SWEN326 Project: Developing an Avionics System

Dr Stuart Marshall

Week 7 – 12

1 Introduction

This assessment is a group project to be completed by students during teaching weeks 7 – 12 of the course. Groups are to consist of 4 – 6 students each, and each group will be tasked with developing a safety-critical software system in the avionics domain.

Groups will need to create a list of requirements for the problem (stated below), design a solution to this problem, and then develop, analyse and test a prototype implementation of this solution. Much of the work will not actually *be in the coding of the prototype* but rather in the documentation and infrastructure that the teams must create to ensure the safety of their code, especially as you'll need to create some software to control the *simulated* hardware.

Following principles of safety-critical software development, your group must be able to trace requirements to design artifacts, code, and tests.

As well as this, you must adhere to coding standards during the development of the prototype.

You will not be assessed *directly* on the code base and documentation that your group ends up with, although these will need to be submitted (one submission per group is fine). Instead, students will be assessed on their critique of both the development approach taken and the documentation/code/tests created. This will be captured in a report that you will write and submit individually.

In six weeks you aren't going to be able to completely develop a truly safe system completely following the entire DO-178C process, and nor can we *teach* the entire standard in a single course alongside other topics. As such, this project is a learning experience on adding parts of a safety-critical approach into the processes you've been using to develop software for the past two+ years.

2 Problem Overview

The group has been tasked with developing a new Avionics Flight Management and Control System. The software will be responsible for managing flight plans and making corrections to the aircraft's control systems and control surfaces, as well as monitoring systems and providing a UI to the pilots.

It's worth noting the system described below doesn't specify absolutely everything that could be specified in a genuine real aircraft system. For example, there isn't as much detail on engine thrust as there may otherwise be. It's worth remembering that the purpose of this assignment is to give you experience developing in a safety-critical domain, and some of the actual specifics of the system can be assumed by groups as there is no real aircraft that the output of this assignment will be deployed to.

2.1 Asking for Clarification

Where there are any ambiguities below that groups want clarification on, groups can ask on the **Project One Discussion Forum** on Nuku during Weeks 7 and 8. Please do not send emails asking for clarification directly to the course coordinator - and use the forum instead.

3 The System

Note: measurements are in imperial units as these are standard across the industry in the Western world for these specific measurement categories.

3.1 Pilot User Interface Specifications

1. **Flight Plan Management:**
   - Input field for entering waypoints (latitude, longitude, altitude).
   - Input fields for speed restrictions and expected times of arrival at each waypoint.
   - A submit button to load and activate the flight plan.
   - Visual display (map) showing current position, planned route, and waypoints.
2. **Autopilot Control Panel:**
   - Buttons to engage/disengage autopilot.
   - Controls for manual override: altitude adjustment, speed, heading.
   - Indicator lights for autopilot status (engaged, disengaged, fault condition).
3. **Sensor Data Display:**
   - Digital readouts for airspeed, altitude, pitch, roll, yaw, and engine parameters.
   - Visual indicators for data update frequency (e.g., colour change or blinking to indicate fresh data).
4. **Hazard Alerts:**
   - Dedicated section of the interface for hazard warnings and mitigation actions.
   - Audible and visual alerts for immediate hazards.
   - A checklist or action plan for emergency procedures.

Note: the visual display map can be fairly basic in this project, and does not need features like zoom or animation. A 2D rectangular backdrop of the world with lines and points and text superimposed on top would be fine

3.2 Sensor Data Simulation and Frequency

1. **Airspeed Sensor:** Simulate data to represent the aircraft's speed relative to the surrounding air. Data format: knots (nautical miles per hour). Update frequency: every second.
2. **Altitude Sensor:** Simulate barometric and GPS altitude data. Data format: feet above mean sea level (AMSL). Update frequency: every 500 milliseconds.
3. **Attitude Sensor:** *(AKA: Artificial Horizon Indicator)* Simulate aircraft's orientation data, including pitch (nose up/down), roll (wing up/down), and yaw (nose left/right). Data format: degrees from the horizon for pitch and roll, magnetic heading for yaw. Update frequency: every 500 milliseconds.
4. **Engine Parameters:** Simulate engine thrust and fuel flow. Data format: thrust in pounds-force (lbf). Update frequency: every second. **Note: we could also consider issues such as temperature or fuel flow, but we will ignore these for this project due to time issues.**

Assume the airspeed sensor, altitude sensor and attitude sensor all operate in a 2oo3 redundant system architecture.

3.3 Control Signal Specifications

1. **Autopilot Control Frequency:** The system should send control signals to the aircraft's control surfaces (elevators, ailerons, rudders) and engine control systems at least every 500 milliseconds.

2. **Execution Check Parameters:** After sending a control signal, the system must verify the execution by reading back the relevant sensor data within 200 milliseconds. Success Criteria: A control signal is considered successfully executed if the sensor data reflects the expected change within a margin of error of $\pm 2\%$ for the control surfaces and $\pm 5\%$ for engine parameters, within 1 second of command issuance. Failure Handling: If the execution check fails, the system must attempt to resend the command up to three times before alerting the pilot to the issue via the user interface.

3.4 Sensor Data Simulation and Value Ranges

- **Airspeed Sensor:** Simulate data to represent the aircraft's speed between 50 to 500 knots. Define operational thresholds and conditions under which values exceed normal operational limits.

- **Altitude Sensor:** Simulate barometric and GPS altitude data from -1,000 to 50,000 feet AMSL, with thresholds for rapid changes indicating potential sensor faults.

- **Attitude Sensor:** Simulate orientation data within typical operational ranges: Pitch: -30 degrees to 30 degrees, Roll: -60 degrees to 60 degrees, Yaw: -180 degrees to 180 degrees. Identify and manage exceedances through system alerts and corrective actions.

Hazard Mitigation Strategies: Implement fault detection, fault tolerance, and fail-safe mechanisms to handle abnormal sensor values, ensuring the system can safely manage potential hazards without catastrophic failures.

3.5 Engine Thrust and Flight Dynamics

Engine thrust is the force generated by the aircraft's engines to propel it forward. This force is critical for achieving and maintaining the aircraft's velocity during various phases of flight. Thrust is produced by accelerating a mass of air or gas to the rear, propelling the aircraft in the opposite direction according to Newton's third law of motion.

3.5.1 Interconnection with Flight Dynamics

Engine thrust plays a vital role in the dynamics of flight, influencing several key parameters:

- **Speed Control:**
  - Thrust is directly proportional to the aircraft's airspeed. An increase in thrust results in an increase in speed, whereas a decrease in thrust lowers the speed, assuming other factors such as air density and drag do not change.
  - Simulation of airspeed should dynamically respond to changes in thrust, especially in scenarios involving speed adjustments to meet flight plan targets or respond to air traffic control.

- **Altitude Adjustment:**
  - Thrust adjustments are necessary for changing altitude. To climb, an aircraft typically increases its thrust; to descend, it reduces thrust.
  - The interplay between speed and pitch (influenced by thrust) is crucial for maintaining or changing altitude, which must be accurately modeled in simulations.

- **Attitude Management:**
  - Changes in thrust can alter the aircraft's pitch—increased thrust can cause the nose to rise (pitch up) and decreased thrust can cause it to lower (pitch down).

  - While roll and yaw are primarily controlled by other surfaces (ailerons and rudders), asymmetric thrust (such as in an engine failure scenario) can significantly impact these attitudes, necessitating simulation scenarios that involve thrust differentials.

3.5.2 Common Aircraft and Their Thrust Levels

The maximum and minimum thrust levels for typical international passenger aircraft vary depending on the aircraft model, engine type, and operating conditions. To provide some context to the somewhat ambiguous issue of *engine thrust* discussed above, consider the following four aircraft which are fairly common on international routes in 2024.

- **Boeing 737-800**
  - *Engine Type:* CFM56-7B
  - *Maximum Thrust:* Approximately 121 to 130 kN per engine
  - *Minimum Thrust:* At idle, thrust can drop to about 5-10% of maximum, depending on specific conditions such as altitude and temperature.
- **Airbus A320**
  - *Engine Types:* CFM56 or V2500
  - *Maximum Thrust:* Ranges from about 98 kN to 120 kN per engine, depending on the engine model.
  - *Minimum Thrust:* At idle, similar to the Boeing 737, thrust typically is around 5-10% of the maximum.
- **Boeing 777-300ER**
  - *Engine Type:* GE90-115B
  - *Maximum Thrust:* Up to 512 kN per engine, one of the most powerful jet engines in commercial service.
  - *Minimum Thrust:* At idle, thrust levels can be around 10-20 kN, depending on operational conditions.
- **Airbus A350**
  - *Engine Type:* Rolls-Royce Trent XWB
  - *Maximum Thrust:* Around 374 kN to 430 kN per engine, depending on the model.
  - *Minimum Thrust:* Idle thrust can range from 10-20 kN.

3.5.3 General Considerations for Thrust Levels

- **Maximum Thrust:** This is used during critical phases such as takeoff and certain climbing operations. The exact value depends on the engine's capacity and the aircraft's weight requirements. **Note: you are free to assume a value and state that assumption in this project.**
- **Minimum Thrust:** Typically observed during idle on the ground or during descent. It's the lowest thrust level at which the engine can operate without shutting down.
- **Cruise Thrust:** Notably lower than maximum thrust, often around 15-30% of the maximum, depending on cruise altitude and aircraft load. **Note: for this project you do not need to consider aircraft load.**

4 Forming Teams

We will use the ECS team signup tool: `https://ecs.wgtn.ac.nz/cgi-bin/teamsignup`. The system will be open on Monday, April 22nd and Tuesday, April 23rd.

Students can self-organise in teams of between 4 – 6 students. Students may submit small groups of 1, 2 or 3 students, however, these small groups will be merged with other small groups to form groups of 4 – 6 students on the morning of Wednesday, April 24th. The final group list will be released at or directly after the lecture on Wednesday, April 24th.

It is not compulsory to do the group project. All students who wish to attempt the group project need to register in the team signup system to signal their intent to do the project. Even if you don't currently have a group, register as a group of one, and you will be added to a group on Wednesday, 24th April. Students who have not registered in the team signup system will be assumed to not attempting the group project and will get a mark of 0% for this assignment. There is no mandatory course requirement to get any marks in this assessment, and you may still pass the course by getting 50% or more of the overall course marks from the teaching-weeks test (which contributes 54%) and the assessment-period test (which contributes 16%).

5 Resources

5.1 Time

Time is your greatest resource in this project, as it is for many others. The assumption is that each individual will commit six hours per week for six weeks to this project, although note that the final week will best be spent working on the individual report submission, so groups should aim to have all of the group work completed by the end of week 11 at the latest.

5.2 AI

You may use AI however you deem fit, however, note that you are responsible for ensuring that the output of the AI tool is correct, and you should cite the tool used and the prompts used in a report appendix.

5.3 Languages

You may attempt this project in either Java or C. The relevant standard is based on the Power of Ten rules, written up at: `https://ieeexplore.ieee.org/document/1642624`. We covered these rules in the labs in week 2, and we will repeat this lab in week 7 as well for those who missed the lab in week 2.

David Pearce wrote up a version of these rules for Java, and this version can be found at: `https://nuku.wgtn.ac.nz/courses/19441/files?preview=2501635`

6 Software Development Process Under DO-178C

The development of a safety-critical avionics system under the DO-178C standard involves several key phases, each critical for ensuring the safety and reliability of the final product:

1. **Requirement Traceability:** Each software requirement must be explicitly linked to design artefacts, code implementations, and corresponding test cases. This ensures that all requirements are accounted for throughout the development process.

2. **Software Development Processes:**

   - **Planning Phase:** Develop comprehensive plans for software requirements, verification, configuration management and quality assurance. You do not need to worry about certification liaisons in this University project.

- **Analysis/Design Phase:** Create a high-level capture of the software requirements and the wider system requirements. Undertake hazard analysis such as fault tree analysis to identify key issues that need addressing, and document a design that the team can then develop against and that is traceable back to the requirements. The design can be purely textual, although teams may also choose to use model techniques such as the Unified Modeling Language (UML), and specifically diagram types within the UML such as class diagrams and state diagrams.
- **Development Phase:** Implement software according to the defined requirements and design, ensuring compliance with coding standards (i.e. "Power of Ten").
- **Integration Phase:** Integrate individual software components in a controlled and systematic manner.
- **Verification Phase:** Conduct thorough testing, including unit tests, integration tests, and system-level tests within simulated environments to verify all aspects of the software.

3. **Verification and Validation:** Emphasise rigorous testing and validation activities to confirm that the software meets all requirements and behaves as expected in simulated environments.

7 Tasks

The assessment objective is to develop a safety-critical software system. To achieve this will require working together as a group to develop multiple software systems.

7.1 Group Agreement

Starting with the last part first, this is a short group project, with only five weeks at most for actual development leaving at least a week for writing the report. As such, it's important that all group members are committed to the group and can contribute regularly, and that group members aren't left waiting for long for other group members to do what they promise to do.

With this in mind, groups should quickly write an agreement on how they will work together, that all group members should sign up to. This should include aspects such as:

- when the group will regularly meet and how the outcome of meetings will be recorded.
- how tasks will be allocated, what information will be stored and how progress or delays on tasks will be reported. For example, this may be either minuted at a group meeting and posted where everyone can see the tasks or may be done via issue-tracking in a GitLab repository.
- a schedule of how many hours per week each member is committing. While the course assumes around 6 – 7 hours per week throughout the project, some group members may have peaks and troughs in their workload elsewhere which means they may need to redistribute these hours differently. This information should be captured and the group's work scheduled planned accordingly.
- an agreement on how variations to the schedule will be communicated and documented.

Group members who are unable to uphold the agreement and don't contribute their share of time and effort to the project will be marked down in the assessment.

At a minimum, groups should use GitLab as a repository of code and documentation. Groups are encouraged to use issue tracking as well.

7.2 Software — Two Systems

The group needs to develop an avionics software system (herein called the *core system*) that does the functionality specified in section **??**. The biggest challenge here is that you as a group do not have access to any hardware to test on and there is no pre-existing simulation framework to plug your new system into! This means that you will need to *not only* develop software that can be deployed to aircraft to solve the

core problem, but you will *also* need to develop a non-trivial amount of software to test the main system by putting it through its paces in realistic ways.

The simulation software will need to simulate: redundant sensors; aircraft control surfaces; and a simple simulation of the engine.

The key thing here is that the simulation software needs to put the core system through its paces and so can't just send only ideal data where everything is perfect and the aircraft is in no danger. The core system needs to be able to mitigate and handle hazards and events on a real aircraft, so the simulation software must be able to replicate things events like: sensor errors; sudden unexpected events such as — for example — loss of altitude; or unexpected delays or errors in responding to control signals by in engine or control surfaces.

7.3 Simulation and DO178-C

The simulation software is distinct from the core system as the simulation software does not need to adhere to the full development process of DO178-C. Normally, something like DO330 Tool Qualification would be used as a standard for software used as a tool to test or verify an avionics software system, but that itself (the tool) isn't intended for actual deployment to the aircraft.

We have not taught DO330 in SWEN326, and as such, you do not need to develop the simulation tool to this standard.

8 Report

The report is a critique of what your group developed and how the development process was followed. The report is to be done individually and is your chance to reflect on what went well and what could have gone better in the development of the system. In scope for discussion are: documentation created, code generated, processes followed, infrastructure constructed, tests performed, and analyses conducted.

This report is what you will be assessed on. The group must make reasonable progress on the development of the system for you to have anything worthwhile to comment on in your reports. It is important to note that reports that simply point out that things the team didn't get around to doing are things the team should have done won't score particularly highly, as that isn't particularly insightful.

8.1 Format

The format of the report needs to follow these formatting principles:

- maximum of 8 pages (not including the appendices - see below).
- single column.
- 12pt font.
- clearly state your: full name; student ID; and group ID at the top of the first page.
- have six section headings: **Documentation; Code; Tests; Infrastructure; Process; Improvements**.
- submit as a single PDF called "StudentID_report.pdf"

8.2 Coverage

Since you have at most eight pages in total to cover all of these six components, there is no expectation that you cover **all** possible issues – both positive and negative – that could be mentioned. Part of the challenge of this report is identifying a couple of issues per report component that are significant and that are worth focussing on.

Note: while I've said this report is a *critique*, it's important to understand that this does not mean you are restricted to only mentioning negative aspects. Something that went well

can also be a perfectly valid thing to state in a critique. The *key* aspect to focus on is to support your claim – whether it be a good, bad or both good and bad aspect of the project – with evidence.

The purpose of the report is neither marketing (e.g. "everything was awesome and the system is great and nothing can possibly go wrong!") nor self-flagellation (e.g. "the system is terrible and a death trap and lots of things went wrong with the group!"), but rather to self-reflect on a learning exercise.

8.3 Appendices

You should include up to two appendices at the end of your report. The first appendix is not needed if you did not use AI tools. The second appendix is needed for all reports and is key for grading your report. Neither appendix counts towards the 8-page limit.

1. [**Appendix A**]: - State how you used AI tools (if you did use AI tools), which tools you used and the prompts that you used.
2. [**Appendix B**]: - State your contribution to the group project and identify how you spent your time on the various group tasks and how you met the group agreement. This should not be longer than one page.

9 Grading

The assignment is worth 30% of your overall course grade, and you will be assessed based on the quality of your critique, moderated by the level of contribution you make to your group.

9.1 Critique

Your report will be assessed on six components of equal weighting:

1. Critique of the documentation created by the team.
2. Critique of the code created by the team.
3. Critique of the tests created by the team.
4. Critique of the infrastructure created by the team to handle the testing / analysis / simulation.
5. Critique of the process followed by the team.
6. A couple of ideas on how to improve for next time, and a brief justification as to why these improvements would meaningfully improve the outcomes.

Each of these components will be graded on a six-point scale of $0 - 5$. These scales roughly equate to the grades E, D, C+, B, A- and A+ respectively.

0 **Well Below**: Either no attempt or the attempt was trivial and didn't include any details.

1 **Poor**: Poor performance overall, some evidence of learning.

2 **Satisfactory**: Satisfactory to good performance.

3 **Good**: Good performance that identifies a non-trivial issue and provides some good supporting evidence that provides some genuine credibility to the argument and claim being made.

4 **Excellent**: Excellent discussion that clearly identifies significant and substantial issues and provides compelling supporting evidence that supports the well-constructed arguments and claims being made.

5 **Outstanding**: Outstanding discussion that exceeds expectations in terms of how evidence and claims are combined to create a truly insightful narrative.

9.2 Effort Multiplier

The final mark is also affected by an assessment of the effort you committed to the group. You will receive a mark in this category on a scale of 0 – 1:

- 0.0 No attempt to contribute to the group work.
- 0.33 Sporadic engagement with the group work for less than 50% of the time, did not stick to the group agreement, and repeatedly and consistently provided late or broken or incomplete output given the tasks allocated to them in the timeframe agreed.
- 0.66 Some effort – although inconsistent and unreliable – for over 50% of the time, and did not substantially stick to the group agreement.
- 1.0 Committed to the group project and put in a reasonable amount of time, sticking to the group agreement for substantial periods of the project duration.

The total mark is the sum of the six component marks (6 components; 5 marks each; maximum total: 30 marks) multiplied by the effort multiplier. The mark will be rounded up to the nearest half mark.

9.3 Example

If someone got "Good" for three components, "Satisfactory" for two components, and "Excellent" for one component, and an effort multiplier of 0.66, they would get a score of:

$(3 + 3 + 3 + 2 + 2 + 4) * 0.66 = 17 * 0.66 = 11.22$

This mark would be rounded up to 11.5 / 30.

10 Submission

One member of each team needs to submit the group's documentation and code to the ECS online submission system: `https://apps.ecs.vuw.ac.nz/submit/SWEN326/Group_Documentation_Code`

Every member of each team needs to submit their report to the ECS online submission system: `https://apps.ecs.vuw.ac.nz/submit/SWEN326/Group_Project`

10.1 Late Penalties

Late submissions will incur an automatic penalty of 10% of the final mark per day. Submissions between 0 and 24 hours late will be counted as one day late; those between 24 and 48 hours late will be counted as two days late, etc. Any request for an extension must be made to the course coordinator before the due date.

The submission date for the individual report is 5:00pm on Friday 31 May and every student needs to submit their own individually-written report.

The submission date for the group documents is 11:59pm on Friday 31 May and only one student from each group should submit the group work for the entire group.