7SEMI

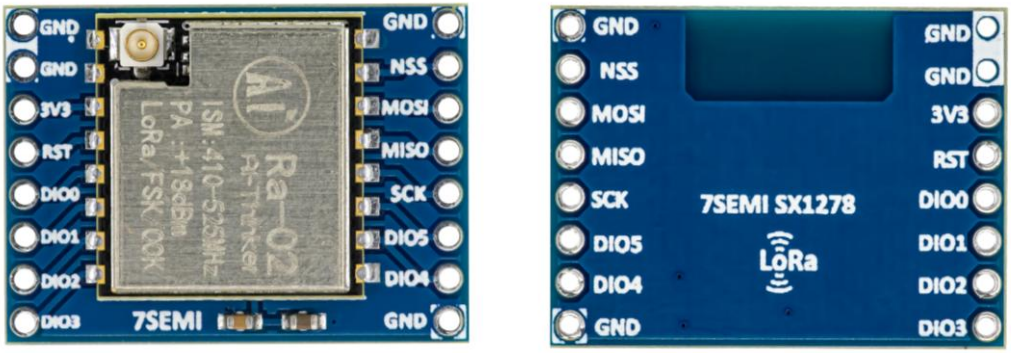
LoRa Ra-02
SX1278
Breakout Module

Version 1.0

7SEMI +91 8655821342 info@7semi.com www.7semi.com

Table of Contents

1.Introduction



The **7Semi LoRa RA-02** module is a wireless communication module based on the **Semtech** SX1278 LoRa transceiver chip. It operates in the **433 MHz frequency band** and is widely used in long-range, low-power IoT applications. LoRa (Long Range) technology enables efficient data transmission over long distances while consuming minimal power.

1.1 Features

- Frequency Range: 410 MHz – 525 MHz (typically used at 433 MHz)
- Modulation: LoRa Spread Spectrum (CSS)
- Data Rate: 0.018 kbps to 37.5 kbps
- Transmission Power: Up to +20 dBm (100 mW)
- Sensitivity: -148 dBm
- Communication Range: Up to 10 km (line-of-sight)
- Power Consumption: Low-power operation suitable for battery-powered devices
- Interface: SPI for communication with microcontrollers

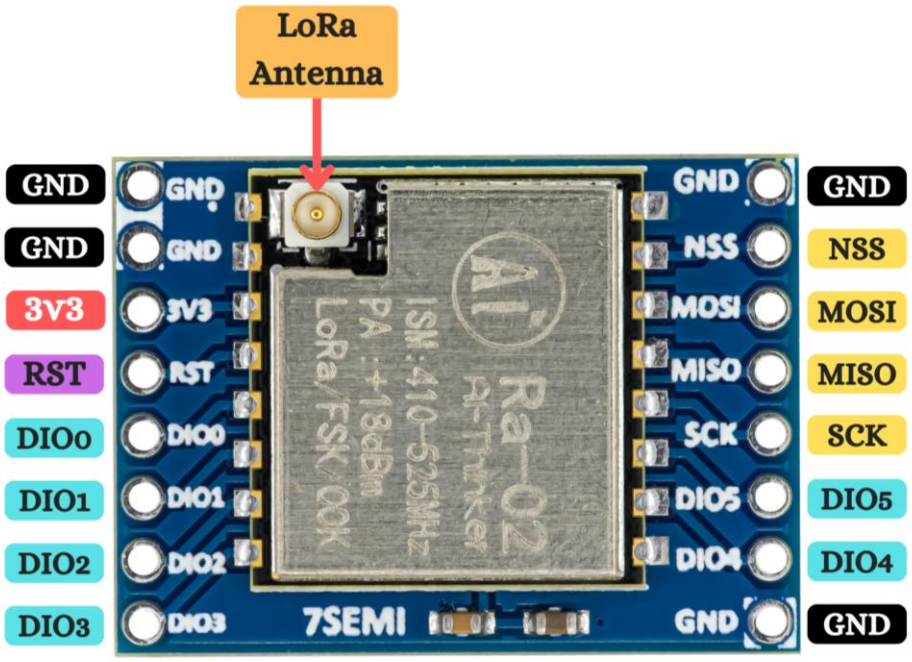
2.Technical Specification

The **Technical Specification** table provides detailed information about **7Semi SX1278 LoRa Ra-02**, including its operating voltage, current consumption, and electrical characteristics. This data helps users understand the power requirements, communication parameters, and performance capabilities of the sensor. It ensures compatibility with different microcontrollers and embedded systems while providing guidelines for efficient integration into various applications.

SX1278 Specifications

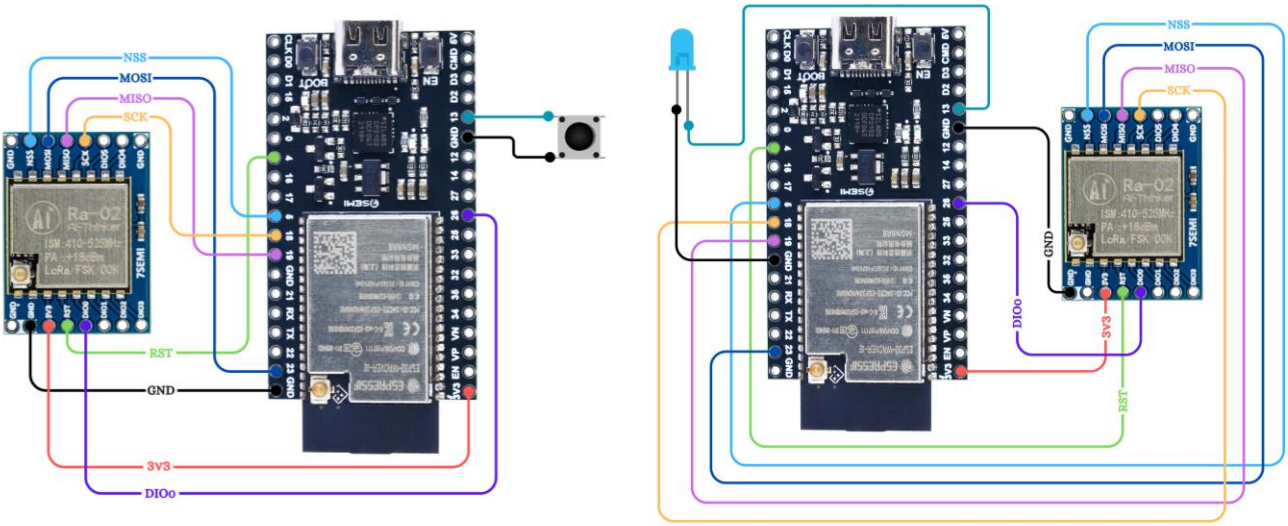
- Chipset: Semtech SX1278
- Frequency Range: 410 MHz – 525 MHz (Typically 433 MHz)
- Transmission Distance: Up to 10 km (in open areas with ideal conditions)
- Operating Voltage: 1.8V – 3.3V (Typically 3.3V)
- Operating Temperature: -40°C to +85°C
- Antenna Interface: IPEX (U.FL) Connector
- Sensitivity: Up to -148 dBm
- Module Breakout Size: 27.7mm x 20.4mm.
- Weight:

3.Pinouts



Pin	Name	Description
1	**GND**	Ground connection
2	**3V3**	Power supply input (3.3V)
3	**RST**	Module Reset
4	**DIO0**	Digital I/O for interrupts
5	**DIO1**	Digital I/O for interrupts
6	**DIO2**	Digital I/O for interrupts
7	**DIO3**	Digital I/O for interrupts
8	**DIO4**	Digital I/O for interrupts
9	**DIO5**	Digital I/O for interrupts
10	**SCK**	SPI Clock
11	**MISO**	SPI Data Output
12	**MOSI**	SPI Data Input
13	**NSS**	SPI Chip Select

4.Hardware Interface



Connection Explanation:

- Connect the **3.3V pin** of both LoRa modules to the **3.3V pin** of their respective ESP32 boards.
- Connect the **GND** pin of both LoRa modules to the **GND** pin of their respective ESP32 boards.
- Connect the **NSS** pin of both LoRa modules to the pin **GPIO 5** of their respective ESP32 boards.
- Connect the **MOSI** pin of both LoRa modules to the pin **GPIO 23** of their respective ESP32 boards.
- Connect the **MISO** pin of both LoRa modules to the pin **GPIO 19** of their respective ESP32 boards.
- Connect the **SCK** pin of both LoRa modules to the pin **GPIO 18** of their respective ESP32 boards.
- Connect the **RST** pin of both LoRa modules to the pin **GPIO 4** of their respective ESP32 boards.
- Connect the **DIO0** pin of both LoRa modules to the pin **GPIO 26** of their respective ESP32 boards.
- Connect **one leg of the push button** to **GPIO 13** of the Transmitter Esp32.
- Connect **positive terminal of the LED** to **GPIO 13** of the Receiver Esp32.
- Connect **anther led of the push button and negative terminal** of the LED to **GND** of the respective Esp32.

5.Example code link

We provide example codes to help you get started with the **7Semi SX1278 LoRa Ra-02**. These examples demonstrate how to communicate with the LoRa Module with push button to blink a LED using SPI protocol. The code is available for two popular platforms: Arduino and ESP32.

Program code for Transmitter:

```
#include <SPI.h> // Include SPI library for communication
#include <LoRa.h> // Include LoRa library
#define BUTTON_PIN 21  // GPIO pin for the button
#define SS 8
#define RST 4
#define DIO0 16

void setup() {
    Serial.begin(115200);
    pinMode(BUTTON_PIN, INPUT_PULLUP);

    SPI.begin(5, 6, 7, 8);  //(MISO=5, MOSI=6, SCK=7, SS=8)
    LoRa.setPins(SS, RST, DIO0);
    // Initialize LoRa at 433 MHz
    if (!LoRa.begin(433E6)) {
        Serial.println("LoRa initialization failed!");
        while (1);
    }
    Serial.println("LoRa Transmitter Initialized.");
}
void loop() {
      // Read the button state
    if (digitalRead(BUTTON_PIN) == LOW) {
        Serial.println("Button Pressed! Sending LED_ON...");
        LoRa.beginPacket();// Start LoRa packet
```

```
        LoRa.print("LED_ON");// Send "LED_ON" command
        LoRa.endPacket();// End packet transmission
        delay(50);
    }
    else{
        // Button is not pressed
        Serial.println("Button released! Sending LED_OFF...");
        LoRa.beginPacket();// Start LoRa packet
        LoRa.print("LED_OFF");// Send "LED_OFF" command
        LoRa.endPacket();// End packet transmission
        delay(50);
    }
}
```

Program code for Receiver:

```
#include <SPI.h> // Include SPI library for communication
#include <LoRa.h> // Include LoRa library

#define LED_PIN 13     // GPIO pin for the LED
#define SS 5
#define RST 4
#define DIO0 26

void setup() {
    Serial.begin(115200);
    pinMode(LED_PIN, OUTPUT);
    digitalWrite(LED_PIN, LOW);
    SPI.begin(18, 19, 23, 5);  //(MISO=5, MOSI=6, SCK=7, SS=8)
    LoRa.setPins(SS, RST, DIO0);
    // Initialize LoRa at 433 MHz
    if (!LoRa.begin(433E6)) {
        Serial.println("LoRa initialization failed!");
        while (1);
    }
    Serial.println("LoRa Receiver Initialized.");
```

```
}
void loop() {
    int packetSize = LoRa.parsePacket();// Check for incoming LoRa packets

    if (packetSize) {
        Serial.print("Received packet: ");

        String receivedData = ""; // Variable to store received message
        while (LoRa.available()) {
            receivedData += (char)LoRa.read();//Read and store received char
        }
        Serial.println(receivedData);
        // Control LED based on received message
        if (receivedData == "LED_ON") {
            digitalWrite(LED_PIN, HIGH);
            Serial.println("LED Turned ON!");
        } else if (receivedData == "LED_OFF") {
            digitalWrite(LED_PIN, LOW);
            Serial.println("LED Turned OFF!");
        }
    }
}
```

Code Explanation

1. **Include Required Libraries:**
   - #include <SPI.h> → Handles SPI communication between ESP32 and LoRa module.
   - #include <LoRa.h> → Provides functions for LoRa data transmission.
2. **Define hardware Connections:**
   - BUTTON_PIN (GPIO21) → Push button input.
   - SS (GPIO8) → LoRa Chip Select (NSS/CS) pin.
   - RST (GPIO4) → LoRa Reset pin.
   - DIO0 (GPIO16) → LoRa Interrupt (DIO0) pin.

3. **Setup Function:**
    - Initializes serial communication at 115200 baud rates for debugging.
    - Initializes SPI communication between ESP32 and LoRa module using SPI.begin().
    - Configures LoRa module pins using LoRa.setPins(SS, RST, DIO0).
    - Starts LoRa communication at 433 MHz using LoRa.begin(433E6).
    - If initialization fails, an error message is displayed, and the program enters an infinite loop.
4. **Loop Function for Transmitter:**
    - Continuously checks the push button state using digitalRead(BUTTON_PIN).
    - **If button is pressed (LOW)**:
        - Prints "Button Pressed! Sending LED_ON..." to the serial monitor.
        - Sends "LED_ON" message using LoRa (LoRa.beginPacket(), LoRa.print(), LoRa.endPacket()).
    - **If button is released (HIGH)**:
        - Prints "Button released! Sending LED_OFF..." to the serial monitor.
        - Sends "LED_OFF" message using LoRa.
        - Adds a debounce delay (delay (50)) to prevent unwanted repeated messages due to switch bouncing.
5. **Loop Function for Receiver:**
    - Continuously checks for incoming LoRa packets using LoRa.parsePacket().
    - If a packet is detected:
        - Reads the incoming LoRa message character by character and stores it in receivedData.
        - Displays the received message on the serial monitor.
6. **ESP32 Example Code**

This example targets ESP32 boards and showcases:

- Configuring the ESP32 SPI interface to communicate with the LoRo module.
- Controlling the LED using Push Button.
- Printing the response to the Serial Monitor.

7. Arduino Example Code

This example is designed for Arduino-compatible boards and demonstrates:

- Initializing the SPI communication with the LoRa module.
- Controlling the LED using Push Button.
- Printing the response to the Serial Monitor.
- **Note: LoRa RA-02 operates at 3.3V logic, while Arduino uses 5V logic. Direct connection may damage the module, use a level shifter for safe communication between Arduino and LoRo Module.**

How to Access the Code

- **Download Link for Transmitter Arduino and ESP32 Example:** Click Here
- **Download Link for Receiver Arduino and ESP32 Example**: Click Here

5.1 Sample Serial Output Arduino

Sample output Image of Arduino:

Transmitter:

```
LoRa Transmitter Initialized.
Button Pressed! Sending LED_ON...
Button Pressed! Sending LED_ON...
Button Pressed! Sending LED_ON...
Button Pressed! Sending LED_ON...
...
```

Receiver:

```
LoRa Receiver Initialized.
Received packet: LED_ON
LED Turned ON!
Received packet: LED_ON
LED Turned ON!
Received packet: LED_OFF
LED Turned OFF!
Received packet: LED_ON
LED Turned ON!
Received packet: LED_OFF
LED Turned OFF!
...
```

6.Mechanical Specification