```
/*
   LoRaLib Receive with Interrupts Example

   This example listens for LoRa transmissions and tries to
   receive them. Once a packet is received, an interrupt is
   triggered. To successfully receive data, the following
   settings have to be the same on both transmitter
   and receiver:
    - carrier frequency
    - bandwidth
    - spreading factor
    - coding rate
    - sync word
    - preamble length

   For more detailed information, see the LoRaLib Wiki
    https://github.com/jgromes/LoRaLib/wiki

   For full API reference, see the GitHub Pages
    https://jgromes.github.io/LoRaLib/
*/

// include the library
#include <LoRaLib.h>

// create instance of LoRa class using SX1278 module
// this pinout corresponds to RadioShield
//  https://github.com/jgromes/RadioShield
// NSS pin:   10 (4 on ESP32/ESP8266 boards)
// DIO0 pin:  2
// DIO1 pin:  3
// IMPORTANT: because this example uses external interrupts,
//            DIO0 MUST be connected to Arduino pin 2 or 3.
//            DIO1 MAY be connected to any free pin
//            or left floating.
SX1278 lora = new LoRa;

void setup() {
  Serial.begin(9600);

  // initialize SX1278 with default settings
  Serial.print(F("Initializing ... "));
  // carrier frequency:           434.0 MHz
  // bandwidth:                   125.0 kHz
  // spreading factor:            9
  // coding rate:                 7
  // sync word:                         0x12
  // output power:                      17 dBm
  // current limit:                     100 mA
  // preamble length:                   8 symbols
  // amplifier gain:                    0 (automatic gain control)
  int state = lora.begin();
  if (state == ERR_NONE) {
    Serial.println(F("success!"));
  } else {
    Serial.print(F("failed, code "));
    Serial.println(state);
    while (true);
  }

  // set the function that will be called
  // when new packet is received
  lora.setDio0Action(setFlag);

  // start listening for LoRa packets
  // NOTE: for spreading factor 6, the packet length
  //       must be known in advance, and provided to both
  //       startReceive() and readData() methods!
  Serial.print(F("Starting to listen ... "));
  state = lora.startReceive();
  if (state == ERR_NONE) {
    Serial.println(F("success!"));
  } else {
    Serial.print(F("failed, code "));
    Serial.println(state);
    while (true);
  }

  // NOTE: 'listen' mode will be disabled
  // automatically by calling any of the
  // following methods:
  //
  // lora.standby()
  // lora.sleep()
  // lora.transmit()
  // lora.receive()
  // lora.scanChannel()
  //
  // LoRa module will not receive any new
  // packets until 'listen' mode is re-enabled
  // by calling lora.startReceive()
}
```

```
// flag to indicate that a packet was received
volatile bool receivedFlag = false;

// disable interrupt when it's not needed
volatile bool enableInterrupt = true;

// this function is called when a complete packet
// is received by the module
// IMPORTANT: this function MUST be 'void' type
//            and MUST NOT have any arguments!
void setFlag(void) {
  // check if the interrupt is enabled
  if(!enableInterrupt) {
    return;
  }

  // we got a packet, set the flag
  receivedFlag = true;
}

void loop() {
  // check if the flag is set
  if(receivedFlag) {
    // disable the interrupt service routine while
    // processing the data
    enableInterrupt = false;

    // reset flag
    receivedFlag = false;

    // you can read received data as an Arduino String
    String str;
    int state = lora.readData(str);

    // you can also read received data as byte array
    /*
      byte byteArr[8];
      int state = lora.readData(byteArr, 8);
    */

    if (state == ERR_NONE) {
      // packet was successfully received
      Serial.println(F("Received packet!"));

      // print data of the packet
      Serial.print(F("Data:\t\t\t"));
      Serial.println(str);

      // print RSSI (Received Signal Strength Indicator)
      Serial.print(F("RSSI:\t\t\t"));
      Serial.print(lora.getRSSI());
      Serial.println(F(" dBm"));

      // print SNR (Signal-to-Noise Ratio)
      Serial.print(F("SNR:\t\t\t"));
      Serial.print(lora.getSNR());
      Serial.println(F(" dB"));

      // print frequency error
      Serial.print(F("Frequency error:\t"));
      Serial.print(lora.getFrequencyError());
      Serial.println(F(" Hz"));

    } else if (state == ERR_CRC_MISMATCH) {
      // packet was received, but is malformed
      Serial.println(F("CRC error!"));

    }

    // we're ready to receive more packets,
    // enable interrupt service routine
    enableInterrupt = true;
  }

}
```