Exercise 5 Report/Write-Up

Dhiraj Bennadi and Patrick Bales-Parks

April 2, 2022

ECEN 5623- Spring 2022

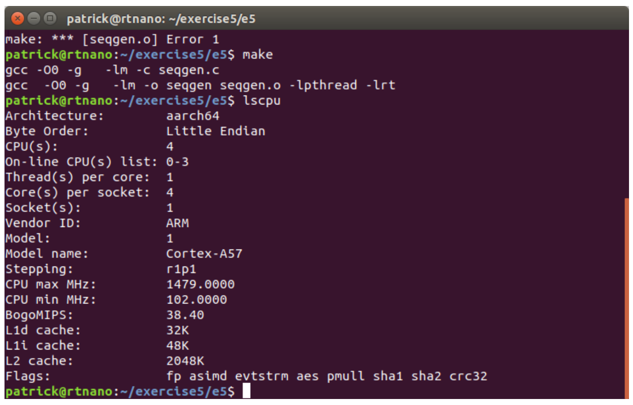
1 Problem 1

1.1 problem

Download seqgen.c and seqgen2x.c (or unzip them from the provided zip file for this exercise) and build them in Linux on the Altera DE1-SOC, Raspberry Pi or Jetson board and execute the code. Describe what it is doing and make sure you understand how to use it to structure an embedded system. Determine the worst case execution time (WCET) for each service by printing or logging timestamps between two points in your code or by use of a profiling tool. Determine D, T, and C for each service and create an RM schedule in Cheddar using your WCET estimates. Calculate the percentage CPU utilization for this system.

1.2 Answer

As was suggested in the comments of the seqgen files, cpu check were preformed on the Jetson Nano prior to initialization.

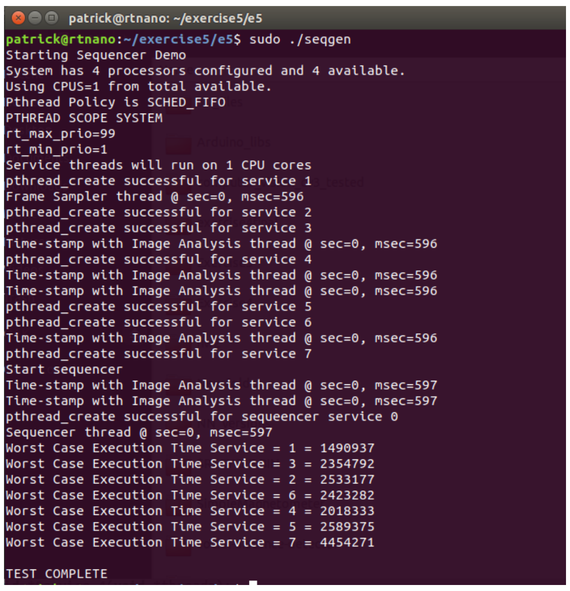


```
make: *** [seqgen.o] Error 1
patrick@rtnano:~/exercise5/e5$ make
gcc -O0 -g   -lm -c seqgen.c
gcc  -O0 -g   -lm -o seqgen seqgen.o -lpthread -lrt
patrick@rtnano:~/exercise5/e5$ lscpu
Architecture:        aarch64
Byte Order:          Little Endian
CPU(s):              4
On-line CPU(s) list: 0-3
Thread(s) per core:  1
Core(s) per socket:  4
Socket(s):           1
Vendor ID:           ARM
Model:               1
Model name:          Cortex-A57
Stepping:            r1p1
CPU max MHz:         1479.0000
CPU min MHz:         102.0000
BogoMIPS:            38.40
L1d cache:           32K
L1i cache:           48K
L2 cache:            2048K
Flags:               fp asimd evtstrm aes pmull sha1 sha2 crc32
patrick@rtnano:~/exercise5/e5$
```

Figure 1: Jetson Nano lscpu

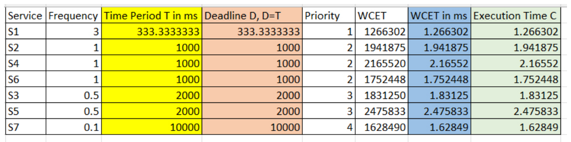
1.2.1 Explanation and Execution of seqgen.c

The seqgen.c file was written to demonstrate the use of sequences in a Real-Time system. The program is designed to call the sequencer function as the main thread with highest priority. The sequences are called as threads, inside the main thread, with priorities lower than the main sequencer thread. These priorities are determined by the RM policy. in the original state, the sequencer operates at 30Hz. The sequencer loops through the services, only activating them at the designated service frequency. These frequencies are: s1 = 3Hz, s2 = 1Hz, s3 = 0.5Hz, s4 = 1Hz, s5 = 0.5Hz, s6 = 1Hz and s7 = 0.1Hz. The sequences are also protected by individual semaphores. Execution of seqgen.c and the values used for analysis can be seen in the figures below.



```
patrick@rtnano: ~/exercise5/e5
patrick@rtnano:~/exercise5/e5$ sudo ./seqgen
Starting Sequencer Demo
System has 4 processors configured and 4 available.
Using CPUS=1 from total available.
Pthread Policy is SCHED_FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
Service threads will run on 1 CPU cores
pthread_create successful for service 1
Frame Sampler thread @ sec=0, msec=596
pthread_create successful for service 2
pthread_create successful for service 3
Time-stamp with Image Analysis thread @ sec=0, msec=596
pthread_create successful for service 4
Time-stamp with Image Analysis thread @ sec=0, msec=596
Time-stamp with Image Analysis thread @ sec=0, msec=596
pthread_create successful for service 5
pthread_create successful for service 6
Time-stamp with Image Analysis thread @ sec=0, msec=596
pthread_create successful for service 7
Start sequencer
Time-stamp with Image Analysis thread @ sec=0, msec=597
Time-stamp with Image Analysis thread @ sec=0, msec=597
pthread_create successful for sequencer service 0
Sequencer thread @ sec=0, msec=597
Worst Case Execution Time Service = 1 = 1490937
Worst Case Execution Time Service = 3 = 2354792
Worst Case Execution Time Service = 2 = 2533177
Worst Case Execution Time Service = 6 = 2423282
Worst Case Execution Time Service = 4 = 2018333
Worst Case Execution Time Service = 5 = 2589375
Worst Case Execution Time Service = 7 = 4454271

TEST COMPLETE
```

Figure 2: seqgen.c execution

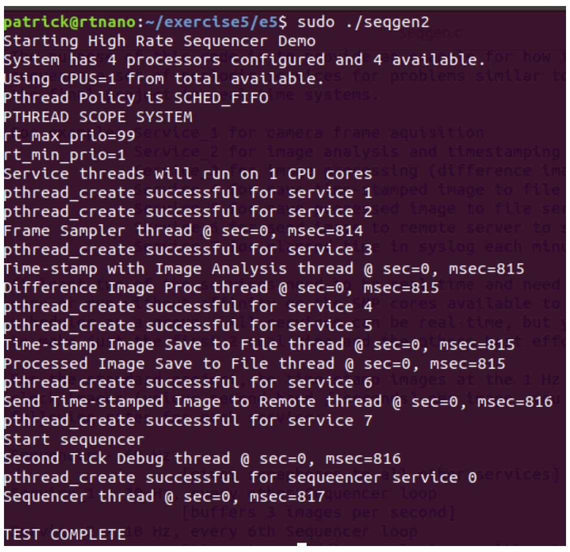


Service	Frequency	Time Period T in ms	Deadline D, D=T	Priority	WCET	WCET in ms	Execution Time C
S1	3	333.3333333	333.3333333	1	1266302	1.266302	1.266302
S2	1	1000	1000	2	1941875	1.941875	1.941875
S4	1	1000	1000	2	2165520	2.16552	2.16552
S6	1	1000	1000	2	1752448	1.752448	1.752448
S3	0.5	2000	2000	3	1831250	1.83125	1.83125
S5	0.5	2000	2000	3	2475833	2.475833	2.475833
S7	0.1	10000	10000	4	1628490	1.62849	1.62849

Figure 3: seqgen.c values

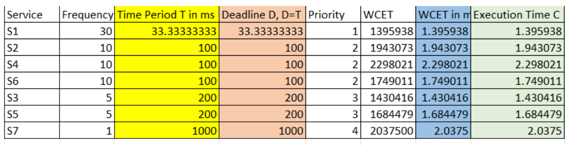
1.2.2 Explanation and Execution of seqgen2.c

The seqgen2.c program demonstrates the same patterns as seqgen.c, but is adapted to operate at twice the frequency of 60Hz. Since the primary thread is the sequencer function, this is the thread changed from 30Hz to 60Hz. This change is accomplished by changing the "timespec delaytime" from 33333333 to 16666666. This changes the delay in nano seconds equivilent to 60Hz. Changing this parameter also means that the service frequencies had to be adjusted. The new service frequncies are: s1 = 30Hz, s2 = 10Hz, s3 = 5Hz, s4 = 10Hz, s5 = 5Hz, s6 = 10Hz and s7 = 1Hz. Execution of seqgen2.c and the values used for analysis can be seen in the figures below.



```
patrick@rtnano:~/exercise5/e5$ sudo ./seqgen2
Starting High Rate Sequencer Demo
System has 4 processors configured and 4 available.
Using CPUS=1 from total available.
Pthread Policy is SCHED_FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
Service threads will run on 1 CPU cores
pthread_create successful for service 1
pthread_create successful for service 2
Frame Sampler thread @ sec=0, msec=814
pthread_create successful for service 3
Time-stamp with Image Analysis thread @ sec=0, msec=815
Difference Image Proc thread @ sec=0, msec=815
pthread_create successful for service 4
pthread_create successful for service 5
Time-stamp Image Save to File thread @ sec=0, msec=815
Processed Image Save to File thread @ sec=0, msec=815
pthread_create successful for service 6
Send Time-stamped Image to Remote thread @ sec=0, msec=816
pthread_create successful for service 7
Start sequencer
Second Tick Debug thread @ sec=0, msec=816
pthread_create successful for sequencer service 0
Sequencer thread @ sec=0, msec=817

TEST COMPLETE
```

Figure 4: seqgen.c execution

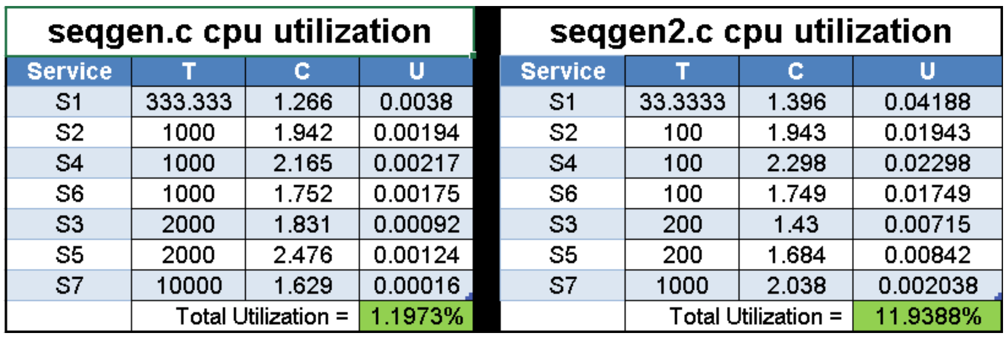


Service	Frequency	Time Period T in ms	Deadline D, D=T	Priority	WCET	WCET in m	Execution Time C
S1	30	33.33333333	33.33333333	1	1395938	1.395938	1.395938
S2	10	100	100	2	1943073	1.943073	1.943073
S4	10	100	100	2	2298021	2.298021	2.298021
S6	10	100	100	2	1749011	1.749011	1.749011
S3	5	200	200	3	1430416	1.430416	1.430416
S5	5	200	200	3	1684479	1.684479	1.684479
S7	1	1000	1000	4	2037500	2.0375	2.0375

Figure 5: seqgen.c values

1.2.3 Cheddar Schedule and CPU Utilization

Cheddar does not allow for the use of decimal values in its parameters. To compensate for this, the values used for execution time C (WCET) were rounded up to the next integer. With a higher execution time, the lower value of the true execution time is still guaranteed to succeed. The total CPU Utilization for each program can be seen in the figure below, followed by the Cheddar simulation task parameters and results.

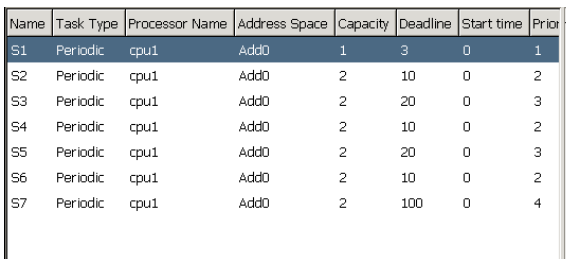


seqgen.c cpu utilization			
Service	T	C	U
S1	333.333	1.266	0.0038
S2	1000	1.942	0.00194
S4	1000	2.165	0.00217
S6	1000	1.752	0.00175
S3	2000	1.831	0.00092
S5	2000	2.476	0.00124
S7	10000	1.629	0.00016
		Total Utilization =	1.1973%

seqgen2.c cpu utilization			
Service	T	C	U
S1	33.3333	1.396	0.04188
S2	100	1.943	0.01943
S4	100	2.298	0.02298
S6	100	1.749	0.01749
S3	200	1.43	0.00715
S5	200	1.684	0.00842
S7	1000	2.038	0.002038
		Total Utilization =	11.9388%

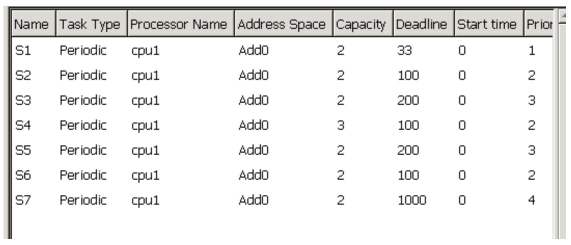
Figure 6: Total CPU Utilization

The Cheddar simulation task parameters and results can be seen here.



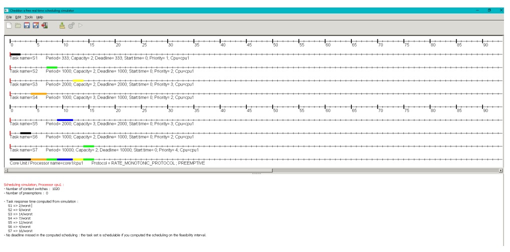
Name	Task Type	Processor Name	Address Space	Capacity	Deadline	Start time	Prior
S1	Periodic	cpu1	Add0	1	3	0	1
S2	Periodic	cpu1	Add0	2	10	0	2
S3	Periodic	cpu1	Add0	2	20	0	3
S4	Periodic	cpu1	Add0	2	10	0	2
S5	Periodic	cpu1	Add0	2	20	0	3
S6	Periodic	cpu1	Add0	2	10	0	2
S7	Periodic	cpu1	Add0	2	100	0	4

Figure 7: Tasks for seqgen.c



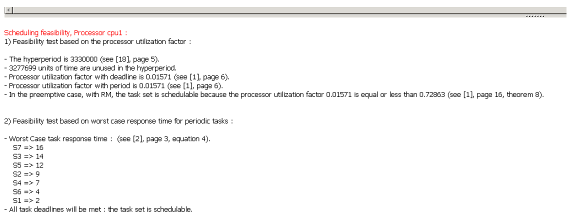
Name	Task Type	Processor Name	Address Space	Capacity	Deadline	Start time	Prior
S1	Periodic	cpu1	Add0	2	33	0	1
S2	Periodic	cpu1	Add0	2	100	0	2
S3	Periodic	cpu1	Add0	2	200	0	3
S4	Periodic	cpu1	Add0	3	100	0	2
S5	Periodic	cpu1	Add0	2	200	0	3
S6	Periodic	cpu1	Add0	2	100	0	2
S7	Periodic	cpu1	Add0	2	1000	0	4

Figure 8: Tasks for seqgen2.c



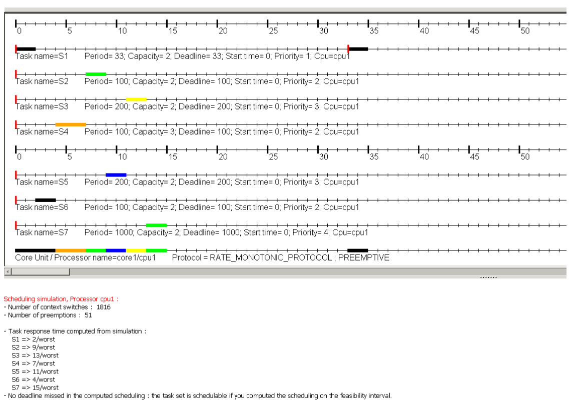
```
Task name=S1    Period= 333; Capacity= 2; Deadline= 333; Start time= 0; Priority= 1; Cpu=cpu1
Task name=S2    Period= 1000; Capacity= 2; Deadline= 1000; Start time= 0; Priority= 2; Cpu=cpu1
Task name=S3    Period= 2000; Capacity= 2; Deadline= 2000; Start time= 0; Priority= 3; Cpu=cpu1
Task name=S4    Period= 1000; Capacity= 3; Deadline= 1000; Start time= 0; Priority= 2; Cpu=cpu1
Task name=S5    Period= 2000; Capacity= 3; Deadline= 2000; Start time= 0; Priority= 3; Cpu=cpu1
Task name=S6    Period= 1000; Capacity= 2; Deadline= 1000; Start time= 0; Priority= 2; Cpu=cpu1
Task name=S7    Period= 10000; Capacity= 2; Deadline= 10000; Start time= 0; Priority= 4; Cpu=cpu1
Core Unit / Processor name=core1/cpu1    Protocol = RATE_MONOTONIC_PROTOCOL ; PREEMPTIVE

Scheduling simulation, Processor cpu1 :
- Number of context switches :  1020
- Number of preemptions :  0

- Task response time computed from simulation :
   S1 => 2/worst
   S2 => 9/worst
   S3 => 14/worst
   S4 => 7/worst
   S5 => 12/worst
   S6 => 4/worst
   S7 => 16/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

Figure 9: RM Execution for seqgen.c



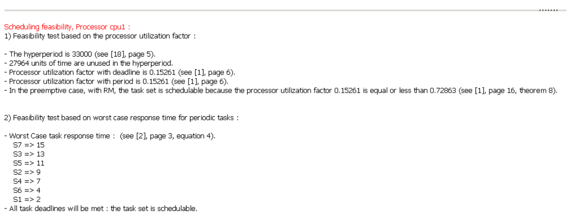
```
Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 3330000 (see [18], page 5).
- 3277699 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.01571 (see [1], page 6).
- Processor utilization factor with period is 0.01571 (see [1], page 6).
- In the preemptive case, with RM, the task set is schedulable because the processor utilization factor 0.01571 is equal or less than 0.72863 (see [1], page 16, theorem 8).


2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time :  (see [2], page 3, equation 4).
   S7 => 16
   S3 => 14
   S5 => 12
   S2 => 9
   S4 => 7
   S6 => 4
   S1 => 2
- All task deadlines will be met : the task set is schedulable.
```

Figure 10: Feasibility test for seqgen.cs



```
Task name=S1    Period= 33; Capacity= 2; Deadline= 33; Start time= 0; Priority= 1; Cpu=cpu1
Task name=S2    Period= 100; Capacity= 2; Deadline= 100; Start time= 0; Priority= 2; Cpu=cpu1
Task name=S3    Period= 200; Capacity= 2; Deadline= 200; Start time= 0; Priority= 3; Cpu=cpu1
Task name=S4    Period= 100; Capacity= 3; Deadline= 100; Start time= 0; Priority= 2; Cpu=cpu1
Task name=S5    Period= 200; Capacity= 2; Deadline= 200; Start time= 0; Priority= 3; Cpu=cpu1
Task name=S6    Period= 100; Capacity= 2; Deadline= 100; Start time= 0; Priority= 2; Cpu=cpu1
Task name=S7    Period= 1000; Capacity= 2; Deadline= 1000; Start time= 0; Priority= 4; Cpu=cpu1
Core Unit / Processor name=core1/cpu1    Protocol = RATE_MONOTONIC_PROTOCOL ; PREEMPTIVE

Scheduling simulation, Processor cpu1 :
- Number of context switches :  1816
- Number of preemptions :  51

- Task response time computed from simulation :
   S1 => 2/worst
   S2 => 9/worst
   S3 => 13/worst
   S4 => 7/worst
   S5 => 11/worst
   S6 => 4/worst
   S7 => 15/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

Figure 11: RM Execution for seqgen2.c



```
Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 33000 (see [18], page 5).
- 27964 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.15261 (see [1], page 6).
- Processor utilization factor with period is 0.15261 (see [1], page 6).
- In the preemptive case, with RM, the task set is schedulable because the processor utilization factor 0.15261 is equal or less than 0.72863 (see [1], page 16, theorem 8).


2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time :  (see [2], page 3, equation 4).
   S7 => 15
   S3 => 13
   S5 => 11
   S2 => 9
   S4 => 7
   S6 => 4
   S1 => 2
- All task deadlines will be met : the task set is schedulable.
```

Figure 12: Feasibility test for seqgen2.c

2 Problem 2

2.1 Problem 2

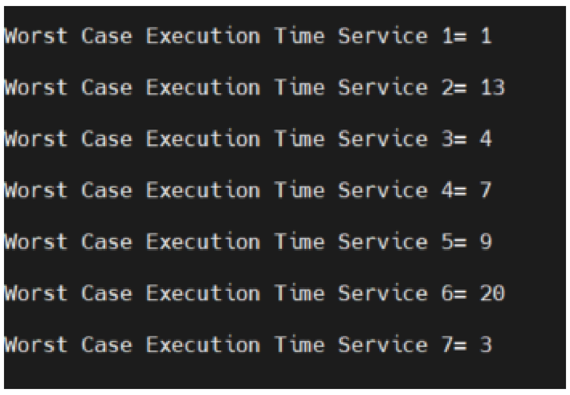
Revise seqgen.c and seqgen2x.c to run under FreeRTOS on the DE1-SoC or TIVA board, by making each service a FreeRTOS task. Use the associated startup file in place of the existing startup file in FreeRTOS. Use an ISR driven by the PIT hardware timer to release each task at the given rate (you could even put the sequencer in the ISR). Build and execute the new code. Determine the worst case execution time (WCET) for each service by printing or logging timestamps between two points in your code or by use of a profiling tool. Determine D, T, and C for each service and create an RM schedule in Cheddar using your WCET estimates. Calculate the Compare this with the results you achieved under Linux in (1).

2.2 Explanation and execution of code on target

The seqgen and the seqgen2x code was ported on the platform TIVA C Servies Launchpad with FreeRTOS as the operating system.

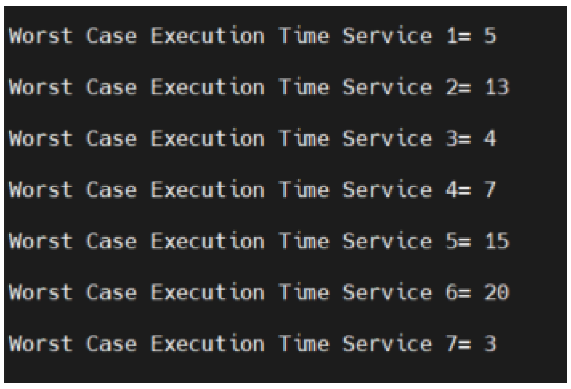
1. The 7 Services were created using the the function xTaskCreate
2. The sequencer was invoked in the ISR using the Timer 0 Timer A
3. UART was used to output on the serial terminal
4. The Tasks were assigned priorities using the RM policy in which the service with the highest frequency was assigned the highest and subsequent priorities in decreasing order of the frequency
5. The Sequencer and the services shared a semaphore among them for signalling
6. The timestamps was recorded using the API xTaskGetTickCount of the FreeRTOS kernel which updates its tick at the frequency configured in FreeRTOSConfig.h file
7. The printing on the serial terminal was protected by a common semaphore shared throughout the system
8. To handling the signal from the sequencer to services, the APIs xSemaphoreGive-FromISR and xSemaphoreTake was used
9. The Timer 0 was configured to occur periodically at the rate of the SystemClock divided the the Sequencer frequency which in this case was 30, 60 and 3000Hz.

2.3 Timestamps and WCET estimates



```
Worst Case Execution Time Service 1= 1
Worst Case Execution Time Service 2= 13
Worst Case Execution Time Service 3= 4
Worst Case Execution Time Service 4= 7
Worst Case Execution Time Service 5= 9
Worst Case Execution Time Service 6= 20
Worst Case Execution Time Service 7= 3
```

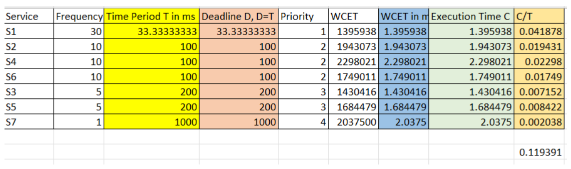
Figure 13: Output on serial terminal for seqgen



```
Worst Case Execution Time Service 1= 5
Worst Case Execution Time Service 2= 13
Worst Case Execution Time Service 3= 4
Worst Case Execution Time Service 4= 7
Worst Case Execution Time Service 5= 15
Worst Case Execution Time Service 6= 20
Worst Case Execution Time Service 7= 3
```

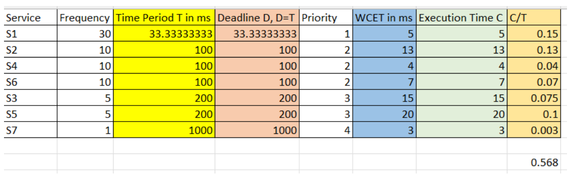
Figure 14: Output on serial terminal for seqgen

2.4 D,C and T Table for all services



Service	Frequency	Time Period T in ms	Deadline D, D=T	Priority	WCET	WCET in m	Execution Time C	C/T
S1	30	33.33333333	33.33333333	1	1395938	1.395938	1.395938	0.041878
S2	10	100	100	2	1943073	1.943073	1.943073	0.019431
S4	10	100	100	2	2298021	2.298021	2.298021	0.02298
S6	10	100	100	2	1749011	1.749011	1.749011	0.01749
S3	5	200	200	3	1430416	1.430416	1.430416	0.007152
S5	5	200	200	3	1684479	1.684479	1.684479	0.008422
S7	1	1000	1000	4	2037500	2.0375	2.0375	0.002038
								0.119391

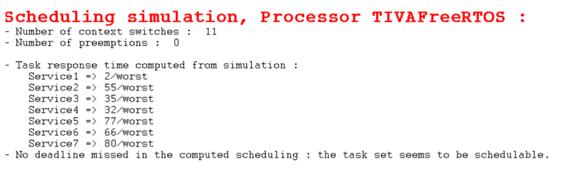
Figure 15: D,C and T Table for all services for Seqgen



Service	Frequency	Time Period T in ms	Deadline D, D=T	Priority	WCET in ms	Execution Time C	C/T
S1	30	33.33333333	33.33333333	1	5	5	0.15
S2	10	100	100	2	13	13	0.13
S4	10	100	100	2	4	4	0.04
S6	10	100	100	2	7	7	0.07
S3	5	200	200	3	15	15	0.075
S5	5	200	200	3	20	20	0.1
S7	1	1000	1000	4	3	3	0.003
							0.568

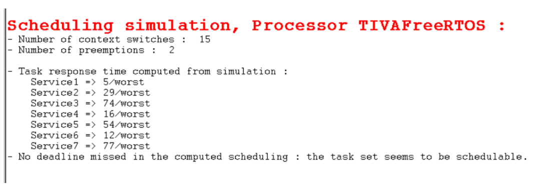
Figure 16: D,C and T Table for all services for Seqgen2X

2.5 Cheddar Schedule



```
Scheduling simulation, Processor TIVAFreeRTOS :
- Number of context switches :  11
- Number of preemptions :  0

- Task response time computed from simulation :
    Service1 => 2/worst
    Service2 => 55/worst
    Service3 => 35/worst
    Service4 => 32/worst
    Service5 => 77/worst
    Service6 => 66/worst
    Service7 => 80/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.
```

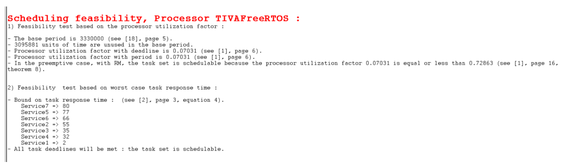
Figure 17: RM Execution for seqgen.c



```
Scheduling simulation, Processor TIVAFreeRTOS :
- Number of context switches :  15
- Number of preemptions :  2

- Task response time computed from simulation :
    Service1 => 5/worst
    Service2 => 29/worst
    Service3 => 74/worst
    Service4 => 16/worst
    Service5 => 54/worst
    Service6 => 12/worst
    Service7 => 77/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.
```

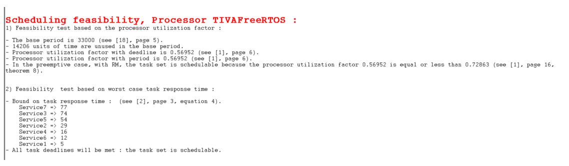
Figure 18: RM Execution for seqgen.c

2.6 % CPU Utilization



```
Scheduling feasibility, Processor TIVAFreeRTOS :
1) Feasibility test based on the processor utilization factor :

- The base period is 3330000 (see [18], page 5).
- 3095881 units of time are unused in the base period.
- Processor utilization factor with deadline is 0.07031 (see [1], page 6).
- Processor utilization factor with period is 0.07031 (see [1], page 6).
- In the preemptive case, with RM, the task set is schedulable because the processor utilization factor 0.07031 is equal or less than 0.72863 (see [1], page 16,
theorem 8).


2) Feasibility  test based on worst case task response time :

- Bound on task response time :  (see [2], page 3, equation 4).
    Service7 => 80
    Service5 => 77
    Service6 => 66
    Service2 => 55
    Service3 => 35
    Service4 => 32
    Service1 => 2
- All task deadlines will be met : the task set is schedulable.
```

Figure 19: Utilization of CPU for seqgen.c



```
Scheduling feasibility, Processor TIVAFreeRTOS :
1) Feasibility test based on the processor utilization factor :

- The base period is 33000 (see [18], page 5).
- 14206 units of time are unused in the base period.
- Processor utilization factor with deadline is 0.56952 (see [1], page 6).
- Processor utilization factor with period is 0.56952 (see [1], page 6).
- In the preemptive case, with RM, the task set is schedulable because the processor utilization factor 0.56952 is equal or less than 0.72863 (see [1], page 16,
theorem 8).


2) Feasibility  test based on worst case task response time :

- Bound on task response time :  (see [2], page 3, equation 4).
    Service7 => 77
    Service3 => 74
    Service5 => 54
    Service2 => 29
    Service4 => 16
    Service6 => 12
    Service1 => 5
- All task deadlines will be met : the task set is schedulable.
```

Figure 20: Utilization of CPU for seqgen2X.c

2.7 Comparision between Linux and FreeRTOS

The Utilization of CPU was found to be different on the Linux and FreeRTOS system. This might be due to the face that a synthetic load was introduced in FreeRTOS due to the resolution of the clocks. The number of context switches might also add to more utilization of CPU in FreeRTOS as the sequencer was handled in the interrupt

3 Problem 3

3.1 Problem

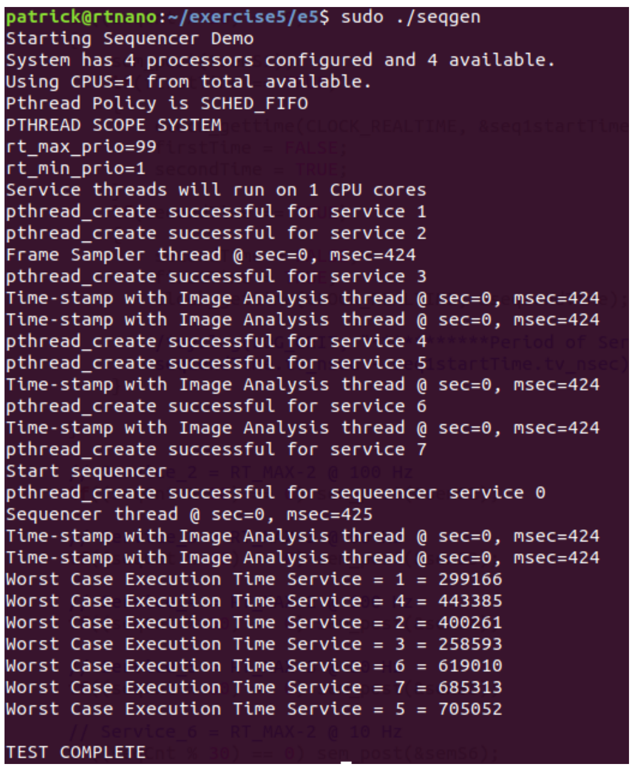
Revise seqgen.c from both previous systems to increase the sequencer frequency and all service frequencies by a factor of 100 (3000 Hz). Build and execute the code under Linux and FreeRTOS on your target boards as before. For both operating systems determine the worst case execution time (WCET) for each service by printing or logging timestamps between two points in your code or by use of a profiling tool. Determine D, T, and C for each service and create an RM schedule in Cheddar using your WCET estimates. Calculate the percentage

CPU utilization for these systems. Compare results between Linux and FreeRTOS in this higher-speed case.

3.2 Answer

3.2.1 Explanation and Execution of code

Linux seqgen.c at 3000Hz As described in section 1, the seqgen.c code runs the sequencer as a primary thread with all services running as parallel threads from it. The difference now is that the sequencer is being run at 3000Hz. This changes the services frequencies to the following: s1 = 300Hz, s2 = 100Hz, s3 = 50Hz, s4 = 100Hz, s5 = 50Hz, s6 = 100Hz and s7 = 10Hz. The figure below shows the execution of seqgen.c at 3000Hz.

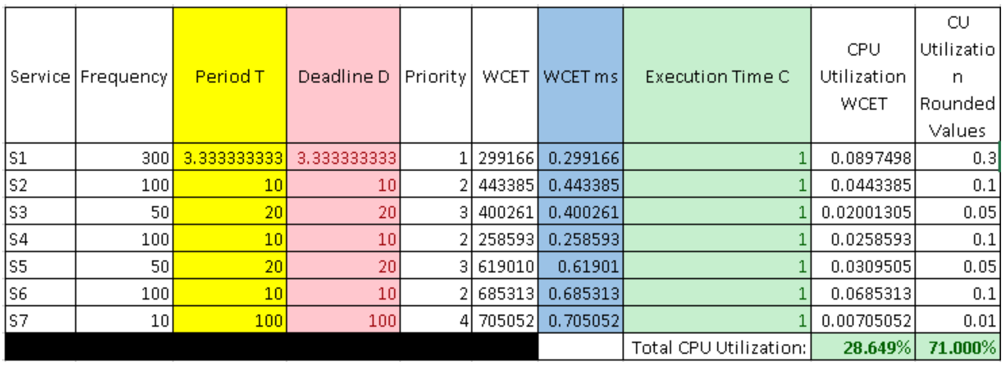


```
patrick@rtnano:~/exercise5/e5$ sudo ./seqgen
Starting Sequencer Demo
System has 4 processors configured and 4 available.
Using CPUS=1 from total available.
Pthread Policy is SCHED_FIFO
PTHREAD SCOPE SYSTEM
rt_max_prio=99
rt_min_prio=1
Service threads will run on 1 CPU cores
pthread_create successful for service 1
pthread_create successful for service 2
Frame Sampler thread @ sec=0, msec=424
pthread_create successful for service 3
Time-stamp with Image Analysis thread @ sec=0, msec=424
Time-stamp with Image Analysis thread @ sec=0, msec=424
pthread_create successful for service 4
pthread_create successful for service 5
Time-stamp with Image Analysis thread @ sec=0, msec=424
pthread_create successful for service 6
Time-stamp with Image Analysis thread @ sec=0, msec=424
pthread_create successful for service 7
Start sequencer
pthread_create successful for sequeencer service 0
Sequencer thread @ sec=0, msec=425
Time-stamp with Image Analysis thread @ sec=0, msec=424
Time-stamp with Image Analysis thread @ sec=0, msec=424
Worst Case Execution Time Service = 1 = 299166
Worst Case Execution Time Service = 4 = 443385
Worst Case Execution Time Service = 2 = 400261
Worst Case Execution Time Service = 3 = 258593
Worst Case Execution Time Service = 6 = 619010
Worst Case Execution Time Service = 7 = 685313
Worst Case Execution Time Service = 5 = 705052

TEST COMPLETE
```

Figure 21: seqgen.c at 3kHz

Cheddar and CPU Utilization As in section 1, the values of C for cheddar simulations were rounded up. This change allows the use of integers for Cheddar simulation and guarantees that any c value lower than those used will be a viable solution. The CPU utilization for the actual WCET values was 28.649% and the Utilization with the rounded values was 71%. These calculations and the Cheddar simulation results are pictured below.

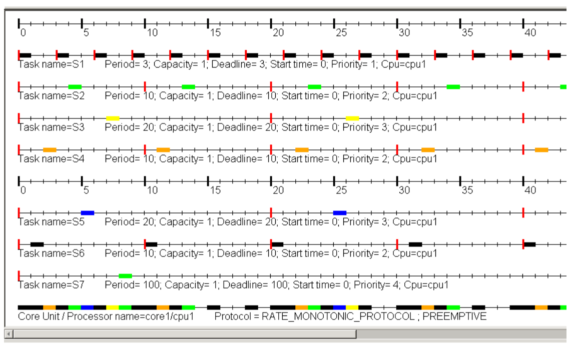
3.3 D,C and T Table for all services for Free RTOS



Service	Frequency	Period T	Deadline D	Priority	WCET	WCET ms	Execution Time C	CPU Utilization WCET	CU Utilization Rounded Values
S1	300	3.333333333	3.333333333	1	299166	0.299166	1	0.0897498	0.3
S2	100	10	10	2	443385	0.443385	1	0.0443385	0.1
S3	50	20	20	3	400261	0.400261	1	0.02001305	0.05
S4	100	10	10	2	258593	0.258593	1	0.0258593	0.1
S5	50	20	20	3	619010	0.61901	1	0.0309505	0.05
S6	100	10	10	2	685313	0.685313	1	0.0685313	0.1
S7	10	100	100	4	705052	0.705052	1	0.00705052	0.01
							Total CPU Utilization:	28.649%	71.000%

Figure 22: Utilization Calculations at 3kHz

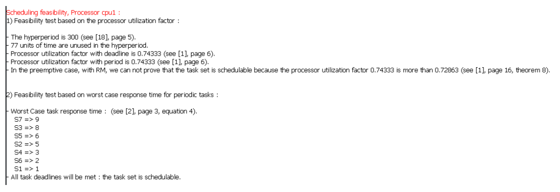
3.4 Cheddar schedule and % CPU Utilization Linux



```
Task name=S1    Period= 3; Capacity= 1; Deadline= 3; Start time= 0; Priority= 1; Cpu=cpu1
Task name=S2    Period= 10; Capacity= 1; Deadline= 10; Start time= 0; Priority= 2; Cpu=cpu1
Task name=S3    Period= 20; Capacity= 1; Deadline= 20; Start time= 0; Priority= 3; Cpu=cpu1
Task name=S4    Period= 10; Capacity= 1; Deadline= 10; Start time= 0; Priority= 2; Cpu=cpu1
Task name=S5    Period= 20; Capacity= 1; Deadline= 20; Start time= 0; Priority= 3; Cpu=cpu1
Task name=S6    Period= 10; Capacity= 1; Deadline= 10; Start time= 0; Priority= 2; Cpu=cpu1
Task name=S7    Period= 100; Capacity= 1; Deadline= 100; Start time= 0; Priority= 4; Cpu=cpu1
Core Unit / Processor name=core1/cpu1    Protocol = RATE_MONOTONIC_PROTOCOL ; PREEMPTIVE
```

```
Scheduling simulation, Processor cpu1 :
- Number of context switches :  199
- Number of preemptions :  0

- Task response time computed from simulation :
   S1 => 1/worst
   S2 => 5/worst
   S3 => 8/worst
   S4 => 3/worst
   S5 => 6/worst
   S6 => 2/worst
   S7 => 9/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

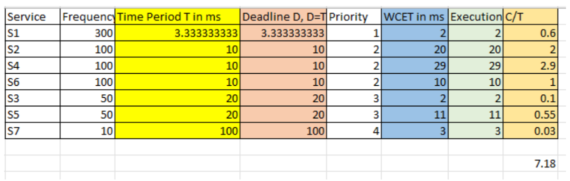
Figure 23: Cheddar Simulation 3kHz



```
Scheduling feasibility, Processor cpu1 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 300 (see [18], page 5).
- 77 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.74333 (see [1], page 6).
- Processor utilization factor with period is 0.74333 (see [1], page 6).
- In the preemptive case, with RM, we can not prove that the task set is schedulable because the processor utilization factor 0.74333 is more than 0.72863 (see [1], page 16, theorem 8).


2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time :  (see [2], page 3, equation 4).
   S7 => 9
   S3 => 8
   S5 => 6
   S2 => 5
   S4 => 3
   S6 => 2
   S1 => 1
- All task deadlines will be met : the task set is schedulable.
```

Figure 24: Cheddar Feasibility 3kHz

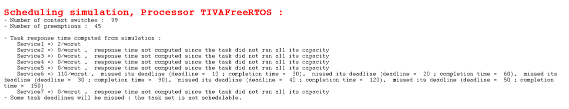
3.5 D,C and T Table for all services for Free RTOS



Service	Frequency	Time Period T in ms	Deadline D, D=T	Priority	WCET in ms	Execution	C/T
S1	300	3.333333333	3.333333333	1	2	2	0.6
S2	100	10	10	2	20	20	2
S4	100	10	10	2	29	29	2.9
S6	100	10	10	2	10	10	1
S3	50	20	20	3	2	2	0.1
S5	50	20	20	3	11	11	0.55
S7	10	100	100	4	3	3	0.03
							7.18

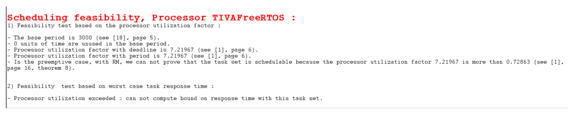
Figure 25: D,T and C Values

3.6 Cheddar schedule and % CPU Utilization



```
Scheduling simulation, Processor TIVAFreeRTOS :
- Number of context switches :  99
- Number of preemptions :  45

- Task response time computed from simulation :
    Service1 => 2/worst
    Service2 => 0/worst ,  response time not computed since the task did not run all its capacity
    Service3 => 0/worst ,  response time not computed since the task did not run all its capacity
    Service4 => 0/worst ,  response time not computed since the task did not run all its capacity
    Service5 => 0/worst ,  response time not computed since the task did not run all its capacity
    Service6 => 110/worst ,  missed its deadline (deadline =  10 ; completion time =  30),  missed its deadline (deadline =  20 ; completion time =  60),  missed its
deadline (deadline =  30 ; completion time =  90),  missed its deadline (deadline =  40 ; completion time =  120),  missed its deadline (deadline =  50 ; completion
time =  150)
    Service7 => 0/worst ,  response time not computed since the task did not run all its capacity
- Some task deadlines will be missed : the task set is not schedulable.
```

Figure 26: Free RTOS Cheddar Simulation 3kHz



```
Scheduling feasibility, Processor TIVAFreeRTOS :
1) Feasibility test based on the processor utilization factor :

- The base period is 3000 (see [18], page 5).
- 0 units of time are unused in the base period.
- Processor utilization factor with deadline is 7.21967 (see [1], page 6).
- Processor utilization factor with period is 7.21967 (see [1], page 6).
- In the preemptive case, with RM, we can not prove that the task set is schedulable because the processor utilization factor 7.21967 is more than 0.72863 (see [1],
page 16, theorem 8).


2) Feasibility  test based on worst case task response time :

- Processor utilization exceeded : can not compute bound on response time with this task set.
```

Figure 27: Free RTOS Cheddar Feasibility 3kHz

3.7 Comparision between Linux and Free RTOS

The services are not schedule in FreeRTOS due to the processing speed. This might also be due to the number of context switches

4 Appendix- Code

Code Repository

Github Link