Technical Test Task Description

Your task is to build a simple dashboard application to monitor the airspeed and altitude of a remotely operated aircraft.

The aircraft transmits telemetry data to a radio base station. The base station exposes the data in real time through a single TCP connection using the WebSocket protocol. In addition, the base station can also be used to transmit control messages to the aircraft. Following, is the specification of all currently supported telemetry data and control messages. All messages are encoded in JSON format.

Telemetry End-point

- ws://interview.dev.ctx.ef.com/telemetry

Telemetry Data

- **Airspeed:** {"airspeed": 0..MAX_SPEED}
- **Altitude:** {"altitude": 0..MAX_ALTITUDE}

Control Messages

- **Landing gear control message:** {"type": "landing_gear", "value": 0..1}
- **Flap adjustment control message:** {"type": "flaps", "value": 0..5}

Since the aircraft is operated by autopilot, these control messages are just a guideline and they may be ignored. You will not get a direct response to control messages.

You will, however, get a similar control message whenever the autopilot either accepts your control messages or when it decides to adjust values.

Important Notes and Considerations:

It’s quite possible that more telemetry data and control messages will be introduced in the future. Also, the base station manufacturer will be deploying a

new firmware within a few months. It is not yet clear what those changes will look like, but so far they've told us they might stop using the WebSocket protocol and they might even use different protocols and connection for control messages and telemetry data.

When the times comes, we will need to update to the new revision. The application therefore needs to be designed so that the impact of this future transition is minimal, so use the appropriate design methodologies to ensure a smooth and pain free transition.

Note: Due to high solar flare activity (but mostly because of a few programming errors), the base station may at times provide invalid data. In addition, disconnections may occur at any time.

Requirements

The user interface has already been specified and all the assets can be found here. The zip file comprises of the following components. They are ordered by priority (highest to lowest):

- **Numeric gauge displaying:** current, minimum, maximum and average speed (in knots)
- **Numeric gauge displaying:** current, minimum, maximum and average altitude (in feet)
- **Switch to extend/retract the landing gear.** The switch also shows the current landing
- **Gear position** (assume retracted by default)
- **Controls to adjust the flaps.** The control also shows the current setting for the flaps (assume 0 by default)
- **Base station connection state** (online/connecting/offline)

Bonus points for the following (entirely optional):

- Needle gauge displaying the instantaneous airspeed. The maximum registered speed is 420 kts (knots)
- Two needle gauge displaying the instantaneous altitude. Works very much like a clock. The small hand measures thousands of feet and the big hand measures hundreds of feet.
    - Example: Small hand pointing to 1 and big hand pointing to 2 should read: 1.200 feet (1 * 1000 + 2*100)

- A line graph plotting the airspeed and altitude changes over a period of five minutes. (requirements dictate that this interval may be user configurable in the future)
- The user interface should be updated in real time, responding to the telemetry data and control message updates.

Additional Considerations

You are free to use whatever frameworks you feel comfortable working with.
We are expecting a single static HTML page and all logic implemented using TypeScript / JavaScript.

Evaluation Criteria

This technical task helps us to evaluate your overall approach to software development, defensive programming and how you interpret requirements.

We don't want to leave you in the dark when it comes to how your task will be evaluated, so here's the evaluation criteria.

- Follows SOLID principles
- Programs to an interface
- Minimal coupling/dependencies between components
- Writes clear, consistent, easy to read and easy to follow code
- Follows good and consistent naming conventions
- Writes unit tests
- Instruments the application

Assuming you pass the test task, you can expect a brief discussion of your test task during the follow up interview along with some requirement changes and how you would implement them.

We hope you have fun implementing this application. We can't wait to see what you come up with.

Good luck!

References

Groundstation URI : interview.dev.ctx.ef.com/telemetry

http://en.wikipedia.org/wiki/WebSocket
http://www.websocket.org/echo.html
http://en.wikipedia.org/wiki/SOLID_(objectoriented_design)
http://www.fatagnus.com/programtoaninterfacenotanimplementation/