# 0 导读

这是一份面向硬件初学者的 **Raspberry Pi 5 ↔ Adafruit BNO085 9-DoF IMU** 全流程指南，目标是 **零烧毁、一次点亮、轻松调试**。内容包含：

- 线路及电气安全设计
- 上电前的逐项检查
- 树莓派 OS 下的软件启用与库安装
- Python 示例与实时数据验证
- 常见故障排查
- INT / RST / SPI / UART 高级扩展

如无特殊说明，所有电压/引脚均以 **3.3 V 逻辑** 为基准。

---

# 1 硬件概览与关键参数

| 模块 | 特性 | 关键电气限制 |
|---|---|---|
| **Raspberry Pi 5** | 与历代 40-pin 排针兼容；I²C-1 默认位于 GPIO2 (SDA) & GPIO3 (SCL)。 | **GPIO 绝对最大 3.6 V**, 典型高电平 ≥ 2.0 V（当 VGPIO_VREF = 3.3 V）<br>filecite turn7file0 |
| **Adafruit BNO085 Breakout** | 板载 LDO：VIN = 3-5 V → 3.3 V；I²C/SPI/UART 三合一双向电平转换；10 kΩ 上拉至 VIN。默认两颗跳线 P0/P1 = LOW → I²C 模式。 | I²C 引脚自带 10 kΩ 上拉至 VIN；若 VIN = 5 V，则 SDA/SCL 会被拉到 5 V filecite turn7file4 |

**要点**：为避免 5 V 上拉损毁 Pi GPIO，**务必让 VIN = 3.3 V**。（下文有可选 5 V 方案及隔离做法。）

---

# 2 接线总表（I²C 模式）

| # | Pi 5 物理引脚 | Pi BCM 名称 | 线色建议 | 接至 BNO085 引脚 | 说明 |
|---|---|---|---|---|---|
| 1 | 3V3 Power | – | **红** | **VIN** | 为板供电并决定上拉电平 |
| 6 | GND | – | **黑** | **GND** | 共地 |
| 3 | GPIO2 | SDA1 | **蓝** | **SDA** | I²C 数据，开漏 |
| 5 | GPIO3 | SCL1 | **黄** | **SCL** | I²C 时钟，开漏 |
| – | (可选) 任意空闲 GPIO | – | 绿 | **INT** | 中断（下降沿） |
| – | (可选) 任意空闲 GPIO | – | 紫 | **RST** | 低电平复位 |

*P0/P1* 无需连线（默认低电平保持 $I^2C$）。若需改地址，将 ***DI*** 引脚拉至 *3.3 V* → 地址变为 *0x4B* *filecite* *turn7file1* 。

---

## 3 电气安全核算

- 并联上拉：1.8 kΩ（Pi 内部） // 10 kΩ（板上）≈ **1.5 kΩ**；高电平电流 ≈ 3.3 V / 1.5 kΩ ≈ 2.2 mA，远低于 Pi 单 GPIO 8 mA 驱动限值 filecite turn7file15。
- GPIO 输入高阈值 2.0 V（@3.3 V Rail），低阈值 0.8 V：10 kΩ 上拉保证高电平 ≥ 3.0 V，余量充足 filecite turn7file0。
- **绝对禁用**：VIN=5 V 且直接连 SDA/SCL → 上拉到 5 V ➜ 过压烧毁（3.6 V 限制）。若必须 5 V 供电：① 断开或拆除板上 10 kΩ；② 使用专用 $I^2C$ 隔离/电平转换器。
- 线材：数据线 < 30 cm；电源线 ≥ 22 AWG；所有操作佩戴防静电腕带，湿度 40–60 %。

---

## 4 上电前 Checklist

1. **断电** 插线——不要热插拔。
2. 用万用表确认 Pi 3V3 与 VIN 间 = 3.3 V；GND 连通。
3. 目测 SDA/SCL 没接反且无焊锡桥。
4. 若使用 INT/RST，确保其空闲电平为 **高**（内部上拉或外部 10 kΩ）。

完成后可通电。

---

## 5 树莓派侧软件配置

```
# 5.1 启用 I²C-1
a. sudo raspi-config  →  Interface Options → I2C → Enable
# 或修改 /boot/firmware/config.txt 中加入
dtparam=i2c_arm=on

# 5.2 装工具与库
sudo apt update
sudo apt install -y i2c-tools python3-pip
pip3 install --upgrade adafruit-blinka adafruit-circuitpython-bno08x

# 5.3 验证设备
sudo i2cdetect -y 1   # 预期看到 0x4a 或 0x4b
```

**遇不到地址？** 立即断电检查线序与 VIN 电压。

### 5.4 I²C 速率设置

- 默认 100 kHz 通常可用。
- 如出现 “remote I/O error”/clock-stretch 超时，将以下行加入 `/boot/firmware/config.txt`：

```
dtparam=i2c_arm_baudrate=400000
```

- BNO085 I²C 实现存在时序偏差；多数 RP2040 / BCM2712 主机在 400 kHz 稳定 filecite turn7file7。

---

## 6 Python 快速测试脚本

```python
import board, adafruit_bno08x
from adafruit_bno08x import BNO_REPORT_ROTATION_VECTOR

i2c = board.I2C()              # 自动使用 /dev/i2c-1
bno = adafruit_bno08x.BNO08X_I2C(i2c, address=0x4A)

bno.enable_feature(BNO_REPORT_ROTATION_VECTOR)
print("Quaternion   w      x      y      z")
while True:
    w, x, y, z = bno.quaternion
    print(f"{w:8.4f} {x:7.4f} {y:7.4f} {z:7.4f}")
```

终端应每 \~25 ms 打印一行四元数。

---

## 7 常见故障排查表

| 现象 | 可能原因 | 对策 |
|---|---|---|
| `i2cdetect` 无地址 | 线序错误 / VIN 无 3.3 V | 断电核线，测 VIN 电压 |
| Python 报 `ValueError: No I2C device` | I²C 启用但设备未应答 | 同上；检查 DI 地址跳线 |
| 数据间歇冻结 | 线长、接触不良、速率不当 | ❶ 压紧杜邦线；❷ 调至 400 kHz；❸ 检查 INT 正常跳变 |
| 总线挂死、需复位 | 时序冲突 | 在代码中监测异常后拉低 RST 100 µs |

---

## 8 高级接口与功能

- **INT 引脚**：低电平表明 FIFO 有新数据。配合 `GPIO.poll()` 可实现低功耗读取。
- **RST 引脚**：软件异常时可由任意 GPIO 拉低 > 100 µs 复位；上电应保持高。
- **SPI 模式**：P0=HIGH、P1=HIGH；需额外接 CS / INT / RST。SPI 最大 8 MHz，最可靠，无 clock-stretch 问题 filecite turn7file9。
- **UART-RVC 模式**：P0=LOW、P1=HIGH；用 TX/RX 两线，可输出航向与加速度文字帧；波特率固定 115200。

---

## 9 附录 A — 上拉电阻计算

```
R_effective = (1 / (1/1.8 kΩ + 1/10 kΩ)) ≈ 1.53 kΩ
I_pull = 3.3 V / 1.53 kΩ ≈ 2.16 mA
```

满足 $I^2C$ Fast-mode (400 kHz) 对 400 pF 总线最大 3 mA 拉电流要求。

## 10 附录 B — 参考文档

- Raspberry Pi Compute Module 5 Datasheet – GPIO 绝对最大值 3.6 V filecite turn7file0
- Adafruit 9-DoF Orientation IMU Breakout BNO085 PDF – 板载 10 kΩ 上拉与 IO 描述 filecite turn7file4
- 同文档 – $I^2C$ 地址跳线 DI filecite turn7file1
- Adafruit 文档 – BNO085 $I^2C$ 协议注意事项 filecite turn7file7
- Raspberry Pi GPIO DC 特性 – 8 mA 驱动电流表 filecite turn7file15

---

**到此，一线一步均已覆盖。祝您调试顺利，传感器再也不用担心被"烟放"！**