

Part 2

by Ruaidhri David Cook

FILE	4148_RUAIDHRI_DAVID_COOK_PART_2_178149_709092849.DOCX (14.4M)		
TIME SUBMITTED	15-APR-2019 09:09PM (UTC+0100)	WORD COUNT	2190
SUBMISSION ID	104825103	CHARACTER COUNT	12405

Critical and Real-Time Application Programming Part 2

Contents

Knowledge and Understanding Skills.....	3
What is a Critical System? (~200 words).....	3
What type of Critical system(s) would software to monitor a patient's heart be? (~300 words).....	3
Implications of building critical systems on the design and development of this device	3
Cognitive and Intellectual Skills	5
Practical Skills.....	9
Research.....	9
Explanation	10
References	12

Table of Figures

Figure 1 Source files involved in serial management, how they are connected and how data flows [5]	11
---	----

Knowledge and Understanding Skills

You have been tasked as part of a development team to write the software for a medical device for monitoring a patient's heart.

What is a Critical System? (~200 words)

A critical system is defined by the consequences that would follow its failure, if the system were to fail and the outcome was inconsequential and not of any great impact then it would not be critical, such an example could be music player software, if this particular software crashed it would merely stop the music from playing. Contrast that with a system such as the software in a space rocket, if the software were to fail the consequences would be catastrophic and could result in deaths if the failure was severe enough. [1][2][3]

Critical systems are split into 3 main types, they are the following:

- Safety critical – These are critical systems that hold the safety of individuals and the environment in the balance, they ensure that harm is not caused.[2]
- Mission critical – These are systems that have the primary purpose of ensuring that the purpose of the overall system is able to be completed/achieved. An example of this kind of system could be the system software used for baggage handling in an airport.
- Business critical – These are critical systems that are associated with economic costs, they are used to prevent interruption of services such as the stock-trading market or the banks.

What type of Critical system(s) would software to monitor a patient's heart be?

Software for a medical device to monitor a patient's heart would primarily be a safety critical system, this is because if it were to fail the patient has the potentiality of death as a result of the doctor's being unable to verify the status of the patient's heart or, even worse, being given false and/or inaccurate information about the heart. This would cause a significant increase in the safety risk of the patient involved, which would imply that it is a safety critical system. [4]

It could also be argued that software to monitor a patient's heart could come under the remit of a mission critical system, this is because the monitoring of a patient's heart is an essential part of the hospital's system. This is because the primary purpose is to keep patients alive and healthy, if the software for the equipment needed to perform that task was not working properly and failed then that primary task could not be achieved, which would have an impact of the death statistics for that hospital for example. [5]

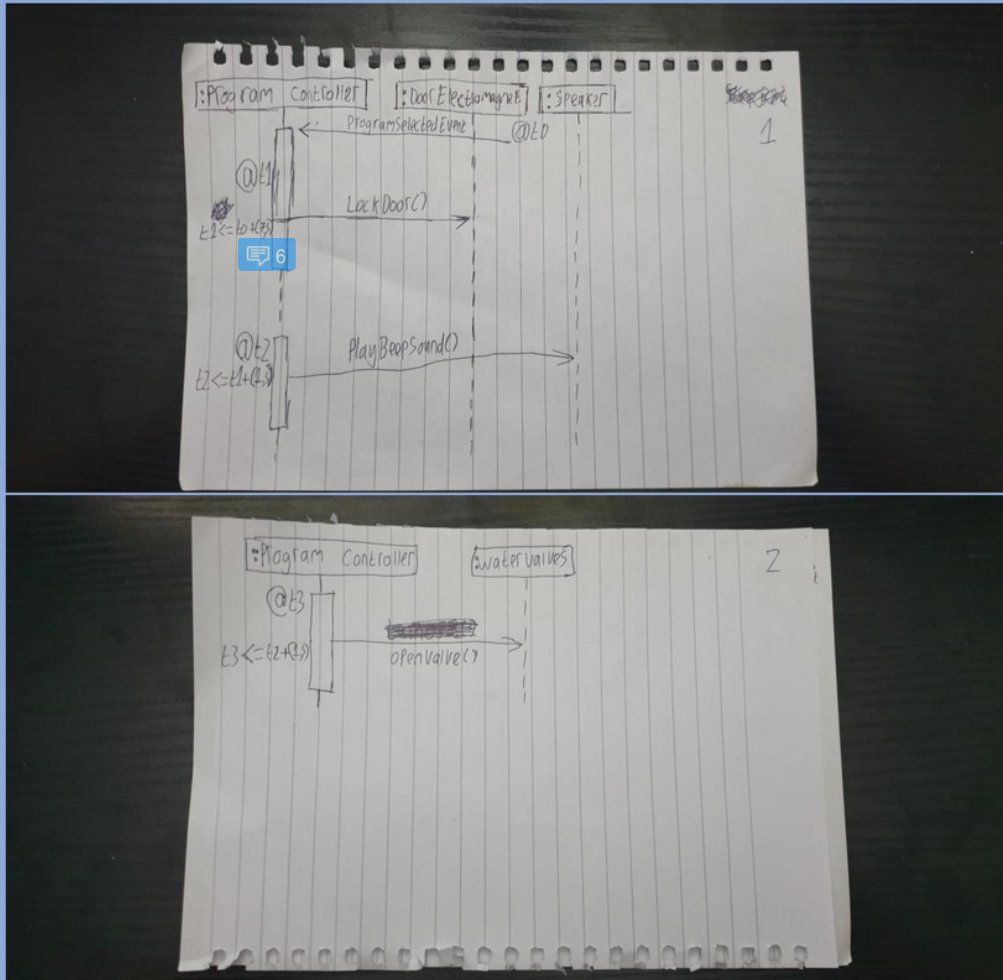
Implications of building critical systems on the design and development of this device

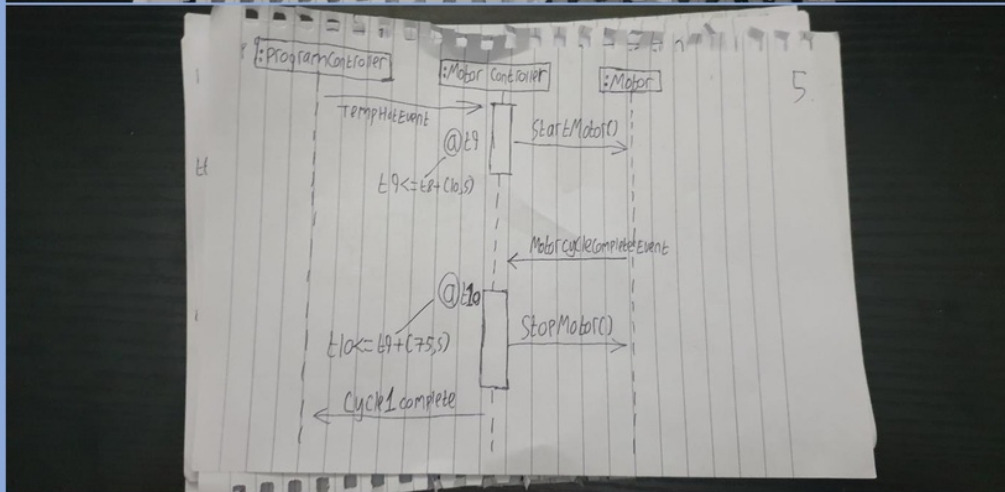
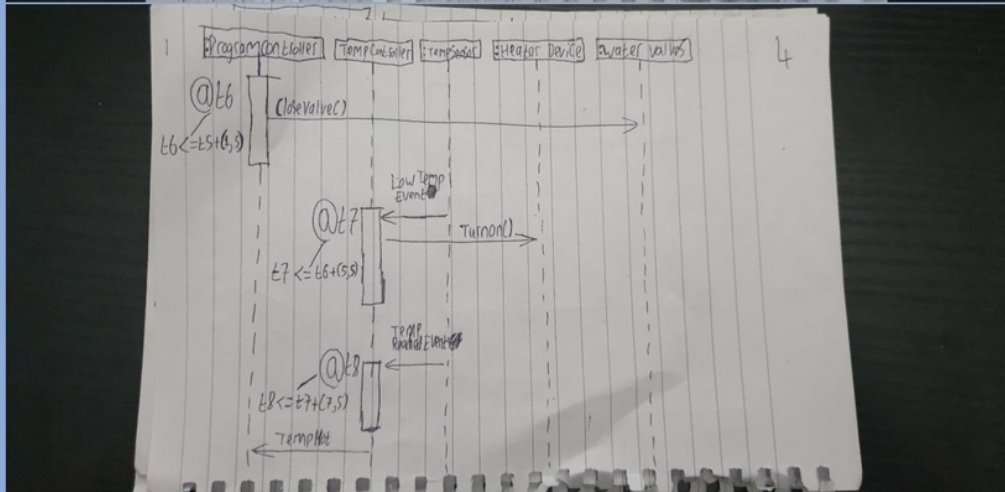
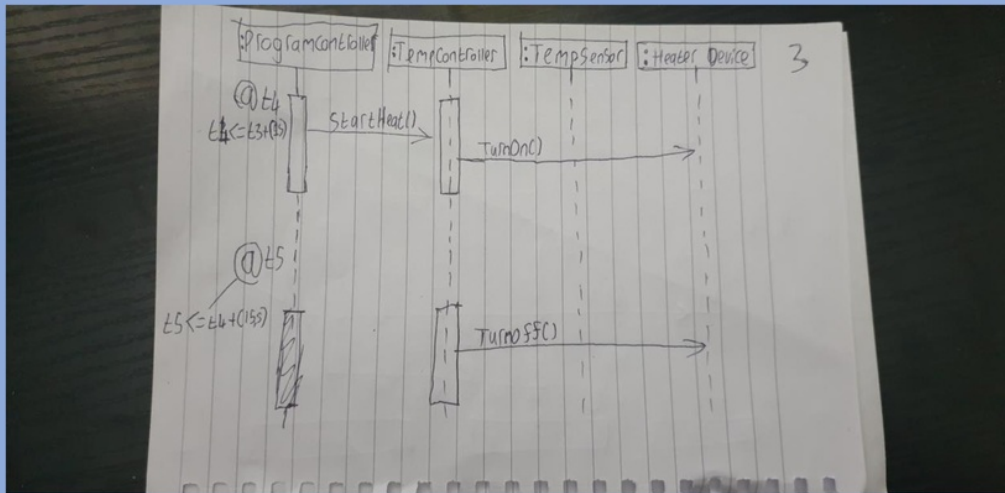
When building critical systems something that must be factored in is the IDE that is used, this is because the software that is used to support the development of safety critical software must be, itself, safety critical, especially the compiler for the code. [6] There is also an increasing problem for safety critical software in the modern world in regard to information security, with security attacks on information systems increasing in frequency, for this reason the security built around a safety critical software must be sufficient enough to ensure the function of the software is not disrupted. For the heart monitor device specifically, it would be important to make sure the network on which it

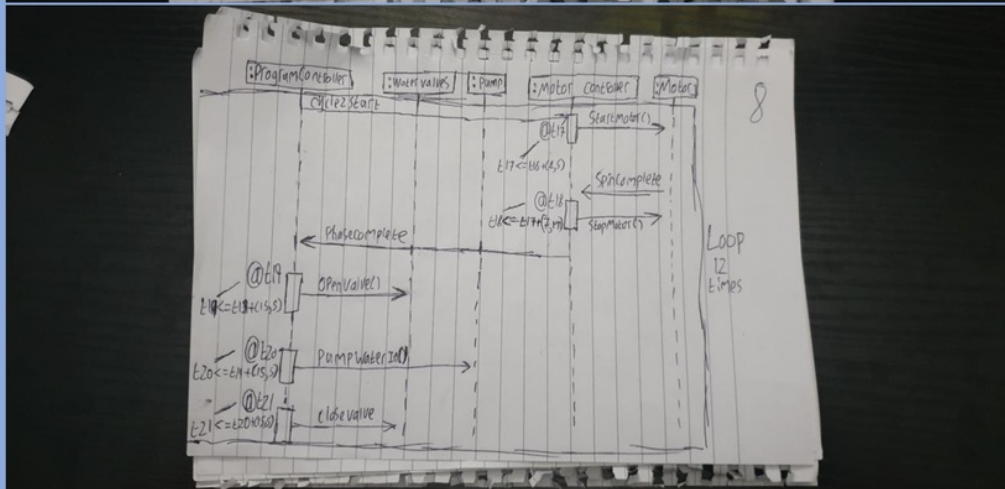
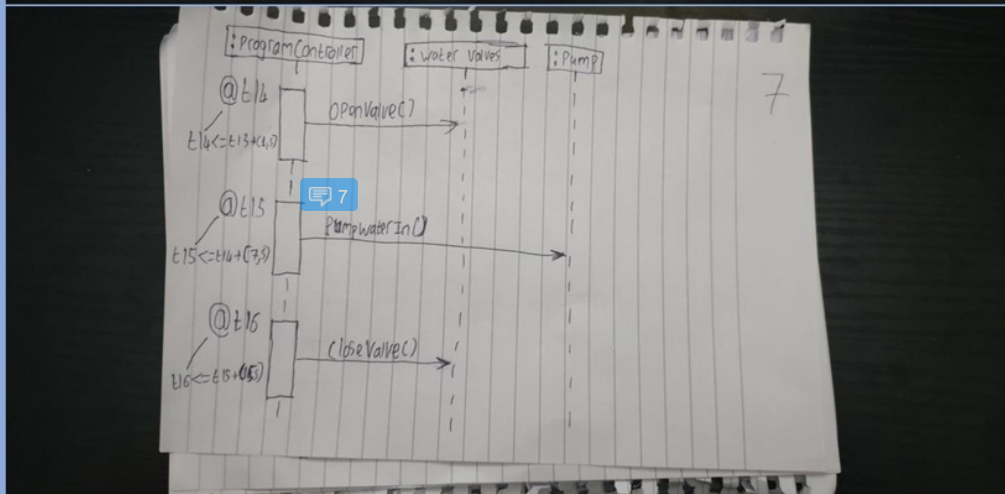
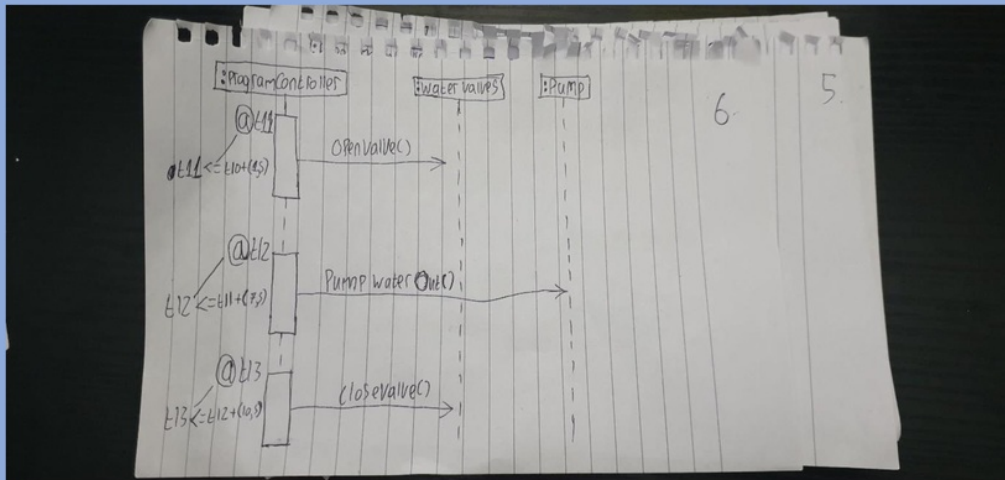
runs is not able to be hacked and the information that is stored on the device is not able to be changed, deleted or corrupted. Something that should be noted is that the increased automation of the software in the heart monitoring device will increase how much of the system is reliant on safety critical computers. It is important to remember that within a hospital there will be multiple safety-critical systems that will need to interact with each other, and therefore the sharing of resources between safety critical systems will be needed, which is a factor that needs to be taken into account when designing a safety critical system. The largest cause of problems and failures in safety critical systems, such as the heart monitoring device, is the intercommunication between the hardware and the software aspects of the system, if the back and forth between the two ceases to function adequately then the appropriate data exchange can no longer occur. When designing systems it is important to take into account whether the system will have a fail-safe state, this is a state where the system will respond to system failure in a manner that minimises the harm that is caused to the device, the environment in which it is being used and the individual associated with the device, however it is important to note that this reduces the reliability of the system at the cost of safety, with this said that would not be a problem if there were multiple of the heart monitoring devices available to use in the event of one device's failure. Building a safety critical system requires a knowledge of the timings, sequences and actions of the device and their associated hazards, so that there is symbiosis between each component of the system.

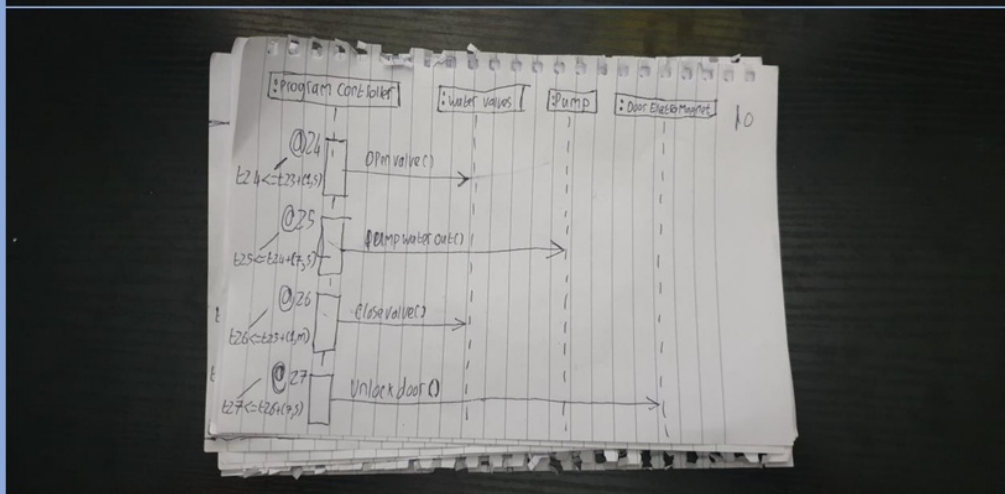
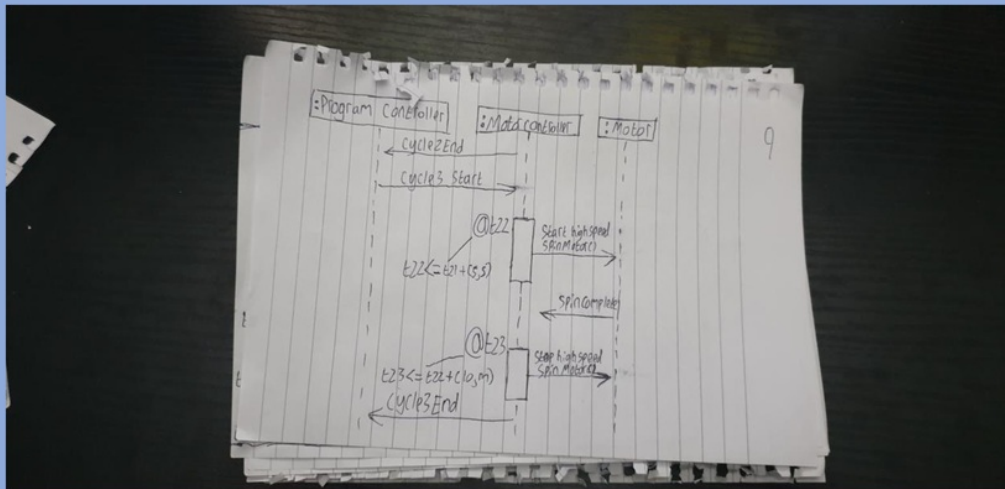
Cognitive and Intellectual Skills

Design the **mainboard** software using **Real-Time UML diagrams** from the point where the user has already set the washing mode and pressed start.









Practical Skills

Research

Comparison and Contrast of RTOS principles/concepts

RTOS stands for Real-time operating system, which is any operating system that's purpose is the running of an application that process data as it arrives in real-time.[5] One concept of RTOS is scheduling and dispatching, scheduling the creation of schedules – organised lists that are used to specify how access to one or more resources is granted when requested by contending entities, and dispatching is the act of granting the access to the contending entity that is most suitable, the suitability is determined by the position of the entity in the schedule, however in a scenario without a schedule the deadline and priority are used to determine suitability.[6]

Another of the concepts, which is linked to the previous one, is schedulable and non-schedulable entities. An example of entities that are scheduled by the scheduler are a thread, process in an application or a task. These are different to the non-schedulable entities that are executed unceasingly, sporadically or in response to an event. The timeliness of these entities is dictated by the system design and implementation rather than the scheduler. Such examples of these non-schedulable entities are OS commands, OS scheduler and interrupt handlers.[7]

Two other related principles are the hard and soft deadline. The hard deadline is a constraint on completion time, much like the soft deadline. The two differ in the fact that with a hard deadline, when the execution point of, for example, the threads reaches the deadline scope the deadline time occurs. This means that the portion of the threads execution that is time constrained is classified as timely – done at the appropriate time. This is different to the soft deadline because if the same scenario were to occur the portion would be classified as timelier, and if the scenario did not occur it would be less timely, rather than being untimely, like with the hard deadline.

Debate drawbacks and benefits of two RTOS

There are different types of Realtime operating systems, two of these are Hard RTOS & soft RTOS. A hard real time operating system is far more strict system because it requires every deadline to be met, these types or systems are a staple in safety critical devices such as nuclear systems and pacemakers. This is different to a soft real time system in the fact that some deadlines can be missed as they are not compulsory for every task every time, however, the reliability of the system is degraded if there are too many missed deadlines. For this reason it could be argued that a soft real time system is more effective because it allows a margin for error, however, it could also be argued that this opens the door to degradation of the system, which is unacceptable in systems that have heavy safety implications. Like the pacemaker example that was given, if the pacemaker was off by a small amount and a deadline was just missed the patient wouldn't instantly die, but if the deadline was consistently delayed and unable to hit the deadline the safety of the device would be compromised.



Explanation

Explain device driver

A device driver can be described as an intermediary between the operating system and the hardware, it is a software application that is designed so that the hardware can have intercommunication between it and the operating system that is interfacing with it. Without a device driver the hardware would fail to work because the information is not being translated for the software, and vice-versa.[8][9][10] Device drivers usually come under one of two generalised categories, these being kernel device drivers and user device drivers.

Kernel device drivers are loaded into a system's RAM with the Operating system, this is because the result is that the device driver can be activated as soon as possible, however, because a kernel device driver is loaded into the system RAM the number of drivers that can run simultaneously is limited, as the performance of the system will be effected.[11]

The other generalised type, user mode device drivers, are used to create an application program interface (API) that is employed at the user level and is triggered by the end-user. This is useful because it allows the interruption of memory management, which avoids any problems that are identified in the aforementioned kernel device drivers. This type of driver can restart without it having an effect of the rest of the system when an error occurs, which makes it particularly useful for devices that make use of USB.[12]

Describe how a serial driver would work

A serial driver is the device driver that is used for serial ports. The serial port is a char driver that is part of the 'tty' abstraction, which is something that allows a serial device driver to integrate into the terminal emulator layer.[13][14][15] There are many types of tty devices, such as the commonly known VGA. The way serial management works is as follows:

In order to register the device drivers, which are each identified using a major number, the file named fs/devices.c is used to export the interface that makes use of the most resources in the system. The file operations used to perform on the tty devices are defined by tty_io.c, which is also used to define the function. Serial.c, and other serial drivers, register an object named struct tty_driver. The driver then declares the major number that was used to register the device driver, in addition to the range of minor numbers that are to be managed and the operations that are supported by said driver.

The scope of the hardware drivers used for the serial ports is determined by the I/O of the data, the interrupt handling and the implementation of the operations. These operations are focused on the input and output of the data, as well as the communication with the higher layers and the associated flow control. The tty layer is used facilitate the data flow between the serial driver and the user space, which means it can employ the functionalities of tty-type device that are most common.

**Source files involved in serial management,
how they are connected and how data flows.**

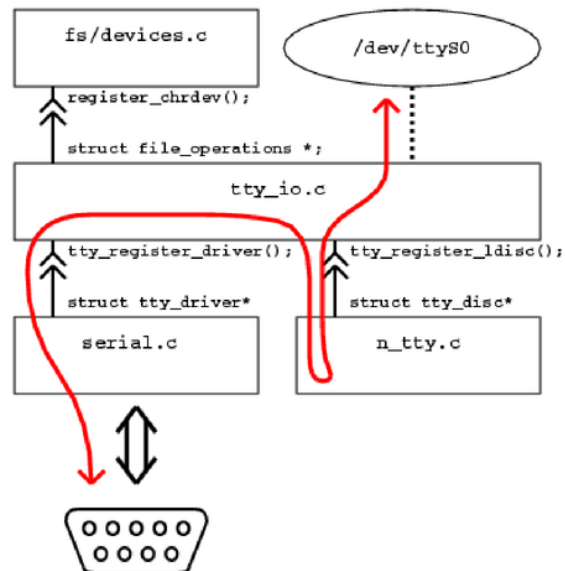


Figure 1 Source files involved in serial management, how they are connected and how data flows [16]

References

- [1] NASA, "Human-Rating Requirements and Guidelines for Space Flight Systems," pp. 1–40, 2008.
- [2] J. A. Carbone, "Managing competence for safety-related systems," pp. 1–20.
- [3] I. Sommerville, "Critical systems." [Online]. Available: <https://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/Dependability/CritSys.html>.
- [4] J. C. Knight, "Safety Critical Systems : Challenges and Directions 3 . WHAT GOES WRONG WITH SAFETY-."
- [5] R. Kamal, "Lesson-12 : Real Time Operating System based Design with the ISRs and Tasks," 2008.
- [6] EmbeddedCraft, "Real Time Operating System Concepts Part 2."
- [7] Massachusetts Institute of Technology, "MIT 6.07 Real Time Operating Systems Lectures •."
- [8] M. Rouse, "Device driver." [Online]. Available: <https://searchenterprisedesktop.techtarget.com/definition/device-driver>. [Accessed: 12-Apr-2019].
- [9] Drivers.com, "What are Device Drivers and why do we need them?" [Online]. Available: <https://www.drivers.com/update/drivers-news/what-device-drivers/>. [Accessed: 11-Apr-2019].
- [10] University West England, "Device Driver Principles."
- [11] S. Hand, "Operating Systems," 2010.
- [12] Microsoft, "Getting started with Windows drivers."
- [13] Microsoft, "Serial Controller Drivers Overview."
- [14] H. Junction, P. Examiner, and D. D. Chang, "(12) United States Patent," vol. 2, no. 12, 2007.
- [15] R. Szabó and A. Gontean, "Creating a serial driver chip for commanding robotic arms," 2013 *Fed. Conf. Comput. Sci. Inf. Syst.*, pp. 667–670, 2013.
- [16] A. Rubini, "Serial Drivers." [Online]. Available: <https://www.linux.it/~rubini/docs/serial/serial.html>. [Accessed: 10-Apr-2019].

Part 2

GRADEMARK REPORT

FINAL GRADE

73/100

GENERAL COMMENTS

Instructor

Critical System (10) 10
Cat heart device (15) 14 Implications on design (15) 12
UML timing diagrams (20) 13
Research RTOS (20) 12
Device driver Int/DMA (20) 12

Very good description of critical systems. Although the discussion of the design implications was as focused and incorporated less relevant ideas. The design is a bit too messy. I think you have the correct approach although you could have illustrated a few different techniques for denoting timings. There were a few errors in the drawing of the sequence diagram itself, classes should not be drawn magically executing methods, either the method is called or invoked by an Actor. However, you have built quite a large sequence diagram containing a lot of detail. Not a lot of detail in the comparison of RTOS types, although although you make some valid points. In discussing the serial driver you kind of dived into the implementation a little too much. How does the hardware/software operate together e.g. what might interrupts be used for here?

PAGE 1

PAGE 2

PAGE 3



Comment 1

Good



Comment 2

I think this is stretching the point a little, but I agree with the the general idea.



Comment 3

I don't quite understand this, perhaps it needs to be rewritten



Comment 4

Is it on a network?

I assume it might be.

PAGE 4



Comment 5

Yes good point

PAGE 5



Comment 6

This is hard to read

PAGE 6

PAGE 7



Comment 7

These activation blocks can be merged because they represent the same method on the ProgramController

PAGE 8

PAGE 9



Comment 8

I was actually expecting you to provide analysis of two RTOS rather than two **types** of RTOS. However,, I will mark this.

PAGE 10

PAGE 11

PAGE 12
