ESP32 Gait Sensor System – Command Protocol Reference

This document defines the complete command protocol for the ESP32-based Wi-Fi Node and Sensor Node system, including UART and MQTT message types.

🟦 Commands: Wi-Fi Node → Sensor Node (UART)

Command	Description
`PING`	Health check sent on boot
`SYNC_TIME:<epoch>`	Send epoch time to sync clocks (every 5s)
`START_CALIBRATION`	Instruct sensor node to begin calibration
`REQ_CAL_STATUS`	Request current calibration status
`CAPTURE_ORIENTATION`	Request to capture initial orientation after wearing the device
`REQ_DATA`	Request a single sample of sensor data

🟩 Messages: Sensor Node → Wi-Fi Node (UART)

Message Type	Format / Example	Description
`ACK`	`"ACK"`	Response to `PING`
`cal_status`	`{ "type": "cal_status", "device_id": "esp-001", "timestamp": 1715776000, "sys": 3, "gyro": 3, ... }`	Sent during calibration and on REQ_CAL_STATUS

Message Type	Format / Example	Description
`sensor_data`	`{ "type": "sensor_data", "device_id": "esp-001", "timestamp": ..., "FSR_1": ..., "yaw": ..., ... }`	Single sensor reading on `REQ_DATA`
`orientation_captured`	`{ "type": "orientation_captured", "device_id": "esp-001", "timestamp": 1715776050, "status": true }`	Confirmation after capturing orientation

🟪 MQTT Commands: Backend → Wi-Fi Node

Topic: `device/{DEVICE_ID}/command`

Command Payload	Description
`{ "command": "check_calibration" }`	Triggers `REQ_CAL_STATUS` to the sensor
`{ "command": "start_calibration" }`	Sends `START_CALIBRATION` to sensor
`{ "command": "capture_orientation" }`	Sends `CAPTURE_ORIENTATION` to sensor
`{ "command": "start_streaming" }`	Starts REQ loop and begins data streaming
`{ "command": "stop_streaming" }`	Stops data loop and sends ALIVE status every 5 Sec

Example Command:

```
{
  "command": "capture_orientation"
}
```

🟧 MQTT Status Updates: Wi-Fi Node → Cloud

Topic	Triggered By	Payload Type
`device/{DEVICE_ID}/status/alive`	Periodically every 30s, and on boot	`{ "type": "device_alive", "device_id": ..., "status": true, "timestamp": ... }`
`device/{DEVICE_ID}/status/calibration`	From calibration updates	`{ "type": "cal_status", "device_id": ..., "status": true/false, "sys": ..., ... }`
`device/{DEVICE_ID}/status/orientation`	From orientation capture response	`{ "type": "orientation_captured", "device_id": ..., "status": true, "timestamp": ... }`
`device/{DEVICE_ID}/sensor_data`	After each sensor `REQ_DATA` response	`{ "type": "sensor_data", "device_id": ..., "timestamp": ..., "FSR_1": ..., ... }`

Example Payloads

- Calibration Status:

```
{
  "type": "cal_status",
  "device_id": "esp-001",
  "timestamp": 1715776000,
  "sys": 3,
  "gyro": 3,
  "accel": 2,
  "mag": 3,
  "status": true
}
```

- Orientation Captured:

```
{
  "type": "orientation_captured",
  "device_id": "esp-001",
  "timestamp": 1715776050,
  "status": true
}
```


- Device Alive:


```
{
  "type": "device_alive",
  "device_id": "esp-001",
  "timestamp": 1715776100,
  "status": true
}
```


- Sensor Data:


```
{
  "type": "sensor_data",
  "device_id": "esp-001",
  "timestamp": 1715776110,
  "FSR_1": 320,
  "FSR_2": 318,
  "yaw": -0.12,
  "pitch": 1.22,
  "roll": -0.87,
  "q0": 0.99,
  "q1": 0.01,
  "q2": -0.02,
  "q3": 0.03,
  "ax": 0.12,
  "ay": -0.05,
  "az": 9.81,
  "gx": 0.03,
  "gy": -0.01,
  "gz": 0.02,
  "sys_cal": 3,
  "gyro_cal": 3,
  "accel_cal": 2,
  "mag_cal": 3
}
```

✅ Session Flow Summary

1. **Boot**
    - Wi-Fi Node connects to Wi-Fi and AWS
    - Sends `PING` to sensor → expects `ACK`
    - Sends `device_alive` heartbeat to cloud every 30s
2. **Start Session** (UI → Backend)
    - Backend sends `check_calibration`
    - If not calibrated → send `start_calibration`
    - Sensor sends `cal_status` periodically until complete
3. **Wear Phase**
    - UI shows "I'm Ready" → backend sends `capture_orientation`
    - Sensor captures orientation → sends `orientation_captured`
4. **Streaming Phase**
    - Backend sends `start_streaming` → Wi-Fi Node starts REQ loop
    - Publishes `sensor_data` every 100ms to MQTT
5. **Stop Phase**
    - UI sends stop → backend sends `stop_streaming`
    - Wi-Fi Node sends `STOP_STREAMING`; system idles
6. **Repeatable**
    - New session → backend always starts with `check_calibration`
    - No persistent calibration state stored on backend; source of truth = sensor node

---

🧩 Notes:

- All JSON messages now include both `device_id` and `timestamp`
- Device ID is defined as a constant in sensor firmware (e.g., `#define DEVICE_ID "esp-001"`)

- Timestamp is generated based on NTP-synced time via Wi-Fi node