riseNRF24L01 documentation

Description

The `RiseNRF24L01` class is a wrapper for the `RF24` library, which provides an interface to work with the NRF24L01+ wireless communication module. This class simplifies the usage of the NRF24L01+ module and provides a set of methods that allow you to send and receive data packets wirelessly. It does not include AES encryption as the padding cannot be set to a fixed size when the packet size is variant.

Class Members

Constructor

The `RiseNRF24L01` class constructor takes three arguments:

1. `ce_pin` - the chip enable (CE) pin to which the NRF24L01+ module is connected.
2. `cs_pin` - the chip select (CSN) pin to which the NRF24L01+ module is connected.
3. `address` - an array of bytes that represents the address of the NRF24L01+ module.

```
RiseNRF24L01(byte ce_pin, byte cs_pin, byte* address);
```

begin()

The `begin()` method initializes the NRF24L01+ module and sets the power amplifier level to low. It also opens a writing pipe and a reading pipe with the given address.

```
void begin();
```

send()

The `send()` method sends a data packet to the specified destination address.

It takes three arguments:

1. `data` - a pointer to the data that you want to send.
2. `size` - the size of the data that you want to send.
3. `dest_address` - an array of bytes that represents the destination address.

```
bool send(void* data, uint8_t size, byte* dest_address);
```

This method returns a boolean value that indicates whether the data packet was sent successfully.

receive()

The `receive()` method checks whether there is any data available to receive from the NRF24L01+ module. If there is data available, it reads the data and stores it in the specified buffer.

It takes two arguments:

1. `data` - a pointer to the buffer where the received data will be stored.
2. `size` - the size of the data that you want to receive.

```
bool receive(void* data, uint8_t size);
```

This method returns a boolean value that indicates whether there was any data available to receive.

isAddressMatched()

The `isAddressMatched()` method checks whether the address of the received data packet matches the expected address.

It takes one argument:

1. `expected_address` - an array of bytes that represents the expected address.

```
bool isAddressMatched(byte* expected_address);
```

This method returns a boolean value that indicates whether the address of the received data packet matches the expected address.

Usage example :

```
#include "RiseTelemetry.h"

// Define a custom struct called MyData with three members: x, y, and z
struct MyData {
  int x;
  int y;
  float z;
};

// Set the address of the NRF24L01 module
byte address[6] = {0x01, 0x02, 0x03, 0x04, 0x05};

// Initialize an instance of the RiseNRF24L01 library on pins 9 and 10 with the specified address
RiseNRF24L01 nrf24l01(9, 10, address);

// Create an instance of the MyData struct to hold the received data
MyData myData;

void setup() {
  // Initialize the serial port for debugging
  Serial.begin(9600);

  // Initialize the NRF24L01 module
  nrf24l01.begin();
}

void loop() {
  // Check if there is any data available on the NRF24L01 module and if the sender's address matches the expected address
  if (nrf24l01.receive(&myData, sizeof(myData)) && nrf24l01.isAddressMatched(address)) {
    // If data is available and the sender's address matches, print the received data to the serial port
    Serial.print("Received data: ");
    Serial.print(myData.x);
    Serial.print(", ");
    Serial.print(myData.y);
    Serial.print(", ");
    Serial.println(myData.z);
  }

  // Generate some random data to send
  myData.x = random(10);
  myData.y = random(10);
  myData.z = random(100) / 10.0;

  // Send the data to the other NRF24L01 module with the specified address
  nrf24l01.send(&myData, sizeof(myData), address);

  // Wait for 1 second before sending more data
  delay(1000);
}
```