Concordia University
Engineering and Computer Science

Report
On
COEN 320 – INTRODUCTION TO REAL-TIME SYSTEMS

# PROJECT: FUNCTIONAL REQUIREMENTS OF AN AIRCRAFT MONITORING SYSTEM

Submitted to

*K. Khorasani*
*Professor's name*

Due Date: December 15th, 2017

By

| | |
|---|---|
| Michael Gravino | 27088140 |
| Name | student ID |
| Matthew Masi | 27039956 |
| Name | student ID |

**"We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality"**

Concordia University

## Table of Contents

## Table of Figures

## Abstract

Aircraft monitoring systems are incredibly important in everyday travel as millions upon millions of people utilize this method of transportation. These monitoring systems must be very precise and methodical as they will ultimately help pilot the aircraft with the help of a human pilot. This report will illustrate the schedulability of tasks that are dedicated to the monitoring of flying conditions of a small engine aircraft. This engine is equipped with sensors that are periodically polled by the system to ensure that it is within safe working range and will notify the pilot when a sensor is out of range.  The system will also provide smoke detection that will be logged onto a magnetic tape.

## Introduction

This system will be designed using C++ with real-time libraries such as threads to properly run the situation. This program will take data requests and output specific data while running its periodic tasks and making sure all jobs meet their appropriate deadlines. The following sections of this report will bring the problem at hand, as well as our assumptions, scheduling model and the development of this system. We will also go through some case studies that will be analyzed using time-demand analysis for verification. These case studies will try and reproduce every scenario, good or bad, of every task completing or missing its deadline and will help ensure that the right scheduling model has been chosen.

## Problem Description

The aircraft monitoring system (to be referred to as AMS for the remainder of this report) will monitor the conditions of the single passenger and engine vehicle. The main attributes that the system will monitor are presented in this figure below:

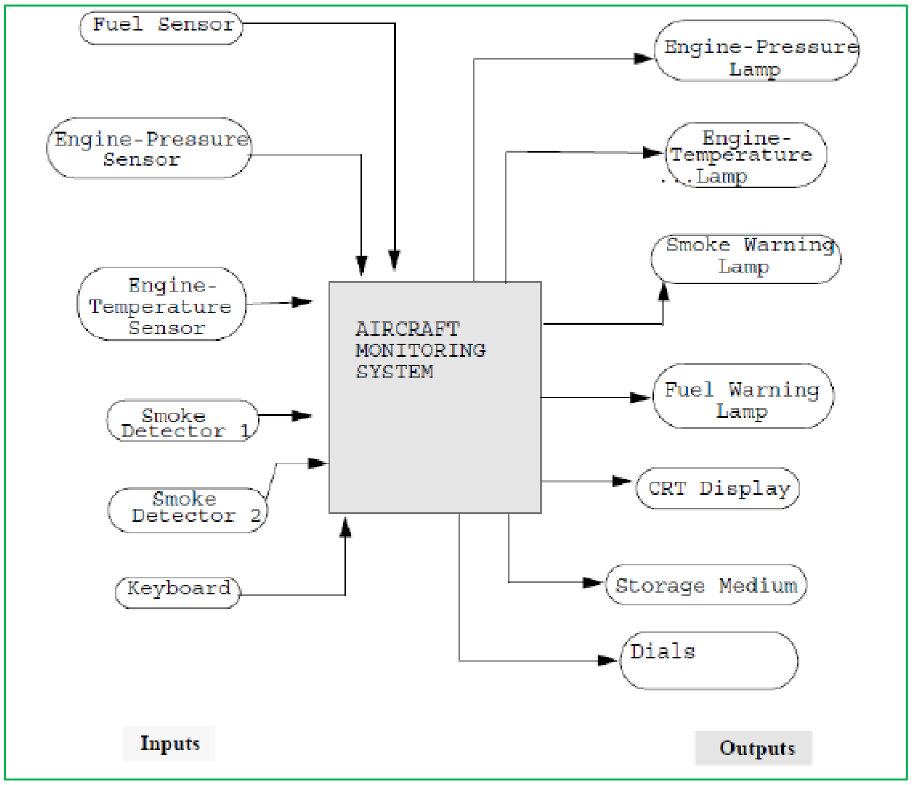


*Figure 1: Aircraft Monitoring System Description*

Figure 1 is the basis of our system in C++. We have implemented all the specifications as described. Other information at hand from the project description is that the data is being read by sensors and the state of the components that are being monitored will be displayed as warning lamps (green and red). When a sensor fails to respond to a poll sequence, a time-out signal is generated, three consecutive time-out signals will change the lamp from green to red.  A message will appear on the CRT if any warnings arise and must be acknowledged by the pilot. The pilot himself can request readings within the system while navigating through the CRT with the help of a keyboard from any sensor, our system is expected to respond accordingly. At the same time, it will keep polling for data while the pilot navigates through the system for his desired information. If ever there is a warning while the pilot is navigating through the AMS, the system will display the warning ahead of his personal request, once he acknowledges the warnings it will be returned to what information he was looking at. This, in a technical term, will be interrupts that our system must handle.

## Scheduling Model

First, we will determine the types of tasks to schedule. Next, we will choose a suitable scheduling algorithm that will meet system requirements, and finally, we will conduct a time-demand analysis of different case studies to see how the system will react.

### Assumptions

For the scheduling model, we will make a few assumptions. We don't possess a special purpose embedded system to perform these simulations so we will instead be using a Windows operated laptop. All context switching will be ignored, and time measurements will be relative; we will not be specifying any unit of measurements other than the ones mentioned within the project manual (polling sensors for example).

### Tasks

Our system will be running three distinct types of tasks in our system: periodic, sporadic and aperiodic. At 1 second intervals, the system polls the three sensors (pressure, temperature, and fuel) and as such, they are scheduled as period tasks. Other tasks (keyboard commands for temperature, pressure, and fuel) are scheduled as sporadic tasks. These sporadic tasks are of less importance and thus are given the lowest priority. These will be implemented by using a periodic server. The periodic server selection will be based on the key needs of our system and the simplicity of the server.

Finally, we have the interrupt handlers. These tasks will alert the pilot through the CRT whenever the sensor readings reveal an abnormality within the system and will change the warning lamp from green to red (and red to green). The pilot will also be alerted when smoke is detected and cleared. The important thing to note is that the interrupt handler must not block the periodic polling of the sensors that occurs every 1 second.  The pilot must also acknowledge the warnings on the CRT screen before being able to execute any other sporadic tasks.

## Periodic Server

To make sure that the sporadic tasks can meet their deadlines (in this case the sporadic tasks that the pilot executes with the keyboard), we will need to choose a periodic server. After reviewing our choices (with specifications and implementation in mind), we brought it down to three periodic servers: polling, sporadic and deferrable server.

The polling server does not have a specific way to quantitatively measure schedulability throughout the system which isn't good for testing on simulations. It also loses its budget for the current period if no aperiodic task is found in the buffer. On the other hand, it is always replenished at the start of each period.

The sporadic server replenishes its budget periodically and keeps it if no aperiodic tasks arrive. It can also quantitatively measure schedulability which will be helpful when testing during simulations.

The deferrable server is comparable to the sporadic server in that it replenishes the budget and will keep it if no aperiodic tasks arrive. It also can quantitatively measure schedulability and is much simpler to implement than the sporadic server. Thus, we've decided to utilize the deferrable server for our system's aperiodic tasks.

## Scheduling

Our system deals with 3 distinct types of tasks as mentioned previously. To schedule the interrupt handlers and periodic tasks, we will be utilizing a rate monotonic scheduling algorithm because of its simple implementation and effectiveness for the task at hand. Within the periodic tasks, we've assigned static priorities such that fuel is the highest, pressure is second, and temperature is the lowest (since they arrive at the same time). The sporadic tasks, as explained previously, will be handled by the deferrable server. The following table displays the tasks that our system will need to process with their respective execution time, deadline, arrival time, and periods:

| Tasks | Description | Type | Arrival | Execution | Period | Deadline |
|---|---|---|---|---|---|---|
| **$T_1$** | Temperature Sensor | Periodic | 0 | 10 | 1000 | 1000 |
| **$T_2$** | Pressure Sensor | Periodic | 0 | 10 | 1000 | 1000 |
| **$T_3$** | Fuel Sensor | Periodic | 0 | 10 | 1000 | 1000 |
| **$T_4$** | Smoke Sensor | Periodic | 0 | 10 | 1000 | 1000 |
| **$T_5$** | Request Fuel Consumption | Sporadic | Input | 20 | - | - |
| **$T_6$** | Request Temp Info | Sporadic | Input | 20 | - | - |
| **$T_7$** | Request Press Info | Sporadic | Input | 20 | - | - |
| **$T_8$** | Temperature Lamp | Interrupt handler/Aperiodic | $T_1$ returns true | 7.5 | - | - |
| **$T_9$** | Pressure Lamp | Interrupt handler/Aperiodic | $T_2$ returns true | 7.5 | - | - |
| **$T_{10}$** | Fuel Lamp | Interrupt handler/Aperiodic | $T_3$ returns true | 7.5 | - | - |
| **$T_{11}$** | Smoke Lamp | Interrupt handler/Aperiodic | $T_4$ returns true | 7.5 | - | - |
| **$T_{12}$** | Deferable Server | Periodic Server | 0 | 20 | 150 | 150 |

*Figure 2: Table of Tasks*

The numbers in the table are all relative to the polling that happens every 1 second and thus they are not exact. The budget was made by testing our system with different values and seeing what would work efficiently.

## Case Study

For the case studies, we will be conducting 4 test cases that our system will encounter and see how it would react regarding our budget choice. In other words, we will conduct a time-demand analysis of different scenarios that may arise up until we find a point of failure. These diagrams were conducted by hand but have also been tested within the system to prove the validity of its results.

**Case 1:** All periodic tasks are executing; no warnings are given and the deferrable server handling the aperiodic tasks is assumed to be always working and to use up its whole allotted budget.

**Results:** Case 1 completes on time and is on schedule; no time-outs and no delays. All four polling tasks were completed within their deadline and were able to successfully obtain the sensor data. The deferrable server was also schedulable with our given budget choice. The following figure represents the timing diagram of this case:

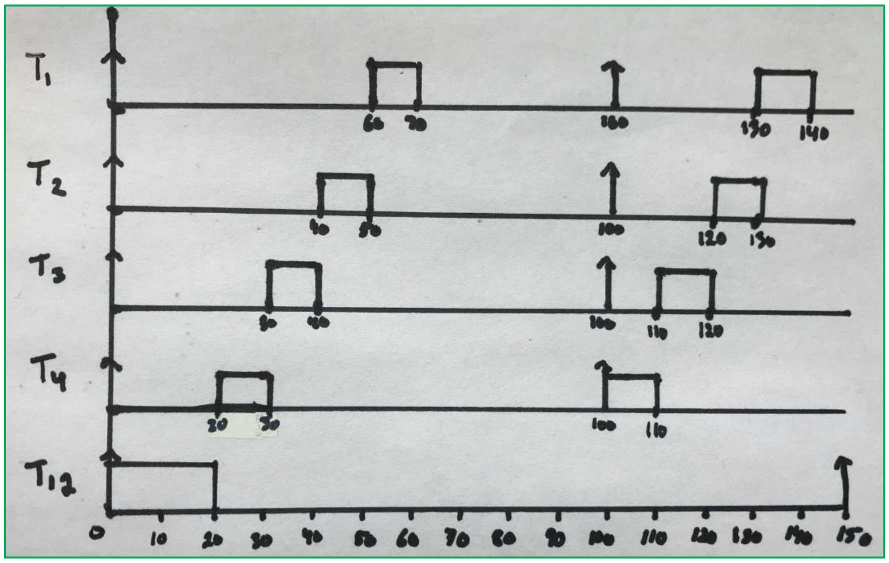


*Figure 3: Case 1 Timing Diagram*

**Case 2:** Here we will have all periodic tasks executing and two warnings that will happen at the same time (the temperature lamp and the pressure lamp). The deferrable server will be used the same as the last case.

**Results:** The interrupts execute first, then the periodic tasks (polling of the sensors), and finally the deferrable server. Case 2 executes successfully; every task completes on time even though we've added the two warnings of the sensors for temperature and pressure. The following diagram represents the situation:

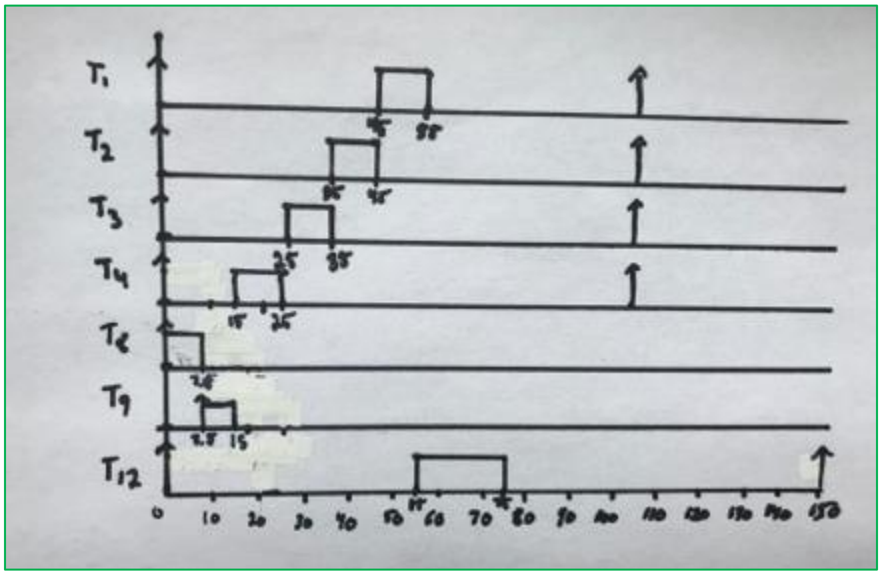


*Figure 4: Case 2 Timing Diagram*

**Case 3:** As with the 2 last cases, we will have all 4 periodic tasks running, as well as letting the deferrable server consume its allotted budget. In this case, however, we will examine how our system reacts when 3 warnings arise at the same time.

**Results:** After scheduling each task by executing it at the appropriate times considering their priorities, we obtain a schedulable system that can be seen in the next figure below. It is important to note that each warning only appears at one time per polling interval.

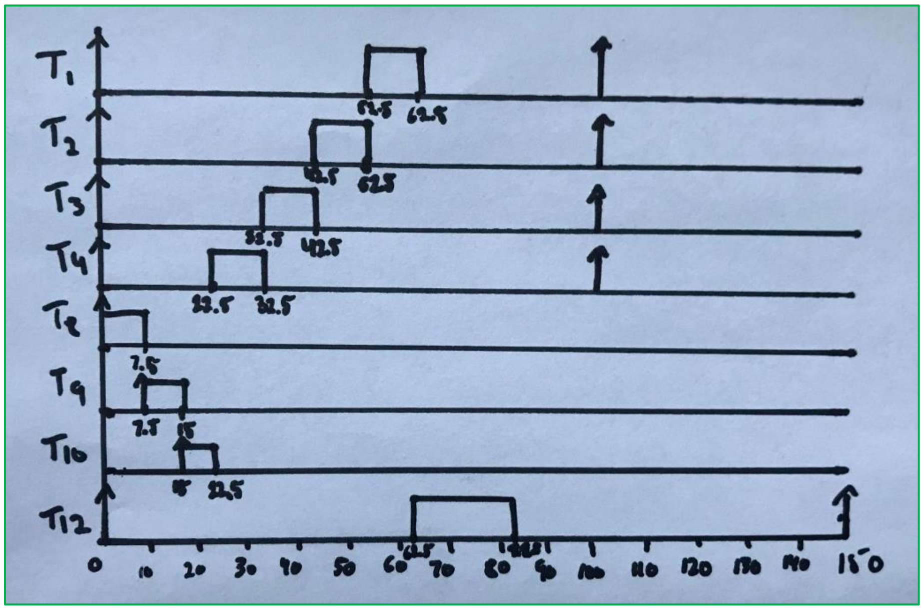


*Figure 5: Case 3 Timing Diagram*

**Case 4:** For the final case, we will consider that all 4 warnings occur within the same polling interval. All periodic tasks will be executed, and the deferrable server will consume all its budget.

**Results:** Like in case 3, we observe from the next figure that if the 4 warnings were to happen all during the same polling interval the system will still be able to function as needed. The deferrable server will also run to completion given its allocated budget.

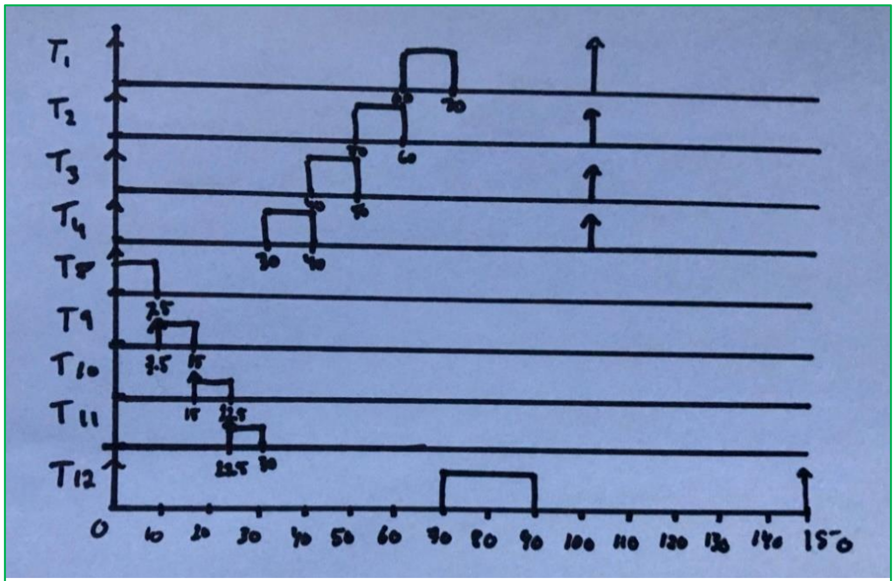


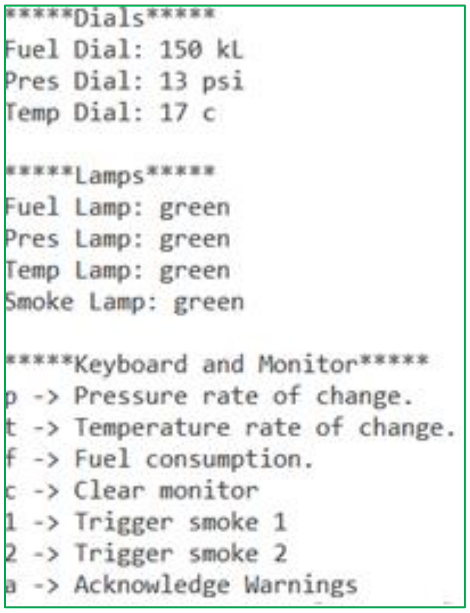
*Figure 6: Case 4 Timing Diagram*

All in all, we are very happy with our results, while the timing diagrams drawn only represent the situation where warnings happen once per polling interval, our system will still function as promised. However, there might be some occurrences that multiple warnings are repeated during the same polling interval. If ever that were the case, we still have some allotted time left to schedule those, even if we generate all 4 warnings, as in case 4. A time-out will only really occur if there is an exaggerated amount of repeated warnings which seems unlikely, but it can surpass the deferrable server as it is very low in priority and won’t be detrimental to the safety of both the aircraft and the pilot.

## Discussion

In this section, we will be discussing results that we've obtained while conducting the test cases presented in the time-demand analysis on the system we have created within C++. This simulation will prove that our design of the real-time software presents a working Aircraft Monitoring System that is completely functional within the requirements presented. The code of this system will be handed in within a USB flash drive with the submission of this report. To begin, we will be looking at the first test case;

**Case 1:** Here we have no warnings, just the periodic tasks running at the same time as the deferrable server:

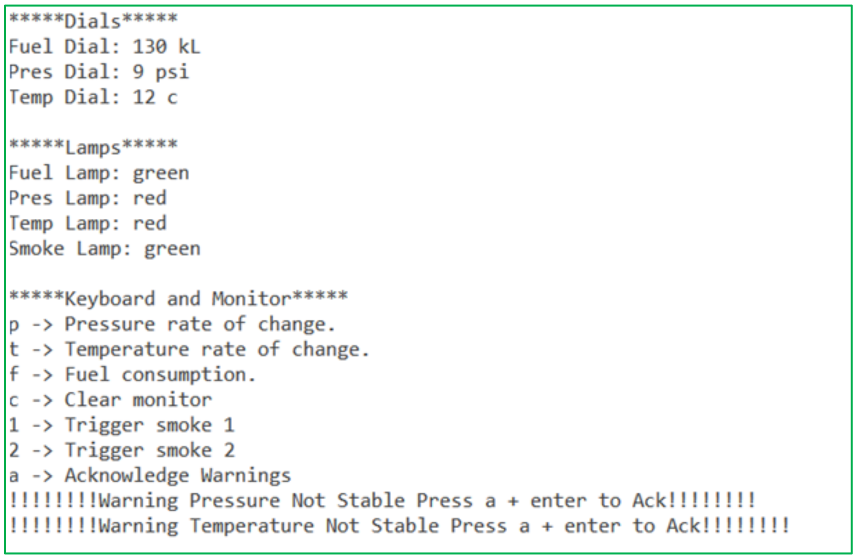


```
*****Dials*****
Fuel Dial: 150 kL
Pres Dial: 13 psi
Temp Dial: 17 c

*****Lamps*****
Fuel Lamp: green
Pres Lamp: green
Temp Lamp: green
Smoke Lamp: green

*****Keyboard and Monitor*****
p -> Pressure rate of change.
t -> Temperature rate of change.
f -> Fuel consumption.
c -> Clear monitor
1 -> Trigger smoke 1
2 -> Trigger smoke 2
a -> Acknowledge Warnings
```

*Figure 7: Case 1 Simulation in the Aircraft Monitoring System*

**System Analysis:** As can be seen in figure 7 just above, the system runs successfully; no warnings are presented as all warning lamps remain green and the pilot has a full view of options on the user interface of the system.

**Case 2:** This case presents two warnings while the periodic tasks and deferrable server are running simultaneously. The warnings that will show will be the Temperature Lamp as well as the Pressure Lamp.

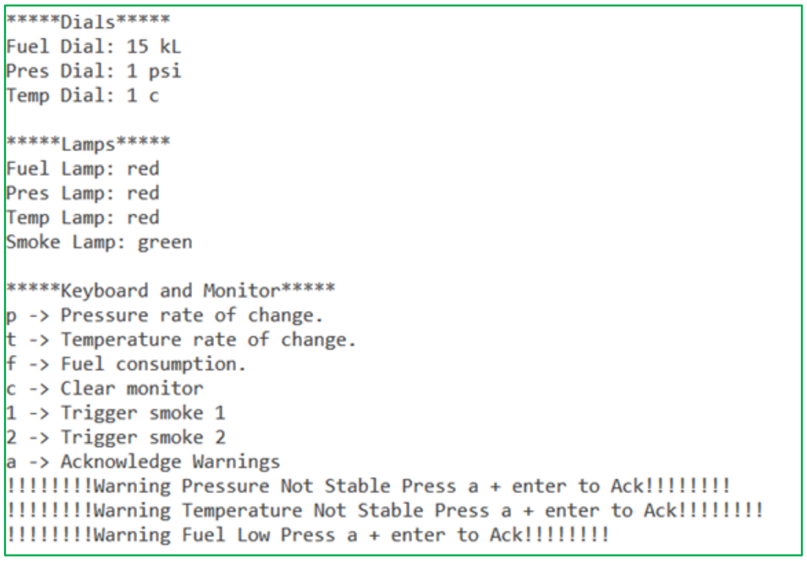


```
*****Dials*****
Fuel Dial: 130 kL
Pres Dial: 9 psi
Temp Dial: 12 c

*****Lamps*****
Fuel Lamp: green
Pres Lamp: red
Temp Lamp: red
Smoke Lamp: green

*****Keyboard and Monitor*****
p -> Pressure rate of change.
t -> Temperature rate of change.
f -> Fuel consumption.
c -> Clear monitor
1 -> Trigger smoke 1
2 -> Trigger smoke 2
a -> Acknowledge Warnings
!!!!!!!!Warning Pressure Not Stable Press a + enter to Ack!!!!!!!!
!!!!!!!!Warning Temperature Not Stable Press a + enter to Ack!!!!!!!!
```

*Figure 8: Case 2 Simulation in the Aircraft Monitoring System*

**System Analysis:** As can be seen in figure 8, both the temperature lamp and the pressure lamp have changed to red and presented a warning at the bottom of the screen that will warn the pilot about the alarming events. Thus, the pilot must acknowledge the warnings by pressing a button on his keyboard before continuing any other navigation through the Aircraft Monitoring system.

**Case 3:** This case will present 3 warnings simultaneously with the periodic tasks as well as the deferrable server. Similarly, this case will alert the pilot that the Temperature, Pressure, and Fuel sensors are detecting alarming changes,

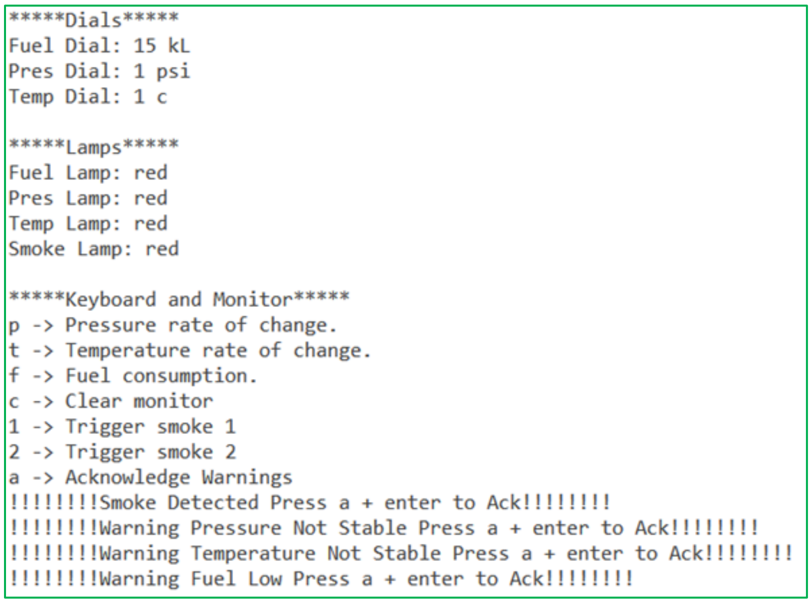


```
*****Dials*****
Fuel Dial: 15 kL
Pres Dial: 1 psi
Temp Dial: 1 c

*****Lamps*****
Fuel Lamp: red
Pres Lamp: red
Temp Lamp: red
Smoke Lamp: green

*****Keyboard and Monitor*****
p -> Pressure rate of change.
t -> Temperature rate of change.
f -> Fuel consumption.
c -> Clear monitor
1 -> Trigger smoke 1
2 -> Trigger smoke 2
a -> Acknowledge Warnings
!!!!!!!!Warning Pressure Not Stable Press a + enter to Ack!!!!!!!!
!!!!!!!!Warning Temperature Not Stable Press a + enter to Ack!!!!!!!!
!!!!!!!!Warning Fuel Low Press a + enter to Ack!!!!!!!!
```

*Figure 9: Case 3 Simulation in the Aircraft Monitoring System*

**System Analysis:** As you can see, the three lamps are indicating a red light and the pilot is being warned about the changes in temperature. The pilot must acknowledge the warnings before being able to navigate elsewhere within the system.

**Case 4:** In this case, we will be simultaneous simulating 4 warnings in the same polling interval as well as run every periodic task and the deferrable server.

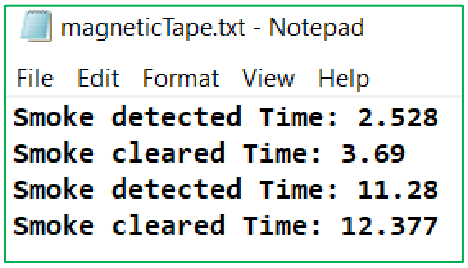


```
*****Dials*****
Fuel Dial: 15 kL
Pres Dial: 1 psi
Temp Dial: 1 c

*****Lamps*****
Fuel Lamp: red
Pres Lamp: red
Temp Lamp: red
Smoke Lamp: red

*****Keyboard and Monitor*****
p -> Pressure rate of change.
t -> Temperature rate of change.
f -> Fuel consumption.
c -> Clear monitor
1 -> Trigger smoke 1
2 -> Trigger smoke 2
a -> Acknowledge Warnings
!!!!!!!!Smoke Detected Press a + enter to Ack!!!!!!!!
!!!!!!!!Warning Pressure Not Stable Press a + enter to Ack!!!!!!!!
!!!!!!!!Warning Temperature Not Stable Press a + enter to Ack!!!!!!!!
!!!!!!!!Warning Fuel Low Press a + enter to Ack!!!!!!!!
```

*Figure 10: Case 4 Simulation in the Aircraft Monitoring System*

**System Analysis:** As you can see, all four lamps have indicated a change from green to red. The pilot is presented with acknowledgments that only he can dismiss by pressing a key on his keyboard.

**Magnetic tape:** This magnetic tape, or magnetic medium, will record all the times that the smoke or no smoke interrupts are received. The following figure presents some examples of the system registering that information.



```
Smoke detected Time: 2.528
Smoke cleared Time: 3.69
Smoke detected Time: 11.28
Smoke cleared Time: 12.377
```

*Figure 11: Magnetic Tape Recording Example*

## Conclusion

This Aircraft Monitoring System was built from the description given from the project manual. With that information, we were able to build a scheduling model that revolved around all the tasks, their type, and priorities regarding the functional requirements. Within these requirements, we had to make additional assumptions as well as properly choose a periodic server to process all sporadic tasks. We chose the deferrable server due to its simplicity and because it preserves its budget when there are no sporadic tasks, as opposed to the polling server. The presentation of the time-demand analysis, as well as the simulation on our software, demonstrate a working system. This project offered a great opportunity to apply theoretical notions learned in class on a real-life situation. The completion of this system has helped us immensely in the understanding of said theoretical notions as well as motivated us to pursue additional information regarding this subject. It is important to note that this system was written in C++ and not in a QNX server as C++ featured all the necessary libraries needed to complete this task as well as being far simpler to operate.