Technical Challenge: Implementation of a driver for 3-Axis Accelerometer Module

Objective:

Develop a driver for a **3-axis Accelerometer (ADXL343)** using the **I2C** communication protocol. The driver should be able to initialize the device, read acceleration data along the X, Y, and Z axes, and configure the accelerometer's range and resolution. Include unit tests to validate the functionality of your driver.

Requirements:

Language: The driver must be written in C or C++.
Communication Protocol: Use the I2C interface for communication between the microcontroller and the accelerometer.
Device Initialization:

- Implement a function to initialize the accelerometer with default settings.
- The initialization function should configure the accelerometer's range and resolution to predefined values.

Data Reading:

- Implement functions to read acceleration data from the X, Y, and Z axes.
- Include error handling for communication failures.

Configuration:

- Implement functions that allow changing the accelerometer's range (e.g., ±2g, ±4g, ±8g, ±16g) and resolution.
- Ensure these settings can be modified at runtime.

Documentation:

- Include comments and documentation in your code to explain your implementation.
- Provide a README file with:
    - Instructions on how to build and run your code.
    - A brief explanation of your approach to designing the driver and tests.
    - Any assumptions made or limitations of your implementation.

Unit Testing (optional):

- Mock the I2C communication to simulate the accelerometer's responses.
- Provide a few test cases covering correct data reading, error handling, and runtime configuration changes.
- Recommendation: Google Test (gtest) for C++ or Unity/Ceedling for C.

Additional information:

Please keep the following points in mind to guide your development and submission process:

- **ADXL343 Datasheet**: The datasheet for the ADXL343 is provided as part of the challenge materials. It contains all necessary technical information about the accelerometer, including pin configurations, register maps, and recommended initialization settings. Reviewing the datasheet will be crucial for successfully implementing the driver.
- **Mocking Hardware Communications**: Since there is no physical hardware available for this challenge, you will use the provided `i2c_write(char* data, size_t length, uint32_t timeout)` and `i2c_read(char* data, size_t length, uint32_t timeout)` functions to simulate communication with the ADXL343 sensor over the I2C interface. You are encouraged to mock these functions in your unit tests to simulate sensor responses and validate the behavior of your driver under various conditions.
- **Time Investment**: We appreciate your interest and efforts in participating in this challenge. To respect your time, you're encouraged to spend no more than a few hours per day on this task. This limitation is designed to focus on the core aspects of driver development and your problem-solving skills rather than exhaustive feature implementation.
- **Submission Guidelines**: You can submit your completed challenge in one of two ways:
    - Git (Preferred): If you're comfortable using Git, we prefer that you submit your work as a Git repository. This approach allows us to review your commit history and better understand your development process. Please provide a link to a public repository (e.g., GitHub, GitLab, Bitbucket) where your work can be accessed.
    - Zip File: If you're not familiar with Git or prefer not to use it, you can submit your work as a zip file. Please ensure your zip file includes all source code, unit tests, and any documentation or README files that describe your implementation and how to run your tests.

Crack open the datasheet, warm up your coding fingers, and may your logic be as solid as your humor. Dive into this challenge and let's see if you can make this accelerometer spin faster with your skills than with gravity. Happy coding, and remember, it's not just about the bytes, it's about the laughs along the way!