Mini-Assessment: Hardware–Software Synchronization Task (IMU + LiDAR)

Real-World Scenario: Warehouse Mobile Robot Sensor Bring-Up

You are part of a robotics team developing an indoor autonomous mobile robot used for **warehouse inventory inspection**.
Before navigation, SLAM, or obstacle avoidance can work, the robot must first have correctly synchronized and calibrated sensor inputs.

Your responsibility in this mini-assessment is to verify that the robot's **IMU** and **LiDAR** are correctly connected and that ROS2 can **fuse their data** to estimate robot motion.

This is the same type of task done by robotics engineers when validating hardware during the early development stages of a real robot.

1. Objective

Validate hardware software integration of **IMU + LiDAR** with ROS2 by:

1. Connecting and streaming IMU + LiDAR data
2. Ensuring timestamps and TF transforms are correct
3. Running a **basic data fusion pipeline** (IMU + LiDAR odometry)
4. Visualizing the fused motion in RViz2

This represents a real industrial milestone called:
"Sensor Bring-up & Fusion Verification"

2. Task Description

Part 1 : Hardware Integration

IMU (e.g., MPU6050, BNO055, MPU9250)

You must:

- Connect the IMU to Arduino over I²C

- Publish IMU data through a ROS2 node using serial communication
- Provide the following topics:
    - `/imu/data_raw`
    - `/imu/temperature` (optional)

LiDAR (RPLiDAR)

You must:

- Connect LiDAR via USB
- Install its ROS2 driver
- Publish:
    - `/scan`

Verification

Run:

```
ros2 topic list
ros2 topic hz /imu/data_raw
ros2 topic hz /scan
```

The two sensors must run **simultaneously without crashing** or timing delays.

Part 2 : TF Tree Setup

You must create a minimal TF transform tree:

```
base_link → imu_link
base_link → laser_frame
```

This TF tree is required for any sensor fusion.

You can use:

- `<node pkg="tf2_ros" exec="static_transform_publisher" ... >`
- Or create a small Python node publishing static transforms.

Part 3 : Time Synchronization

Sensors must publish **meaningful timestamps** using:

- `builtin_interfaces/msg/Time`

You must check:

- IMU timestamp increments correctly with each read
- LiDAR timestamp matches real time
- No sensor publishes messages older than 200 ms
- No drift when both run together

This is important for real robots because non-synchronized sensors produce wrong odometry and unstable SLAM.

Part 4 : Data Fusion Pipeline

Since you have **no camera**, your fusion pipeline focuses on:

Minimum fusion requirement

Use **robot_localization** to fuse:

Sensor	Data Used
IMU	orientation (quaternion), angular velocity, linear acceleration
LiDAR (optional)**	lidar-based odometry (if you have RPLidar + SLAM or scan matcher running)

If LiDAR does not provide odometry, fuse IMU only and visualize orientation.

Fusion Node Output

The system must produce:

- `/odometry/filtered`
- TF: `odom → base_link`

This confirms that the robot can estimate motion.

Part 5 : Visualization in RViz2

You must create an RViz2 config that shows:

Required Displays

- **Laser Scan** (`/scan`)
- **IMU Orientation** (`/imu/data_raw`)
- **TF Tree**
- **Filtered Odometry** (path)
- Axes on:
    - `base_link`
    - `laser_frame`

    - `imu_link`

Final Test

Move the robot (or the IMU+LiDAR board) by hand:

You should observe:

- LiDAR scan rotates correctly as you rotate the device
- IMU orientation updates smoothly
- Odometry path updates in real time

This proves hardware–software synchronization.

3. Deliverables

You must submit:

✔ **ROS2 workspace** (src folder zipped)
✔ **Launch file** starting IMU, LiDAR, static transforms, and fusion
✔ **YAML fusion configuration**
✔ **RViz2 config file**