Title: Design a testing interface for a sample IMU

Objective:

To demonstrate hands-on knowledge of coding and integration using Python/C++/Rust, and your ability to understand, design, and build systems involving software integrated with hardware.

Overview:

You are to design and implement a simple system that involves a hardware data stub communicating with a software application. The hardware device could be a simple sensor that can is to be simulated.

Tasks:

Part 1: Hardware Data stub (10 points)

Design a simulation that acts as a data stub for an STIM300. This stub will expose a serial port that works at 115200 baud.

Implement the following as part of the hardware data stub:

1. "Normal Mode datagram" as specified in the section 5.5.6 of the datasheet. The stub should send a sample packet at a user defined frequency (in Hz) specified at the beginning of the simulator. This will be "auto mode". It should also send this information when queried using appropriate command structure specified in section 8 of the datasheet. This will be "normal mode".
2. "Serial Number datagram" as specified in the section 5.5.2 of the datasheet. The stub should send a sample serial number packet on start of the sim and echo it on the terminal. It should also send this information when queried using appropriate command structure specified in section 8 of the datasheet.

Part 2: Software Application (30 points)

Design and implement a "testing interface" for testing the hardware simulator. The testing interface must

1. have a GUI that user can use to interact with the software.
2. Give the user to start the hardware simulator through the GUI, with an option to provide any initial data required by the simulator.
3. Give user a choice to run various tests. These are – "Test serial command", "Test normal mode", "Test auto mode". User must be able to provide a sequence and select the tests they want to run.
4. Give the user an interface that presents the information from the hardware simulator in a human readable format, that refreshes automatically when the data comes into the simulator.

Deliverables:

1. Link to the GitHub repository containing all the files.
2. The repo must contain a launch script with setup, run, and clean-up options.
3. A README file that explains how to run the code and how the system works, with instructions to set up the environment. Assume a clean Ubuntu 22.04 install.
4. A SYSTEM_DESIGN.md file that describes the design of the system, including a block diagram showing the main components of the system and how they interact.