

LORA 32: CONNESSIONI INTELLIGENTI, OVUNQUE



By Morgan Petrelli & Giuseppe Cialdella



Il potere del LoRa

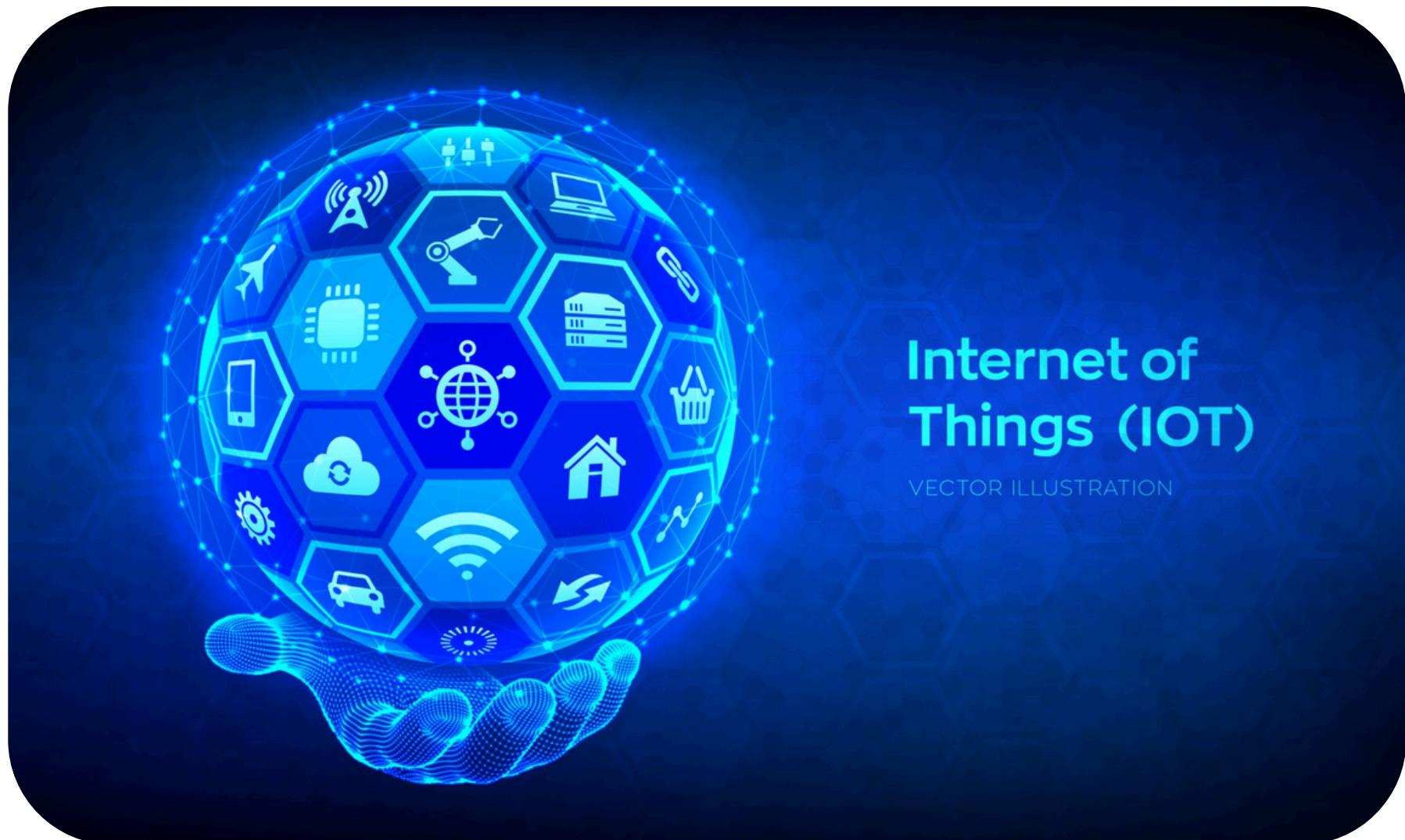
“Da piccoli chip derivano grandi connessioni”

Viviamo in un mondo dove tutto può comunicare.

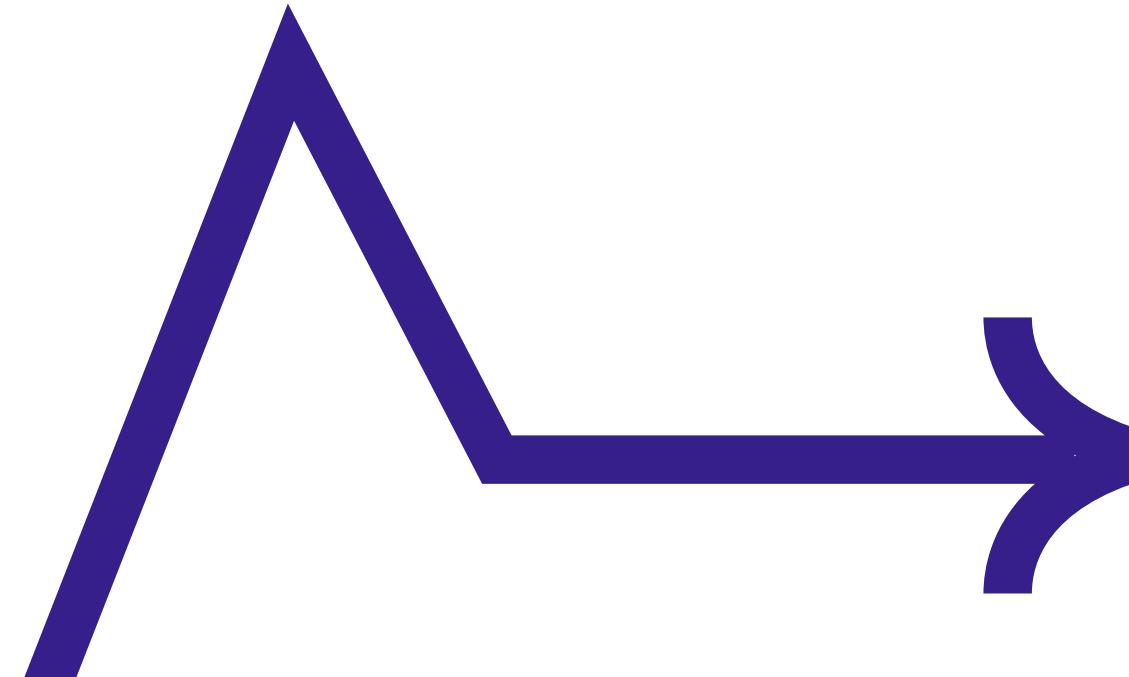
Come connettere dispositivi in aree isolate, senza Wi-Fi o rete cellulare?

👉 La risposta è LoRa 32:

- ◆ Basso consumo
- ◆ Lunga distanza
- ◆ Ideale per l'IoT



Cos'è LoRa?



È un microcontrollore con:

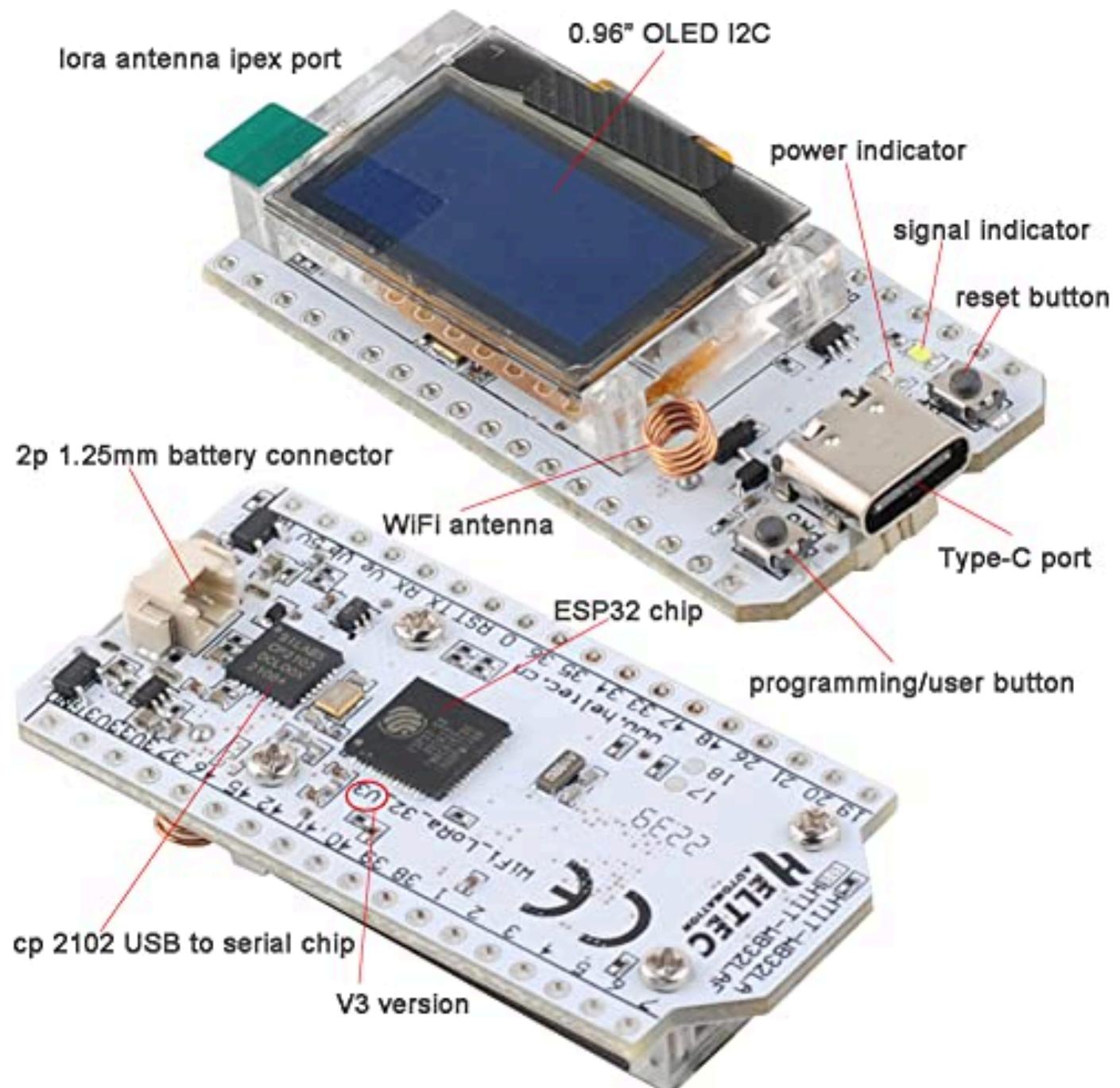
- Wi-Fi + Bluetooth
 - Modulo LoRa (Long Range)
 - ESP32 dual-core
- ✓ Comunica a chilometri di distanza
 - ✓ Perfetto per ambienti remoti
 - ✓ Funziona senza Internet



◆ Specifiche Tecniche e Funzionalità del Dispositivo

Oltre a integrare Wi-Fi, Bluetooth e un modulo radio LoRa,
offre anche:

- Processore ESP32 dual-core.
- GPIO multipli.
- Alimentazione tramite batteria.
- Porta microUSB per programmazione e alimentazione.
- Supporto per deep sleep mode.
- Possibilità di espansione con antenne esterne.



◆ Come funziona LoRa?

- Trasmissione a pacchetti via radiofrequenza
- Modalità:
 - Punto-punto
 - Con infrastruttura LoRaWAN (gateway + server)
- Portata: fino a 15 km in campo aperto



◆ Esempi di utilizzo

Reti di sensori ambientali:

Permette di collegare sensori per rilevare dati come temperatura, umidità, qualità dell'aria o livelli di inquinamento.

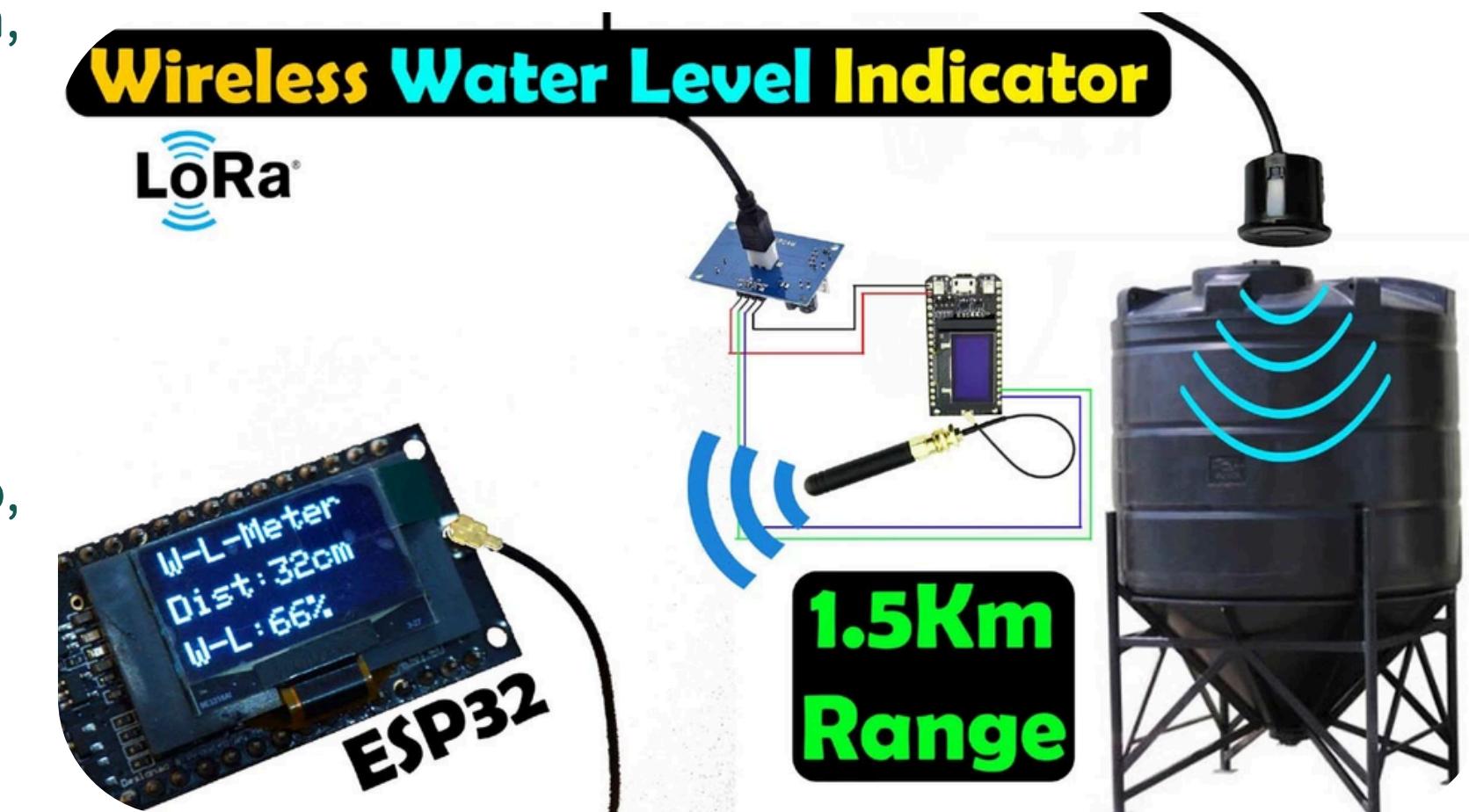
Monitoraggio agricolo (smart farming):

In agricoltura, viene utilizzato per monitorare umidità del suolo, condizioni meteorologiche o livelli dei serbatoi d'acqua.



Domotica:

In ambito smart home, può connettere dispositivi distribuiti su grandi aree, permettendo il controllo remoto di luci, allarmi, serrature e altri sistemi.



◆ Vantaggi e Svantaggi di LoRa

✓ Vantaggi

- Basso consumo energetico.
- Costo contenuto.
- Uso senza licenza: utilizza bande ISM (libere).

✗ Svantaggi

- Bassa velocità di trasmissione.
- Non supporta audio o video.
- Limiti regolatori: in alcune regioni ci sono restrizioni sull'uso delle frequenze e sul tempo di trasmissione.



Vantaggi



Lunga
distanza di
trasmissione



Basso
consumo
energetico



Elevata
capacità di
resistenza alle
interferenze



Sicurezza
dei dati



Svantaggi



Bassa
velocità di
trasmissione



Capacita
limitata
della rete



Dipendenza
dalla topologia



Limitazioni
normative

Comunicazione tra LoRa Sender & Receiver

Due dispositivi comunicano via LoRa: un Sender, che invia periodicamente messaggi , e un Receiver, che li riceve e visualizza. Tutto funziona senza internet, solo tramite radiofrequenze.

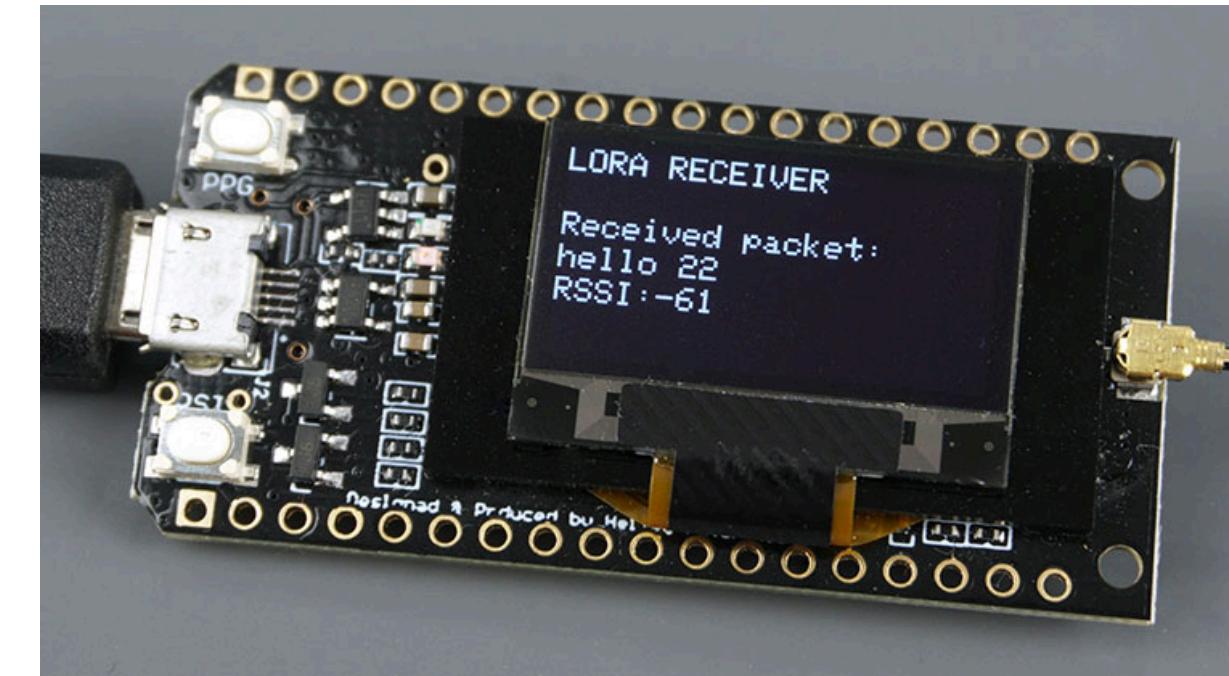
SENDER:

- ⬆️ Invia messaggi ogni 5s
- ⟳ Comandi: start / stop
- 📺 Mostra stato su display



RECEIVER:

- ⬇️ Riceve messaggi
- 📺 Li mostra su OLED
- 🔧 Gestisce timeout ed errori



Analisi script sender



The screenshot shows the Arduino IDE interface with the sketch named "sender_START_STOP.ino". The code is written in C++ and uses several libraries: RadioLib.h, HT_SSD1306Wire.h, and HT_SSD1306Wire.h. It includes definitions for pins LORA_CS, LORA_RST, LORA_BUSY, and LORA_DIO1. The code initializes a SX1262 module and an SSD1306 display. It sets up the serial port at 115200 bps and configures pins. It then initializes the display and prints "LoRa Sender...". It checks if the LoRa module is initialized and prints an error message if not. It then enters a loop where it reads commands from the serial monitor ('start' or 'stop'), updates the display, and sends a message every 5 seconds via LoRa. The message contains the current value of the counter.

```
#include <RadioLib.h>
#include "HT_SSD1306Wire.h"

// Pin SX1262 Heltec V3
#define LORA_CS 8
#define LORA_RST 12
#define LORA_BUSY 13
#define LORA_DIO1 14

SX1262 lora = new Module(LORA_CS, LORA_DIO1, LORA_RST, LORA_BUSY);

// OLED Heltec
SSD1306Wire mydisplay(0x3C, 400000, SDA_OLED, SCL_OLED, GEOMETRY_128_64, RST_OLED);

int counter = 0;
unsigned long lastSend = 0;
bool sendingEnabled = true;

void setup() {
  Serial.begin(115200);
  pinMode(Vext, OUTPUT);
  digitalWrite(Vext, LOW); // Accende LoRa + OLED
  delay(100);

  mydisplay.init();
  mydisplay.setFont(ArialMT_Plain_10);
  mydisplay.clear();
  mydisplay.drawString(0, 0, "LoRa Sender...");
  mydisplay.display();

  int state = lora.begin();
  if (state != RADIOLIB_ERR_NONE) {
    Serial.print("Errore LoRa: ");
    Serial.println(state);
    while (true);
  }
}

mydisplay.drawString(0, 20, "LoRa inizializzato!");
mydisplay.display();

Serial.println("Digita 'stop' per fermare l'invio o 'start' per riattivarlo.");
}

void loop() {
  // Lettura comandi da Serial Monitor
  if (Serial.available()) {
    String cmd = Serial.readStringUntil('\n');
    cmd.trim(); // Rimuove spazi e newline

    if (cmd.equalsIgnoreCase("start")) {
      sendingEnabled = true;
      Serial.println("Invio abilitato.");
      mydisplay.clear();
      mydisplay.drawString(0, 0, "Invio ON");
      mydisplay.display();
    } else if (cmd.equalsIgnoreCase("stop")) {
      sendingEnabled = false;
      Serial.println("Invio disabilitato.");
      mydisplay.clear();
      mydisplay.drawString(0, 0, "Invio OFF");
      mydisplay.display();
    } else {
      Serial.println("Comando non riconosciuto. Usa 'start' o 'stop'.");
    }
  }

  // Invio LoRa solo se abilitato
  if (sendingEnabled && (millis() - lastSend > 5000)) {
    lastSend = millis();
    String msg = "Messaggio #" + String(counter++);
    Serial.print("Invio: ");
    Serial.println(msg);

    int state = lora.transmit(msg);
    mydisplay.clear();
    if (state == RADIOLIB_ERR_NONE) {
      mydisplay.drawString(0, 0, "Invia:");
      mydisplay.drawString(0, 10, msg);
    } else {
      mydisplay.drawString(0, 0, "Errore TX!");
    }
    mydisplay.display();
  }
}
```

🔧 Inizializzazione:

- **Librerie:** Include le librerie necessarie per la comunicazione LoRa (<SPI.h>, <LoRa.h>) e per il display OLED se utilizzato.
- **Configurazione dei Pin:** Definisce i pin GPIO utilizzati per la comunicazione SPI con il modulo LoRa.
- **Frequenza:** Imposta la frequenza di trasmissione (ad esempio, 433E6 per 433 MHz).

⟳ Loop Principale:

- **Contatore:** Mantiene un contatore per tracciare il numero di messaggi inviati.
- **Comandi Seriali:** Ascolta comandi dalla porta seriale per avviare o fermare la trasmissione.
- **Invio Messaggi:** Ogni 5 secondi, invia un messaggio contenente il contatore incrementato.
- **Display OLED:** Aggiorna il display con lo stato corrente e il numero di messaggi inviati.

Analisi script receiver



The screenshot shows the Arduino IDE interface with the file 'receiver_def.ino' open. The code is written in C++ and defines a LoRa receiver setup. It includes RadioLib and HT_SSD1306Wire libraries, configures pins for SX1262 Heltec V3, initializes an SSD1306 OLED display, and sets up the LoRa module. The setup() function initializes the serial port at 115200 bps, sets pin Vext as an output, and prints "LoRa Receiver..." to the display. The loop() function checks for received messages, prints them to the serial port, and displays them on the OLED screen. It also handles errors and implements a timeout mechanism.

```
#include <RadioLib.h>
#include "HT_SSD1306Wire.h"

// Pin SX1262 Heltec V3
#define LORA_CS 8
#define LORA_RST 12
#define LORA_BUSY 13
#define LORA_DIO1 14

SX1262 lora = new Module(LORA_CS, LORA_DIO1, LORA_RST, LORA_BUSY);

// OLED Heltec
SSD1306Wire mydisplay(0x3C, 400000, SDA_OLED, SCL_OLED, GEOMETRY_128_64, RST_OLED);

void setup() {
    Serial.begin(115200);
    pinMode(Vext, OUTPUT);
    digitalWrite(Vext, LOW);
    delay(100);

    mydisplay.init();
    mydisplay.setFont(ArialMT_Plain_10);
    mydisplay.clear();
    mydisplay.drawString(0, 0, "LoRa Receiver...");
    mydisplay.display();

    int state = lora.begin();
    if (state != RADIOLIB_ERR_NONE) {
        Serial.print("Errore LoRa: ");
        Serial.println(state);
        mydisplay.drawString(0, 20, "Errore LoRa");
        mydisplay.display();
        while (true);
    }

    mydisplay.drawString(0, 20, "In ascolto...");
    mydisplay.display();
}

void loop() {
    String msg;
    int state = lora.receive(msg);

    // Se è stato ricevuto correttamente
    if (state == RADIOLIB_ERR_NONE) {
        Serial.print("[Ricevuto] ");
        Serial.println(msg);
        mydisplay.clear();
        mydisplay.drawString(0, 0, "Ricevuto:");
        mydisplay.drawString(0, 10, msg);
        mydisplay.display();
    }
    // Se timeout, prova di nuovo
    else if (state == RADIOLIB_ERR_RX_TIMEOUT) {
    }
    else {
        Serial.println("Errore ricezione");
    }

    // Ricezione continua (in caso di timeout, si riprova senza bloccare)
    delay(10);
}
```

🔧 Inizializzazione:

- **Librerie:** Include le stesse librerie del Sender.
- **Configurazione dei Pin:** Imposta i pin GPIO per la comunicazione con il modulo LoRa.
- **Frequenza:** Assicura che la frequenza impostata corrisponda a quella del Sender.

⟳ Loop Principale:

- **Ricezione Pacchetti:** Controlla continuamente la presenza di nuovi pacchetti LoRa.
- **Lettura Dati:** Se un pacchetto è disponibile, legge il contenuto e lo visualizza sul display OLED e/o sulla porta seriale.
- **Gestione Errori:** Implementa timeout o altri meccanismi per gestire la perdita di pacchetti o errori di comunicazione.

END Conclusion



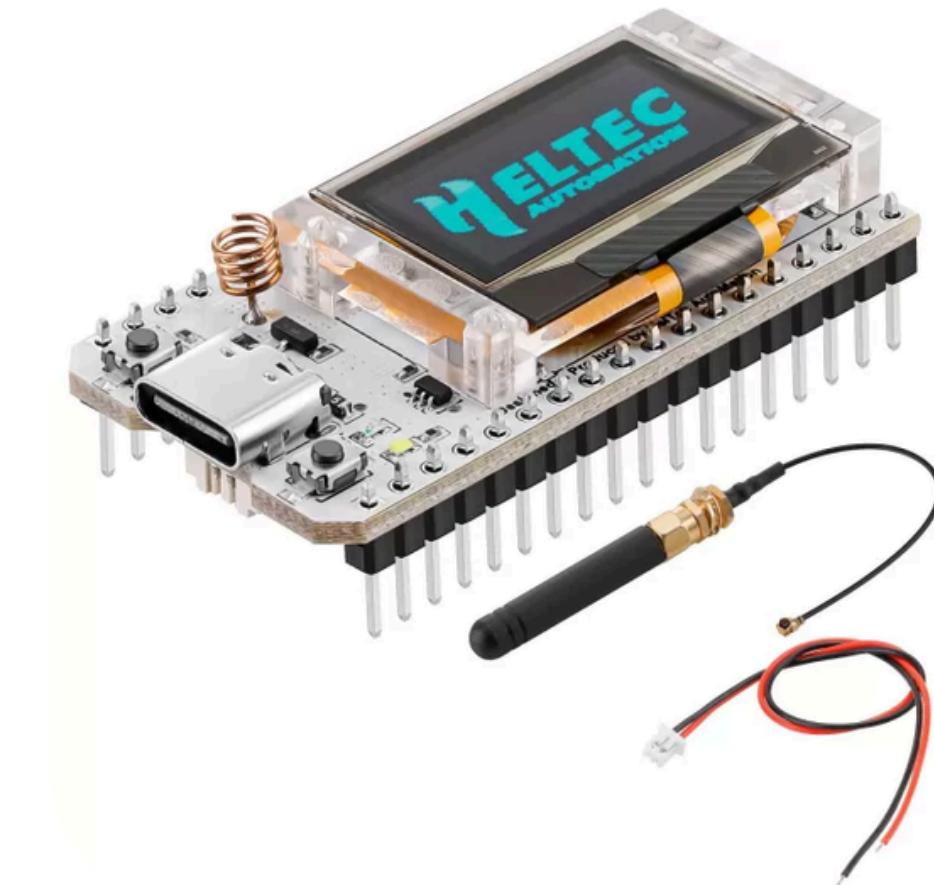
LoRa 32 non è solo un modulo di comunicazione: è una chiave per

abilitare soluzioni IoT in contesti difficili e remoti.

Grazie alla sua efficienza energetica, ampia copertura e semplicità d'uso, è

oggi una delle tecnologie più promettenti per connettere il mondo reale al

digitale.



**“In un mondo sempre più connesso, la vera
innovazione nasce dove la connessione sembra
impossibile.”**