

# CritRealAppProg Sem 2 Robert McIntyre

*by* Robert McIntyre

---

|                |                                                                                             |                 |       |
|----------------|---------------------------------------------------------------------------------------------|-----------------|-------|
| FILE           | 2783_ROBERT_MCINTYRE_CRITREALAPPPROG_SEM_2_ROBERT_MCINTYRE_178149_1383396733.DOCX (203.66K) |                 |       |
| TIME SUBMITTED | 15-APR-2019 08:12AM (UTC+0100)                                                              | WORD COUNT      | 2964  |
| SUBMISSION ID  | 104776910                                                                                   | CHARACTER COUNT | 18229 |

**Critical Real-time Application Programming**

**Part 2**

**Robert McIntyre**

**M6X01090**

**Mike Dacey**

## Table of Contents

|                                          |   |
|------------------------------------------|---|
| Introduction .....                       | 3 |
| Critical Systems .....                   | 3 |
| Safety Critical .....                    | 3 |
| Mission Critical .....                   | 3 |
| Business Critical .....                  | 3 |
| Heart Device Example .....               | 3 |
| Design and Development of Device .....   | 4 |
| UML Diagram of the Washing Machine ..... | 4 |
| Washing Cycle Diagram.....               | 5 |
| Key Diagram of the washing cycle .....   | 5 |
| Rinsing Cycle .....                      | 6 |
| Key Diagram of the Rinsing cycle .....   | 6 |
| Spin Cycle Diagram.....                  | 7 |
| Key Diagram of the Spin cycle.....       | 7 |
| Real Time operating systems .....        | 7 |
| Principals and Concepts .....            | 8 |
| Multitasking .....                       | 8 |
| Interrupt Latency .....                  | 8 |
| Context Switching .....                  | 8 |
| Scheduling and dispatching .....         | 8 |
| Examples of RTOS .....                   | 8 |
| Device Driver .....                      | 8 |
| Serial Driver.....                       | 9 |
| References .....                         | 9 |

## Introduction

### Critical Systems

Critical systems are computer systems where there may be a serious consequence that cause an injury or death to a human being if they fail. There are three main types of critical systems: Safety, Mission and Business. [1] Small errors in the system can quickly multiply and result in a major failure. These could be there is a stack overflow or missing memory which may be a large error but could cause an unexpected result of the system that could lead to incorrect analysis of the data.

#### Safety Critical

Safety critical systems are where if a system fails this may result in an injury or destroys the environment. [2] Safety critical systems are designed to put the system in a safe state if there is a situation that may lead to a life-threatening consequence. Sometimes it would mean that the system stops immediately and switches itself off but if this is not possible, the system must choose the best possible outcome to minimize the risk. If it would be dangerous to stop then procedures must be in place to cope with the failure while continuing to work. An example of this is the car window which if it detects an object when it is closing will immediately reopen.

#### Mission Critical

Mission critical systems are when data could be lost the result of the main goal is then changed. An example of this is aircraft navigation systems. If the aircraft navigation system loses its data then the data needs to be regained for the aircraft to find its relative position. [2] A mission critical system is when the system is essential for the function of the activity to be completed. These systems are less well documented and standardized than the safety critical systems. [3] Mission critical system are often used in defence and security systems.

#### Business Critical

Business critical systems are systems which if a failure occurs would result in a large economic loss to a company. [2] A failure can result in a in high cost for the business-critical system when the system fails. Examples would include systems used in bank where if the system went down for a short time this would cause financial problems.[4]

For critical systems there need to be a well-defined method for developing the system that usually includes a specification with hazard and risk analysis. The methods used for the critical systems should be trusted and tested extensively due to the high costs of the failure of the systems. These may include some software engineering techniques such as redundancy or fault detection and correction that would not be used on non-critical systems because of their cost. Critical systems require a large amount of verification and validation to uncover as many errors as possible before implementation of the system.

#### Heart Device Example

The software in the heart device can be categorized as a safety critical system but its use for monitoring a patient's heart could also defined in a few cases as a mission critical system as well.

The first is the safety critical system as the device is measuring the heart output of the patient which is needed to determine the state of the patient. A failure in this could cause a life-threatening incident. Also, any data that is missed could lead to an inappropriate diagnosis of the patient and the

possibility that incorrect treatment takes place. It is important that the failure rates are extremely low so that the data collected can be relied on. Any data that is missed may miss critical symptoms that are not repeatable.

The other type is mission critical system. As the heart device is for monitoring it is important that the data is collected when required and if any data is lost it will mean that the monitoring is not complete and therefore the measurement process will have to be repeated. This can mean delays to operations of patients when it was being used as an aid to determine if a patient is healthy in a patient test but if it was in a life monitoring situation it would be a safety critical system.

## Design and Development of Device

When designing a critical system, it is important to investigate key areas such as the specification, architecture, verification and process. Also, the information security of the device may need to be investigated. When designing a system, safety is always kept in mind. Computers are used widely in medical systems, but this is not always recognized. When a critical system is designed the code still must be compiled and therefore the compiler is also a safety critical device and therefore needs to meet the requirements of the critical system. There may be other systems that are used when developing the device and it must be considered whether these will also be critical systems. When there is a failure there needs to be a lesson learnt from the mistake. Ideally this will be carried out in the verification and validation of the code before it is implemented. It is not the number of mistakes that have occurred it is the checking of the failure that is important. For a system to be implemented the specification will identify the reliability that is required of the device and this will need to be tested for to ensure it passes.

It is possible that the heart monitor may be connected to the internet to transfer the collected data to be analysed by doctors remotely. If this is the case, then in designing the device information security needs to be considered. If the device is to be operated successfully the software engineering needs to make sure the system is not vulnerable to be attacked through defects in the system.

Communication between software engineering and system engineering is important so that assumptions are not made which could lead to errors being introduced. Making clear specifications for the device is important to prevent failures later in development.

Breakthroughs in technology may make it more feasible to design safety critical heart device software to meet the patient needs. One challenge will be that once the high-speed communication has been solved there will be a large increase in the amount of information systems that are used. Even more systems will become safety critical requiring more development, verification and validation.

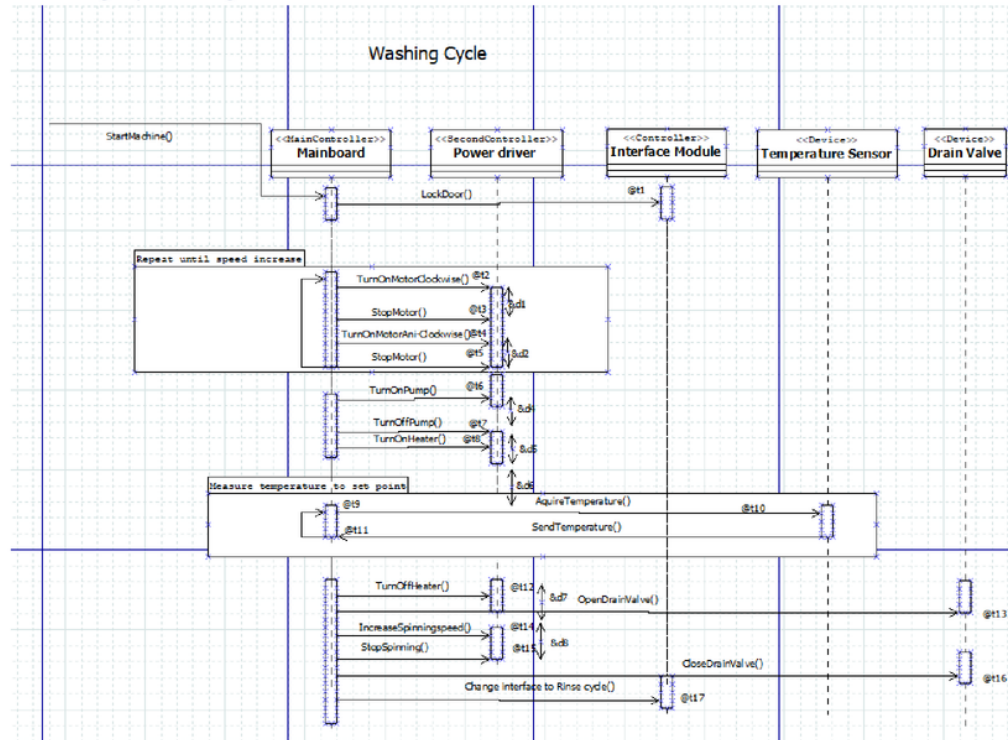
Development of future proofed systems is not always possible as there is a cost of money and time involved. Especially when there may only be a small advantage gained in increasing the lifetime of the device.[5]

## UML Diagram of the Washing Machine

From the timings in the first assignment of the wash and dry cycle of the washing machine a UML MARTE diagram has been produced. UML MARTE is a unified modelling language format.

There were three stages to the washing machine cycle; the wash cycle, the spin cycle and the rinse cycle. These are split into 3 diagrams using the UML diagram specification.

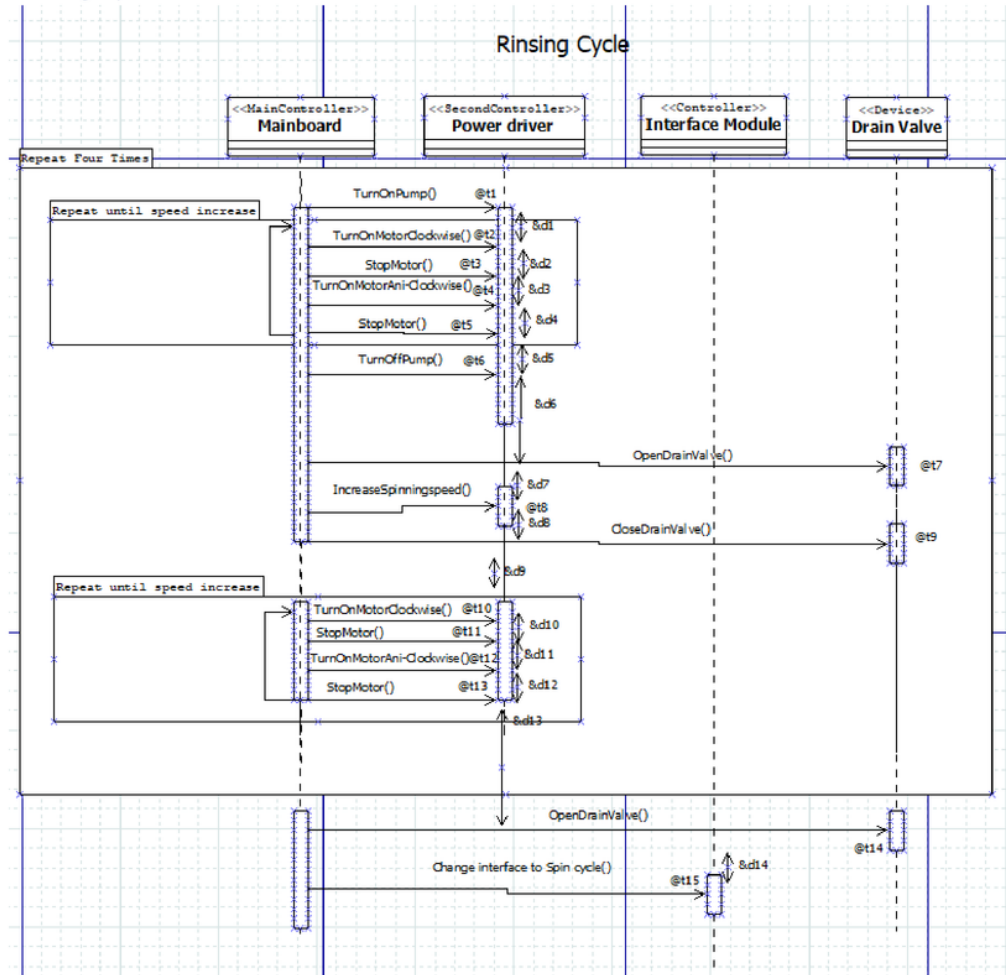
## Washing Cycle Diagram



## Key Diagram of the washing cycle

|                            |                         |    |                  |
|----------------------------|-------------------------|----|------------------|
| T1 - Lockingdoor           | {t1 <= [t0 +(1,s)]}     | D1 | {d1 < (5,min)}   |
| T2 - MotorClockwise        | {t2 <= [t1 +(1,min)]}   | D2 | {d2 < (5,min)}   |
| T3 - StopMotor             | {t3 <= [t2 +(1,s)]}     | D3 | {d3 < (5,s)}     |
| T4 - Motorantiwise         | {t4 <= [t3 +(1,min)]}   | D4 | {d4 < (5,min)}   |
| T5 - StopMotor             | {t5 <= [t4 +(1,s)]}     | D5 | {d5 < (5,s)}     |
| T6 – TurnOnPump            | {t6 <= [t5 +(30 ,min)]} | D6 | {d6 < (1 ,min)}  |
| T7 – TurnOffPump           | {t7 <= [t6 +(1,s)]}     | D7 | {d7 < (10 ,min)} |
| T8 - TurnOnHeater          | {t8 <= [t7 +(1,s)]}     | D8 | {d8 < (1,s)}     |
| T9 – acquireTemp           | {t9 <= [t8 +(5,min)]}   |    |                  |
| T10 – MeasureTemp          | {t10 <= [t9 +(30,min)]} |    |                  |
| T11 – Send Temp            | {t11 <= [t10 +(1,s)]}   |    |                  |
| T12 – TurnOffHeater        | {t12 <= [t11 +(1,s)]}   |    |                  |
| T13 – OpenDrain            | {t13 <= [t12 +(1,s)]}   |    |                  |
| T14 – IncreaseSpin         | {t14 <= [t13 +(1,s)]}   |    |                  |
| T15 -StopSpinning          | {t15 <= [t14 +(1,s)]}   |    |                  |
| T16 – CloseDrainValve      | {t16 <= [t15 +(1,s)]}   |    |                  |
| T17 – ChangeinterfaceRinse | {t17 <= [t16 +(1,s)]}   |    |                  |

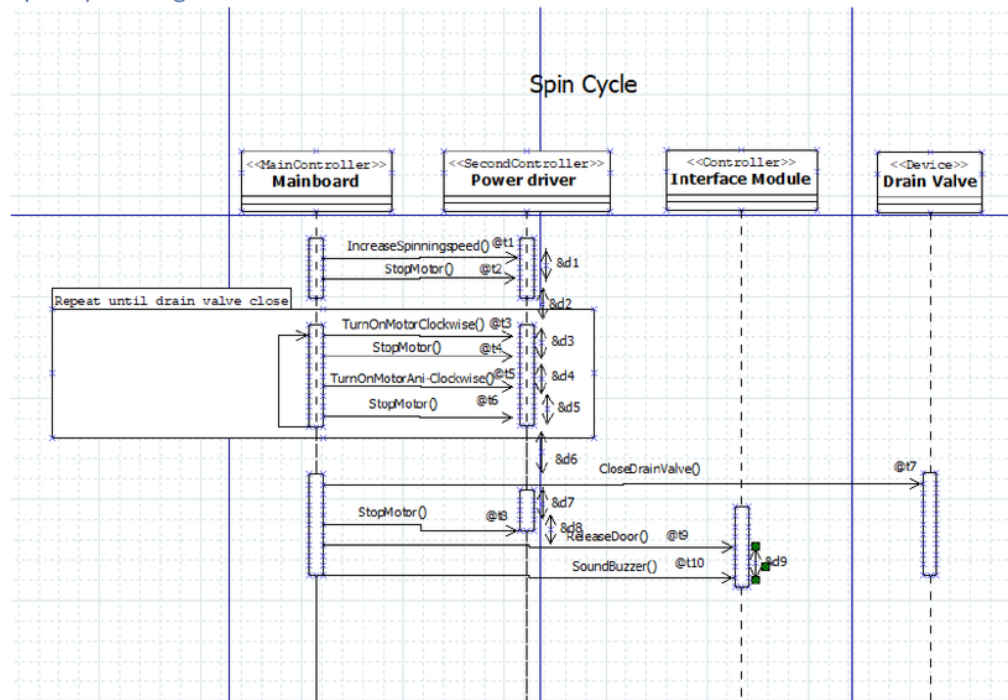
## Rinsing Cycle



Key Diagram of the Rinsing cycle

|                           |                          |     |                  |
|---------------------------|--------------------------|-----|------------------|
| T1 - TurnOnPump           | {t1 <= [t0 +(10,min)]}   | D1  | {d1 < (2,min)}   |
| T2 - MotorClockwise       | {t2 <= [t1 +(1,min)]}    | D2  | {d2 < (5,min)}   |
| T3 - StopMotor            | {t3 <= [t2 +(1,s)]}      | D3  | {d3 < (5,s)}     |
| T4 - Motorantiwise        | {t4 <= [t3 +(1,min)]}    | D4  | {d4 < (5,min)}   |
| T5 - StopMotor            | {t5 <= [t4 +(1,s)]}      | D5  | {d5 < (5,s)}     |
| T6 – TurnOffPump          | {t6 <= [t5 +(1 ,sec)]}   | D6  | {d6 < (10 ,min)} |
| T7 – OpenDrainValve       | {t7 <= [t6 +(5 ,min)]}   | D7  | {d7 < (1,sec)}   |
| T8 - IncreaseSpinSpeed    | {t8 <= [t7 +(5 ,min)]}   | D8  | {d8 < (3,sec)}   |
| T9 – CloseDrainValve      | {t9 <= [t8 +(1,sec)]}    | D9  | {d9 < (30,sec)}  |
| T10 – MotorClockwise      | {t10 <= [t9 +(1,min)]}   | D10 | {d10 < (5,min)}  |
| T11 – StopMotor           | {t11 <= [t10 +(1,s)]}    | D11 | {d11 < (5,s)}    |
| T12 – Motorantiwise       | {t12 <= [t11 +(1,min)]}  | D12 | {d12< (5,min)}   |
| T13 - StopMotor           | {t13 <= [t12 +(1,s)]}    | D13 | {d13 < (5,s)}    |
| T14 – OpenDrainValve      | {t14 <= [t13 +(5 ,min)]} | D14 | {d14 < (1,sec)}  |
| T15 - ChangeSpinInterface | {t15 <= [t14+(1,s)]}     |     |                  |

## Spin Cycle Diagram



## Key Diagram of the Spin cycle

|                        |                         |    |                 |
|------------------------|-------------------------|----|-----------------|
| T1 - IncreaseSpinSpeed | {t1 <= [t0 +(10,min)]}  | D1 | {d1 < (3,sec)}  |
| T2 - StopMotor         | {t2 <= [t1 +(1,s)]}     | D2 | {d2 < (5,s)}    |
| T3 - MotorClockwise    | {t3 <= [t2 +(1,min)]}   | D3 | {d3 < (5,min)}  |
| T4 - StopMotor         | {t4 <= [t3+(1,s)]}      | D4 | {d4 < (5,s)}    |
| T5 - Motorantiwise     | {t5 <= [t4 +(1,min)]}   | D5 | {d5 < (5,min)}  |
| T6 - StopMotor         | {t6 <= [t5 +(1,s)]}     | D6 | {d6 < (5,s)}    |
| T7 – CloseDrainValve   | {t7<= [t6 +(1,sec)]}    | D7 | {d7 < (30,sec)} |
| T8 - StopMotor         | {t8 <= [t7 +(1,s)]}     | D8 | {d8 < (5,s)}    |
| T9 – ReleaseDoor       | {t9 <= [t8 +(1 ,sec)]}  | D9 | {d9 < (1,min)}  |
| T10 – SoundBuzzer      | {t10 <= [t9 +(1 ,min)]} |    |                 |

## Real Time operating systems

Real-time operating system (RTOS) are mainly used in embedded systems. They deliver operations with a time constraint. They are needed where not only should the task have the correct result but there is also a timeframe for it to happen. These systems can process multiple tasks and the resources are managed explicitly to operate in real time. There is a balance between resource management and the hardware capacity. There are different types of RTOS from hard where the deadline must be met with no adjustment to a soft one where there is some flexibility in the missing of some deadlines.[6]

## Principals and Concepts

There are various principals for RTOS which include Multitasking, Interrupt Latency, Context Switching, Memory Management, Scheduling, Timer Resolution Inter and Task Communication. The concepts for RTOS include Scheduling, Deadlines, Hard/Soft Real Time and Predictability.

### Multitasking

The ability to multitask allows the CPU to be kept busy by splitting applications into multiple tasks. A conventional processor can only execute a single task a time whereas in a multitasking system by constantly switching it appears that many tasks are executing at the same time.[7] The kernel within the operating system allows the users to use the computer by executing multiple applications concurrently. [8]

### Interrupt Latency

The interrupt latency of the computer is the time that is needed for an interrupt to be generated. That is the time from the processor being signalled, carrying out the instruction and executing the system code and transferring it to the device handler interrupt. The shorter this is the more efficient the system is. [9]

### Context Switching

This is the time it takes to recognize that an event has arrived and to execute it. It should be kept to a minimum. It is important that the context of the system is the same for a task when it resumes after a suspension as otherwise an error may occur when the task is executed. This is an essential feature in RTOS because of multitasking. [10]

### Scheduling and dispatching

Scheduling is an ordered list that specifies how tasks are used in an order. There are criteria that are used for this such as timeliness. The list is designed so that the smallest amount of time is spent in the critical section. Algorithms are used to decide which tasks are executed, delayed or must wait for another task to be completed. Dispatching is where access is granted to the most appropriate task depending on its priority and deadline. [10]

### Examples of RTOS

Two examples of RTOS are VxWorks and FreeRTOS. Both VxWorks and FreeRTOS both have a small kernel size and use a small amount of RAM. VxWorks is a paid for software whereas FreeRTOS is open source. The VxWorks has a faster interrupt latency than FreeRTOS which is important if many interrupts are needed for many events. FreeRTOS supports many microprocessor platforms whereas VxWorks supports fewer but is one of the widely used commercial systems. [11][12][13]

The main features for VxWorks are it uses pre-emptive and round robin algorithms and a fast interrupt response. It produces high performance, determinism and flexibility because all the memory is accessible to developers. The memory is isolated in real time process from other user mode applications to protect it. FreeRTOS main features are that it supports many platforms and has an efficient timer that doesn't use CPU cycles unless it is invoked. The queues used for interprocess communication are different to other RTOSs. The data is passed by values instead of references therefore it is simple. Memory management can be integrated using C libraries of free and allocating memory. [13]

### Device Driver

A device driver is a piece of software that has specific code that is used to carry out actions to run a device. The device driver acts as a translator between the device and the operating system. Instead

of accessing the device itself the operating system calls specific functions in the driver software to carry out the tasks on the device. A device driver is also used because it moves potentially dangerous operations out of the user code and into the system code.

### Serial Driver

A serial driver is a 16550 universal asynchronous receiver-transmitter(UART).It is used in many devices because it is highly reliable and can be used over long distances. [14]A serial driver is used for devices attached to serial ports to either send or request data on the input/output signals of the processor or for reading or writing from specific memory locations. They also intercept interrupt signals from the ports that indicate that data is available. A device driver is used because the resources are shared between different ports and there is a need to control them to avoid conflicts. [15]The UART uses a standard protocol of RS232. The UART consist of first in first out (FIFO) transmitter and receiver, baudrate generator logic, modem signals logic and register groups. With these the data sent from the operating system and the data coming from the device can be interpreted. [14] In the UART when it receives or sends data interrupts are used so that the data can be buffered until it is ready to be processed. In a serial interrupt there are two flags that can be triggered, the receive interrupt and the transmit interrupt, depending on whether the data was received or sent. This however takes up some of the CPU time and by using a DMA controller large amounts of data can be put through the system without using the CPU leaving it free to perform other tasks. The DMA uses a pointer from the start of a source buffer and a pointer to a destination buffer and the amount of data to be transferred. DMA is more complicated if the data is not in one block. It is also not as good if the size of the data to be transferred is not known.[16]

### References

- [1] "Critical systems – Software Engineering 10th edition." [Online]. Available: <https://iansommerville.com/software-engineering-book/web/critical-systems/>. [Accessed: 10-Apr-2019].
- [2] I. Sommerville, *Software engineering*. Pearson, 2016.
- [3] K. Houliotis, P. Oikonomidis, P. Charchalakis, and E. Stipidis, "Mission-Critical Systems Design Framework," *Adv. Sci. Technol. Eng. Syst. J.*, vol. 3, no. 2, pp. 128–137, Mar. 2018.
- [4] "Mission-critical vs. Business-critical applications: HUH? | ActiveState." [Online]. Available: <https://www.activestate.com/blog/mission-critical-vs-business-critical-applications-huh/>. [Accessed: 14-Apr-2019].
- [5] J. C. Knight, "Safety critical systems: challenges and directions," pp. 547–550, 2003.
- [6] "Why RTOS and What is RTOS?" [Online]. Available: <https://www.freertos.org/about-RTOS.html>. [Accessed: 09-Apr-2019].
- [7] "Difference between Multiprogramming, Multitasking, Multithreading and Multiprocessing | 8 BIT AVENUE." [Online]. Available: <http://www.8bitavenue.com/2012/10/difference-between-multiprogramming-multitasking-multithreading-and-multiprocessing/>. [Accessed: 20-Dec-2017].
- [8] "RTOS kernel rapidly switches between tasks." [Online]. Available: <https://www.freertos.org/implementation/a00004.html>. [Accessed: 14-Apr-2019].
- [9] J. Yin, *The Definitive Guide to the ARM Cortex-M3*. Elsevier, 2010.

- [10] P. Zhang, *Advanced Industrial Control Technology*. Elsevier, 2010.
- [11] "FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions." [Online]. Available: <https://www.freertos.org/>. [Accessed: 09-Apr-2019].
- [12] H. Shah, "How to select the right RTOS for your IoT needs: RTOS comparison," *Simform*, 2017. [Online]. Available: <https://www.simform.com/iot-rtos-selection/>. [Accessed: 14-Apr-2019].
- [13] S. Jape, M. Kulkarni, and P. Pawade, "Comparison of Contemporary Real Time Operating Systems," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 11, pp. 416–419, 2015.
- [14] Zhichao Zhang and Wuchen Wu, "UART integration in OR1200 based SOC design," in *2010 2nd International Conference on Computer Engineering and Technology*, 2010, pp. V7-331-V7-334.
- [15] "Serial Controller Drivers Overview (Windows Drivers)." [Online]. Available: [https://technet.microsoft.com/en-us/jj241971\(v=vs.96\)](https://technet.microsoft.com/en-us/jj241971(v=vs.96)). [Accessed: 14-Apr-2019].
- [16] Huimei Yuan, Junyou Yang, and Peipei Pan, "Optimized design of UART IP soft core based on DMA mode," in *2010 5th IEEE Conference on Industrial Electronics and Applications*, 2010, pp. 1907–1910.

### FINAL GRADE

85/100

### GENERAL COMMENTS

#### Instructor

Critical System (10) 10  
Cat heart device (15) 13 Implications on design (15) 9  
UML diagrams (20) 18  
Research RTOS (20) 17  
Device driver Int/DMA (20) 18

Very good description of critical systems. Although the discussion of the design implications was a bit too vague. What can be done at the design stage to help ensure the design is safe? Really nice UML diagrams with a lot of timing information shown. They might even be a little overcomplicated, but this is a minor criticism. Very nice evaluation of RTOS. The comparison of the RTOSs might have been clearer if you had first identified key features for the comparison. Nice description of serial driver.

---

PAGE 1

---

PAGE 2

---

PAGE 3

---

PAGE 4



#### Comment 1

Hmm, OK I guess it might be connected to the Internet

---

PAGE 5

---

PAGE 6

---

PAGE 7

---

PAGE 8

---

PAGE 9

---

