**1. ADXL345 (Accelerometer)**

**1.1 Supply & I/O Voltage**

* **Supply Voltage (VS):** 2.0–3.6 V
* **I/O Voltage (VDD I/O):** 1.7 V to VS
* On STM32, we typically power the ADXL345 with **3.3 V** from the Nucleo board.

**1.2 Communication Interface**

* Supports **I2C** (7-bit address) **or** **SPI** (3-wire or 4-wire).
* If using **I2C**, typical 7-bit address can be **0x53** (when SDO/ALT ADDRESS = GND) or **0x1D** (when SDO/ALT ADDRESS = VDD I/O).
* For **SPI**, we’ll configure up to 5 MHz clock (the datasheet recommends max 5 MHz for stable communication).

**1.3 Data Rates & Bandwidth**

* Output Data Rate (ODR) can be set from **0.1 Hz up to 3200 Hz**.
* We configure this in the **BW\_RATE** register.
* Higher ODR = higher power consumption.

**1.4 Resolution & Range**

* We can choose ±2 g, ±4 g, ±8 g, or ±16 g.
* “Full resolution” mode can give up to **13-bit** data at ±16 g.
* For typical rocket or high-dynamic usage, we might pick ±16 g to avoid saturations.

**1.5 Interrupts**

* ADXL345 has two interrupt pins (INT1, INT2).
* We can enable **Data Ready**, **Tap**, **Free-Fall**, or **Activity** interrupts, etc.
* If we want to use an interrupt line on the STM32, we’ll configure a GPIO input in **EXTI** mode.

**1.6 .ioc Configuration Tips**

* If using **I2C**:
  + Pick an I2C peripheral (e.g., I2C1) at **100 kHz** or **400 kHz**.
  + In CubeMX, enable I2C on pins that connect to ADXL345 SDA, SCL.
  + Make sure to set the correct slave address (0x53 or 0x1D).
* If using **SPI**:
  + Choose a dedicated SPI peripheral (e.g., SPI1).
  + Configure up to 5 MHz clock, CPOL=1, CPHA=1 is typical for ADXL345 (datasheet recommends mode 3).
  + One Chip Select (CS) pin to ADXL345’s CS.

**2. NEO-6M (GPS)**

**2.1 Supply & I/O Voltage**

* Typically operates at **3.0–3.6 V** (3.3 V is standard).
* On the Nucleo-F446RE, we’d supply 3.3 V to VCC, and ensure the serial pins are also 3.3 V.

**2.2 Communication Interface**

* Most common interface is **UART** (TX/RX).
* Default baud rate often **9600 or 38400** bps, but can be reconfigured.
* The module can also support SPI or DDC (I2C-like), but the **UART** is by far the most typical for NEO-6M.

**2.3 GPS Data Format**

* By default, outputs **NMEA** sentences (ASCII) once per second (1 Hz).
* We can also enable **UBX** binary protocol for advanced configuration.

**2.4 Pinout Highlights**

* **TX** → wer MCU’s **RX**.
* **RX** → wer MCU’s **TX**.
* **GND** → Nucleo GND.
* **VCC** → 3.3 V from Nucleo.

**2.5 .ioc Configuration Tips**

* Enable one of the Nucleo’s **UART** peripherals (e.g., USART2) in CubeMX.
* Set baud rate to match the NEO-6M default (e.g., 9600 bps).
* Optionally enable **DMA** for receive if we want continuous reading without blocking the CPU.

**2.6 Antenna & PPS**

* The module often has a **PPS** (pulse per second) pin that toggles once per second for time synchronization.
* If we need that for Kalman or timing, configure a GPIO input in the STM32 (EXTI or polling).

**3. DPS310 (Barometric Sensor)**

**3.1 Supply & I/O Voltage**

* **VDD:** 1.7–3.6 V
* **VDDIO:** 1.2–3.6 V
* On Nucleo, typically both get **3.3 V**.

**3.2 Communication Interface**

* Supports **I2C** or **SPI**.
* For **I2C**, default address is **0x77** (7-bit) if SDO = VDD; or **0x76** if SDO = GND (check wer breakout or datasheet).
* For **SPI**, typical max frequency ~8 MHz (datasheet says up to ~10 MHz but recommended ~8 MHz for stable comms).

**3.3 Measurement & ODR**

* Pressure range: **300–1200 hPa**.
* Configurable oversampling (up to 128×) for better resolution vs. speed trade-off.
* The sensor can measure temperature too.

**3.4 FIFO & Interrupts**

* Up to **32-level FIFO** to buffer measurements.
* Interrupt pin can be used to signal data ready or FIFO full.

**3.5 .ioc Configuration Tips**

* For **I2C**: same approach as ADXL345.
  + Pick the same or different I2C bus. We can put ADXL345 and DPS310 on the same I2C bus if addresses don’t clash.
  + Set address (0x76 or 0x77).
* For **SPI**: same approach as ADXL345 (choose a separate Chip Select if we share the same SPI bus).

**4. STM32 NUCLEO-F446RE Board**

**4.1 MCU Overview**

* **STM32F446RE** has an ARM Cortex-M4 core @ up to 180 MHz.
* Has multiple **I2C**, **SPI**, **UART**, and **DMA** channels.

**4.2 Power Rails**

* The Nucleo typically provides **3.3 V** on the morpho and Arduino headers.
* We can also get 5 V from USB input, but for these sensors, 3.3 V is standard.

**4.3 Pin Assignments**

* **Arduino UNO–like connectors** (CN7, CN10) or **ST morpho** connectors for extra pins.
* Common I2C pins (Arduino style) are **PB7 (SDA)** and **PB6 (SCL)** or we can route them differently in the .ioc.
* Common UART pins are usually on **PA2 (TX)**, **PA3 (RX)** for USART2 if we want to use the onboard ST-Link Virtual COM Port. Or we can use other UART blocks (USART1, USART3, etc.) for the GPS module.

**4.4 CubeMX (IOC) Configuration Steps**

1. **Enable desired peripherals**:
   * For ADXL345 & DPS310 on I2C: Enable e.g. **I2C1** at 400 kHz (Fast Mode).
   * For NEO-6M on UART: Enable e.g. **USART2** at 9600 bps.
2. **Configure DMA** (optional but recommended) for:
   * ADC or other tasks.
   * If we want GPS data via DMA, set “DMA for Rx” in the USART config.
3. **Pin Remapping**: The F446RE has multiple alternate functions for each pin. Ensure the .ioc pin assignments match the physical wiring.
4. **Clock Configuration**: Make sure the bus frequencies support the I2C or SPI speeds we desire.

**4.5 Debug / Serial Output**

* Typically, the Nucleo board’s **ST-Link** interface is on **USART2** (PA2, PA3).
* If we want to watch debug logs while reading GPS on the same UART, we’ll need a second UART or reroute the GPS to a different peripheral (like UART4).

**4.6 Practical Pin Usage Example**

* **I2C1**:
  + PB6 → SCL
  + PB7 → SDA
  + Connect ADXL345 and DPS310 in parallel with different addresses.
  + Pull-ups ~4.7 kΩ on SCL, SDA.
* **USART2** (Hardware ST-Link Virtual COM) for debug or if not used, we can put GPS here.
* **UART4** or **USART3** for NEO-6M if we want to keep ST-Link on USART2 for debugging.

**5. Putting It All Together**

1. **Power**
   * All sensors can run at 3.3 V from the Nucleo.
   * Double-check current consumption to ensure the 3.3 V rail can supply it (usually no issue for these low-power sensors).
2. **I2C or SPI Choice**
   * ADXL345 and DPS310 both support I2C/SPI.
   * Typically, it’s simpler to put them on a single I2C bus with different addresses.
   * If we want maximum speed or are worried about I2C bus congestion, we could use separate SPI lines.
3. **GPS**
   * Easiest to connect the NEO-6M via **UART** (TX ↔ RX).
   * Check the default baud rate (often 9600 or 38400).
   * If using the same UART as ST-Link Virtual COM, we lose easy debugging. So consider a second UART.
4. **FreeRTOS**
   * We’ll likely create tasks for each sensor (e.g., an “IMU task,” a “Barometer task,” a “GPS task”).
   * Or one combined “Sensor task” reading all data, then queueing it to a “Kalman filter task.”
5. **Interrupts**
   * If we want Data Ready interrupts from ADXL345 or DPS310, define a GPIO input in EXTI mode in the .ioc.
   * The GPS can also provide a PPS interrupt if we want precise timing.