University of Colorado Boulder

EXERCISE 6

BY
Amreeta Sengupta and Ridhi Shah

04/18/2019

QUESTION 1

Therac-25

Therac-25, developed by Atomic Energy of Canada Limited (AECL), was a radiation therapy machine which was used to destroy cancerous tissues and tumors by acceleration of electrons to generate energy beams. Therac-25 was preceded by Therac-6 and Therac-20, each of which were designed to use hardware interlocks to monitor safety critical issues. Therac-25 on the other hand was designed to be a solely computer-controlled device and it handled the safety critical issues using software in order to save time and money. Therac-25 had three modes of operation, Electron beam therapy, Megavolt X-ray therapy and Field Light mode. The software was implemented using assembly language and had several loopholes which resulted in high current electron beam being directly delivered to patients. The software faults included the incorrect setting of the electron beam for X-ray mode without the target being in place when an operator selected the X-ray mode by mistake before switching to electron mode. Another fault was the erroneous activation of the electron beam during the field light mode, during which the beam scanner is inactive, or the target isn't in place. The high current electron beam struck the patients with approximately 100 times the required doses of radiation which led to fatal consequences and it involved at least six casualties where human lives were lost. The root causes of failure included:

- The software lacked self-checks and error detection which would point out the inconsistencies and did not contain any check for analyzing whether the machine and the software were operating correctly. The reactions of the patients were the only indication of the severity of the problem. The error codes were not well documented in the manual.
- Poor software reuse from older modules which was assumed to guarantee the safety of the system due to its extensive use previously. The versions preceding this had hardware interlocks and were insufficient to prevent the software defects.
- There was no proper synchronization between the equipment control task and the operator interface which led to occurrence of race conditions on swift switching between device modes.
- The software used flag variables by incrementing it instead of setting it to a fixed value which eventually lead to overflow and the flag variable was again set to zero which lead to bypass of the safety checks. Overall, the software was not designed to take care of the worst-case scenarios.

ARIANE 5 Flight 501 Failure

The Ariane 5 launcher which was launched on 4 June 1996 ended in failure. It exploded when it reached the altitude of 3700m, only about 40 seconds after the launch. The weather was perfect during the launch day and also there was no risk of lightning, the causes of failure known after investigation was that after 36 seconds of normal behavior of the system, the backup inertial reference system and the active inertial reference system failed. Inertial reference system (SRI) is a system used to measure the altitude and the movements of the launcher in space. The SRI data was transferred to the On Board Computer (OBC) which executes the flight program.2 SRIs are used i.e. one is active and one is for backup in case the other fails. The main reason for the failure was that the SRI 2 did not send correct data to the OBC since there was failure due to some software exception. The backup SRI also could not be used since it was ceased earlier due to the same reason.
This SRI exception was caused during the conversion from 64-bit floating point to 16-bit signed value, the 64-bit number was converted to a larger value than the value that could be represented by 16 bit which resulted in an operand error. The piece of code where this error occurred was not required after liftoff, but it was kept as a continuation to Arian 4 which required this code for 40 seconds after liftoff, this operand error occurred due to some unprotected variables in the code and an unexpected high value of the inertial function .
To Summarize, the main causes of failure were that the backup system failed and the erroneous data which occurred as a result of unprotected variables, software exceptions and overflow of the data value, was sent to the OBC which caused the flight to change its path and lead to high aerodynamic load which caused the failure of the system.

Mars Climate Orbiter

The mars climate orbiter was a 338 kg space probe launched by NASA to study the Martian Climate and Martian Atmosphere. The communication with the spacecraft was lost because the spacecraft went into orbital insertion due to the error in the ground-based software, which produce an output in non SI units instead of SI units which was specified in a contract between NASA and Lockheed.
During the trajectory correction Maneuver-4 it was decided that the spacecraft would be placed at an optimal position for an insertion so that the spacecraft would come to an altitude of 226km from Mars. but later during a week before the orbital insertion the it was indicated that the altitude should be around 10 to 170Km and 24 hours before the insertion , it was said that the orbiter be placed at 110Km which was the altitude at which the mars obiter was thought to be capable of surviving. But the spacecraft failed, and it was destroyed in the atmosphere or re-entered in the heliocentric space and after the failure it was noticed that this was at 57km.
The main cause of this failure was that the ground software supplied by Lockheed martin gave results in the United states customary unit, which was contrary to the Software interface specification, the second system which used this data expected it to be in the SI units specified by the Software interface specifications. Another software that was used to calculate the total impulse produced by thruster firings gave results in Pound-force seconds and finally the trajectory calculation that updated the predicted position of the spacecraft used all these results and expected these values to be in Newton seconds i.e. SI unit. All this discrepancy between the calculated positions resulted in the discrepancy in the orbital insertion.
Thus, to summarize, the differences in the Units required by different softwares to calculate the position of spacecraft in real time caused the spacecraft to fail.

Ranking:

Out of all these we rank the 3 worst real time mission critical errors in the following order:

- The worst real time mission critical error occurred in Therac-25 where human lives were lost as a result of poor programming practices. Software interlocks, without taking care of all bugs, were used in order to save resources in such a critical device at the cost of human lives.
- The 2nd worst mission critical failure was caused by a simple software exception and overflow error in Ariane 5 which lead to a lot of destruction and a loss of millions of dollars
- The 3rd worst failure was the Mars orbited caused to due errors in the measuring units which resulted in the entire spacecraft to collapse.

QUESTION 2

Key findings on the USS Yorktown

- The Yorktown Smart Ship was built with the purpose of reducing manpower, maintenance and cost. This smart ship was designed such that the applications were implemented using Microsoft Windows NT which required fewer sailors for the control of the key ship functions.
- However, the entry of faulty data into the system database resulted in an overflow which crashed the entire program and crippled the operations of the entire ship. This left the Yorktown Smart Ship stranded and dead in the middle of water for about two hours and 45 minutes and had to be later towed into the Naval base at Norfolk.
- The major reason of the crash was found to be the fact that the Yorktown's Standard Monitoring Control System administrator had entered a zero into the data field for the Remote Data Base Manager which resulted in an overflow in the database and crashed al the LAN consoles and miniature remote terminal units. Consequently, the control of the propulsion system was lost as the computers were not able to divide a number by zero.
- Later the Atlantic Fleet officials were reported to have said that henceforth, the program administrators were trained to bypass such bad data field error and the change the value on occurrence of any such problem.
- However, according to DiGiorgio, who serviced the automated control systems on the Navy ships, the source of the ship's computer problems was the NT operating system which is known to have some failure modes. Ron Redman, deputy technical director of the Fleet Introduction Division of the Aegis Program Executive Office, is also of the opinion that NT is not fully refined and has resulted in numerous software shutdowns. It is a better system while transferring data and information while for control of equipment and machinery, Unix is a better and more reliable system.

Alternate key findings

- According to https://www.wired.com/1998/07/sunk-by-windows-nt/, the software glitches that occurred on the Yorktown Smart Ship appeared to be more political than technical. The selection of Windows NT operating system in such a critical environment was singularly decided by a small group of engineers without much knowledge. Things were rushed and there was no real prototype that was implemented prior to this and the engineers just tried to make things work along the way.
- On the other hand, Singley believed NT was chosen because it was thought to have a more friendly graphical user interface as compared to Unix systems. Bill Gates even nominated the Smart Ship program for the ComputerWorld/Smithsonian Awards Program.
- John Kirch, a networking consultant and Microsoft certified professional, in his white paper, mentioned Windows NT Server 4.0 is no match for any Unix operating system and Unix is much more reliable as compared to Windows NT and also offers free, open-aourse Linux operating system.
- According to https://medium.com/@bishr_tabbaa/when-smart-ships-divide-by-zer0-uss-yorktown-4e53837f75b2, Captain Richard Rushton, the commanding officer of the Yorktown, believed that similar program crashes had occurred twice before since the Smart Ship installation due to incorrect values entered into the Remote Database Manager (RDM) and on restarting the system, the RDM values were reset and the ship's system restarted.

Root cause analysis based on reading

- We believe that the main cause of the fatal accident was that 0 was entered in the database and there was no mechanism to check the values in the database, this error could have been avoided if the values that were entered were checked beforehand.

- We do not think that it was entirely the operator's fault, the operator entered a value which he though suited best for resetting the valve, we think that there should have been a checking mechanism in the Windows NT OS to check for the divide by 0 error.

- The windows OS did not provide a protection against a divide by 0 error and since the OS is tightly coupled, if one subsystem fails, then it causes the entire system to fail. If the system was built such that it was loosely coupled, then the system failure could be avoided even if one subsystem failed.

Would RedHawk Linux help for large scale mission critical systems like the USS Yorktown?

- We believe that the RedHawk Linux OS would be better than the windows OS and would help in reducing certain errors in the system.
- Though the windows NT OS provides a user-friendly interface, the RedHawk Linux provides a better performance when it comes to meeting the deadlines in the Real-Time systems.
- The RedHawk Linux OS provides high speed and performance since it has a single kernel programming environment which helps in giving a better performance to the system and provides direct control to the operations. On the other hand, windows NT OS uses hybrid kernel which is not as efficient as Linux OS for real time applications

Final exercise proposal

Overview and Design description

We aim to develop a system for the detection and tracking of a circular object. The system will first detect the shape of the object, determine its color and then classify the object as a golf ball based on the previous shape and color detection using the OpenCV libraries. It is also the secondary objective to develop a ball following mechanism by tracking the position of the circular object in the frame and driving two motors to follow the golf ball.

First the logitech camera and the openCV libraries are configured on the Jetson TK1 board. We use OpenCV for detection of shape and color of the object. To determine the shape, we use the hough elliptical transform and develop the algorithm for detecting the circular shape. Then we detect the colour of the object, if white color is detected then we determine that it is the object to be tracked and the robot has to follow the object if it is moving. For determining the direction in which the robot has to move, we calculate the offset of the circle from the entire image captured, i.e. we take the difference between the center of the image and the center of the circle and depending on its value a decision is made about the direction in which the robot will move. If the offset is positive, then the robot moves in the right direction and if it is negative the robot turns left otherwise it goes straight.

We have chosen the Rate monotonic policy for the implementation of services, the application has been divided into two services, one service is used for the image processing part where the shape and color are detected, and the offset value is computed. This offset value is used in the Motor control task where the motor uses this and determines the direction of the movement of the robot. We have calculated the WCET and the deadlines based on how much time will it take to execute the Hough elliptical algorithm, the color detection algorithm and also calculating offset of the image. Similarly, we have calculated WCET for the motor control task based on the decision-making algorithm which uses the offset data and determines the direction. We run cheddar using the WCET, T and D for the system. We can see that this service set is schedulable. The 2 tasks share the data of the offset and the tasks have to be synchronized, this synchronization will be taken care of in the implementation.

Overall, the system working includes the following steps:

1. The Logitech C200n camera is used to detect an object.
2. Once, an object is detected, its shape is determined.
3. If a circle is detected, the system then proceeds to determine its color.
4. If a white circle is detected (which is basically a golf ball), the offset of the circle is calculated from the center of the image which in turn will issue turning orders to the motor.
5. To control the speed of the motor, the PWM is configured on the GPIO pins.
6. If a moving object is detected, the motors are configured and driven in a manner such that the robot follows the object. Thus, it helps in keeping track of the object.

Citations

- https://pdfs.semanticscholar.org/549a/2bff5d595e759156fb2cb9bad14d5a2fc892.pdf:This paper tells us about the use of different Scheduling policies.
- https://www.researchgate.net/publication/4277229_PARTES_Performance_Analysis_of_Real-Time_Embedded_Systems: This paper tells us about the performance analysis of Real-Time Embedded Systems.
- http://www.allresearchjournal.com/archives/2015/vol1issue9/PartG/1-9-20.pdf: This paper tells us about use of OpenCV in object tracking and detection.

Personal contribution

The OpenCV configuration must be done and algorithm for detection of shape and color of the object has to be developed. After detecting that the object is a white ball, the offset is calculated by taking the difference between the center of the circular object and the center of the entire image is calculated. The offset calculated is used to determine whether the object has to turn left or right, and the motor is controlled according to it.

Ridhi - Shape detection algorithm uses the Hough transform to detect the circle. The coordinates of the center of the circle are also displayed. If the shape detected is a circle, then we go ahead and detect the color of the object. The color detection of the object and determining the color depending on the grayscale and setting up of other parameters will be done and coded accordingly, then a code is developed which computes the difference between the center of the entire image. The synchronization between the 2 tasks, i.e. the shape and color detection and the motor control will also be taken care of.

Amreeta - According to the offset values computed, the motor has to be controlled. Initially, the motor configuration has to be done and then a power supply is given to the motor. Next, the motor driver PWM pins are configured using GPIO. The proper functioning of the motor and movements of the motor, according to the PWM input, will be tested first. Then the offset values determined previously, will be used and the code is developed to control the motor according to these values. A proportional control algorithm will be used for this and depending on the offset values, we will determine the direction in which the motor will move i.e. left or right. If the offset value is positive, then the robot moves to the right and if it is negative the robot moves to the left.

Functional (capability) requirements for system design

- Shape Detection
  The circular shape is detected by using the Hough Elliptical Transform algorithm using OpenCV. The Hough transform is the feature extraction method used in image processing, The Hough transform was originally used to identify the lines in the image, but the Hough transform has been extended to identifying arbitrary shapes, most commonly circles or ellipses. HoughCircles function can be used to find the circles in a greyscale image.

- Color Detection
  This is done by first defining the upper- and lower-pixel values and then inRange function is used to return a mask which determines the pixels that fall in the defined range. Bitwise_and function can be used to apply the mask on the image and determine the color.

- Object Tracking
  The camera is used to capture the image of the object at each instance. Hence, as the object moves, the coordinates of the object change. We can continuously check for this change to calculate the offset of the circle from the center of the image and determine the direction in which the motor will be driven.

- Motor Control
  Motor Driver TB834A2 is a dual motor driver which is used to interface two DC motors. It is used to drive the motor to change the direction of the robot. For taking turns, one wheel is stopped while the other wheel will be turned in the desired direction.

- Proportional Controller
  On tracking the object, the distance between the center of the circle and the center of the image is calculated to determine the offset value. If this value is less than zero, then the proportional controller will decide to make a left turn and otherwise it will take a right turn. The proportional controller basically decides the direction in which the motor will be driven.

- Synchronization between Tasks
  The code runs a sequencer with two services. The sequencer runs at HCF of request periods of the two tasks. The priority is assigned based on Rate Monotonic Policy which states that higher priority will be given to a service with higher frequency. According to the frequency of the services, we make an algorithm to post the semaphores to maintain the schedule. The semaphores get posted and then the sequencer goes to sleep. When the sequencer is in sleep, the semaphores which are posted run to completion. This cycle goes on and on for the sequencer period.

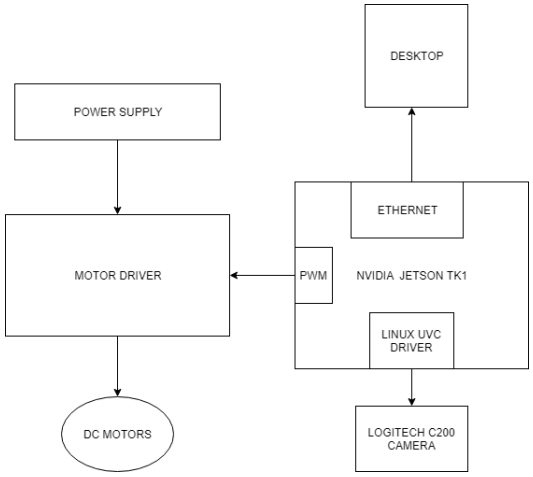
Completeness:

The Rate monotonic policy is used here since 2 tasks are independent. The motor control task depends on the image processing task if the values are to be updated, otherwise it is an independent task. The

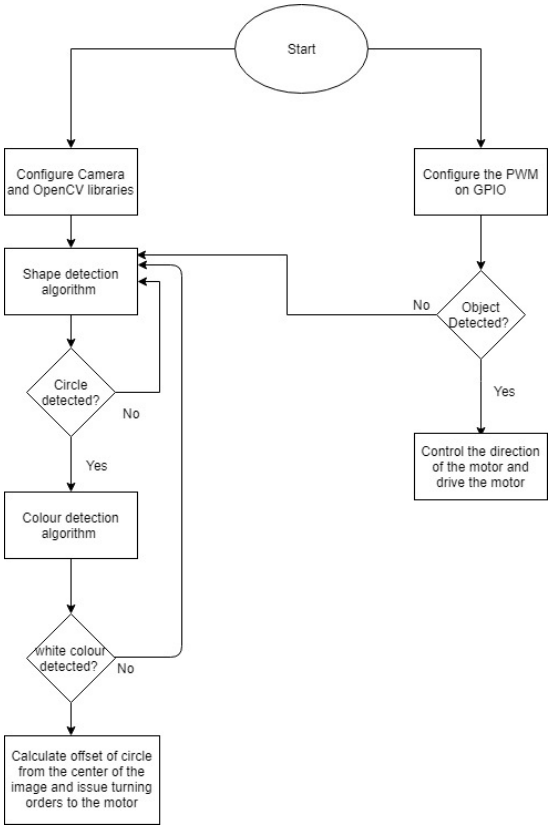
WCET, T and D are calculated accordingly and scheduled. The 2 service sets are schedulable. Thus, the system is designed to meet the requirements.

High level system and software design

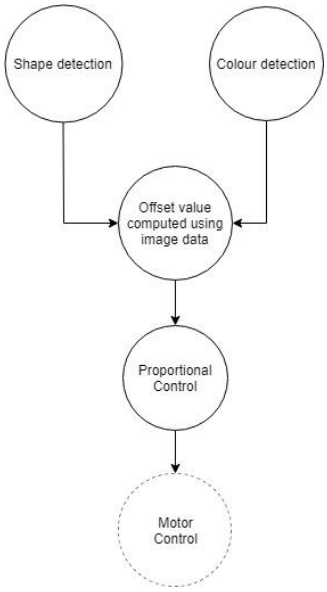
BLOCK DIAGRAM



FLOWCHART



DATAFLOW DIAGRAM



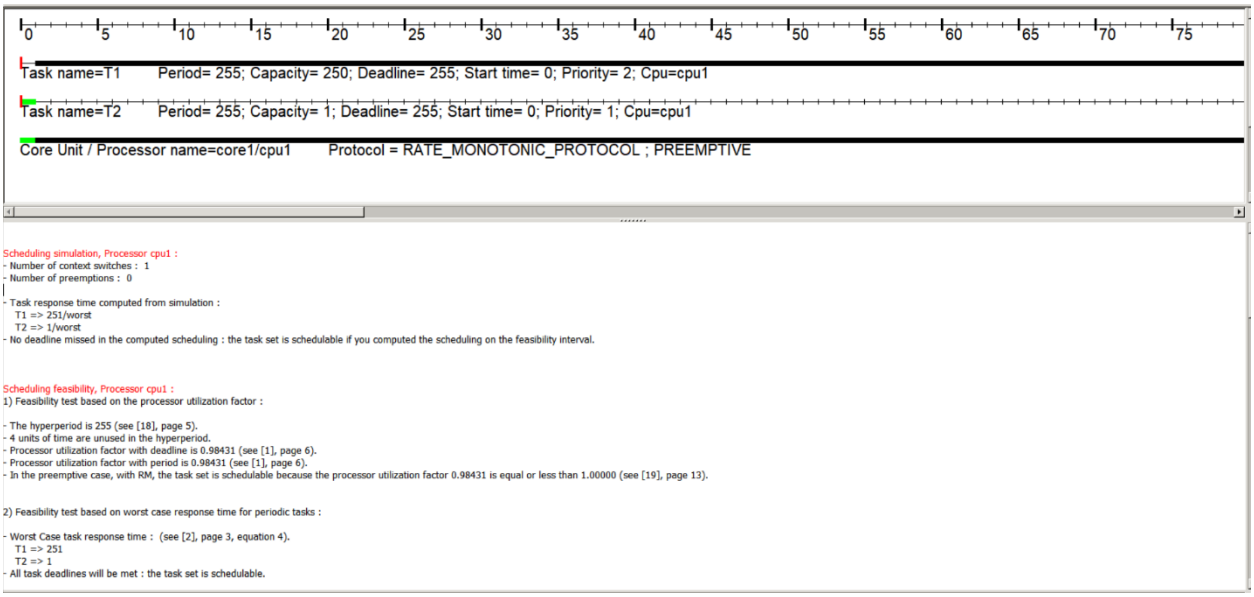
Functional Description

The camera is used to capture the image of the object at each instance. From this, the color and shape of the object is detected through the use of the color and shape detection algorithms. As the object moves, the coordinates of the object change. We can continuously check for this change to calculate the offset of the circle from the center of the image. This offset value is basically the distance between the center of the circle and the center of the image. The value will help the proportional controller determine the direction in which the motor should turn. If this value is less than zero, then the proportional controller will decide to make a left turn. Otherwise, it will take a right turn. The motor will then be driven by the motor controller in the direction determined. This will result in the robot following the golf ball.

Real-Time Services and Requirements

Services

Tasks (S_i)	T_i	WCET (C_i)	D_i
Image processing	255 ms	250 ms	255 ms
Motor Control	255 ms	1 ms	255 ms



Service 1 is the image processing task which is basically used to detect the object and identify its shape to determine if it's a circle. If its shape is circular, then its color is determined and if its is identified to be white, then golf ball is detected. On detecting the object, the distance between the center of the circle and the center of the image is calculated to determine the offset value.

Service 2 use is basically used to configure and control the motor. It uses the offset value determined above to control and drive the motor. If this value is less than zero, then the proportional controller will decide to make a left turn. Otherwise, it will take a right turn. This will basically result in the robot following the golf ball.

C_i and WCET descriptions specification

Motor control thread only executes PWM and control signals for H-bridge. Any update in motor control is issued by the image processing task, in which case few register writes are required to change the GPIOs and PWM registers. Jetson, having its clock in GHz range, makes 1 ms a sufficient overestimation for few register writes. Estimation of Hough Elliptical Transform from previous exercise is 90 ms for 640 X 480 resolution on Jetson. This would further require color detection algorithm and proportional control which makes 250 ms an appropriate WCET estimate. This value will be updated with actual data in final report. Also, execution time of these algorithms depends on image complexity like number of circles in an image and the gradient of the RGB value of neighboring pixels. To estimate WCET, it is required to assume maximum number of possible circles in the image and color variation. Here, we will

assume two circles i.e. two golf balls and sharp color contrast as the ball will be white in green background of grass.

T_i and D_i specification

Schedule is implemented using RM Policy. Hence, request time and deadlines are same. Estimated WCET for image processing task is 250 ms and motor control task has WCET of 1 ms only. These values suggest that request period above 251 ms will make the overall schedule feasible even though RM LUB test will fail because of processor utilization being almost one. This is not an issue as RM LUB is only a sufficient test, not a necessary one. So, we have set request period for image processing task as 255 ms which will also account for kernel preemption in Linux. Also, any update in motor control won't occur until the Image processing task is computed. Thus, request period for motor control task is also set to 255ms.

REFERENCES

- https://en.wikipedia.org/wiki/Therac-25
- https://bohr.wlu.ca/cp164/therac/therac25.htm
- sunnyday.mit.edu/accidents/Ariane5accidentreport.html
- https://en.wikipedia.org/wiki/Mars_Climate_Orbiter
- https://gcn.com/Articles/1998/07/13/Software-glitches-leave-Navy-Smart-Ship-dead-in-the-water.aspx?Page=1
- https://canvas.colorado.edu/courses/24814/files/2521713?module_item_id=1108496
- https://medium.com/@bishr_tabbaa/when-smart-ships-divide-by-zer0-uss-yorktown-4e53837f75b2
- https://www.wired.com/1998/07/sunk-by-windows-nt/
- https://docs.opencv.org/3.4.2/dd/d1a/group__imgproc__feature.html#ga47849c3be0d0406ad3ca45db65a25d2d
- https://www.pyimagesearch.com/2014/08/04/opencv-python-color-detection/
- https://www.pololu.com/product/713
- https://www.pyimagesearch.com/2016/02/08/opencv-shape-detection/