Firmware BNO085 – Transmissor ESP-NOW

Visão Geral

`esp32s3_codes/BNO085/BNO085.ino` roda em um ESP32 (ou ESP32-S3) dedicado à leitura do IMU **SparkFun BNO08x**. Ele obtém quaternions + ângulos de Euler diretamente do sensor, aplica offsets de zero (recalibração por software) e envia os dados para o ESP32-S3 principal (controle PID) por ESP-NOW. A telemetria inclui roll/pitch/yaw compensados e o quaternion completo, permitindo que o backend escolha qual representação usar.

Fluxo

1. **Inicialização**
   - Configura I2C (`Wire.begin(21, 22)`, 400 kHz).
   - Inicializa `BNO08x` no endereço padrão `0x4B` e habilita o *Rotation Vector* a cada 50 ms (~20 Hz).
   - Configura ESP-NOW em modo `WIFI_STA`, adiciona o peer do receptor (`receiverMac[]`) e registra o callback `onDataRecv`.
2. **Loop**
   - Aguarda eventos do BNO08x com `imu.getSensorEvent()`.
   - Lê os quaternions (`quatI/J/K/Real`) e converte para Euler com `quaternionToEuler`.
   - Aplica offsets (`offsetRoll/Pitch/Yaw`). Offsets são atualizados quando:
     - É a primeira leitura (`temOffset == false`).
     - Chega um comando `recalibra` via ESP-NOW (flag `pedidoRecalibra`).
   - Preenche o struct `TelemetryData` com Euler + quaternions e envia a cada `sendIntervalMs` (50 ms) utilizando `esp_now_send`.

Estruturas ESP-NOW

```
typedef struct {
  float roll, pitch, yaw;
  bool  recalibra;   // comando vindo do S3
} CommandData;

typedef struct {
  float roll, pitch, yaw;   // já compensados
  float qw, qx, qy, qz;     // quaternion bruto
  bool  recalibra;          // sempre false (mantido por compatibilidade)
} TelemetryData;
```

- **Recepção**: `onDataRecv` copia o `CommandData`. Se `recalibra == true`, levanta a flag `pedidoRecalibra`, tratada no `loop`.
- **Envio**: `ori` (TelemetryData) é atualizado com os ângulos compensados antes de chamar `esp_now_send`.

Recalibração

- Quando o controlador marca “Recalibrar” (página `accelerometer.html`), o backend envia `recalibra` para o firmware PID, que repassa o comando via ESP-NOW ao transmissor BNO.
- Este firmware apenas “zera” os offsets atuais (armazenados em `offsetRoll/Pitch/Yaw`), sem precisar recalibrar o sensor fisicamente, garantindo resposta rápida.

Configurações Importantes

Constante	Descrição
`receiverMac[]`	MAC do ESP32-S3 principal (ajuste conforme o hardware).
`sendIntervalMs`	Intervalo de envio (50 ms ≈ 20 Hz).
`BNO08x_DEFAULT_ADDRESS`	Troque para `0x4A` se o breakout estiver configurado nesse endereço.
Pinos I2C	`Wire.begin(21, 22)` – altere conforme seu hardware.

Integração com o Backend

- O backend (`accelerometer.js` ou `kinematics.js`) consome os dados convertidos via `/ws/telemetry`. Quando o firmare PID encaminha uma request `recalibra`, ela chega até este transmissor, reiniciando a referência.
- A disponibilidade do quaternion permite que o frontend exiba tanto a forma “roll/pitch/yaw” quanto `q = w + xi + yj + zk`.

Testes Rápidos

1. Compile/flash `BNO085.ino` com a biblioteca `SparkFun_BNO08x_Arduino_Library` (v1.0.6).
2. Abra o Serial Monitor (115200) e verifique:
   - MAC do transmissor.
   - Mensagem “BNO08x configurado!”.
   - Log “ESP-NOW pronto…”.
3. Mova o sensor – o Serial exibirá os ângulos enviados, e o firmware PID deverá começar a receber quaternions (observe a telemetria em `actuators.html` ou `accelerometer.html`).

Arquivos Relacionados

- PID-CONTROL-FILTER-SPIKE-BNO.md – descreve como o receptores utiliza esta telemetria.
- ACCELEROMETER-README.md – mostra a interface que consome os dados do BNO085.