FIRESTORM

Take-Home Software Exercise

Simple Android GCS application

Objective

You are asked to develop a simplified Ground Control Station (GCS) application prototype that allows a user to monitor and control a simulated uncrewed aerial system (UAS). This application should include both a backend written in C++ and a frontend using QML, using Qt framework capabilities. The application should be able to run, at least, on an Android emulator.

Requirements

- **Framework:** Qt 5.15.X or above.
- **Backend:** C++.
- **Frontend:** QML
- **Deployment:** The app should be deployable on an Android emulator (Android 10 or above).
- **Documentation:** Provide a README.md file with instructions on how to compile and run the application, along with explanations of the approaches taken.
- **Unit Tests:** Include unit tests for the backend code.

Exercise description

You are to develop a basic GCS application prototype that includes the following functionalities:

1. Dashboard View:

- Display the current status of the UAS (e.g., battery level, altitude, speed).
- Display a map showing the current location of the UAS.

2. Control Panel:

- Include basic controls to simulate actions such as takeoff, land, and return to home.

3. Telemetry simulation:

- Simulate the UAS telemetry data in the backend and update the frontend in real-time. The simulated data should include, but not limited to:
  - Battery level (percentage)
  - Altitude (meters)
  - Speed (meters/second)
  - GPS coordinates (latitude and longitude)
- Simulate the responses of the UAS to the commands in backend and update the frontend in real-time. The backend should respond to, at least, the following commands:

- Take-off.
- Go-To.
- Land.

Functional Requirements

1. Backend (C++/Qt):

- Implement a *TelemetrySimulator* class that generates random data for the UAS status.
- Use signals and slots to update the frontend with the simulated telemetry data.
- Implement basic command handling for takeoff, land, and Go-To actions.

2. Frontend (QML):

- Create a dashboard to display the UAS status.
- Use a map component to show the UAS location.
- Implement buttons for takeoff, land, and go-to actions.
- Update the dashboard in real-time as the location of the drone changes and telemetry data is received.

Instructions

1. Setup:

- Create a new Qt project using Qt Creator IDE.
- Organize the project structure following best practices (e.g., separate folders inside the src folder for backend and frontend code, a folder for tests, etc.).

2. Backend development:

- Create a *TelemetrySimulator* class in C++ that emits signals with telemetry data.
- Implement slots for handling takeoff, land, and go-to commands.
- Write unit tests for the *TelemetrySimulator* class.

3. Frontend development:

- Develop the QML interface to display telemetry data and a map.
- Bind the QML components to the backend using Qt's signal and slot mechanism.
- Implement the control buttons and connect them to the backend slots.

4. Deployment:

- Configure the project to build and deploy on an Android emulator.
- Provide detailed instructions in README.md on how to set up the environment, build, and run the application in the emulator.

5. Testing:

- Include at least one unit test for the *TelemetrySimulator* class.
- Provide instructions in the README.md on how to run the tests.

Deliverables

The solution should be sent to nuno@launchfirestorm.com, with careers@launchfirestorm.com in CC, in the space of a week after the reception of the email with these instructions, including:

- Source code of the GCS application in a .zip file, with the name format **<FirstName_LastName>_Mobile_App_TakeHome.zip** (unless otherwise this is a GitHub repo – check the Bonus Points section for details).
- README.md with detailed instructions on how to compile, run, and test the application, along with explanations of the approaches taken.

Evaluation Criteria

- **Code quality:** Clean, readable, and well-documented code.
- **Functionality:** The application should meet all the specified requirements, besides, of course, run and do what is expected. Applications that do not even run will not be considered for evaluation.
- **Architecture:** Proper use of OOP approaches, design patterns, modularity and Qt framework features, including signals and slots.
- **UI/UX:** Although simple, we look for an Intuitive and responsive user interface.
- **Documentation:** Clear and comprehensive instructions and comments. Doxygen-enabled documentation is welcomed.
- **Testing:** Presence and quality of unit tests.

Bonus Points

To distinguish your solution further and showcase your expertise, consider incorporating the following bonus points:

1. Integration of the MAVLink protocol:

- Add a MAVLinkInterface class to the backend to receive UDP data from a simulated drone outside the app.
- That simulated drone can be created in Python using pymavlink, but other solutions can be taken.
- The code should be updated so that if telemetry data is received from MAVLink, then the app should stop "listening" to the TelemetrySimulator and instead fetch telemetry data from the MAVLinkInterface.
- Provide documentation in the README.md on how to run the simulated drone script and the app.

2. Enhanced user interface:

- Improve the user interface with advanced QML features such as animations, transitions, and more detailed visualizations of UAS data.
- Include user feedback mechanisms such as status notifications and error messages.

3. GitHub repository:

- Create a private GitHub repository for the project, following the best practices for a project that should allow others to contribute to. Add a LICENSE file to it as well.
- Share, at least, read access to the repository to the **TSC21** username, and then the repository link to nuno@launchfirestorm.com, ensuring it includes all the source code, README.md, and any other relevant documentation.

4. Continuous Integration (CI) setup:

- Implement a CI pipeline using GitHub Actions or another CI tool.
- Configure automated builds and tests for the project .

5. Detailed documentation:

- Extend the README.md with a section on potential improvements and next steps for future development.

For any questions or clarifications, please reach out to nuno@launchfirestorm.com.
