXD | Output | TTL UART outputs, connects to external RXD (MCU, PC) inputpin. Can be configured as open-drain or push-pull output |
| 5 | AUX | Output | Per indicare lo stato di funzionamento del modulo e riattivare l’MCU esterno. Durante la procedura di inizializzazione di autocontrollo, il pin emette una bassa tensione. Può essere configurato come uscita open-drain o output push-pull (è consentito non metterlo a terra, ma se hai problemi, ad esempio ti si freeze il dispositivo è preferibile mettere una restistenza di pull-up da 4.7k o meglio collegarlo al dispositivo). |
| 6 | VCC |  | Power supply 2.3V~5.5V DC |
| 7 | GND |  | Ground |

As you can see you can set various modes via M0 and M1 pins.

| **Mode** | **M1** | **M0** | **Explanation** |
| --- | --- | --- | --- |
| Normal | 0 | 0 | UART and wireless channel are open, transparent transmission is on (Supports configuration over air via special command) |
| WOR Mode | 0 | 1 | Can be defined as WOR transmitter and WOR receiver |
| Configuration mode | 1 | 0 | Users can access the register through the serial port to control the working state of the module |
| Deep sleep mode | 1 | 1 | Sleep mode |

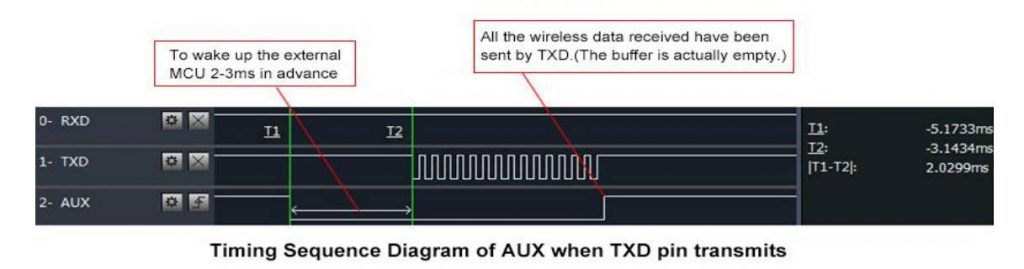
There are some pins that can be use in a static way, but If you connect It to the microcontroller and configure they in the library you gain in performance and you can control all mode via software, but we are going to explain better next.

Fully connected schema

As I already say It’s not important to connect all pin to the output of microcontroller, you can put M0 and M1 pins to HIGH or LOW to get desidered configuration, and **if you don’t connect AUX the library set a reasonable delay to be sure that the operation is complete** (**If you have trouble** **like freeze device, you must put a pull-up 4.7k resistor or better connect to the device.** ).

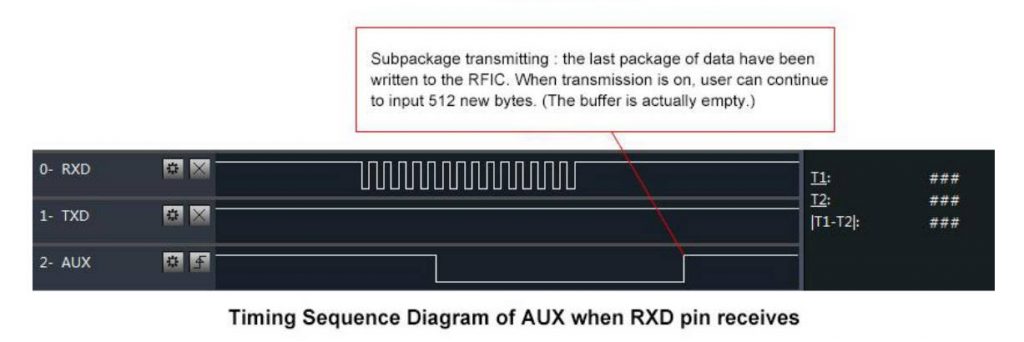
AUX pin

When transmitting data can be used to wake up external MCU and return HIGH on data transfer finish.

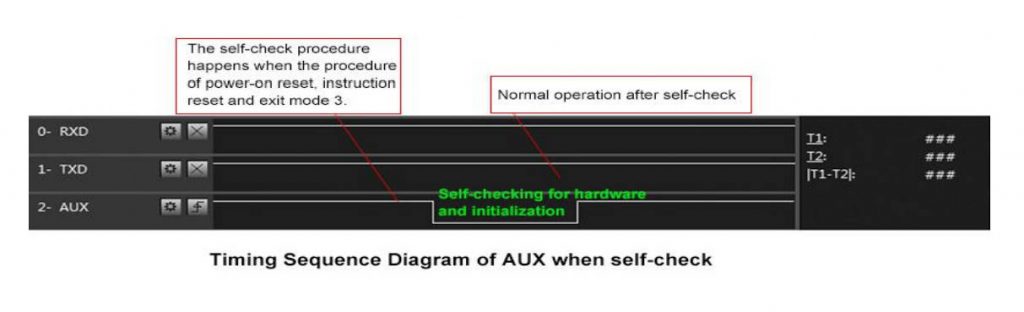
[](https://mischianti.org/wp-content/uploads/2019/10/e32auxPinOnTransmission.jpg)LoRa E32 AUX Pin on transmission

When receiving AUX going LOW and return HIGH when buffer is empty.

[](https://www.sunfounder.com/?ref=fz59AUVD)

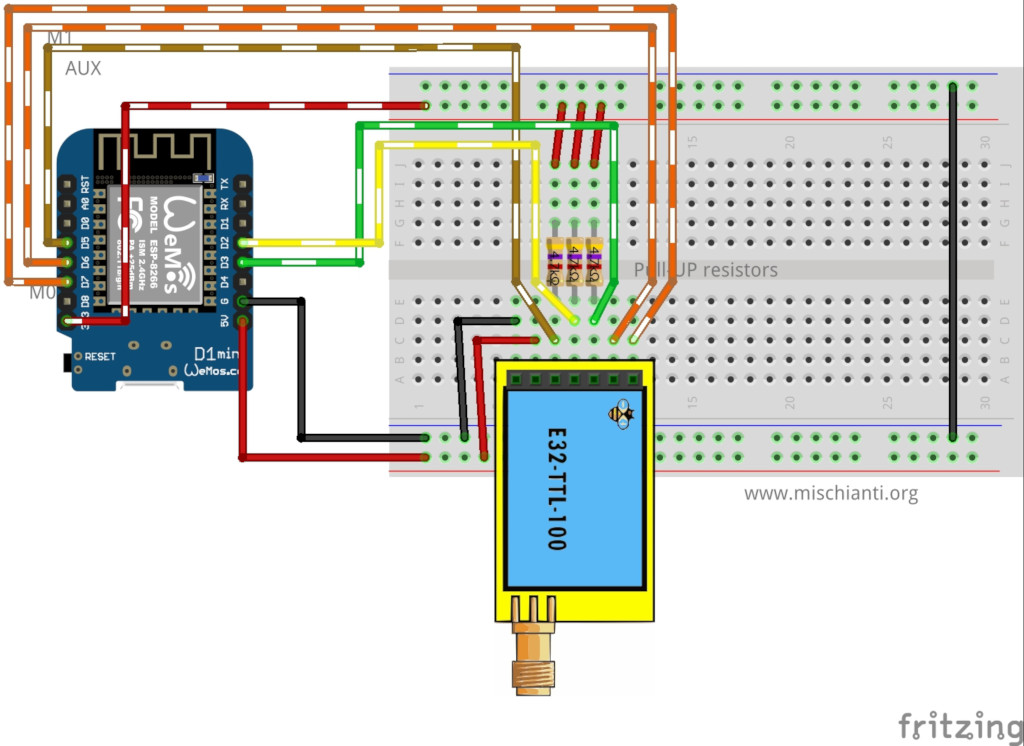
[](https://mischianti.org/wp-content/uploads/2019/10/e32auxPinOnReception.jpg)LoRa e32 AUX pin on reception

It’s also used for self checking to restore normal operation (on power-on and sleep/program mode).

[](https://mischianti.org/wp-content/uploads/2019/10/e32auxPinOnSelfCheck.jpg)LoRa e32 AUX pin on self-check

esp8266

esp8266 connection schema is more simple because It work at the same voltage of logical communications (3.3v).

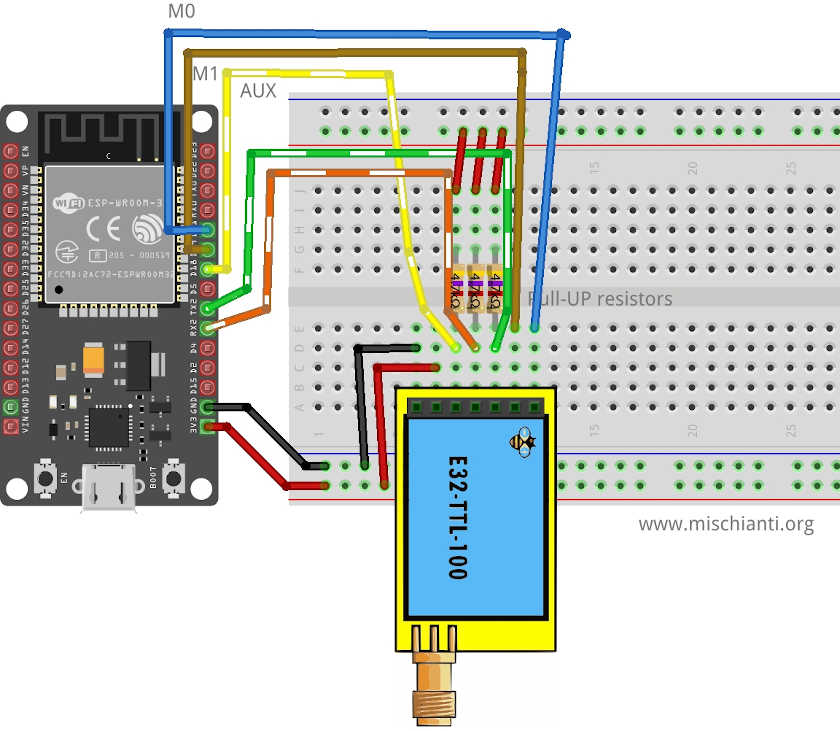
[](https://mischianti.org/wp-content/uploads/2019/10/LoRa_E32-TTL-100_WemosD1_VD_PU_FullyConnected_bb-e1570517387323.jpg)LoRa E32 TTL 100 Wemos D1 fully connected

It’s important to add pull-up resistor (4,7Kohm) to get good stability.

| **E22** | **esp8266** |
| --- | --- |
| M0 | D7 |
| M1 | D6 |
| TX | PIN D2 (PullUP 4,7KΩ) |
| RX | PIN D3 (PullUP 4,7KΩ) |
| AUX | PIN D5 (PullUP 4,7KΩ) |
| VCC | 5V (but work with less power in 3.3v) |
| GND | GND |

esp32

Similar connection schema for esp32, but for RX and TX we use RX2 and TX2, because by default esp32 doesn’t have SoftwareSerial but have 3 Serial.

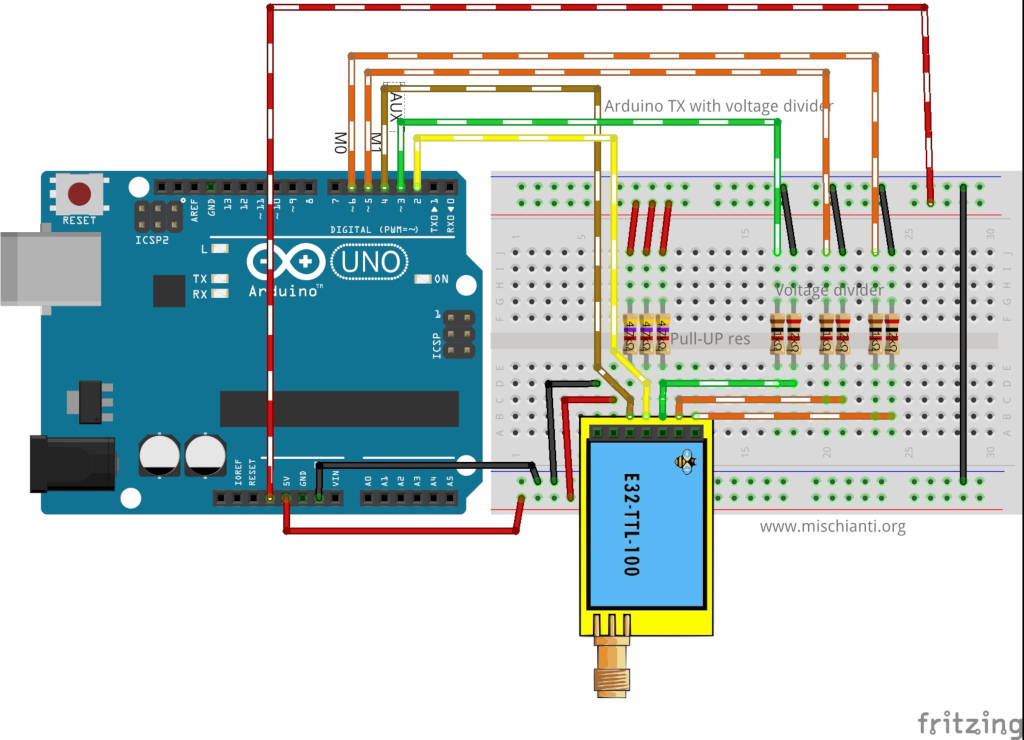
[](https://mischianti.org/wp-content/uploads/2020/08/Ebyte-LoRa-E22-device-esp32-dev-kit-v1-breadboard-full-connection.jpg)Ebyte LoRa E22 device esp32 dev kit v1 breadboard full connection

| **E22** | **esp32** |
| --- | --- |
| M0 | D21 |
| M1 | D19 |
| TX | PIN RX2 (PullUP 4,7KΩ) |
| RX | PIN TX3 (PullUP 4,7KΩ) |
| AUX | PIN D18 (PullUP 4,7KΩ) |
| VCC | 5V (but work with less power in 3.3v) |
| GND | GND |

Arduino

Arduino working voltage is 5v, so we need to add a voltage divider on RX pin M0 and M1 of LoRa module to prevent damage, you can get more information here [Voltage divider: calculator and application](https://mischianti.org/voltage-divider-calculator-and-application/).

You can use a 2Kohm resistor to GND and 1Kohm from signal than put together on RX.

[](https://mischianti.org/wp-content/uploads/2019/10/LoRa_E32-TTL-100_Arduino_VD_PU_FullyConnected_bb-e1570517268668.jpg)LoRa E32 TTL 100 Arduino fully connected

|  |  |
| --- | --- |
| M0 | 7 (voltage divider) |
| M1 | 6 (voltage divider) |
| TX | PIN 2 (PullUP 4,7KΩ) |
| RX | PIN 3 (PullUP 4,7KΩ & Voltage divider) |
| AUX | PIN 5 (PullUP 4,7KΩ) |
| VCC | 5V |
| GND | GND |

Arduino MKR WiFi 1010

Ebyte LoRa Exx Arduino MKR WiFi 1010 Fully connected breadboard

|  |  |
| --- | --- |
| M0 | 2 (voltage divider) |
| M1 | 3 (voltage divider) |
| TX | PIN 14 Tx (PullUP 4,7KΩ) |
| RX | PIN 13 Rx (PullUP 4,7KΩ) |
| AUX | PIN 1 (PullUP 4,7KΩ) |
| VCC | 5V |
| GND | GND |

Constructor

I made a set of quite numerous constructors, because we can have more options and situations to manage.

|  |  |
| --- | --- |
| 1  2  3 | LoRa\_E22(byte txE22pin, byte rxE22pin, UART\_BPS\_RATE bpsRate = UART\_BPS\_RATE\_9600);  LoRa\_E22(byte txE22pin, byte rxE22pin, byte auxPin, UART\_BPS\_RATE bpsRate = UART\_BPS\_RATE\_9600);  LoRa\_E22(byte txE22pin, byte rxE22pin, byte auxPin, byte m0Pin, byte m1Pin, UART\_BPS\_RATE bpsRate = UART\_BPS\_RATE\_9600); |

First set of constructor are create to delegate the manage of Serial and other pins to the library.

* txE22pin and rxE22pin is the pin to connect to UART and they are **mandatory**.
* auxPin is a pin that check the operation, transmission and receiving status (we are going to explain better next), that pin **It isn’t mandatory**, if you don’t set It I apply a delay to permit the operation to complete itself (with latency, i**f you have trouble, like freeze device, you must put a pull-up 4.7k resistor or better connect to the device** ).
* m0pin and m1Pin are the pins to change operation MODE (see the table upper), I think **this pins in “production” are going to connect directly HIGH or LOW**, but for test they are usefully to be managed by the library.
* bpsRate is the boudrate of SoftwareSerial normally is 9600 (the only baud rate in programmin/sleep mode)

A simple example is

|  |  |
| --- | --- |
| 1  2  3  4 | #include "LoRa\_E22.h"    LoRa\_E32 e22ttl100(2, 3);  // e22 TX e22 RX  // LoRa\_E32 e32ttl100(2, 3, 5, 6, 7);  // e22 TX e22 RX |

We can use directly a SoftwareSerial with another constructor

|  |  |
| --- | --- |
| 1  2  3 | LoRa\_E22(HardwareSerial\* serial, UART\_BPS\_RATE bpsRate = UART\_BPS\_RATE\_9600);  LoRa\_E22(HardwareSerial\* serial, byte auxPin, UART\_BPS\_RATE bpsRate = UART\_BPS\_RATE\_9600);  LoRa\_E22(HardwareSerial\* serial, byte auxPin, byte m0Pin, byte m1Pin, UART\_BPS\_RATE bpsRate = UART\_BPS\_RATE\_9600); |

The example upper with this constructor can be do like so.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | #include <SoftwareSerial.h>  #include "LoRa\_E22.h"    SoftwareSerial mySerial(2, 3); // e22 TX e22 RX  LoRa\_E22 e22ttl100(&mySerial);  // LoRa\_E22 e22ttl100(&mySerial, 5, 7, 6); |

The last set of constructor is to permit to use an HardwareSerial instead of SoftwareSerial.

|  |  |
| --- | --- |
| 1  2  3 | LoRa\_E22(SoftwareSerial\* serial, UART\_BPS\_RATE bpsRate = UART\_BPS\_RATE\_9600);  LoRa\_E22(SoftwareSerial\* serial, byte auxPin, UART\_BPS\_RATE bpsRate = UART\_BPS\_RATE\_9600);  LoRa\_E22(SoftwareSerial\* serial, byte auxPin, byte m0Pin, byte m1Pin, UART\_BPS\_RATE bpsRate = UART\_BPS\_RATE\_9600); |

For esp32 you have 3 additional constructor to permit to manage pins for HardWare serial

|  |  |
| --- | --- |
| 1  2  3 | LoRa\_E22(byte txE22pin, byte rxE22pin, HardwareSerial\* serial, UART\_BPS\_RATE bpsRate = UART\_BPS\_RATE\_9600, uint32\_t serialConfig = SERIAL\_8N1);  LoRa\_E22(byte txE22pin, byte rxE22pin, HardwareSerial\* serial, byte auxPin, UART\_BPS\_RATE bpsRate = UART\_BPS\_RATE\_9600, uint32\_t serialConfig = SERIAL\_8N1);  LoRa\_E22(byte txE22pin, byte rxE22pin, HardwareSerial\* serial, byte auxPin, byte m0Pin, byte m1Pin, UART\_BPS\_RATE bpsRate = UART\_BPS\_RATE\_9600, uint32\_t serialConfig = SERIAL\_8N1); |

Begin

The begin command is used to startup Serial and pins in input and output mode.

|  |  |
| --- | --- |
| 1 | **void** begin(); |

in execution is

[](https://www.pcbway.com/?from=mischianti723)

|  |  |
| --- | --- |
| 1  2 | // Startup all pins and UART  e22ttl100.begin(); |

Configuration and information method

There a set of methods for manage configuration and get information of the device.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | ResponseStructContainer getConfiguration();  ResponseStatus setConfiguration(Configuration configuration, PROGRAM\_COMMAND saveType = WRITE\_CFG\_PWR\_DWN\_LOSE);    ResponseStructContainer getModuleInformation();  **void** printParameters(**struct** Configuration configuration);  ResponseStatus resetModule(); |

Response container

To simplify the manage of response I create a set of container, for me very usefully to manage errors and return generic data.

ResponseStatus

This is a status container and have 2 simple entry point, with this you can get the status code and the description of status code

|  |  |
| --- | --- |
| 1  2 | Serial.println(c.getResponseDescription()); // Description of code  Serial.println(c.code); // 1 if Success |

The code are

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SUCCESS = 1,  ERR\_E22\_UNKNOWN,  ERR\_E22\_NOT\_SUPPORT,  ERR\_E22\_NOT\_IMPLEMENT,  ERR\_E22\_NOT\_INITIAL,  ERR\_E22\_INVALID\_PARAM,  ERR\_E22\_DATA\_SIZE\_NOT\_MATCH,  ERR\_E22\_BUF\_TOO\_SMALL,  ERR\_E22\_TIMEOUT,  ERR\_E22\_HARDWARE,  ERR\_E22\_HEAD\_NOT\_RECOGNIZED |

ResponseContainer

This container is created to manage String response and have 2 entry point.

data with the string returned from message and status an instance of RepsonseStatus.

|  |  |
| --- | --- |
| 1  2  3  4  5 | ResponseContainer rs = e22ttl.receiveMessage();  String message = rs.data;    Serial.println(rs.status.getResponseDescription());  Serial.println(message); |

but this command go to read all the data in the buffer, if you receive 3 message you are going to read all 3 message in one time, my simple solution is to use an end character to send at the end of message, to default I use \0 (null character)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | ResponseContainer rs = e22ttl.receiveMessageUntil();          // You can specify a custom delimiter also  // ResponseContainer rs = e22ttl.receiveMessageUntil('|');    String message = rs.data;    Serial.println(rs.status.getResponseDescription());  Serial.println(message); |

This version of device support RSSI also, to read that parameter (if you specify in the configuration that you want send also that), you can use

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | ResponseContainer rc = e22ttl.receiveMessageRSSI();  String message = rs.data;    Serial.println(rs.status.getResponseDescription());  Serial.println(message);          Serial.print("RSSI: "); Serial.println(rc.rssi, DEC); |

ResponseStructContainer

This is the more “complex” container, I use this to manage structure, It has the same entry point of ResponseContainer but data is a void pointer to manage complex structure.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | ResponseStructContainer c;  c = e22ttl100.getConfiguration();  // It's important get configuration pointer before all other operation  Configuration configuration = \*(Configuration\*) c.data;  Serial.println(c.status.getResponseDescription());  Serial.println(c.status.code);      c.close(); |

If you receive a structure message with RSSI you can use

[](https://www.sunfounder.com/?ref=fz59AUVD)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | ResponseStructContainer rsc = e22ttl.receiveMessageRSSI(**sizeof**(Message));  Serial.println(rsc.status.getResponseDescription());  **struct** Message message = \*(Message\*) rsc.data;  Serial.println(message.type);  Serial.println(message.message);  Serial.println(\*(**float**\*)(message.temperature));  Serial.print("RSSI: "); Serial.println(rsc.rssi, DEC);          rsc.close(); |

Every time you use a ResponseStructContainer you must close It with close()

getConfiguration and setConfiguration

The first method is getConfiguration, you can use It to retrieve all data stored on the device.

|  |  |
| --- | --- |
| 1 | ResponseStructContainer getConfiguration(); |

Here an usage example.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | ResponseStructContainer c;  c = e32ttl100.getConfiguration();  // It's important get configuration pointer before all other operation  Configuration configuration = \*(Configuration\*) c.data;  Serial.println(c.status.getResponseDescription());  Serial.println(c.status.code);      Serial.println(configuration.SPED.getUARTBaudRate());       c.close(); |

Structure of configuration have all data of settings, and I add a series of function to get all description of single data.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | configuration.ADDL = 0x03; // First part of address  configuration.ADDH = 0x00; // Second part  configuration.NETID = 0x00; // NETID used for repeater function    configuration.CHAN = 23; // Communication channel    configuration.SPED.uartBaudRate = UART\_BPS\_9600; // Serial baud rate  configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24; // Air baud rate  configuration.SPED.uartParity = MODE\_00\_8N1; // Parity bit    configuration.OPTION.subPacketSetting = SPS\_240\_00; // Packet size  configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED; // Need to send special command  configuration.OPTION.transmissionPower = POWER\_22; // Device power    configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_DISABLED; // Enable RSSI info  configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_TRANSPARENT\_TRANSMISSION; // Transmission type  configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED; // Enable repeater mode  configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED; // Check interference  configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_RECEIVER; // Enable WOR mode  configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011; // WOR timing |

You have the equivalent function for all attribute to get all description:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | DEBUG\_PRINT(F("HEAD : "));  DEBUG\_PRINT(configuration.COMMAND, HEX);DEBUG\_PRINT(" ");DEBUG\_PRINT(configuration.STARTING\_ADDRESS, HEX);DEBUG\_PRINT(" ");DEBUG\_PRINTLN(configuration.LENGHT, HEX);  DEBUG\_PRINTLN(F(" "));  DEBUG\_PRINT(F("AddH : "));  DEBUG\_PRINTLN(configuration.ADDH, HEX);  DEBUG\_PRINT(F("AddL : "));  DEBUG\_PRINTLN(configuration.ADDL, HEX);  DEBUG\_PRINT(F("NetID : "));  DEBUG\_PRINTLN(configuration.NETID, HEX);  DEBUG\_PRINTLN(F(" "));  DEBUG\_PRINT(F("Chan : "));  DEBUG\_PRINT(configuration.CHAN, DEC); DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.getChannelDescription());  DEBUG\_PRINTLN(F(" "));  DEBUG\_PRINT(F("SpeedParityBit     : "));  DEBUG\_PRINT(configuration.SPED.uartParity, BIN);DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.SPED.getUARTParityDescription());  DEBUG\_PRINT(F("SpeedUARTDatte     : "));  DEBUG\_PRINT(configuration.SPED.uartBaudRate, BIN);DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.SPED.getUARTBaudRateDescription());  DEBUG\_PRINT(F("SpeedAirDataRate   : "));  DEBUG\_PRINT(configuration.SPED.airDataRate, BIN);DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.SPED.getAirDataRateDescription());  DEBUG\_PRINTLN(F(" "));  DEBUG\_PRINT(F("OptionSubPacketSett: "));  DEBUG\_PRINT(configuration.OPTION.subPacketSetting, BIN);DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.OPTION.getSubPacketSetting());  DEBUG\_PRINT(F("OptionTranPower    : "));  DEBUG\_PRINT(configuration.OPTION.transmissionPower, BIN);DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.OPTION.getTransmissionPowerDescription());  DEBUG\_PRINT(F("OptionRSSIAmbientNo: "));  DEBUG\_PRINT(configuration.OPTION.RSSIAmbientNoise, BIN);DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.OPTION.getRSSIAmbientNoiseEnable());  DEBUG\_PRINTLN(F(" "));  DEBUG\_PRINT(F("TransModeWORPeriod : "));  DEBUG\_PRINT(configuration.TRANSMISSION\_MODE.WORPeriod, BIN);DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.TRANSMISSION\_MODE.getWORPeriodByParamsDescription());  DEBUG\_PRINT(F("TransModeTransContr: "));  DEBUG\_PRINT(configuration.TRANSMISSION\_MODE.WORTransceiverControl, BIN);DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.TRANSMISSION\_MODE.getWORTransceiverControlDescription());  DEBUG\_PRINT(F("TransModeEnableLBT : "));  DEBUG\_PRINT(configuration.TRANSMISSION\_MODE.enableLBT, BIN);DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.TRANSMISSION\_MODE.getLBTEnableByteDescription());  DEBUG\_PRINT(F("TransModeEnableRSSI: "));  DEBUG\_PRINT(configuration.TRANSMISSION\_MODE.enableRSSI, BIN);DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.TRANSMISSION\_MODE.getRSSIEnableByteDescription());  DEBUG\_PRINT(F("TransModeEnabRepeat: "));  DEBUG\_PRINT(configuration.TRANSMISSION\_MODE.enableRepeater, BIN);DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.TRANSMISSION\_MODE.getRepeaterModeEnableByteDescription());  DEBUG\_PRINT(F("TransModeFixedTrans: "));  DEBUG\_PRINT(configuration.TRANSMISSION\_MODE.fixedTransmission, BIN);DEBUG\_PRINT(" -> "); DEBUG\_PRINTLN(configuration.TRANSMISSION\_MODE.getFixedTransmissionDescription()); |

In the same way, setConfiguration wants a configuration structure, so I think the better way to manage configuration is to retrieve the current one, apply the only change you need and set It again.

|  |  |
| --- | --- |
| 1 | ResponseStatus setConfiguration(Configuration configuration, PROGRAM\_COMMAND saveType = WRITE\_CFG\_PWR\_DWN\_LOSE); |

configuration is the structure previously shown, saveType permit to you to choose if the change becomes permanently of only for the current session.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | ResponseStructContainer c;  c = e32ttl100.getConfiguration();  // It's important get configuration pointer before all other operation  Configuration configuration = \*(Configuration\*) c.data;  Serial.println(c.status.getResponseDescription());  Serial.println(c.status.code);    printParameters(configuration);  configuration.ADDL = 0x03; // First part of address  configuration.ADDH = 0x00; // Second part  configuration.NETID = 0x00; // NETID used for repeater function    configuration.CHAN = 23; // Communication channel    configuration.SPED.uartBaudRate = UART\_BPS\_9600; // Serial baud rate  configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24; // Air baud rate  configuration.SPED.uartParity = MODE\_00\_8N1; // Parity bit    configuration.OPTION.subPacketSetting = SPS\_240\_00; // Packet size  configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED; // Need to send special command  configuration.OPTION.transmissionPower = POWER\_22; // Device power    configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_DISABLED; // Enable RSSI info  configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_TRANSPARENT\_TRANSMISSION; // Transmission type  configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED; // Enable repeater mode  configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED; // Check interference  configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_RECEIVER; // Enable WOR mode  configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011; // WOR timing    // Set configuration changed and set to not hold the configuration  ResponseStatus rs = e32ttl100.setConfiguration(configuration, WRITE\_CFG\_PWR\_DWN\_LOSE);  Serial.println(rs.getResponseDescription());  Serial.println(rs.code);  printParameters(configuration);      c.close() |

The parameter are all managed as constant:

Basic configuration option

| **Name** | **Description** | **Address** |
| --- | --- | --- |
| ADDH | High address byte of module (the default 00H) | 00H |
| ADDL | Low address byte of module (the default 00H) | 01H |
| SPED | Information about data rate parity bit and Air data rate | 02H |
| OPTION | Type of transmission, packet size, allow special message | 03H |
| CHAN | Communication channel（410M + CHAN\*1M）, default 17H (433MHz), **valid only for 433MHz device** check below to check the correct frequency of your device | 04H |
| TRANSMISSION\_MODE | A lot of parameter that specify the transmission modality | 06H |
| CRYPT | Encryption to avoid interception | 07H |

SPED detail

UART Parity bit: *UART mode can be different between communication parties*

SPED detail

UART Parity bit: *UART mode can be different between communication parties*

| **UART parity bit** | **Constant value** |
| --- | --- |
| 8N1 (default) | MODE\_00\_8N1 |
| 8O1 | MODE\_01\_8O1 |
| 8E1 | MODE\_10\_8E1 |
| 8N1 (equal to 00) | MODE\_11\_8N1 |

UART baud rate: UART baud rate can be different between communication parties, The UART baud rate has nothing to do with wireless transmission parameters & won’t affect the wireless transmit / receive features.

| **TTL UART baud rate（bps）** | **Constant value** |
| --- | --- |
| 1200 | UART\_BPS\_1200 |
| 2400 | UART\_BPS\_2400 |
| 4800 | UART\_BPS\_4800 |
| 9600 (default) | UART\_BPS\_9600 |
| 19200 | UART\_BPS\_19200 |
| 38400 | UART\_BPS\_38400 |
| 57600 | UART\_BPS\_57600 |
| 115200 | UART\_BPS\_115200 |

Air data rate: The lower the air data rate, the longer the transmitting distance, better anti- interference performance and longer transmitting time, The air data rate must keep the same for both communication parties.

| **Air data rate（bps）** | **Constant value** |
| --- | --- |
| 0.3k | AIR\_DATA\_RATE\_000\_03 |
| 1.2k | AIR\_DATA\_RATE\_001\_12 |
| 2.4k (default) | AIR\_DATA\_RATE\_010\_24 |
| 4.8k | AIR\_DATA\_RATE\_011\_48 |
| 9.6k | AIR\_DATA\_RATE\_100\_96 |
| 19.2k | AIR\_DATA\_RATE\_101\_192 |
| 38.4k | AIR\_DATA\_RATE\_110\_384 |
| 62.5k | AIR\_DATA\_RATE\_111\_625 |

OPTION detail

Sub packet setting

This is the max length of the packet.

When the data is smaller than the sub packet length, the serial output of the receiving end is an uninterrupted continuous output. When the data is larger than the sub packet length, the receiving end serial port will output the sub packet.

| **Packet size** | **Constant value** |
| --- | --- |
| 240bytes (default) | SPS\_240\_00 |
| 128bytes | SPS\_128\_01 |
| 64bytes | SPS\_064\_10 |
| 32bytes | SPS\_032\_11 |

RSSI Ambient noise enable

This command can enable/disable the management type of RSSI, It’s important to manage the remote configuration, pay attention isn’t the RSSI parameter in the message.

When enabled, the C0 C1 C2 C3 command can be sent in the transmitting mode or WOR transmitting mode to read the register. Register 0x00: Current ambient noise rssi Register 0X01: rssi when the data was received last time.

| **RSSI Ambient noise enable** | **Constant value** |
| --- | --- |
| Enable | RSSI\_AMBIENT\_NOISE\_ENABLED |
| Disable (default) | RSSI\_AMBIENT\_NOISE\_DISABLED |

Transmission power

You can change this set of constant by apply a define like so:

|  |  |
| --- | --- |
| 1 | #define E22\_22 // default value without set |

Applicable for **E22 with 22dBm as max power.**  
Low power transmission is not recommended due to its low power supply efficiency.

| **Transmission power (approximation)** | **Constant value** |
| --- | --- |
| 22dBm (default) | POWER\_22 |
| 17dBm | POWER\_17 |
| 13dBm | POWER\_13 |
| 10dBm | POWER\_10 |

Applicable for **E22 with 30dBm as max power.**  
Low power transmission is not recommended due to its low power supply efficiency.

|  |  |  |
| --- | --- | --- |
| 1 | #define E22\_30 | |
| **Transmission power (approximation)** | **Constant value** |
| 30dBm (default) | POWER\_30 |
| 27dBm | POWER\_27 |
| 24dBm | POWER\_24 |
| 21dBm | POWER\_21 |

You can configure Channel frequency also with this define:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | // One of  #define FREQUENCY\_433  #define FREQUENCY\_170  #define FREQUENCY\_470  #define FREQUENCY\_868  #define FREQUENCY\_915 |

TRANSMISSION\_MODE Detail

Enable RSSI

When enabled, the module receives wireless data and it will follow an RSSI strength byte after output via the serial port TXD

| **Enable RSSI** | **Constant value** |
| --- | --- |
| Enable | RSSI\_ENABLED |
| Disable (default) | RSSI\_DISABLED |

Transmission type

Transmission mode: in fixed transmission mode, the first three bytes of each user’s data frame can be used as high/low address and channel. The module changes its address and channel when transmit. And it will revert to original setting after complete the process.

| **Fixed transmission enabling bit** | **Constant value** |
| --- | --- |
| Fixed transmission mode | FT\_FIXED\_TRANSMISSION |
| Transparent transmission mode (default) | FT\_TRANSPARENT\_TRANSMISSION |

Enable repeater function

| **Enable repeater** | **Constant value** |
| --- | --- |
| Enable repeater | REPEATER\_ENABLED |
| Disable repeater (default) | REPEATER\_DISABLED |

Monitor data before transmission

When enabled, wireless data will be monitored before it is transmitted, which can avoid interference to a certain extent, but may cause data delay.

| **LBT enable byte** | **Constant value** |
| --- | --- |
| Enable | LBT\_ENABLED |
| Disable (default) | LBT\_DISABLED |

WOR

**WOR transmitter:** the module receiving and transmitting functions are turned on, and a wake-up code is added when transmitting data. Receiving is turned on.

**WOR receiver:** the module is unable to transmit data and works in WOR monitoring mode. The monitoring period is as follows (WOR cycle), which can save a lot of power.

| **WOR** | **Constant value** |
| --- | --- |
| WOR transmitter | WOR\_TRANSMITTER |
| WOR receiver (default) | WOR\_RECEIVER |

WOR cycle

If WOR is transmitting: after the WOR receiver receives the wireless data and outputs it through the serial port, it will wait for 1000ms before entering the WOR again. Users can input the serial port data and return it via the wireless during this period. Each serial byte will be refreshed for 1000ms. Users must transmit the first byte within 1000ms.

* Period T = (1 + WOR) \* 500ms, maximum 4000ms, minimum 500ms
* The longer the WOR monitoring interval period, the lower the average power consumption, but
* the greater the data delay
* **Both the transmitter and the receiver must be the same (very important).**

| **Wireless wake-up time** | **Constant value** |
| --- | --- |
| 500ms | WAKE\_UP\_500 |
| 1000ms | WAKE\_UP\_1000 |
| 1500ms | WAKE\_UP\_1500 |
| 2000ms (default) | WAKE\_UP\_2000 |
| 2500ms | WAKE\_UP\_2500 |
| 3000ms | WAKE\_UP\_3000 |
| 3500ms | WAKE\_UP\_3500 |
| 4000ms | WAKE\_UP\_4000 |

Check buffer

First we must introduce a simple but usefully method to check if something is in the receiving buffer

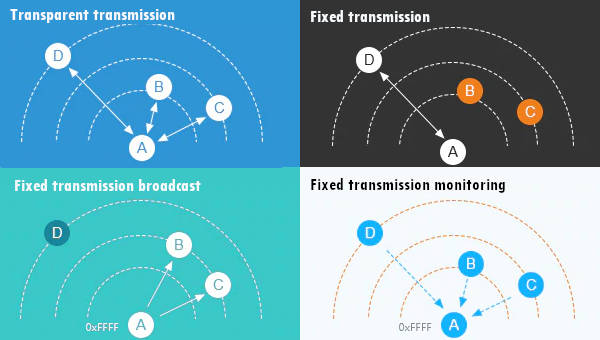
|  |  |
| --- | --- |
| 1 | **int** available(); |

It’s simply return how many bytes you have in the current stream.

Send receive messages

Normal transmission mode

Normal/Transparent transmission mode is used to send messages to all device with same address and channel.

LoRa E32 transmitting scenarios, lines are channels

There are a lot of method to send/receive message, we are going to explain in detail:

|  |  |
| --- | --- |
| 1  2 | ResponseStatus sendMessage(**const** String message);  ResponseContainer receiveMessage(); |

First method is sendMessage and is used to send a String to a device in **Normal mode**.

|  |  |
| --- | --- |
| 1  2 | ResponseStatus rs = e22ttl.sendMessage("Prova");  Serial.println(rs.getResponseDescription()); |

The other device simply do on the loop

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | **if** (e32ttl.available()  > 1){      ResponseContainer rs = e32ttl.receiveMessage();      String message = rs.data; // First ever get the data      Serial.println(rs.status.getResponseDescription());      Serial.println(message);  } |

Pay attention if you receive multiple message in the buffer and you don’t want read all in one time you must use ResponseContainer rs = e32ttl.receiveMessageUntil(); with a delimiter put on the end of sending message.

If you enabled the RSSI you must use receiveMessageRSSI.

Manage structure

If you want send a complex structure you can use this method

|  |  |
| --- | --- |
| 1  2 | ResponseStatus sendMessage(**const** **void** \*message, **const** **uint8\_t** size);  ResponseStructContainer receiveMessage(**const** **uint8\_t** size); |

It’s used to send structure, for example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | **struct** Messaggione {  **char** type[5];  **char** message[8];  **bool** mitico;  };  **struct** Messaggione messaggione = {"TEMP", "Peple", **true**};      ResponseStatus rs = e22ttl.sendMessage(&messaggione, **sizeof**(Messaggione));  Serial.println(rs.getResponseDescription()); |

and the other side you can receive the message so

|  |  |
| --- | --- |
| 1  2  3  4  5 | ResponseStructContainer rsc = e22ttl.receiveMessage(**sizeof**(Messaggione));  **struct** Messaggione messaggione = \*(Messaggione\*) rsc.data;  Serial.println(messaggione.message);  Serial.println(messaggione.mitico);          rsc.close(); |

If you enabled the RSSI you must use receiveMessageRSSI.

Read partial structure

If you want to read the first part of the message to manage more types of structure you can use this method.

|  |  |
| --- | --- |
| 1 | ResponseContainer receiveInitialMessage(**const** **uint8\_t** size); |

I create It to receive a string with type or other to identify the strucuture to load.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | **struct** Messaggione { // Partial structure without type  **char** message[8];  **bool** mitico;  };    **char** type[5]; // first part of structure  ResponseContainer rs = e32ttl.receiveInitialMessage(**sizeof**(type));          // Put string in a char array (not needed)  **memcpy** ( type, rs.data.c\_str(), **sizeof**(type) );    Serial.println("READ TYPE: ");  Serial.println(rs.status.getResponseDescription());  Serial.println(type);            // Read the rest of structure  ResponseStructContainer rsc = e32ttl.receiveMessage(**sizeof**(Messaggione));  **struct** Messaggione messaggione = \*(Messaggione\*) rsc.data;          rsc.close(); |

Fixed mode instead of normal mode

At same manner I create a set of method to use with fixed transmission

Fixed transmission

**You need to change only the sending method, because the destination device don’t receive the preamble with Address and Channel quando settato il fixed mode.**

So for String message you have

|  |  |
| --- | --- |
| 1  2 | ResponseStatus sendFixedMessage(byte ADDH, byte ADDL, byte CHAN, **const** String message);  ResponseStatus sendBroadcastFixedMessage(byte CHAN, **const** String message); |

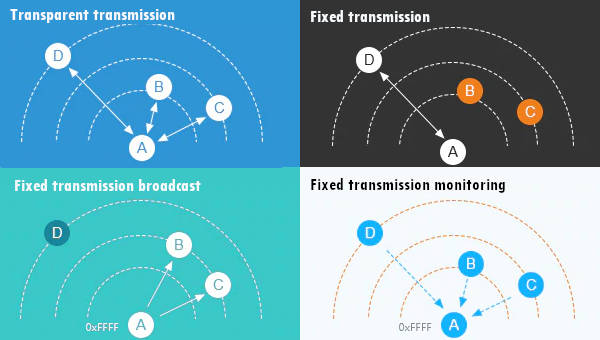
and for structure you have

|  |  |
| --- | --- |
| 1  2 | ResponseStatus sendFixedMessage(byte ADDH, byte ADDL, byte CHAN, **const** **void** \*message, **const** **uint8\_t** size);  ResponseStatus sendBroadcastFixedMessage(byte CHAN, **const** **void** \*message, **const** **uint8\_t** size ); |

Here a simple example

|  |  |
| --- | --- |
| 1  2 | ResponseStatus rs = e22ttl.sendFixedMessage(0, 0, 0x17, &messaggione, **sizeof**(Messaggione));  //  ResponseStatus rs = e22ttl.sendFixedMessage(0, 0, 0x17, "Ciao"); |

Fixed transmission have more scenarios

LoRa E32 transmitting scenarios, lines are channels

If you send to a specific device (second scenarios Fixed transmission) you must add ADDL, ADDH and CHAN to identify It directly.

|  |  |
| --- | --- |
| 1 | ResponseStatus rs = e22ttl.sendFixedMessage(2, 2, 0x17, "Message to a device"); |

If you want send a message to all device in a specified Channel you can use this method.

|  |  |
| --- | --- |
| 1 | ResponseStatus rs = e22ttl.sendBroadcastFixedMessage(0x17, "Message to a devices of a channel"); |

If you want receive all broadcast message in the network you must set your ADDH and ADDL with BROADCAST\_ADDRESS.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | ResponseStructContainer c;  c = e22ttl100.getConfiguration();  // It's important get configuration pointer before all other operation  Configuration configuration = \*(Configuration\*) c.data;  Serial.println(c.status.getResponseDescription());  Serial.println(c.status.code);    printParameters(configuration);  configuration.ADDL = BROADCAST\_ADDRESS;  configuration.ADDH = BROADCAST\_ADDRESS;    // Set configuration changed and set to not hold the configuration  ResponseStatus rs = e32ttl100.setConfiguration(configuration, WRITE\_CFG\_PWR\_DWN\_LOSE);  Serial.println(rs.getResponseDescription());  Serial.println(rs.code);  printParameters(configuration);      c.close(); |

Wireless configuration

This device supports wireless configuration with special commands but seems not to work, I ask to EBYTE but no response was received.

I implement a command that send the packet in the correct way (tested with logic analyzer) but seems not to work.

By the way, first you muset activate RSSI noise environment, than you can use the command like so:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | Configuration configuration;      configuration.ADDL = 0x13;    configuration.ADDH = 0x13;    configuration.NETID = 0x00;      configuration.CHAN = 23;      configuration.SPED.uartBaudRate = UART\_BPS\_9600;    configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24;    configuration.SPED.uartParity = MODE\_00\_8N1;      configuration.OPTION.subPacketSetting = SPS\_240\_00;    configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED;    configuration.OPTION.transmissionPower = POWER\_22;      configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_DISABLED;    configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_FIXED\_TRANSMISSION;    configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED;    configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED;    configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_TRANSMITTER;    configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011;      // Send message  ResponseStatus rs = e22ttl100.sendConfigurationMessage(0, DESTINATION\_ADDL, 23, &configuration);  // Check If there is some problem of successfully send  Serial.println(rs.getResponseDescription()); |

Thanks

Ebyte LoRa E22 device for Arduino, esp32 or esp8266: configuration – 3

BY [RENZO MISCHIANTI](https://mischianti.org/author/reef/) · PUBLISHED 29 MARCH 2022 · UPDATED 16 JUNE 2023

**Spread the love**

**LoRa or Long Range wireless data telemetry** is a technology pioneered by Semtech that operates at a lower frequency than NRF24L01 (433 MHz, 868 MHz, or 916 MHz against 2.4 GHz for the NRF24L01) but at thrice the distance (from 4000m to 10000m).

Now we try to examine better how to configure the Ebyte E22 UART LoRa device based on SX1262/SX1268 Wireless Modules.

[Support Forum](https://mischianti.org/forums/forum/mischiantis-libraries/ebyte-lora-e22-uart-devices-sx1262-sx1268/)

Ebyte LoRa E22 device for Arduino, esp32 or esp8266 Configuration

You can find here [AliExpress (433MHz 5.5Km)](https://s.click.aliexpress.com/e/_DlLGLbF) - [AliExpress (433MHz 10Km)](https://s.click.aliexpress.com/e/_DCGzJIt) - [AliExpress (868MHz 915Mhz 5.5Km)](https://s.click.aliexpress.com/e/_DczWHrb) - [AliExpress (868MHz 915Mhz 10Km)](https://s.click.aliexpress.com/e/_DeoQrNn)

I created this library to manage EBYTE E22 especially to simplify the configuration process because can be very tedious.

[](https://mischianti.org/wp-content/uploads/2019/09/ebyte-lora-exx-pinout-1.jpg)sx1278 sx1276 wireless lora uart module serial 3000m arduino 433 rf

**If you have device freezing problems, you must put a pull-up 4.7k resistor or better connect to the device AUX pin.**

* [Library](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-configuration-3/#Library)
* [Connection schemas for programming](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-configuration-3/#Connection_schemas_for_programming)
  + [Normal configuration (transparent)](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-configuration-3/#Normal_configuration_transparent)
  + [Wiring for programming/sleep mode](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-configuration-3/#Wiring_for_programmingsleep_mode)
* [Basic configuration option](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-configuration-3/#Basic_configuration_option)
  + [Get configuration](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-configuration-3/#Get_configuration)
  + [Set configuration](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-configuration-3/#Set_configuration)
* [Thanks](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-configuration-3/#Thanks)

Library

You can find my library [here](https://github.com/xreef/EByte_LoRa_E22_Series_Library), and It’s available on Arduino IDE library manager.

EByte LoRa E22 E32 Arduino library manager

To download.

Click the [DOWNLOADS](https://github.com/xreef/EByte_LoRa_E22_Series_Library/archive/master.zip)button in the top right corner, rename the uncompressed folder LoRa\_E22.

Check that the LoRa\_E22 folder contains LoRa\_E22.cpp and LoRa\_E22.h.

Place the LoRa\_E22 library folder in your /libraries/ folder.

You may need to create the libraries subfolder if it’s your first library.

Restart the IDE.

Connection schemas for programming

For the basic usage, we had used a specified configuration for Arduino. Still, you are working only in “Normal mode” in that configuration. Now we will manage only the needed pins dynamic (RX, TX) to simplify the programming process and the others in a static way.

| **Mode** | **M1** | **M0** | **Explanation** |
| --- | --- | --- | --- |
| Normal | 0 | 0 | UART and wireless channel are open, transparent transmission is on (Supports configuration over air via special command) |
| WOR Mode | 0 | 1 | Can be defined as WOR transmitter and WOR receiver |
| Configuration mode | 1 | 0 | Users can access the register through the serial port to control the working state of the module |
| Deep sleep mode | 1 | 1 | Sleep mode |

Normal configuration (transparent)

Ebyte LoRa E32 E22 E220 Arduino UNO normal mode breadboard

|  |  |
| --- | --- |
| M0 | GND (Set normal mode) |
| M1 | GND (Set normal mode) |
| TX | PIN 2 (PullUP 4,7KΩ) |
| RX | PIN 3 (PullUP 4,7KΩ & Voltage divider) |
| AUX | Not connected |
| VCC | 5v |
| GND | GND |

and this configuration for Wemos D1 mini:

Ebyte LoRa E32 E22 E220 Wemos D1 normal mode breadboard

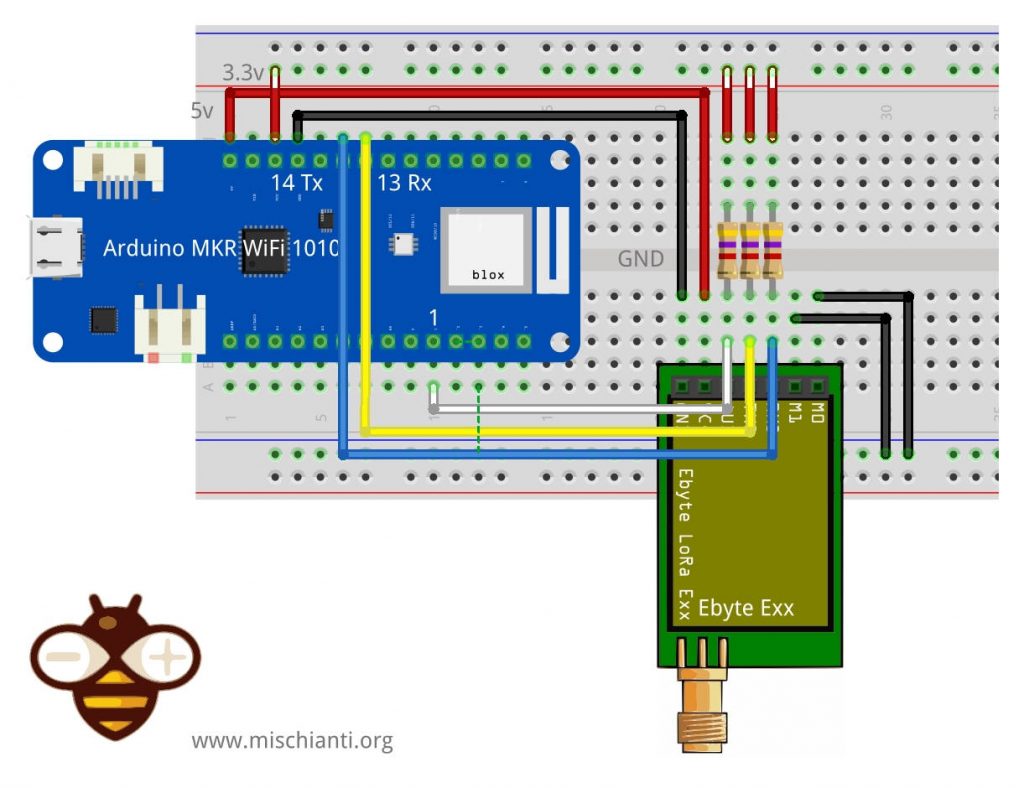
|  |  |
| --- | --- |
| M0 | GND (Set normal mode) |
| M1 | GND (Set normal mode) |
| TX | PIN D2 (PullUP 4,7KΩ) |
| RX | PIN D3 (PullUP 4,7KΩ) |
| AUX | Not connected |
| VCC | 3.3v/5v |
| GND | GND |

ESP-32:

Ebyte LoRa E32 E22 E220 ESP32 DEV KIT V1 normal mode breadboard

|  |  |
| --- | --- |
| M0 | GND (Set normal mode) |
| M1 | GND (Set normal mode) |
| RX | TX2 (PullUP 4,7KΩ) |
| TX | RX2 (PullUP 4,7KΩ) |
| AUX | Not connected (PullUP 4,7KΩ) |
| VCC | 3.3v/5v |
| GND | GND |

Arduino MKR WiFi 1010:

[](https://mischianti.org/wp-content/uploads/2021/12/Ebyte_LoRa_Exx_Arduino_MKR_WiFi_1010_normal_mode_connected_breadboard.jpg)Ebyte LoRa Exx Arduino MKR WiFi 1010 normal mode connected breadboard

|  |  |
| --- | --- |
| M0 | GND (Set normal mode) |
| M1 | GND (Set normal mode) |
| TX | PIN 14 Tx (PullUP 4,7KΩ) |
| RX | PIN 13 Rx (PullUP 4,7KΩ) |
| AUX | Not connected (PullUP 4,7KΩ) |
| VCC | 3.3v/5v |
| GND | GND |

Wiring for programming/sleep mode

**To configure It, you must set M0 to GND and M1 to high (remember to use 3.3v).**

But If you connect all pins, the library sets HIGH or LOW the pins as needed without a problem.

Ebyte LoRa E22 Wemos D1 configuration mode breadboard

|  |  |
| --- | --- |
| M0 | GND (Set programming/sleep mode) |
| M1 | VCC (Set programming/sleep mode) |
| TX | PIN D2 (PullUP 4,7KΩ) |
| RX | PIN D3 (PullUP 4,7KΩ) |
| AUX | Not connected |
| VCC | 3.3v/5v |
| GND | GND |

Ebyte LoRa E22 Arduino UNO configuration mode breadboard

|  |  |
| --- | --- |
| M0 | GND (Set programming/sleep mode) |
| M1 | 3.3v (Set programming/sleep mode) |
| TX | PIN 2 (PullUP 4,7KΩ) |
| TX | PIN 3 (PullUP 4,7KΩ & Voltage divider) |
| AUX | Not connected |
| VCC | 5v |
| GND | GND |

Ebyte LoRa E22 ESP32 DEV KIT V1 configuration mode breadboard

|  |  |
| --- | --- |
| M0 | GND (Set programming/sleep mode) |
| M1 | 3.3v (Set programming/sleep mode) |
| RX | TX2 (PullUP 4,7KΩ) |
| TX | RX2 (PullUP 4,7KΩ) |
| AUX | Not connected |
| VCC | 3.3v/5v |
| GND | GND |

LoRa E22 Arduino MKR WiFi 1010 sleep config mode breadboard

|  |  |
| --- | --- |
| M0 | GND (Set programming/sleep mode) |
| M1 | 3.3v (Set programming/sleep mode) |
| TX | PIN 14 Tx (PullUP 4,7KΩ) |
| RX | PIN 13 Rx (PullUP 4,7KΩ) |
| AUX | Not connected (PullUP 4,7KΩ) |
| VCC | 3.3v/5v |
| GND | GND |

In this mode, you can manage the configuration of the device

Basic configuration option

| **Name** | **Description** | **Address** |
| --- | --- | --- |
| ADDH | High address byte of the module (the default 00H) | 00H |
| ADDL | Low address byte of the module (the default 00H) | 01H |
| NETID | Network address, used to distinguish the network. | 02H |
| SPED | Information about data rate parity bit and Air data rate | 03H |
| OPTION | Type of transmission, pull-up settings, wake-up time, FEC, Transmission power | 04H |
| CHAN | Communication channel（410M + CHAN\*1M）, default 17H (433MHz), **valid only for 433MHz device** | 05H |
| TRANSMISSION\_MODE | All transmission parameters | 06H |
| CRTYPT\_H | User encryption | 07H |
| CRTYPT\_L | User encryption | 08H |

You can find configuration options in the Library article.

Get configuration

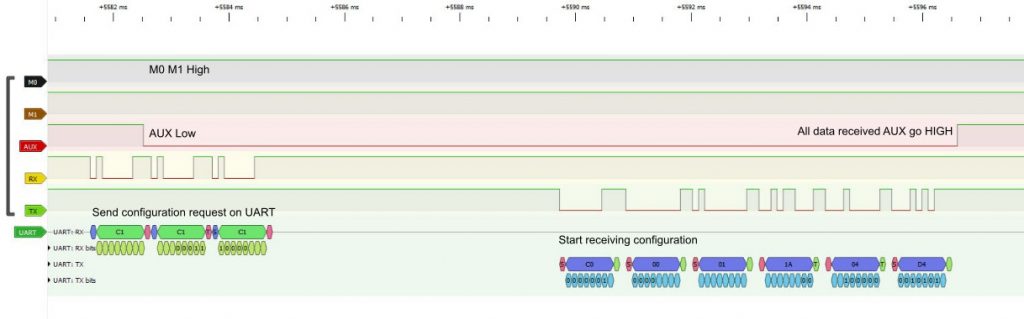
Arduino example sketch

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123 | /\*   \* EByte LoRa E22   \* Get configuration.   \* [https://mischianti.org](https://mischianti.org/)   \*   \* E22        ----- Arduino   \* M0         ----- 7 (or GND)   \* M1         ----- 6 (or HIGH)   \* RX         ----- 4 (PullUP)   \* TX         ----- 5 (PullUP)   \* AUX        ----- 3  (PullUP)   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/  #include "Arduino.h"  #include "LoRa\_E22.h"    // ---------- esp8266 pins --------------  //LoRa\_E22 e22ttl(RX, TX, AUX, M0, M1);  // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(D3, D4, D5, D7, D6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(D2, D3); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(D2, D3); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, D5, D7, D6); // AUX M0 M1  // -------------------------------------    // ---------- Arduino pins --------------  LoRa\_E22 e22ttl(4, 5, 3, 7, 6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(4, 5); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(4, 5); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, 3, 7, 6); // AUX M0 M1  // -------------------------------------    // ------------- Arduino MKR WiFi 1010 -------------  // LoRa\_E220 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ---------- esp32 pins --------------  //LoRa\_E22 e22ttl(&Serial2, 18, 21, 19); //  RX AUX M0 M1    //LoRa\_E22 e22ttl(&Serial2, 22, 4, 18, 21, 19, UART\_BPS\_RATE\_9600); //  esp32 RX <-- e22 TX, esp32 TX --> e22 RX AUX M0 M1  // -------------------------------------    **void** printParameters(struct Configuration configuration);  **void** printModuleInformation(struct ModuleInformation moduleInformation);    **void** **setup**() {      Serial.begin(9600);  **while**(**!**Serial){};      delay(500);        Serial.println();          // Startup all pins and UART      e22ttl.begin();        ResponseStructContainer c;      c **=** e22ttl.getConfiguration();      // It's important get configuration pointer before all other operation      Configuration configuration **=** **\***(Configuration**\***) c.data;      Serial.println(c.status.getResponseDescription());      Serial.println(c.status.code);        printParameters(configuration);        ResponseStructContainer cMi;      cMi **=** e22ttl.getModuleInformation();      // It's important get information pointer before all other operation      ModuleInformation mi **=** **\***(ModuleInformation**\***)cMi.data;        Serial.println(cMi.status.getResponseDescription());      Serial.println(cMi.status.code);        printModuleInformation(mi);  }    **void** **loop**() {    }  **void** printParameters(struct Configuration configuration) {      Serial.println("----------------------------------------");        Serial.print(F("HEAD : "));  Serial.print(configuration.COMMAND, HEX);Serial.print(" ");Serial.print(configuration.STARTING\_ADDRESS, HEX);Serial.print(" ");Serial.println(configuration.LENGHT, HEX);      Serial.println(F(" "));      Serial.print(F("AddH : "));  Serial.println(configuration.ADDH, HEX);      Serial.print(F("AddL : "));  Serial.println(configuration.ADDL, HEX);      Serial.print(F("NetID : "));  Serial.println(configuration.NETID, HEX);      Serial.println(F(" "));      Serial.print(F("Chan : "));  Serial.print(configuration.CHAN, DEC); Serial.print(" -> "); Serial.println(configuration.getChannelDescription());      Serial.println(F(" "));      Serial.print(F("SpeedParityBit     : "));  Serial.print(configuration.SPED.uartParity, BIN);Serial.print(" -> "); Serial.println(configuration.SPED.getUARTParityDescription());      Serial.print(F("SpeedUARTDatte     : "));  Serial.print(configuration.SPED.uartBaudRate, BIN);Serial.print(" -> "); Serial.println(configuration.SPED.getUARTBaudRateDescription());      Serial.print(F("SpeedAirDataRate   : "));  Serial.print(configuration.SPED.airDataRate, BIN);Serial.print(" -> "); Serial.println(configuration.SPED.getAirDataRateDescription());      Serial.println(F(" "));      Serial.print(F("OptionSubPacketSett: "));  Serial.print(configuration.OPTION.subPacketSetting, BIN);Serial.print(" -> "); Serial.println(configuration.OPTION.getSubPacketSetting());      Serial.print(F("OptionTranPower    : "));  Serial.print(configuration.OPTION.transmissionPower, BIN);Serial.print(" -> "); Serial.println(configuration.OPTION.getTransmissionPowerDescription());      Serial.print(F("OptionRSSIAmbientNo: "));  Serial.print(configuration.OPTION.RSSIAmbientNoise, BIN);Serial.print(" -> "); Serial.println(configuration.OPTION.getRSSIAmbientNoiseEnable());      Serial.println(F(" "));      Serial.print(F("TransModeWORPeriod : "));  Serial.print(configuration.TRANSMISSION\_MODE.WORPeriod, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getWORPeriodByParamsDescription());      Serial.print(F("TransModeTransContr: "));  Serial.print(configuration.TRANSMISSION\_MODE.WORTransceiverControl, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getWORTransceiverControlDescription());      Serial.print(F("TransModeEnableLBT : "));  Serial.print(configuration.TRANSMISSION\_MODE.enableLBT, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getLBTEnableByteDescription());      Serial.print(F("TransModeEnableRSSI: "));  Serial.print(configuration.TRANSMISSION\_MODE.enableRSSI, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getRSSIEnableByteDescription());      Serial.print(F("TransModeEnabRepeat: "));  Serial.print(configuration.TRANSMISSION\_MODE.enableRepeater, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getRepeaterModeEnableByteDescription());      Serial.print(F("TransModeFixedTrans: "));  Serial.print(configuration.TRANSMISSION\_MODE.fixedTransmission, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getFixedTransmissionDescription());          Serial.println("----------------------------------------");  }  **void** printModuleInformation(struct ModuleInformation moduleInformation) {      Serial.println("----------------------------------------");      Serial.print(F("HEAD: "));  Serial.print(moduleInformation.COMMAND, HEX);Serial.print(" ");Serial.print(moduleInformation.STARTING\_ADDRESS, HEX);Serial.print(" ");Serial.println(moduleInformation.LENGHT, DEC);        Serial.print(F("Model no.: "));  Serial.println(moduleInformation.model, HEX);      Serial.print(F("Version  : "));  Serial.println(moduleInformation.version, HEX);      Serial.print(F("Features : "));  Serial.println(moduleInformation.features, HEX);      Serial.println("----------------------------------------");    } |

Here is the result of the sketch

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | Success  1  ----------------------------------------  HEAD : C1 0 9    AddH : 0  AddL : 3  NetID : 0    Chan : 23 -> 433MHz    SpeedParityBit     : 0 -> 8N1 (Default)  SpeedUARTDatte     : 11 -> 9600bps (default)  SpeedAirDataRate   : 10 -> 2.4kbps (default)    OptionSubPacketSett: 0 -> 240bytes (default)  OptionTranPower    : 0 -> 22dBm (Default)  OptionRSSIAmbientNo: 0 -> Disabled (default)    TransModeWORPeriod : 0 -> 500ms  TransModeTransContr: 0 -> WOR Receiver (default)  TransModeEnableLBT : 0 -> Disabled (default)  TransModeEnableRSSI: 0 -> Disabled (default)  TransModeEnabRepeat: 0 -> Disabled (default)  TransModeFixedTrans: 1 -> Fixed transmission (first three bytes can be used as high/low address and channel)  ----------------------------------------  Success  1  ----------------------------------------  HEAD: C1 80 7  Model no.: 16  Version  : A  Features : 0  ---------------------------------------- |

As you can see, I add a lot of constructors for every device, if you change the first configuration you can switch the device with the same code.

[](https://mischianti.org/wp-content/uploads/2019/12/E32_request_configuration_logic_analyzer.jpg)LoRa E32 request configuration logic analyzer Arduino

To get the correct information, I add some #define to change the device type (same #define manage more other devices, I create only one for type for simplicity).

|  |  |
| --- | --- |
| 1  2 | #define E22\_22  #define E22\_30 |

**You can select only one of them**. The parameter changes the Transmission power constant as described in the configuration schema.

In the same manner, you can select one reference frequences

|  |  |
| --- | --- |
| 1  2  3  4  5 | #define FREQUENCY\_433  #define FREQUENCY\_170  #define FREQUENCY\_470  #define FREQUENCY\_868  #define FREQUENCY\_915 |

**You can choose** only one of them. The parameter changes the reference frequencies only for display purposes,

Set configuration

Naturally, when you have a configuration you want to change It for your purpose, I think you can get the configuration from a device, modify what you want and set It.

But in the example sketch, I add a set of configuration commented, you can uncomment the config you want, and in the other examples in the library, there is a reference to the relative configuration to apply.

Ricorda che il parametro saveType è fondamentale per mantenere le opzioni al riavvio del dispositivo, WRITE\_CFG\_PWR\_DWN\_LOSE naturalmente perderai le impostazioni con WRITE\_CFG\_PWR\_DWN\_SAVE non perderai le impostazioni.

Here is an Arduino Nano 33 IoT sketch:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373 | /\*   \* LoRa E22-TTL-100   \* Get configuration.   \* [https://mischianti.org](https://mischianti.org/)   \*   \* E22        ----- Arduino Nano 33 IoT   \* M0         ----- 4 (or GND)   \* M1         ----- 6 (or HIGH)   \* RX         ----- TX (PullUP)   \* TX         ----- RX (PullUP)   \* AUX        ----- 2  (PullUP)   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/  #include "Arduino.h"  #include "LoRa\_E22.h"    // ---------- esp8266 pins --------------  //LoRa\_E22 e22ttl(RX, TX, AUX, M0, M1);  // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(D3, D4, D5, D7, D6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(D2, D3); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(D2, D3); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, D5, D7, D6); // AUX M0 M1  // -------------------------------------    // ---------- Arduino pins --------------  // LoRa\_E22 e22ttl(4, 5, 3, 7, 6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(4, 5); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(4, 5); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, 3, 7, 6); // AUX M0 M1  // -------------------------------------    // ------------- Arduino Nano 33 IoT -------------  LoRa\_E22 e22ttl(**&**Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ------------- Arduino MKR WiFi 1010 -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ---------- esp32 pins --------------  //LoRa\_E22 e22ttl(&Serial2, 18, 21, 19); //  RX AUX M0 M1    //LoRa\_E22 e22ttl(&Serial2, 22, 4, 18, 21, 19, UART\_BPS\_RATE\_9600); //  esp32 RX <-- e22 TX, esp32 TX --> e22 RX AUX M0 M1  // -------------------------------------    **void** printParameters(struct Configuration configuration);  **void** printModuleInformation(struct ModuleInformation moduleInformation);    **void** **setup**() {      Serial.begin(9600);  **while**(**!**Serial){};      delay(500);        Serial.println();          // Startup all pins and UART      e22ttl.begin();        ResponseStructContainer c;      c **=** e22ttl.getConfiguration();      // It's important get configuration pointer before all other operation      Configuration configuration **=** **\***(Configuration**\***) c.data;      Serial.println(c.status.getResponseDescription());      Serial.println(c.status.code);        printParameters(configuration);        configuration.ADDL **=** 0x03;      configuration.ADDH **=** 0x00;      configuration.NETID **=** 0x00;        configuration.CHAN **=** 23;          //  ----------------------- DEFAULT TRANSPARENT -----------------------          configuration.ADDL **=** 0x03;          configuration.ADDH **=** 0x00;          configuration.NETID **=** 0x00;            configuration.CHAN **=** 23;            configuration.SPED.uartBaudRate **=** UART\_BPS\_9600;          configuration.SPED.airDataRate **=** AIR\_DATA\_RATE\_010\_24;          configuration.SPED.uartParity **=** MODE\_00\_8N1;            configuration.OPTION.subPacketSetting **=** SPS\_240\_00;          configuration.OPTION.RSSIAmbientNoise **=** RSSI\_AMBIENT\_NOISE\_DISABLED;          configuration.OPTION.transmissionPower **=** POWER\_22;            configuration.TRANSMISSION\_MODE.enableRSSI **=** RSSI\_DISABLED;          configuration.TRANSMISSION\_MODE.fixedTransmission **=** FT\_TRANSPARENT\_TRANSMISSION;          configuration.TRANSMISSION\_MODE.enableRepeater **=** REPEATER\_DISABLED;          configuration.TRANSMISSION\_MODE.enableLBT **=** LBT\_DISABLED;          configuration.TRANSMISSION\_MODE.WORTransceiverControl **=** WOR\_RECEIVER;          configuration.TRANSMISSION\_MODE.WORPeriod **=** WOR\_2000\_011;      //  ----------------------- DEFAULT TRANSPARENT WITH RSSI -----------------------      //  configuration.ADDL = 0x03;      //  configuration.ADDH = 0x00;      //  configuration.NETID = 0x00;      //      //  configuration.CHAN = 23;      //      //  configuration.SPED.uartBaudRate = UART\_BPS\_9600;      //  configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24;      //  configuration.SPED.uartParity = MODE\_00\_8N1;      //      //  configuration.OPTION.subPacketSetting = SPS\_240\_00;      //  configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED;      //  configuration.OPTION.transmissionPower = POWER\_22;      //      //  configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_ENABLED;      //  configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_TRANSPARENT\_TRANSMISSION;      //  configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED;      //  configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED;      //  configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_RECEIVER;      //  configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011;      //  ----------------------- FIXED SENDER -----------------------  //          configuration.ADDL = 0x02;  //          configuration.ADDH = 0x00;  //          configuration.NETID = 0x00;  //  //          configuration.CHAN = 23;  //  //          configuration.SPED.uartBaudRate = UART\_BPS\_9600;  //          configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24;  //          configuration.SPED.uartParity = MODE\_00\_8N1;  //  //          configuration.OPTION.subPacketSetting = SPS\_240\_00;  //          configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED;  //          configuration.OPTION.transmissionPower = POWER\_22;  //  //          configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_DISABLED;  //          configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_FIXED\_TRANSMISSION;  //          configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED;  //          configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED;  //          configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_TRANSMITTER;  //          configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011;      //      //  ----------------------- FIXED RECEIVER -----------------------  //          configuration.ADDL = 0x03;  //          configuration.ADDH = 0x00;  //          configuration.NETID = 0x00;  //  //          configuration.CHAN = 23;  //  //          configuration.SPED.uartBaudRate = UART\_BPS\_9600;  //          configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24;  //          configuration.SPED.uartParity = MODE\_00\_8N1;  //  //          configuration.OPTION.subPacketSetting = SPS\_240\_00;  //          configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED;  //          configuration.OPTION.transmissionPower = POWER\_22;  //  //          configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_DISABLED;  //          configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_FIXED\_TRANSMISSION;  //          configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED;  //          configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED;  //          configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_RECEIVER;  //          configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011;      //  ----------------------- FIXED SENDER RSSI -----------------------      //      configuration.ADDL = 0x02;      //      configuration.ADDH = 0x00;      //      configuration.NETID = 0x00;      //      //      configuration.CHAN = 23;      //      //      configuration.SPED.uartBaudRate = UART\_BPS\_9600;      //      configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24;      //      configuration.SPED.uartParity = MODE\_00\_8N1;      //      //      configuration.OPTION.subPacketSetting = SPS\_240\_00;      //      configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED;      //      configuration.OPTION.transmissionPower = POWER\_22;      //      //      configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_ENABLED;      //      configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_FIXED\_TRANSMISSION;      //      configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED;      //      configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED;      //      configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_TRANSMITTER;      //      configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011;      //      //  ----------------------- FIXED RECEIVER RSSI -----------------------      //      configuration.ADDL = 0x03;      //      configuration.ADDH = 0x00;      //      configuration.NETID = 0x00;      //      //      configuration.CHAN = 23;      //      //      configuration.SPED.uartBaudRate = UART\_BPS\_9600;      //      configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24;      //      configuration.SPED.uartParity = MODE\_00\_8N1;      //      //      configuration.OPTION.subPacketSetting = SPS\_240\_00;      //      configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED;      //      configuration.OPTION.transmissionPower = POWER\_22;      //      //      configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_ENABLED;      //      configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_FIXED\_TRANSMISSION;      //      configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED;      //      configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED;      //      configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_RECEIVER;      //      configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011;        //  ----------------------- WOR SENDER -----------------------      //      configuration.ADDL = 0x02;      //      configuration.ADDH = 0x00;      //      configuration.NETID = 0x00;      //      //      configuration.CHAN = 23;      //      //      configuration.SPED.uartBaudRate = UART\_BPS\_9600;      //      configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24;      //      configuration.SPED.uartParity = MODE\_00\_8N1;      //      //      configuration.OPTION.subPacketSetting = SPS\_240\_00;      //      configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED;      //      configuration.OPTION.transmissionPower = POWER\_22;      //      //      configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_DISABLED;      //      configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_FIXED\_TRANSMISSION;      //      configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED;      //      configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED;      //      configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_TRANSMITTER;      //      configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011;      //      //  ----------------------- WOR RECEIVER -----------------------  //          configuration.ADDL = 0x03;  //          configuration.ADDH = 0x00;  //          configuration.NETID = 0x00;  //  //          configuration.CHAN = 23;  //  //          configuration.SPED.uartBaudRate = UART\_BPS\_9600;  //          configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24;  //          configuration.SPED.uartParity = MODE\_00\_8N1;  //  //          configuration.OPTION.subPacketSetting = SPS\_240\_00;  //          configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED;  //          configuration.OPTION.transmissionPower = POWER\_22;  //  //          configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_DISABLED;  //          configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_FIXED\_TRANSMISSION;  //          configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED;  //          configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED;  //          configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_RECEIVER;  //          configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_500\_000;      //  ----------------------- BROADCAST MESSAGE 1 -----------------------  //          configuration.ADDL = 0x04;  //          configuration.ADDH = 0x00;  //          configuration.NETID = 0x00;  //  //          configuration.CHAN = 23;  //  //          configuration.SPED.uartBaudRate = UART\_BPS\_9600;  //          configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24;  //          configuration.SPED.uartParity = MODE\_00\_8N1;  //  //          configuration.OPTION.subPacketSetting = SPS\_240\_00;  //          configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED;  //          configuration.OPTION.transmissionPower = POWER\_22;  //  //          configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_DISABLED;  //          configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_FIXED\_TRANSMISSION;  //          configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED;  //          configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED;  //          configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_RECEIVER;  //          configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011;      //  ----------------------- BROADCAST MESSAGE 2 -----------------------  //          configuration.ADDL = 0x05;  //          configuration.ADDH = 0x00;  //          configuration.NETID = 0x00;  //  //          configuration.CHAN = 23;  //  //          configuration.SPED.uartBaudRate = UART\_BPS\_9600;  //          configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24;  //          configuration.SPED.uartParity = MODE\_00\_8N1;  //  //          configuration.OPTION.subPacketSetting = SPS\_240\_00;  //          configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED;  //          configuration.OPTION.transmissionPower = POWER\_22;  //  //          configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_DISABLED;  //          configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_FIXED\_TRANSMISSION;  //          configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED;  //          configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED;  //          configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_RECEIVER;  //          configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011;      //  ----------------------- BROADCAST MESSAGE 3 -----------------------      //      configuration.ADDL = 0x06;      //      configuration.ADDH = 0x00;      //      configuration.NETID = 0x00;      //      //      configuration.CHAN = 23;      //      //      configuration.SPED.uartBaudRate = UART\_BPS\_9600;      //      configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24;      //      configuration.SPED.uartParity = MODE\_00\_8N1;      //      //      configuration.OPTION.subPacketSetting = SPS\_240\_00;      //      configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED;      //      configuration.OPTION.transmissionPower = POWER\_22;      //      //      configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_DISABLED;      //      configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_FIXED\_TRANSMISSION;      //      configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED;      //      configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED;      //      configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_RECEIVER;      //      configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011;        // Set configuration changed and set to not hold the configuration      ResponseStatus rs **=** e22ttl.setConfiguration(configuration, WRITE\_CFG\_PWR\_DWN\_SAVE);      Serial.println(rs.getResponseDescription());      Serial.println(rs.code);        c **=** e22ttl.getConfiguration();      // It's important get configuration pointer before all other operation      configuration **=** **\***(Configuration**\***) c.data;      Serial.println(c.status.getResponseDescription());      Serial.println(c.status.code);        printParameters(configuration);  }    **void** **loop**() {    }  **void** printParameters(struct Configuration configuration) {      Serial.println("----------------------------------------");        Serial.print(F("HEAD : "));  Serial.print(configuration.COMMAND, HEX);Serial.print(" ");Serial.print(configuration.STARTING\_ADDRESS, HEX);Serial.print(" ");Serial.println(configuration.LENGHT, HEX);      Serial.println(F(" "));      Serial.print(F("AddH : "));  Serial.println(configuration.ADDH, HEX);      Serial.print(F("AddL : "));  Serial.println(configuration.ADDL, HEX);      Serial.print(F("NetID : "));  Serial.println(configuration.NETID, HEX);      Serial.println(F(" "));      Serial.print(F("Chan : "));  Serial.print(configuration.CHAN, DEC); Serial.print(" -> "); Serial.println(configuration.getChannelDescription());      Serial.println(F(" "));      Serial.print(F("SpeedParityBit     : "));  Serial.print(configuration.SPED.uartParity, BIN);Serial.print(" -> "); Serial.println(configuration.SPED.getUARTParityDescription());      Serial.print(F("SpeedUARTDatte     : "));  Serial.print(configuration.SPED.uartBaudRate, BIN);Serial.print(" -> "); Serial.println(configuration.SPED.getUARTBaudRateDescription());      Serial.print(F("SpeedAirDataRate   : "));  Serial.print(configuration.SPED.airDataRate, BIN);Serial.print(" -> "); Serial.println(configuration.SPED.getAirDataRateDescription());      Serial.println(F(" "));      Serial.print(F("OptionSubPacketSett: "));  Serial.print(configuration.OPTION.subPacketSetting, BIN);Serial.print(" -> "); Serial.println(configuration.OPTION.getSubPacketSetting());      Serial.print(F("OptionTranPower    : "));  Serial.print(configuration.OPTION.transmissionPower, BIN);Serial.print(" -> "); Serial.println(configuration.OPTION.getTransmissionPowerDescription());      Serial.print(F("OptionRSSIAmbientNo: "));  Serial.print(configuration.OPTION.RSSIAmbientNoise, BIN);Serial.print(" -> "); Serial.println(configuration.OPTION.getRSSIAmbientNoiseEnable());      Serial.println(F(" "));      Serial.print(F("TransModeWORPeriod : "));  Serial.print(configuration.TRANSMISSION\_MODE.WORPeriod, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getWORPeriodByParamsDescription());      Serial.print(F("TransModeTransContr: "));  Serial.print(configuration.TRANSMISSION\_MODE.WORTransceiverControl, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getWORTransceiverControlDescription());      Serial.print(F("TransModeEnableLBT : "));  Serial.print(configuration.TRANSMISSION\_MODE.enableLBT, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getLBTEnableByteDescription());      Serial.print(F("TransModeEnableRSSI: "));  Serial.print(configuration.TRANSMISSION\_MODE.enableRSSI, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getRSSIEnableByteDescription());      Serial.print(F("TransModeEnabRepeat: "));  Serial.print(configuration.TRANSMISSION\_MODE.enableRepeater, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getRepeaterModeEnableByteDescription());      Serial.print(F("TransModeFixedTrans: "));  Serial.print(configuration.TRANSMISSION\_MODE.fixedTransmission, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getFixedTransmissionDescription());          Serial.println("----------------------------------------");  }  **void** printModuleInformation(struct ModuleInformation moduleInformation) {      Serial.println("----------------------------------------");      Serial.print(F("HEAD: "));  Serial.print(moduleInformation.COMMAND, HEX);Serial.print(" ");Serial.print(moduleInformation.STARTING\_ADDRESS, HEX);Serial.print(" ");Serial.println(moduleInformation.LENGHT, DEC);        Serial.print(F("Model no.: "));  Serial.println(moduleInformation.model, HEX);      Serial.print(F("Version  : "));  Serial.println(moduleInformation.version, HEX);      Serial.print(F("Features : "));  Serial.println(moduleInformation.features, HEX);      Serial.println("----------------------------------------");    } |

Here is the result on the console

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93 | Connetti alla porta seriale COM19 a 115200    Success  1  ----------------------------------------  HEAD : C1 0 9    AddH : 0  AddL : 3  NetID : 0    Chan : 23 -> 433MHz    SpeedParityBit     : 0 -> 8N1 (Default)  SpeedUARTDatte     : 11 -> 9600bps (default)  SpeedAirDataRate   : 10 -> 2.4kbps (default)    OptionSubPacketSett: 0 -> 240bytes (default)  OptionTranPower    : 0 -> 22dBm (Default)  OptionRSSIAmbientNo: 0 -> Disabled (default)    TransModeWORPeriod : 0 -> 500ms  TransModeTransContr: 0 -> WOR Receiver (default)  TransModeEnableLBT : 0 -> Disabled (default)  TransModeEnableRSSI: 0 -> Disabled (default)  TransModeEnabRepeat: 0 -> Disabled (default)  TransModeFixedTrans: 1 -> Fixed transmission (first three bytes can be used as high/low address and channel)  ----------------------------------------  Success  1  ----------------------------------------  HEAD: C1 80 7  Model no.: 16  Version  : A  Features : 0  ----------------------------------------    Connetti alla porta seriale COM19 a 115200    Success  1  ----------------------------------------  HEAD : C1 0 9    AddH : 0  AddL : 3  NetID : 0    Chan : 23 -> 433MHz    SpeedParityBit     : 0 -> 8N1 (Default)  SpeedUARTDatte     : 11 -> 9600bps (default)  SpeedAirDataRate   : 10 -> 2.4kbps (default)    OptionSubPacketSett: 0 -> 240bytes (default)  OptionTranPower    : 0 -> 22dBm (Default)  OptionRSSIAmbientNo: 0 -> Disabled (default)    TransModeWORPeriod : 0 -> 500ms  TransModeTransContr: 0 -> WOR Receiver (default)  TransModeEnableLBT : 0 -> Disabled (default)  TransModeEnableRSSI: 0 -> Disabled (default)  TransModeEnabRepeat: 0 -> Disabled (default)  TransModeFixedTrans: 1 -> Fixed transmission (first three bytes can be used as high/low address and channel)  ----------------------------------------  Success  1  Success  1  ----------------------------------------  HEAD : C1 0 9    AddH : 0  AddL : 3  NetID : 0    Chan : 23 -> 433MHz    SpeedParityBit     : 0 -> 8N1 (Default)  SpeedUARTDatte     : 11 -> 9600bps (default)  SpeedAirDataRate   : 10 -> 2.4kbps (default)    OptionSubPacketSett: 0 -> 240bytes (default)  OptionTranPower    : 0 -> 22dBm (Default)  OptionRSSIAmbientNo: 0 -> Disabled (default)    TransModeWORPeriod : 11 -> 2000ms (default)  TransModeTransContr: 0 -> WOR Receiver (default)  TransModeEnableLBT : 0 -> Disabled (default)  TransModeEnableRSSI: 0 -> Disabled (default)  TransModeEnabRepeat: 0 -> Disabled (default)  TransModeFixedTrans: 0 -> Transparent transmission (default)  ---------------------------------------- |

The library is quite simple, but in the next chapter, we will test various device options.

Thanks

Ebyte LoRa E22 device for Arduino, esp32 or esp8266: fixed transmission, broadcast, monitor, and RSSI – 4

BY [RENZO MISCHIANTI](https://mischianti.org/author/reef/) · 4 APRIL 2022

**Spread the love**

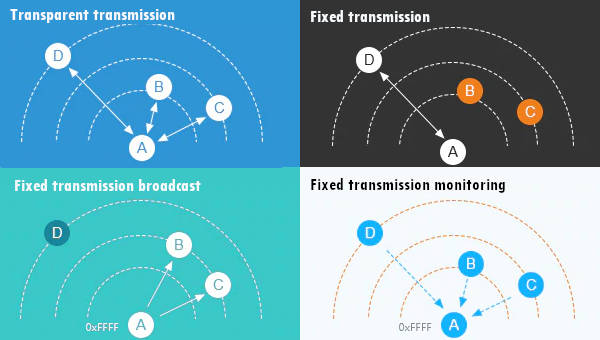
We will now understand the various transmission types of our E22 UART LoRa device based on popular Semtech’s SX1262 and SX1268 RF chips.

[Support Forum](https://mischianti.org/forums/forum/mischiantis-libraries/ebyte-lora-e22-uart-devices-sx1262-sx1268/)

Ebyte LoRa E22 device for Arduino, esp32 or esp8266 fixed transmission and RSSI

Here a selection of LoRa devices [AliExpress (433MHz 5.5Km)](https://s.click.aliexpress.com/e/_DlLGLbF) - [AliExpress (433MHz 10Km)](https://s.click.aliexpress.com/e/_DCGzJIt) - [AliExpress (868MHz 915Mhz 5.5Km)](https://s.click.aliexpress.com/e/_DczWHrb) - [AliExpress (868MHz 915Mhz 10Km)](https://s.click.aliexpress.com/e/_DeoQrNn)

In the first part, we’ve used a transparent transmission, so we send to all and receive from all that have the same address e channel.

LoRa E32 transmitting scenarios

But It isn’t a standard scenario, and we usually want to send to a specified point and receive a response.

**If you have trouble with the device’s freeze, you must put a pull-up 4.7k resistor or better connect to the device AUX pin.**

* [Normal mode](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-fixed-transmission-broadcast-monitor-and-rssi-4/#Normal_mode)
  + [Transparent transmission](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-fixed-transmission-broadcast-monitor-and-rssi-4/#Transparent_transmission)
  + [Transparent transmission and RSSI](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-fixed-transmission-broadcast-monitor-and-rssi-4/#Transparent_transmission_and_RSSI)
* [Fixed transmission](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-fixed-transmission-broadcast-monitor-and-rssi-4/#Fixed_transmission)
  + [Fixed transmission: point to point](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-fixed-transmission-broadcast-monitor-and-rssi-4/#Fixed_transmission_point_to_point)
  + [Fixed transmission with RSSI:](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-fixed-transmission-broadcast-monitor-and-rssi-4/#Fixed_transmission_with_RSSI)
  + [Fixed transmission: broadcast](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-fixed-transmission-broadcast-monitor-and-rssi-4/#Fixed_transmission_broadcast)
  + [Fixed transmission: monitoring](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-fixed-transmission-broadcast-monitor-and-rssi-4/#Fixed_transmission_monitoring)
* [Thanks](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-fixed-transmission-broadcast-monitor-and-rssi-4/#Thanks)

Normal mode

For normal transmission, you must set M0 and M1 to LOW, and It’s better if you connect the AUX pin to have a better synchronization, but not needed. You can check the connection of the AUX pin in the library complete example connection, and you only must add the PIN to the constructor.

Ebyte LoRa E32 E22 E220 Arduino UNO normal mode breadboard

|  |  |
| --- | --- |
| M0 | GND (Set normal mode) |
| M1 | GND (Set normal mode) |
| TX | PIN 2 (PullUP 4,7KΩ) |
| RX | PIN 3 (PullUP 4,7KΩ & Voltage divider) |
| AUX | Not connected (PullUP 4,7KΩ) |
| VCC | 5v |
| GND | GND |

and this configuration for Wemos D1 mini:

Ebyte LoRa E32 E22 E220 Wemos D1 normal mode breadboard

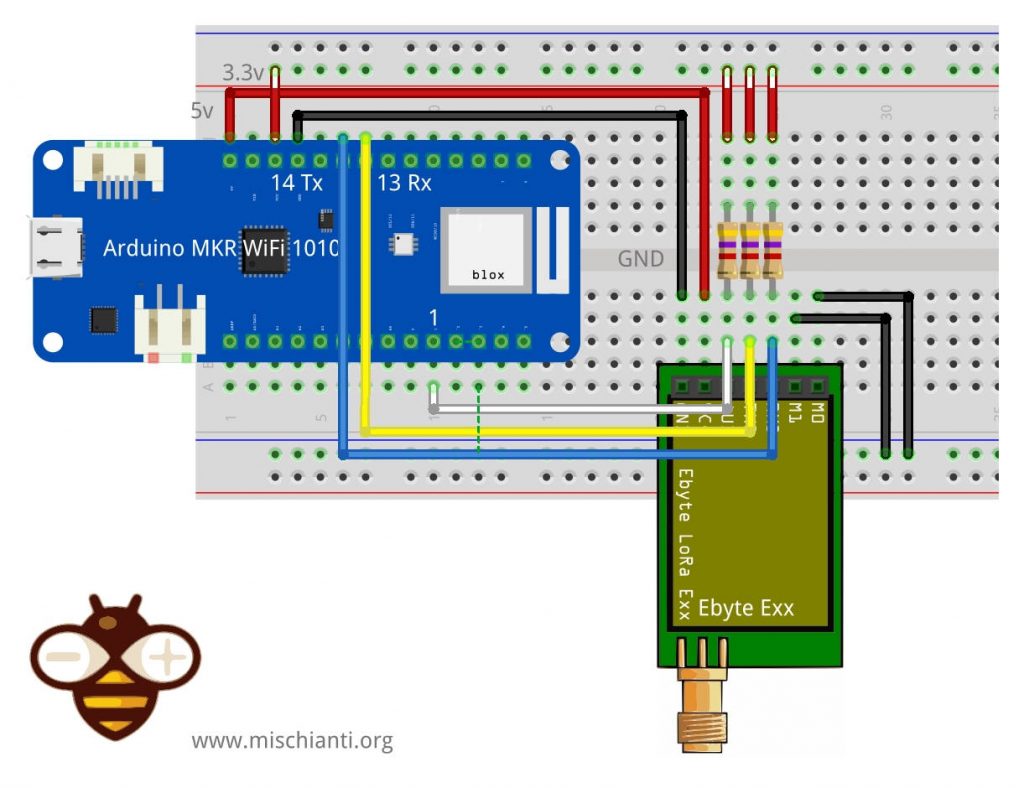
|  |  |
| --- | --- |
| M0 | GND (Set normal mode) |
| M1 | GND (Set normal mode) |
| TX | PIN D2 (PullUP 4,7KΩ) |
| RX | PIN D3 (PullUP 4,7KΩ) |
| AUX | Not connected (PullUP 4,7KΩ) |
| VCC | 3.3v/5v |
| GND | GND |

ESP-32:

Ebyte LoRa E32 E22 E220 ESP32 DEV KIT V1 normal mode breadboard

|  |  |
| --- | --- |
| M0 | GND (Set normal mode) |
| M1 | GND (Set normal mode) |
| RX | TX2 (PullUP 4,7KΩ) |
| TX | RX2 (PullUP 4,7KΩ) |
| AUX | Not connected (PullUP 4,7KΩ) |
| VCC | 3.3v/5v |
| GND | GND |

Arduino MKR WiFi 1010:

[](https://mischianti.org/wp-content/uploads/2021/12/Ebyte_LoRa_Exx_Arduino_MKR_WiFi_1010_normal_mode_connected_breadboard.jpg)Ebyte LoRa Exx Arduino MKR WiFi 1010 normal mode connected breadboard

|  |  |
| --- | --- |
| M0 | GND (Set normal mode) |
| M1 | GND (Set normal mode) |
| TX | PIN 14 Tx (PullUP 4,7KΩ) |
| RX | PIN 13 Rx (PullUP 4,7KΩ) |
| AUX | Not connected (PullUP 4,7KΩ) |
| VCC | 3.3v/5v |
| GND | GND |

So you must only instantiate the constructor.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | // ---------- esp8266 pins --------------  //LoRa\_E22 e22ttl(RX, TX, AUX, M0, M1);  // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(D3, D4, D5, D7, D6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(D2, D3); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(D2, D3); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, D5, D7, D6); // AUX M0 M1  // -------------------------------------    // ---------- Arduino pins --------------  LoRa\_E22 e22ttl(4, 5, 3, 7, 6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(4, 5); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(4, 5); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, 3, 7, 6); // AUX M0 M1  // -------------------------------------    // ------------- Arduino Nano 33 IoT -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ------------- Arduino MKR WiFi 1010 -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ---------- esp32 pins --------------  //LoRa\_E22 e22ttl(&Serial2, 18, 21, 19); //  RX AUX M0 M1    //LoRa\_E22 e22ttl(&Serial2, 22, 4, 18, 21, 19, UART\_BPS\_RATE\_9600); //  esp32 RX <-- e22 TX, esp32 TX --> e22 RX AUX M0 M1  // ------------------------------------- |

Transparent transmission

Generic sketch, select the correct constructor for every device:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90 | /\*   \* LoRa E22   \* send a transparent message, you must check that the transmitter and receiver have the same   \* CHANNEL ADDL and ADDH   \*   \* Renzo Mischianti <[https://mischianti.org](https://mischianti.org/)>   \* <https://mischianti.org/category/my-libraries/ebyte-lora-e22-devices/>   \*   \* E22        ----- Arduino   \* M0         ----- 7 (or GND)   \* M1         ----- 6 (or GND)   \* RX         ----- 4 (PullUP)   \* TX         ----- 5 (PullUP)   \* AUX        ----- 3  (PullUP)   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/    #include "Arduino.h"  #include "LoRa\_E22.h"    // ---------- esp8266 pins --------------  //LoRa\_E22 e22ttl(RX, TX, AUX, M0, M1);  // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(D3, D4, D5, D7, D6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(D2, D3); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(D2, D3); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, D5, D7, D6); // AUX M0 M1  // -------------------------------------    // ---------- Arduino pins --------------  LoRa\_E22 e22ttl(4, 5, 3, 7, 6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(4, 5); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(4, 5); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, 3, 7, 6); // AUX M0 M1  // -------------------------------------    // ---------- esp32 pins --------------  // LoRa\_E22 e22ttl(&Serial2, 18, 21, 19); //  RX AUX M0 M1    //LoRa\_E22 e22ttl(&Serial2, 22, 4, 18, 21, 19, UART\_BPS\_RATE\_9600); //  esp32 RX <-- e22 TX, esp32 TX --> e22 RX AUX M0 M1  // -------------------------------------    **void** **setup**() {    Serial.begin(9600);    delay(500);      // Startup all pins and UART    e22ttl.begin();    //  If you have ever change configuration you must restore It  //  ResponseStructContainer c;  //  c = e22ttl.getConfiguration();  //  Configuration configuration = \*(Configuration\*) c.data;  //  Serial.println(c.status.getResponseDescription());  //  configuration.CHAN = 0x17;  //  configuration.OPTION.fixedTransmission = FT\_TRANSPARENT\_TRANSMISSION;  //  e22ttl.setConfiguration(configuration, WRITE\_CFG\_PWR\_DWN\_SAVE);      Serial.println("Hi, I'm going to send message!");      // Send message    ResponseStatus rs **=** e22ttl.sendMessage("Hello, world?");    // Check If there is some problem of succesfully send    Serial.println(rs.getResponseDescription());  }    **void** **loop**() {      // If something available  **if** (e22ttl.available()>1) {        // read the String message      ResponseContainer rc **=** e22ttl.receiveMessage();      // Is something goes wrong print error  **if** (rc.status.code**!=**1){          Serial.println(rc.status.getResponseDescription());      }**else**{          // Print the data received          Serial.println(rc.status.getResponseDescription());          Serial.println(rc.data);      }    }  **if** (Serial.available()) {        String input **=** Serial.readString();        e22ttl.sendMessage(input);    }  } |

If you have already changed the configuration, you must restore the default parameters:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | //  If you have ever change configuration you must restore It      ResponseStructContainer c;      c **=** e22ttl.getConfiguration();      Configuration configuration **=** **\***(Configuration**\***) c.data;      Serial.println(c.status.getResponseDescription());      configuration.ADDL **=** 0x03;      configuration.ADDH **=** 0x00;      configuration.NETID **=** 0x00;        configuration.CHAN **=** 23;        configuration.SPED.uartBaudRate **=** UART\_BPS\_9600;      configuration.SPED.airDataRate **=** AIR\_DATA\_RATE\_010\_24;      configuration.SPED.uartParity **=** MODE\_00\_8N1;        configuration.OPTION.subPacketSetting **=** SPS\_240\_00;      configuration.OPTION.RSSIAmbientNoise **=** RSSI\_AMBIENT\_NOISE\_DISABLED;      configuration.OPTION.transmissionPower **=** POWER\_22;        configuration.TRANSMISSION\_MODE.enableRSSI **=** RSSI\_DISABLED;      configuration.TRANSMISSION\_MODE.fixedTransmission **=** FT\_TRANSPARENT\_TRANSMISSION;      configuration.TRANSMISSION\_MODE.enableRepeater **=** REPEATER\_DISABLED;      configuration.TRANSMISSION\_MODE.enableLBT **=** LBT\_DISABLED;      configuration.TRANSMISSION\_MODE.WORTransceiverControl **=** WOR\_RECEIVER;      configuration.TRANSMISSION\_MODE.WORPeriod **=** WOR\_2000\_011;      e22ttl.setConfiguration(configuration, WRITE\_CFG\_PWR\_DWN\_SAVE);      c.close(); |

Transparent transmission and RSSI

[](https://mischianti.org/forums/topic/constructor-difference-and-hardwareserial-and-softwareserial-difference/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-3-devices-module-smd/)Ebyte LoRa E22 device for Arduino, esp32, or esp8266 three devices module SMD

In [telecommunications](https://en.wikipedia.org/wiki/Telecommunications), a **received signal strength indicator (RSSI) measures** the [power](https://en.wikipedia.org/wiki/Electric_power) present in a received [radio](https://en.wikipedia.org/wiki/Radio) signal.

RSSI is usually invisible to a user of a receiving device. However, because signal strength can vary greatly and affect functionality in [wireless networking](https://en.wikipedia.org/wiki/Wireless_LAN), [IEEE 802.11](https://en.wikipedia.org/wiki/IEEE_802.11) devices often make the measurement available to users.

This device supports RSSI, and in some situations, It’s very useful. To use It, you must activate the configuration flag.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | //  If you have ever change configuration you must restore It      ResponseStructContainer c;      c **=** e22ttl.getConfiguration();      Configuration configuration **=** **\***(Configuration**\***) c.data;      Serial.println(c.status.getResponseDescription());      configuration.ADDL **=** 0x03;      configuration.ADDH **=** 0x00;      configuration.NETID **=** 0x00;        configuration.CHAN **=** 23;        configuration.SPED.uartBaudRate **=** UART\_BPS\_9600;      configuration.SPED.airDataRate **=** AIR\_DATA\_RATE\_010\_24;      configuration.SPED.uartParity **=** MODE\_00\_8N1;        configuration.OPTION.subPacketSetting **=** SPS\_240\_00;      configuration.OPTION.RSSIAmbientNoise **=** RSSI\_AMBIENT\_NOISE\_DISABLED;      configuration.OPTION.transmissionPower **=** POWER\_22;        configuration.TRANSMISSION\_MODE.enableRSSI **=** RSSI\_ENABLED;      configuration.TRANSMISSION\_MODE.fixedTransmission **=** FT\_TRANSPARENT\_TRANSMISSION;      configuration.TRANSMISSION\_MODE.enableRepeater **=** REPEATER\_DISABLED;      configuration.TRANSMISSION\_MODE.enableLBT **=** LBT\_DISABLED;      configuration.TRANSMISSION\_MODE.WORTransceiverControl **=** WOR\_RECEIVER;      configuration.TRANSMISSION\_MODE.WORPeriod **=** WOR\_2000\_011;      e22ttl.setConfiguration(configuration, WRITE\_CFG\_PWR\_DWN\_SAVE);      c.close(); |

After configuration, we can go to use the method created to manage the RSSI value: receiveMessageRSSI().

For the transmitter, all the process was managed by configuration; only the receiver must use a “special” method to retrieve the value. So the previous sketch becomes like so:

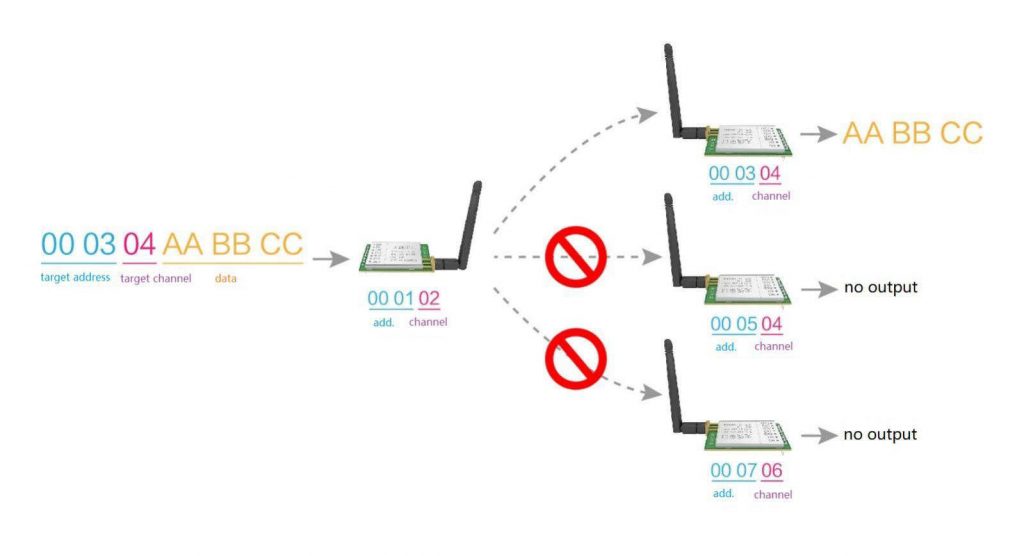
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85 | /\*   \* LoRa E22   \* send a transparent message, you must check that the transmitter and receiver have the same   \* CHANNEL ADDL and ADDH   \*   \* Pai attention e22 support RSSI, if you want use that functionality you must enable RSSI on configuration   \* configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_ENABLED;   \*   \* Renzo Mischianti <[https://mischianti.org](https://mischianti.org/)>   \* <https://mischianti.org/category/my-libraries/ebyte-lora-e22-devices/>   \*   \*/    #include "Arduino.h"  #include "LoRa\_E22.h"    // ---------- esp8266 pins --------------  //LoRa\_E22 e22ttl(RX, TX, AUX, M0, M1);  // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  // LoRa\_E22 e22ttl(D3, D4, D5, D7, D6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(D2, D3); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(D2, D3); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, D5, D7, D6); // AUX M0 M1  // -------------------------------------    // ---------- Arduino pins --------------  LoRa\_E22 e22ttl(4, 5, 3, 7, 6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(4, 5); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(4, 5); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, 3, 7, 6); // AUX M0 M1  // -------------------------------------    // ------------- Arduino Nano 33 IoT -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ------------- Arduino MKR WiFi 1010 -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ---------- esp32 pins --------------  //LoRa\_E22 e22ttl(&Serial2, 18, 21, 19); //  RX AUX M0 M1    //LoRa\_E22 e22ttl(&Serial2, 22, 4, 18, 21, 19, UART\_BPS\_RATE\_9600); //  esp32 RX <-- e22 TX, esp32 TX --> e22 RX AUX M0 M1  // -------------------------------------    **void** **setup**() {    Serial.begin(9600);    delay(500);      // Startup all pins and UART    e22ttl.begin();      Serial.println("Hi, I'm going to send message!");      // Send message    ResponseStatus rs **=** e22ttl.sendMessage("Hello, world?");    // Check If there is some problem of succesfully send    Serial.println(rs.getResponseDescription());  }    **void** **loop**() {      // If something available  **if** (e22ttl.available()>1) {        // read the String message      ResponseContainer rc **=** e22ttl.receiveMessageRSSI();      // Is something goes wrong print error  **if** (rc.status.code**!=**1){          Serial.println(rc.status.getResponseDescription());      }**else**{          // Print the data received          Serial.println(rc.status.getResponseDescription());          Serial.println(rc.data);            Serial.print("RSSI: "); Serial.println(rc.rssi, DEC);      }    }  **if** (Serial.available()) {        String input **=** Serial.readString();        e22ttl.sendMessage(input);    }  } |

Fixed transmission

For fixed transmission, you must set M0 and M1 to LOW, and It’s better to connect the AUX pin to have a better synchronization.

Fixed transmission: point to point

To use this type of transmission, we must set a parameter on configuration and set a specified address for every device.

[](https://mischianti.org/wp-content/uploads/2019/10/fixedMessageToASpecifiedDevice.jpg)LoRa E32 Fixed message to a specified device

So first, we must set M0 to LOW and M1 pin to HIGH to enter on program/sleep mode and set the correct address and fixed transmission flag.

If we want to replicate the condition of the sender in the upper image, we must do this configuration.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | ResponseStructContainer c;  c **=** e22ttl.getConfiguration();  Configuration configuration **=** **\***(Configuration**\***) c.data;  configuration.ADDL **=** 0x02;  configuration.ADDH **=** 0x00;  configuration.NETID **=** 0x00;    configuration.CHAN **=** 23;    configuration.SPED.uartBaudRate **=** UART\_BPS\_9600;  configuration.SPED.airDataRate **=** AIR\_DATA\_RATE\_010\_24;  configuration.SPED.uartParity **=** MODE\_00\_8N1;    configuration.OPTION.subPacketSetting **=** SPS\_240\_00;  configuration.OPTION.RSSIAmbientNoise **=** RSSI\_AMBIENT\_NOISE\_DISABLED;  configuration.OPTION.transmissionPower **=** POWER\_22;    configuration.TRANSMISSION\_MODE.enableRSSI **=** RSSI\_DISABLED;  configuration.TRANSMISSION\_MODE.fixedTransmission **=** FT\_FIXED\_TRANSMISSION;  configuration.TRANSMISSION\_MODE.enableRepeater **=** REPEATER\_DISABLED;  configuration.TRANSMISSION\_MODE.enableLBT **=** LBT\_DISABLED;  configuration.TRANSMISSION\_MODE.WORTransceiverControl **=** WOR\_TRANSMITTER;  configuration.TRANSMISSION\_MODE.WORPeriod **=** WOR\_2000\_011;  e22ttl.setConfiguration(configuration, WRITE\_CFG\_PWR\_DWN\_SAVE);  c.close(); |

Then for the receiver device, we must set this configuration.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | ResponseStructContainer c;  c **=** e22ttl.getConfiguration();  Configuration configuration **=** **\***(Configuration**\***) c.data;  configuration.ADDL **=** 0x03;  configuration.ADDH **=** 0x00;  configuration.NETID **=** 0x00;    configuration.CHAN **=** 23;    configuration.SPED.uartBaudRate **=** UART\_BPS\_9600;  configuration.SPED.airDataRate **=** AIR\_DATA\_RATE\_010\_24;  configuration.SPED.uartParity **=** MODE\_00\_8N1;    configuration.OPTION.subPacketSetting **=** SPS\_240\_00;  configuration.OPTION.RSSIAmbientNoise **=** RSSI\_AMBIENT\_NOISE\_DISABLED;  configuration.OPTION.transmissionPower **=** POWER\_22;    configuration.TRANSMISSION\_MODE.enableRSSI **=** RSSI\_DISABLED;  configuration.TRANSMISSION\_MODE.fixedTransmission **=** FT\_FIXED\_TRANSMISSION;  configuration.TRANSMISSION\_MODE.enableRepeater **=** REPEATER\_DISABLED;  configuration.TRANSMISSION\_MODE.enableLBT **=** LBT\_DISABLED;  configuration.TRANSMISSION\_MODE.WORTransceiverControl **=** WOR\_RECEIVER;  configuration.TRANSMISSION\_MODE.WORPeriod **=** WOR\_2000\_011;  e22ttl.setConfiguration(configuration, WRITE\_CFG\_PWR\_DWN\_SAVE);  c.close(); |

Now we can send a message to the specified device.

|  |  |
| --- | --- |
| 1  2  3  4  5 | #define DESTINATION\_ADDL 2  // Send message  ResponseStatus rs **=** e22ttl.sendFixedMessage(0, DESTINATION\_ADDL, 23, "Hello, world?");  // Check If there is some problem of succesfully send  Serial.println(rs.getResponseDescription()); |

The receiver is like the transparent one because the device manages the address and channel.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | **void** **loop**()  {      // If something available  **if** (e22ttl.available()>1) {        // read the String message      ResponseContainer rc **=** e22ttl.receiveMessage();      // Is something goes wrong print error  **if** (rc.status.code**!=**1){          Serial.println(rc.status.getResponseDescription());      }**else**{          // Print the data received          Serial.println(rc.status.getResponseDescription());          Serial.println(rc.data);      }    }  } |

Pay attention to these lines of code:

|  |  |
| --- | --- |
| 15  16  17  18  19 | // With FIXED SENDER configuration  // #define DESTINATION\_ADDL 3    // With FIXED RECEIVER configuration  #define DESTINATION\_ADDL 2 |

You must uncomment the correct DESTINATION\_ADDL for the sender and the receiver.

Arduino MKR WiFi 1010 on the breadboard with Ebyte LoRa E22

Sender and receive sketch:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134 | /\*   \* EBYTE LoRa E22   \* Send a string message to a fixed point ADDH ADDL CHAN 0 2 23   \*   \* Write a string on serial monitor or reset to resend default value.   \*   \* Send a fixed point message, you must check that the transmitter and receiver have different   \* CHANNEL ADDL or ADDH, check down the correct configuration   \*   \* Renzo Mischianti <[https://mischianti.org](https://mischianti.org/)>   \* <https://mischianti.org/category/my-libraries/ebyte-lora-e22-devices/>   \*   \*/    // With FIXED SENDER configuration  // #define DESTINATION\_ADDL 3    // With FIXED RECEIVER configuration  #define DESTINATION\_ADDL 2    #include "Arduino.h"  #include "LoRa\_E22.h"    // ---------- esp8266 pins --------------  //LoRa\_E22 e22ttl(RX, TX, AUX, M0, M1);  // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  // LoRa\_E22 e22ttl(D3, D4, D5, D7, D6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(D2, D3); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(D2, D3); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, D5, D7, D6); // AUX M0 M1  // -------------------------------------    // ---------- Arduino pins --------------  LoRa\_E22 e22ttl(4, 5, 3, 7, 6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(4, 5); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(4, 5); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, 3, 7, 6); // AUX M0 M1  // -------------------------------------    // ------------- Arduino Nano 33 IoT -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ------------- Arduino MKR WiFi 1010 -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ---------- esp32 pins --------------  //LoRa\_E22 e22ttl(&Serial2, 18, 21, 19); //  RX AUX M0 M1    //LoRa\_E22 e22ttl(&Serial2, 22, 4, 18, 21, 19, UART\_BPS\_RATE\_9600); //  esp32 RX <-- e22 TX, esp32 TX --> e22 RX AUX M0 M1  // -------------------------------------    **void** printParameters(struct Configuration configuration);    **void** **setup**() {    Serial.begin(9600);    delay(500);      // Startup all pins and UART    e22ttl.begin();        ResponseStructContainer c;      c **=** e22ttl.getConfiguration();      // It's important get configuration pointer before all other operation      Configuration configuration **=** **\***(Configuration**\***) c.data;      Serial.println(c.status.getResponseDescription());      Serial.println(c.status.code);        printParameters(configuration);      c.close();        Serial.println("Hi, I'm going to send message!");      // Send message    ResponseStatus rs **=** e22ttl.sendFixedMessage(0, DESTINATION\_ADDL, 23, "Hello, world?");    // Check If there is some problem of succesfully send    Serial.println(rs.getResponseDescription());  }    **void** **loop**() {      // If something available  **if** (e22ttl.available()>1) {        // read the String message      ResponseContainer rc **=** e22ttl.receiveMessage();        // Is something goes wrong print error  **if** (rc.status.code**!=**1){          Serial.println(rc.status.getResponseDescription());      }**else**{          // Print the data received          Serial.println(rc.status.getResponseDescription());          Serial.println(rc.data);      }    }  **if** (Serial.available()) {        String input **=** Serial.readString();        e22ttl.sendFixedMessage(0, DESTINATION\_ADDL, 23, input);    }  }    **void** printParameters(struct Configuration configuration) {      Serial.println("----------------------------------------");        Serial.print(F("HEAD : "));  Serial.print(configuration.COMMAND, HEX);Serial.print(" ");Serial.print(configuration.STARTING\_ADDRESS, HEX);Serial.print(" ");Serial.println(configuration.LENGHT, HEX);      Serial.println(F(" "));      Serial.print(F("AddH : "));  Serial.println(configuration.ADDH, HEX);      Serial.print(F("AddL : "));  Serial.println(configuration.ADDL, HEX);      Serial.print(F("NetID : "));  Serial.println(configuration.NETID, HEX);      Serial.println(F(" "));      Serial.print(F("Chan : "));  Serial.print(configuration.CHAN, DEC); Serial.print(" -> "); Serial.println(configuration.getChannelDescription());      Serial.println(F(" "));      Serial.print(F("SpeedParityBit     : "));  Serial.print(configuration.SPED.uartParity, BIN);Serial.print(" -> "); Serial.println(configuration.SPED.getUARTParityDescription());      Serial.print(F("SpeedUARTDatte     : "));  Serial.print(configuration.SPED.uartBaudRate, BIN);Serial.print(" -> "); Serial.println(configuration.SPED.getUARTBaudRateDescription());      Serial.print(F("SpeedAirDataRate   : "));  Serial.print(configuration.SPED.airDataRate, BIN);Serial.print(" -> "); Serial.println(configuration.SPED.getAirDataRateDescription());      Serial.println(F(" "));      Serial.print(F("OptionSubPacketSett: "));  Serial.print(configuration.OPTION.subPacketSetting, BIN);Serial.print(" -> "); Serial.println(configuration.OPTION.getSubPacketSetting());      Serial.print(F("OptionTranPower    : "));  Serial.print(configuration.OPTION.transmissionPower, BIN);Serial.print(" -> "); Serial.println(configuration.OPTION.getTransmissionPowerDescription());      Serial.print(F("OptionRSSIAmbientNo: "));  Serial.print(configuration.OPTION.RSSIAmbientNoise, BIN);Serial.print(" -> "); Serial.println(configuration.OPTION.getRSSIAmbientNoiseEnable());      Serial.println(F(" "));      Serial.print(F("TransModeWORPeriod : "));  Serial.print(configuration.TRANSMISSION\_MODE.WORPeriod, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getWORPeriodByParamsDescription());      Serial.print(F("TransModeTransContr: "));  Serial.print(configuration.TRANSMISSION\_MODE.WORTransceiverControl, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getWORTransceiverControlDescription());      Serial.print(F("TransModeEnableLBT : "));  Serial.print(configuration.TRANSMISSION\_MODE.enableLBT, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getLBTEnableByteDescription());      Serial.print(F("TransModeEnableRSSI: "));  Serial.print(configuration.TRANSMISSION\_MODE.enableRSSI, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getRSSIEnableByteDescription());      Serial.print(F("TransModeEnabRepeat: "));  Serial.print(configuration.TRANSMISSION\_MODE.enableRepeater, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getRepeaterModeEnableByteDescription());      Serial.print(F("TransModeFixedTrans: "));  Serial.print(configuration.TRANSMISSION\_MODE.fixedTransmission, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getFixedTransmissionDescription());          Serial.println("----------------------------------------");  } |

Fixed transmission with RSSI:

[](https://mischianti.org/ebyte-lora-e22-device-for-arduino-esp32-or-esp8266-rssi-signal-strength/)Ebyte LoRa E22 device for Arduino, esp32 or esp8266 RSSI signal strength

To manage RSSI, It’s must be enabled via configuration, so the configuration for the sender becomes like this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | ResponseStructContainer c;  c **=** e22ttl.getConfiguration();  Configuration configuration **=** **\***(Configuration**\***) c.data;    configuration.ADDL **=** 0x02;  configuration.ADDH **=** 0x00;  configuration.NETID **=** 0x00;    configuration.CHAN **=** 23;    configuration.SPED.uartBaudRate **=** UART\_BPS\_9600;  configuration.SPED.airDataRate **=** AIR\_DATA\_RATE\_010\_24;  configuration.SPED.uartParity **=** MODE\_00\_8N1;    configuration.OPTION.subPacketSetting **=** SPS\_240\_00;  configuration.OPTION.RSSIAmbientNoise **=** RSSI\_AMBIENT\_NOISE\_DISABLED;  configuration.OPTION.transmissionPower **=** POWER\_22;    configuration.TRANSMISSION\_MODE.enableRSSI **=** RSSI\_ENABLED;  configuration.TRANSMISSION\_MODE.fixedTransmission **=** FT\_FIXED\_TRANSMISSION;  configuration.TRANSMISSION\_MODE.enableRepeater **=** REPEATER\_DISABLED;  configuration.TRANSMISSION\_MODE.enableLBT **=** LBT\_DISABLED;  configuration.TRANSMISSION\_MODE.WORTransceiverControl **=** WOR\_TRANSMITTER;  configuration.TRANSMISSION\_MODE.WORPeriod **=** WOR\_2000\_011;    e22ttl.setConfiguration(configuration, WRITE\_CFG\_PWR\_DWN\_SAVE);  c.close(); |

and for the receiver:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | ResponseStructContainer c;  c **=** e22ttl.getConfiguration();  Configuration configuration **=** **\***(Configuration**\***) c.data;    configuration.ADDL **=** 0x03;  configuration.ADDH **=** 0x00;  configuration.NETID **=** 0x00;    configuration.CHAN **=** 23;    configuration.SPED.uartBaudRate **=** UART\_BPS\_9600;  configuration.SPED.airDataRate **=** AIR\_DATA\_RATE\_010\_24;  configuration.SPED.uartParity **=** MODE\_00\_8N1;    configuration.OPTION.subPacketSetting **=** SPS\_240\_00;  configuration.OPTION.RSSIAmbientNoise **=** RSSI\_AMBIENT\_NOISE\_DISABLED;  configuration.OPTION.transmissionPower **=** POWER\_22;    configuration.TRANSMISSION\_MODE.enableRSSI **=** RSSI\_ENABLED;  configuration.TRANSMISSION\_MODE.fixedTransmission **=** FT\_FIXED\_TRANSMISSION;  configuration.TRANSMISSION\_MODE.enableRepeater **=** REPEATER\_DISABLED;  configuration.TRANSMISSION\_MODE.enableLBT **=** LBT\_DISABLED;  configuration.TRANSMISSION\_MODE.WORTransceiverControl **=** WOR\_RECEIVER;  configuration.TRANSMISSION\_MODE.WORPeriod **=** WOR\_2000\_011;    e22ttl.setConfiguration(configuration, WRITE\_CFG\_PWR\_DWN\_SAVE);  c.close(); |

Also, in this sketch, pay attention to these lines of code:

|  |  |
| --- | --- |
| 15  16  17  18  19 | // With FIXED SENDER configuration  // #define DESTINATION\_ADDL 3    // With FIXED RECEIVER configuration  #define DESTINATION\_ADDL 2 |

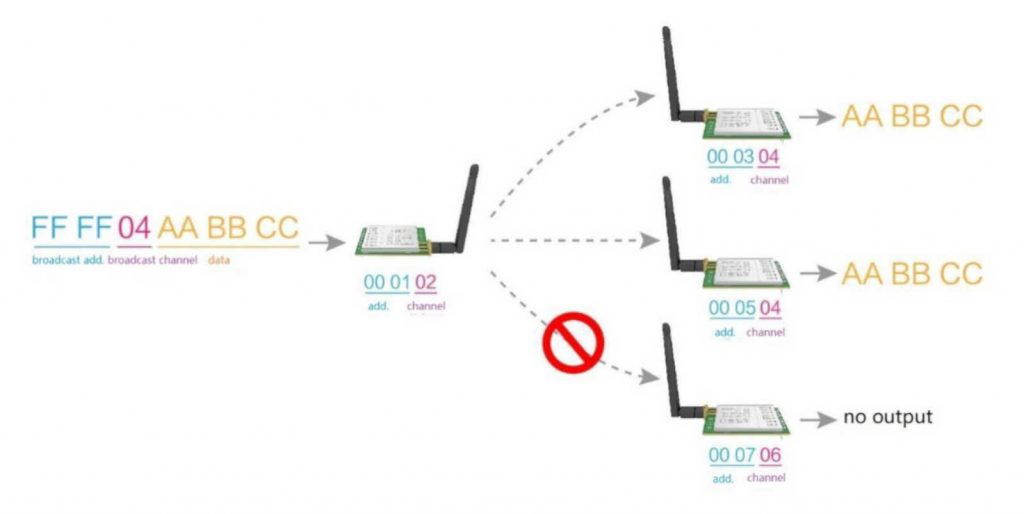
You must uncomment the correct DESTINATION\_ADDL for the sender and the receiver.

And here is the sketch with RSSI:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137 | /\*   \* EBYTE LoRa E22   \* Send a string message to a fixed point ADDH ADDL CHAN 0 2 23   \*   \* Write a string on serial monitor or reset to resend default value.   \*   \* Send a fixed point message, you must check that the transmitter and receiver have different   \* CHANNEL ADDL or ADDH, check down the correct configuration   \*   \* Pai attention e22 support RSSI, if you want use that functionality you must enable RSSI on configuration   \* configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_ENABLED;   \*   \* Renzo Mischianti <[https://mischianti.org](https://mischianti.org/)>   \* <https://mischianti.org/category/my-libraries/ebyte-lora-e22-devices/>   \*   \*/    // With FIXED SENDER configuration  // #define DESTINATION\_ADDL 3    // With FIXED RECEIVER configuration  #define DESTINATION\_ADDL 2    #include "Arduino.h"  #include "LoRa\_E22.h"    // ---------- esp8266 pins --------------  //LoRa\_E22 e22ttl(RX, TX, AUX, M0, M1);  // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  // LoRa\_E22 e22ttl(D3, D4, D5, D7, D6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(D2, D3); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(D2, D3); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, D5, D7, D6); // AUX M0 M1  // -------------------------------------    // ---------- Arduino pins --------------  LoRa\_E22 e22ttl(4, 5, 3, 7, 6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(4, 5); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(4, 5); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, 3, 7, 6); // AUX M0 M1  // -------------------------------------    // ------------- Arduino Nano 33 IoT -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ------------- Arduino MKR WiFi 1010 -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ---------- esp32 pins --------------  //LoRa\_E22 e22ttl(&Serial2, 18, 21, 19); //  RX AUX M0 M1    //LoRa\_E22 e22ttl(&Serial2, 22, 4, 18, 21, 19, UART\_BPS\_RATE\_9600); //  esp32 RX <-- e22 TX, esp32 TX --> e22 RX AUX M0 M1  // -------------------------------------    **void** printParameters(struct Configuration configuration);    **void** **setup**() {    Serial.begin(9600);    delay(500);      // Startup all pins and UART    e22ttl.begin();        ResponseStructContainer c;      c **=** e22ttl.getConfiguration();      // It's important get configuration pointer before all other operation      Configuration configuration **=** **\***(Configuration**\***) c.data;      Serial.println(c.status.getResponseDescription());      Serial.println(c.status.code);        printParameters(configuration);      c.close();        Serial.println("Hi, I'm going to send message!");      // Send message    ResponseStatus rs **=** e22ttl.sendFixedMessage(0, DESTINATION\_ADDL, 23, "Hello, world?");    // Check If there is some problem of succesfully send    Serial.println(rs.getResponseDescription());  }    **void** **loop**() {      // If something available  **if** (e22ttl.available()>1) {        // read the String message      ResponseContainer rc **=** e22ttl.receiveMessageRSSI();      // Is something goes wrong print error  **if** (rc.status.code**!=**1){          Serial.println(rc.status.getResponseDescription());      }**else**{          // Print the data received          Serial.println(rc.status.getResponseDescription());          Serial.println(rc.data);          Serial.print("RSSI: "); Serial.println(rc.rssi, DEC);      }    }  **if** (Serial.available()) {        String input **=** Serial.readString();        e22ttl.sendFixedMessage(0, DESTINATION\_ADDL, 23, input);    }  }    **void** printParameters(struct Configuration configuration) {      Serial.println("----------------------------------------");        Serial.print(F("HEAD : "));  Serial.print(configuration.COMMAND, HEX);Serial.print(" ");Serial.print(configuration.STARTING\_ADDRESS, HEX);Serial.print(" ");Serial.println(configuration.LENGHT, HEX);      Serial.println(F(" "));      Serial.print(F("AddH : "));  Serial.println(configuration.ADDH, HEX);      Serial.print(F("AddL : "));  Serial.println(configuration.ADDL, HEX);      Serial.print(F("NetID : "));  Serial.println(configuration.NETID, HEX);      Serial.println(F(" "));      Serial.print(F("Chan : "));  Serial.print(configuration.CHAN, DEC); Serial.print(" -> "); Serial.println(configuration.getChannelDescription());      Serial.println(F(" "));      Serial.print(F("SpeedParityBit     : "));  Serial.print(configuration.SPED.uartParity, BIN);Serial.print(" -> "); Serial.println(configuration.SPED.getUARTParityDescription());      Serial.print(F("SpeedUARTDatte     : "));  Serial.print(configuration.SPED.uartBaudRate, BIN);Serial.print(" -> "); Serial.println(configuration.SPED.getUARTBaudRateDescription());      Serial.print(F("SpeedAirDataRate   : "));  Serial.print(configuration.SPED.airDataRate, BIN);Serial.print(" -> "); Serial.println(configuration.SPED.getAirDataRateDescription());      Serial.println(F(" "));      Serial.print(F("OptionSubPacketSett: "));  Serial.print(configuration.OPTION.subPacketSetting, BIN);Serial.print(" -> "); Serial.println(configuration.OPTION.getSubPacketSetting());      Serial.print(F("OptionTranPower    : "));  Serial.print(configuration.OPTION.transmissionPower, BIN);Serial.print(" -> "); Serial.println(configuration.OPTION.getTransmissionPowerDescription());      Serial.print(F("OptionRSSIAmbientNo: "));  Serial.print(configuration.OPTION.RSSIAmbientNoise, BIN);Serial.print(" -> "); Serial.println(configuration.OPTION.getRSSIAmbientNoiseEnable());      Serial.println(F(" "));      Serial.print(F("TransModeWORPeriod : "));  Serial.print(configuration.TRANSMISSION\_MODE.WORPeriod, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getWORPeriodByParamsDescription());      Serial.print(F("TransModeTransContr: "));  Serial.print(configuration.TRANSMISSION\_MODE.WORTransceiverControl, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getWORTransceiverControlDescription());      Serial.print(F("TransModeEnableLBT : "));  Serial.print(configuration.TRANSMISSION\_MODE.enableLBT, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getLBTEnableByteDescription());      Serial.print(F("TransModeEnableRSSI: "));  Serial.print(configuration.TRANSMISSION\_MODE.enableRSSI, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getRSSIEnableByteDescription());      Serial.print(F("TransModeEnabRepeat: "));  Serial.print(configuration.TRANSMISSION\_MODE.enableRepeater, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getRepeaterModeEnableByteDescription());      Serial.print(F("TransModeFixedTrans: "));  Serial.print(configuration.TRANSMISSION\_MODE.fixedTransmission, BIN);Serial.print(" -> "); Serial.println(configuration.TRANSMISSION\_MODE.getFixedTransmissionDescription());          Serial.println("----------------------------------------");  } |

Fixed transmission: broadcast

We can test the broadcast communication with the exact configuration of the address and channel.

[](https://mischianti.org/wp-content/uploads/2019/10/broadcastMessageToASpecifiedChannelDevice.jpg)Broadcast message to a set of channel devices

To do a test, you can create three devices with these three configurations:

Device 1:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | configuration.ADDL **=** 0x04;  configuration.ADDH **=** 0x00;  configuration.NETID **=** 0x00;    configuration.CHAN **=** 23;    configuration.SPED.uartBaudRate **=** UART\_BPS\_9600;  configuration.SPED.airDataRate **=** AIR\_DATA\_RATE\_010\_24;  configuration.SPED.uartParity **=** MODE\_00\_8N1;    configuration.OPTION.subPacketSetting **=** SPS\_240\_00;  configuration.OPTION.RSSIAmbientNoise **=** RSSI\_AMBIENT\_NOISE\_DISABLED;  configuration.OPTION.transmissionPower **=** POWER\_22;    configuration.TRANSMISSION\_MODE.enableRSSI **=** RSSI\_DISABLED;  configuration.TRANSMISSION\_MODE.fixedTransmission **=** FT\_FIXED\_TRANSMISSION;  configuration.TRANSMISSION\_MODE.enableRepeater **=** REPEATER\_DISABLED;  configuration.TRANSMISSION\_MODE.enableLBT **=** LBT\_DISABLED;  configuration.TRANSMISSION\_MODE.WORTransceiverControl **=** WOR\_RECEIVER;  configuration.TRANSMISSION\_MODE.WORPeriod **=** WOR\_2000\_011; |

Device 2:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | configuration.ADDL **=** 0x05;  configuration.ADDH **=** 0x00;  configuration.NETID **=** 0x00;    configuration.CHAN **=** 23;    configuration.SPED.uartBaudRate **=** UART\_BPS\_9600;  configuration.SPED.airDataRate **=** AIR\_DATA\_RATE\_010\_24;  configuration.SPED.uartParity **=** MODE\_00\_8N1;    configuration.OPTION.subPacketSetting **=** SPS\_240\_00;  configuration.OPTION.RSSIAmbientNoise **=** RSSI\_AMBIENT\_NOISE\_DISABLED;  configuration.OPTION.transmissionPower **=** POWER\_22;    configuration.TRANSMISSION\_MODE.enableRSSI **=** RSSI\_DISABLED;  configuration.TRANSMISSION\_MODE.fixedTransmission **=** FT\_FIXED\_TRANSMISSION;  configuration.TRANSMISSION\_MODE.enableRepeater **=** REPEATER\_DISABLED;  configuration.TRANSMISSION\_MODE.enableLBT **=** LBT\_DISABLED;  configuration.TRANSMISSION\_MODE.WORTransceiverControl **=** WOR\_RECEIVER;  configuration.TRANSMISSION\_MODE.WORPeriod **=** WOR\_2000\_011; |

Device 3:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | configuration.ADDL = 0x06;  configuration.ADDH = 0x00;  configuration.NETID = 0x00;    configuration.CHAN = 23;    configuration.SPED.uartBaudRate = UART\_BPS\_9600;  configuration.SPED.airDataRate = AIR\_DATA\_RATE\_010\_24;  configuration.SPED.uartParity = MODE\_00\_8N1;    configuration.OPTION.subPacketSetting = SPS\_240\_00;  configuration.OPTION.RSSIAmbientNoise = RSSI\_AMBIENT\_NOISE\_DISABLED;  configuration.OPTION.transmissionPower = POWER\_22;    configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_DISABLED;  configuration.TRANSMISSION\_MODE.fixedTransmission = FT\_FIXED\_TRANSMISSION;  configuration.TRANSMISSION\_MODE.enableRepeater = REPEATER\_DISABLED;  configuration.TRANSMISSION\_MODE.enableLBT = LBT\_DISABLED;  configuration.TRANSMISSION\_MODE.WORTransceiverControl = WOR\_RECEIVER;  configuration.TRANSMISSION\_MODE.WORPeriod = WOR\_2000\_011; |
| 1  2  3  4 | // Send message  ResponseStatus rs **=** e22ttl.sendBroadcastFixedMessage(23, "Hello, world?");  // Check If there is some problem of succesfully send  Serial.println(rs.getResponseDescription()); |

Or

|  |  |
| --- | --- |
| 1  2  3 | // Send message  ResponseStatus rs **=** e22ttl.sendFixedMessage(BROADCAST\_ADDRESS, BROADCAST\_ADDRESS, 4, "Send message to channel 4");    Serial.println(rs.getResponseDescription()); |

The receiver, as described, has the same code because the device manages the preamble with Address and Channel.

Here is the sender sketch:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118 | /\*   \* EBYTE LoRa E22   \*   \* Send a string message to the all devices of channel 23   \*   \* Write a string on serial monitor or reset to resend default value.   \*   \* Send a fixed message, you must check that the transmitter and receiver have different   \* CHANNEL ADDL or ADDH, check the configuration down   \*   \* Pai attention e22 support RSSI, if you want use that functionality you must enable RSSI on configuration   \* configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_ENABLED;   \*   \* and uncomment #define ENABLE\_RSSI true in this sketch   \*   \* You must select 3 devices, and you must use the configuration   \* BROADCAST MESSAGE 1   \* BROADCAST MESSAGE 2   \* BROADCAST MESSAGE 3   \*   \* Renzo Mischianti <[https://mischianti.org](https://mischianti.org/)>   \* <https://mischianti.org/category/my-libraries/ebyte-lora-e22-devices/>   \*   \* E22        ----- Arduino   \* M0         ----- 7 (or GND)   \* M1         ----- 6 (or GND)   \* RX         ----- 4 (PullUP)   \* TX         ----- 5 (PullUP)   \* AUX        ----- 3  (PullUP)   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/    // If you want use RSSI uncomment //#define ENABLE\_RSSI true  // and use relative configuration with RSSI enabled  //#define ENABLE\_RSSI true    #include "Arduino.h"  #include "LoRa\_E22.h"    // ---------- esp8266 pins --------------  //LoRa\_E22 e22ttl(RX, TX, AUX, M0, M1);  // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  // LoRa\_E22 e22ttl(D3, D4, D5, D7, D6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(D2, D3); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(D2, D3); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, D5, D7, D6); // AUX M0 M1  // -------------------------------------    // ---------- Arduino pins --------------  LoRa\_E22 e22ttl(4, 5, 3, 7, 6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(4, 5); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(4, 5); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, 3, 7, 6); // AUX M0 M1  // -------------------------------------    // ------------- Arduino Nano 33 IoT -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ------------- Arduino MKR WiFi 1010 -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ---------- esp32 pins --------------  //LoRa\_E22 e22ttl(&Serial2, 18, 21, 19); //  RX AUX M0 M1    //LoRa\_E22 e22ttl(&Serial2, 22, 4, 18, 21, 19, UART\_BPS\_RATE\_9600); //  esp32 RX <-- e22 TX, esp32 TX --> e22 RX AUX M0 M1  // -------------------------------------    **void** **setup**() {    Serial.begin(9600);    delay(500);      // Startup all pins and UART    e22ttl.begin();      Serial.println("Hi, I'm going to send message!");      // Send message    ResponseStatus rs **=** e22ttl.sendBroadcastFixedMessage(23, "Hello, world?");    // Check If there is some problem of succesfully send    Serial.println(rs.getResponseDescription());  }    **void** **loop**() {      // If something available  **if** (e22ttl.available()>1) {        // read the String message  #ifdef ENABLE\_RSSI      ResponseContainer rc **=** e22ttl.receiveMessageRSSI();  #else      ResponseContainer rc **=** e22ttl.receiveMessage();  #endif      // Is something goes wrong print error  **if** (rc.status.code**!=**1){          Serial.println(rc.status.getResponseDescription());      }**else**{          // Print the data received          Serial.println(rc.status.getResponseDescription());          Serial.println(rc.data);  #ifdef ENABLE\_RSSI          Serial.print("RSSI: "); Serial.println(rc.rssi, DEC);  #endif      }    }  **if** (Serial.available()) {        String input **=** Serial.readString();        ResponseStatus rs **=** e22ttl.sendBroadcastFixedMessage(23, input);        // Check If there is some problem of succesfully send        Serial.println(rs.getResponseDescription());      }  } |

Fixed transmission: monitoring

As you have already seen on the first schema, you can hear all messages in the specified channel, and you must configure your address like so, with an address equal to 0xFFFF.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | configuration.ADDL **=** BROADCAST\_ADDRESS;  configuration.ADDH **=** BROADCAST\_ADDRESS;  configuration.NETID **=** 0x00;    configuration.CHAN **=** 23;    configuration.SPED.uartBaudRate **=** UART\_BPS\_9600;  configuration.SPED.airDataRate **=** AIR\_DATA\_RATE\_010\_24;  configuration.SPED.uartParity **=** MODE\_00\_8N1;    configuration.OPTION.subPacketSetting **=** SPS\_240\_00;  configuration.OPTION.RSSIAmbientNoise **=** RSSI\_AMBIENT\_NOISE\_DISABLED;  configuration.OPTION.transmissionPower **=** POWER\_22;    configuration.TRANSMISSION\_MODE.enableRSSI **=** RSSI\_DISABLED;  configuration.TRANSMISSION\_MODE.fixedTransmission **=** FT\_FIXED\_TRANSMISSION;  configuration.TRANSMISSION\_MODE.enableRepeater **=** REPEATER\_DISABLED;  configuration.TRANSMISSION\_MODE.enableLBT **=** LBT\_DISABLED;  configuration.TRANSMISSION\_MODE.WORTransceiverControl **=** WOR\_RECEIVER;  configuration.TRANSMISSION\_MODE.WORPeriod **=** WOR\_2000\_011; |

To receive the messages:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110 | /\*   \* EBYTE LoRa E22   \*   \* Write a string on serial monitor or reset to resend default value.   \*   \* Send a fixed message, you must check that the transmitter and receiver have different   \* CHANNEL ADDL or ADDH, check the configuration down   \*   \* Pai attention e22 support RSSI, if you want use that functionality you must enable RSSI on configuration   \* configuration.TRANSMISSION\_MODE.enableRSSI = RSSI\_ENABLED;   \*   \* and uncomment #define ENABLE\_RSSI true in this sketch   \*   \* Renzo Mischianti <[https://mischianti.org](https://mischianti.org/)>   \* <https://mischianti.org/category/my-libraries/ebyte-lora-e22-devices/>   \*   \* E22        ----- Arduino   \* M0         ----- 7 (or GND)   \* M1         ----- 6 (or GND)   \* RX         ----- 4 (PullUP)   \* TX         ----- 5 (PullUP)   \* AUX        ----- 3  (PullUP)   \* VCC        ----- 3.3v/5v   \* GND        ----- GND   \*   \*/    #define DESTINATION\_ADDL 3    // If you want use RSSI uncomment //#define ENABLE\_RSSI true  // and use relative configuration with RSSI enabled  //#define ENABLE\_RSSI true    #include "Arduino.h"  #include "LoRa\_E22.h"    // ---------- esp8266 pins --------------  //LoRa\_E22 e22ttl(RX, TX, AUX, M0, M1);  // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  // LoRa\_E22 e22ttl(D3, D4, D5, D7, D6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(D2, D3); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(D2, D3); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, D5, D7, D6); // AUX M0 M1  // -------------------------------------    // ---------- Arduino pins --------------  LoRa\_E22 e22ttl(4, 5, 3, 7, 6); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX AUX M0 M1  //LoRa\_E22 e22ttl(4, 5); // Config without connect AUX and M0 M1    //#include <SoftwareSerial.h>  //SoftwareSerial mySerial(4, 5); // Arduino RX <-- e22 TX, Arduino TX --> e22 RX  //LoRa\_E22 e22ttl(&mySerial, 3, 7, 6); // AUX M0 M1  // -------------------------------------    // ------------- Arduino Nano 33 IoT -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ------------- Arduino MKR WiFi 1010 -------------  // LoRa\_E22 e22ttl(&Serial1, 2, 4, 6); //  RX AUX M0 M1  // -------------------------------------------------    // ---------- esp32 pins --------------  //LoRa\_E22 e22ttl(&Serial2, 18, 21, 19); //  RX AUX M0 M1    //LoRa\_E22 e22ttl(&Serial2, 22, 4, 18, 21, 19, UART\_BPS\_RATE\_9600); //  esp32 RX <-- e22 TX, esp32 TX --> e22 RX AUX M0 M1  // -------------------------------------    **void** **setup**() {    Serial.begin(9600);    delay(500);      // Startup all pins and UART    e22ttl.begin();    //  Serial.println("Hi, I'm going to send message!");  //  //  // Send message  //  ResponseStatus rs = e22ttl.sendFixedMessage(0, DESTINATION\_ADDL, 23, "Hello, world?");  //  // Check If there is some problem of succesfully send  //  Serial.println(rs.getResponseDescription());  }    **void** **loop**() {      // If something available  **if** (e22ttl.available()>1) {        // read the String message  #ifdef ENABLE\_RSSI      ResponseContainer rc **=** e22ttl.receiveMessageRSSI();  #else      ResponseContainer rc **=** e22ttl.receiveMessage();  #endif      // Is something goes wrong print error  **if** (rc.status.code**!=**1){          Serial.println(rc.status.getResponseDescription());      }**else**{          // Print the data received          Serial.println(rc.status.getResponseDescription());          Serial.println(rc.data);  #ifdef ENABLE\_RSSI          Serial.print("RSSI: "); Serial.println(rc.rssi, DEC);  #endif      }    }  **if** (Serial.available()) {        String input **=** Serial.readString();        e22ttl.sendFixedMessage(0, DESTINATION\_ADDL, 23, input);    }  } |

Thanks