ESP32 BLE + iBeacon Scanner: Updated Documentation & Workflow

System Overview

Your setup now does **both iBeacon transmission and BLE scanning simultaneously**, and sends scanned data to your Node.js backend.

Components:

Component	Role	
ESP32	Transmits iBeacon & scans BLE devices simultaneously	
Node.js Backend	Receives scanned data via REST API & stores it in SQLite	
SQLite DB	Stores devices and readings	
Frontend (optional)	Can visualize live BLE scans via Socket.IO	

ESP32 Workflow

Setup Phase (setup())

- 1. WiFi Connection @
 - o Connect to specified SSID and password
 - o Print IP address and MAC
- 2. BLE iBeacon Initialization 📝
 - Beacon continuously advertises:

- UUID: E2C56DB5-DFFB-48D2-B060-D0F5A71096E0
- Major: 1
- Minor: 1
- TX Power: -59
- Non-connectable beacon (ADV_TYPE_NONCONN_IND)
- 3. BLE Scan Setup 🔍
 - Active scan mode
 - Set scan interval/window
 - Prepare internal buffer foundArr[MAX_FOUND]

Loop Phase (loop())

- 1. WiFi Check & Reconnect 🔄
 - Ensures ESP32 stays online
- 2. BLE Scanning 🔍
 - Scan duration: SCAN_SECONDS (4 sec)
 - For each detected device:
 - Skip own MAC
 - Store: MAC, name, RSSI, advertisement data (advHex), protocol, tx beacon name
 - Store in circular buffer foundArr (replace if stronger RSSI)

3. Build JSON Payload 🗐

Convert foundArr to JSON:

```
{
    "mac": "xx:xx:xx:xx",
    "name": "BLE Device",
   "type": "BLE",
    "protocol": "BLE",
    "uuid": null,
    "major": 1,
    "minor": 1,
    "tx_power": -59,
    "tx_beacon_name": "ESP32_Beacon",
    "rssi": -70,
    "adv_data": "FF01AA...",
    "ts": 123456789
 }
]
```

o Major, Minor, TX Power always included

4. Send JSON to Backend ...

- o POST request to http://192.168.1.122:3000/api/scan
- Check HTTP response for confirmation

5. Clear Scan Results /

- Prevent duplicate processing
- Keep beacon running continuously

6. Loop Delay 👸

Small delay (POST_DELAY_MS = 200ms) to avoid overwhelming the backend

Node.js Backend Workflow

Setup

- Dependencies:
 - o express, socket.io, better-sqlite3, cors, body-parser, dotenv
- Database: SQLite (devices + readings tables)

Tables

devices:

- id INTEGER PK
- mac TEXT UNIQUE
- name TEXT
- type TEXT
- protocol TEXT
- first_seen INTEGER
- last_seen INTEGER

readings:

- id INTEGER PK
- device_id INTEGER FK
- rssi INTEGER
- adv_data TEXT
- ts INTEGER
- tx_beacon_name TEXT
- major INTEGER
- minor INTEGER
- tx_power INTEGER

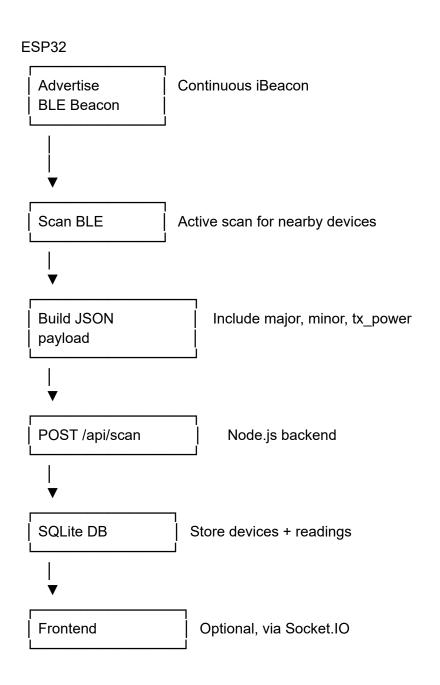
API Endpoints

Endpoint	Method	Description
/api/scan	POST	Receive ESP32 JSON payload, insert/update devices and readings
/api/devices	GET	List all devices
/api/devices/:mac/rea	GET	List all readings for a specific device

Data Handling

- 1. Insert device if not exists (INSERT OR IGNORE)
- 2. Update last_seen and protocol
- 3. Insert reading with numeric fields:
 - o major, minor, tx_power
 - o rssi
 - o adv_data
 - o tx_beacon_name
 - o timestamp
- 4. Emit batch to frontend via **Socket.IO** (scan:batch)

Data Flow Diagram



Key Features After Update

- Continuous iBeacon transmission
- **☑** BLE scanning runs simultaneously
- JSON payload includes numeric fields: major, minor, tx_power
- Z Data stored reliably in SQLite backend
- Socket.IO emits live scan updates
- Z Easy to expand for OTA, metrics, or more metadata