Code Documentation

ESP32 + RAK3172 (Ear-Tag Transmitter)

1. Overview

This firmware runs on an **ESP32** microcontroller connected to a **RAK3172 LoRa module**.
The purpose is to collect **chicken health and activity data** from multiple sensors and transmit it live to a **BOM (Base-of-Monitoring) device** via **LoRa Peer to Peer communication** at **866 MHz** (legal ISM frequency band in India).

2. Connected Sensors

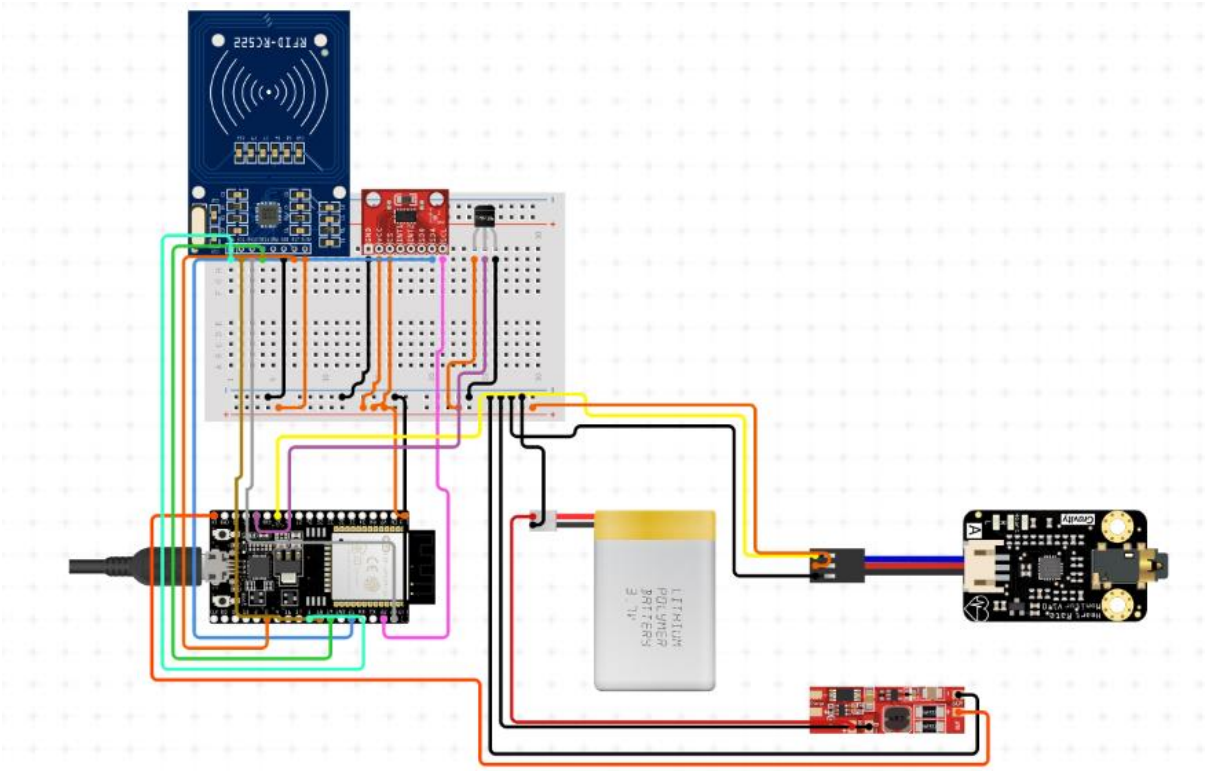
- **RFID (RC522, SPI)** → Each chicken has a unique RFID tag; identifies the individual.
- **DHT22** → Measures ambient temperature (°C).
- **MPU6050** → 3-axis accelerometer for movement/activity tracking.
- **MAX3010x (e.g., MAX30102/05)** → Optical sensor for Heart Rate (BPM) and SpO_2 (%).

3. Communication Interfaces

- **SPI** → Used for RFID module (RC522).
- **I2C** → Shared bus for MPU6050 (accelerometer) and MAX3010x (heart/SpO_2).
- **UART (Serial1)** → Used between ESP32 and RAK3172 LoRa module.

ESP32 Pins Used:

- `RST_PIN = 4`, `SS_PIN = 5` → RFID
- `DHTPIN = 15` → DHT22
- `I2C SDA=21, SCL=22` → MPU6050 + MAX3010x
- `RAK_RX_PIN = 16`, `RAK_TX_PIN = 17` → LoRa UART link



4. LoRa Configuration

LoRa is set in **P2P (Peer-to-Peer) mode** using AT commands:

```
sendAT("AT+NWM=0");                    // Switch to P2P mode
sendAT("AT+P2P=866000000:7:125:1:8:"); // Frequency 866 MHz (India legal band),
                                       // SF7, BW 125 kHz, CR 4/5, Preamble=8
sendAT("AT+SYNCWORD=34");              // Set sync word for network separation
sendAT("AT+PRECV=65533"); // Enable continuous RX while allowing TX
```

- **Why 866 MHz?**
  The 865–867 MHz band is the **license-free LoRa band in India**, so the firmware is tuned accordingly.

5. Data Workflow

1. **Read sensors:**

a. RFID → Read UID if a tag is present.
b. DHT22 → Read temperature (°C).
c. MPU6050 → Read accelerometer values (X, Y, Z in m/s^2).
d. MAX3010x → Collect heart rate & SpO_2 (simplified algorithm for demo).

2. **Format payload:**
   Create a JSON string:

```
{
  "id":"ABC123",
  "t":28.50,
  "ax":0.12,
  "ay":-0.03,
  "az":9.70,
  "hr":72,
  "spo2":97.2
}
```

3. **Convert to HEX:**
   Since RAK3172 requires hex-formatted payload for `AT+PSEND`, the JSON string is converted into a hex string.
4. **Transmit via LoRa:**
   Use `AT+PSEND=<hexPayload>` to send the packet.
   Retries are performed if the RAK3172 is busy.
5. **Wait interval:**
   Respect **3-second interval** (or configured value) before sending the next update.

6. Important Functions

- **`sendAT(cmd)`** → Sends AT command string to RAK3172 over UART.
- **`readRakResponse()`** → Reads and prints RAK3172 response.
- **`readRFIDOnce()`** → Reads tag UID (updates global variable).
- **`readTemperature()`** → Reads temperature (handles NAN cases).
- **`readAccelerometer()`** → Fetches X, Y, Z accelerometer readings.
- **`readHeartSpO2()`** → Collects samples from MAX3010x (demo version, placeholder values in current code).
- **`toHex()`** → Converts JSON payload into hex string for LoRa transmission.

7. Limitations / Notes

- **MAX3010x readings:** The heart rate and SpO_2 functions currently use placeholder/demo logic; in production, a validated algorithm/library must be used.
- **RFID reading:** Only updates when a new tag is present, otherwise retains last value.
- **Duty cycle regulations:** LoRa transmissions in ISM bands must respect duty-cycle limits (regional compliance).
- **Error handling:** Retries for busy LoRa module are basic; advanced handling can be added.

8. System Flow (High-Level)

1. Initialize sensors and LoRa module.
2. Loop:
    a. Read all sensor values.
    b. Create JSON payload.
    c. Convert JSON to hex string.
    d. Send via LoRa (`AT+PSEND`).
    e. Retry if busy.
    f. Wait until next send interval.