ESB-P: ESP-NOW SENSOR BROADCAST PROTOCOL DRAFT 1.1

Status : Experimental | Editors : Embedded Systems Team
Date : 2025-05-17 | Expires : 2026-05-17

---

0 Document Structure (Quick Guide for GPT Agents)
yaml
Copiar
Editar
§1 Introduction
§2 Conventions & Terminology
§3 Frame Format (generic, device-agnostic)
§4 Typical Frame Construction (step-by-step recipe)
§5 Receiver Behaviour & /devices JSON schema
§6 Project-Specific Frames
├─ 6.1 Reloj : FORM
├─ 6.2 Reloj : HR
├─ 6.3 GSR : VMEDIDO
├─ 6.4 Gafas : ACCEL
└─ 6.5 Pecho : ACCEL / FLEX
§7 Extensibility & Versioning
§8 Reference Source Code Pointers

---

1 Introduction
ESB-P defines one fixed-length ESP-NOW frame that every ESP32-class Sensor Device broadcasts.
A single Receiver ESP32 listens, de-multiplexes by deviceId + broadcastType, and makes the latest values available via HTTP/JSON.

2 Conventions & Terminology

Term	Meaning
deviceId	Null-terminated ASCII ≤ 15 chars identifying the board.
broadcastType	Null-terminated ASCII ≤ 15 chars describing the payload kind (“HR”, “ACCEL”…).
frame	Exactly 88 bytes transmitted with esp_now_send().
Sensor Device	Any ESP32 that sends ESB-P frames.
Receiver	The ESP32 that receives all ESB-P traffic and serves /devices JSON.

Channel: All nodes must share the same Wi-Fi channel (default 1).

3 Frame Format (generic)
c
Copiar

Editar
```
/* Total size: 88 bytes */
typedef struct  __attribute__((packed)) {
  char  deviceId[16];        // §2
  char  broadcastType[16];   // §2
  uint8_t payload[56];       // device-specific use
} espMessage;
```
Why 56 bytes? 16 + 16 + 56 = 88, which fits comfortably inside one ESP-NOW MAC payload (limit ≈ 250 B).

All devices send the same 88-byte struct – they just lay out the 56-byte payload differently.

4 Typical Frame Construction (Recipe)
Clear and identify

c
Copiar
Editar
```
espMessage msg = {};                    // zero-fill
strncpy(msg.deviceId, "MySensor01", 15);
strncpy(msg.broadcastType, "TEMP",   15);
```
Encode payload
Example: store a little-endian 32-bit float at byte 0.

c
Copiar
Editar
```
float T = 23.7f;
memcpy(&msg.payload[0], &T, sizeof(T));   // 4 bytes
```
Send

c
Copiar
Editar
```
const uint8_t bc[6] = {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};
esp_now_send(bc, (uint8_t*)&msg, sizeof(msg));
```
5 Receiver Behaviour & JSON Schema
Mode: WIFI_AP_STA ➜ SoftAP “ESP_Receiver” (channel 1) plus ESP-NOW STA.
Callback: On every frame, group by deviceId, then by broadcastType, store the latest payload.

5.1 /devices Response
json

Copiar
Editar

```
{
  "devices":[
    {
      "id":"Clock",
      "FORM": { "edad":"25","altura":"170","peso":"70", ... },
      "HR"  : 72,
      "ACCEL": { "ax":-12,"ay":34,"az":1000 }
    },
    {
      "id":"GSR",
      "VMEDIDO": 0.523
    }
  ]
}
```

Rules

Every top-level object == one unique deviceId.

Keys inside are broadcastType values.

Receiver may attach timestamps or extra meta-fields in future.

6 Project-Specific Frames

6.1 Reloj → “FORM”

Field	Offset	Bytes	Encoding	Example
edad	0	4	ASCII	"25"
altura	4	4	ASCII	"170"
peso	8	4	ASCII	"70"
vasos	12	4	ASCII	" 8"
hrs	16	4	ASCII	" 7"
nivel	20	16	ASCII	"Moderado"
padding		36	20	zero n/a

6.2 Reloj → “HR”

less
Copiar
Editar

```
payload[0..3] : int32  little-endian  (beats per minute)
payload[4..55]: zero padding
```

6.3 GSR → “VMEDIDO”

less
Copiar

Editar
payload[0..3] : 32-bit float (skin voltage, volts)
payload[4..55]: padding
6.4 Gafas → “ACCEL”
less
Copiar
Editar
payload[0..1] : int16 ax
payload[2..3] : int16 ay
payload[4..5] : int16 az
payload[6..55]: padding
6.5 Pecho → “ACCEL” & “FLEX”
ACCEL = same layout as 6.4.

FLEX : payload[0..1] = uint16 resistance (Ω); rest padding.

A device may broadcast multiple broadcastTypes back-to-back.

7 Extensibility & Versioning
A new sensor only needs to pick:

ini
Copiar
Editar
deviceId   = "YourBoardName"
broadcastType = "NEWTYPE"
payload   = your ≤56-byte layout
Receiver must add a case for "NEWTYPE" if it wants to parse it; otherwise it can store the raw 56-byte blob.

Future protocol versions may add uint8_t version as the first byte of the frame; receivers should ignore unknown versions safely.

8 Reference Source Code
Component File Notes
Clock (form + HR) ClockDevice.ino LVGL UI, periodic HR, sends "FORM" & "HR"
GSR probe GSR.ino Reads ADC, sends "VMEDIDO"
Gafas accel Gafas.ino Reads MPU-6050, sends "ACCEL"
Pecho accel/flex Pecho.ino Two frames: "ACCEL" & "FLEX"
Receiver ReceiverAP.ino SoftAP+ESP-NOW, serves /devices JSON

All sketches compile under Arduino-ESP32 core ≥ 2.0.12.

End of ESB-P v1.1