SPRINTS

Healthcare System

System design case study using RTOS

Submitted by:

Mahmoud Hamdy

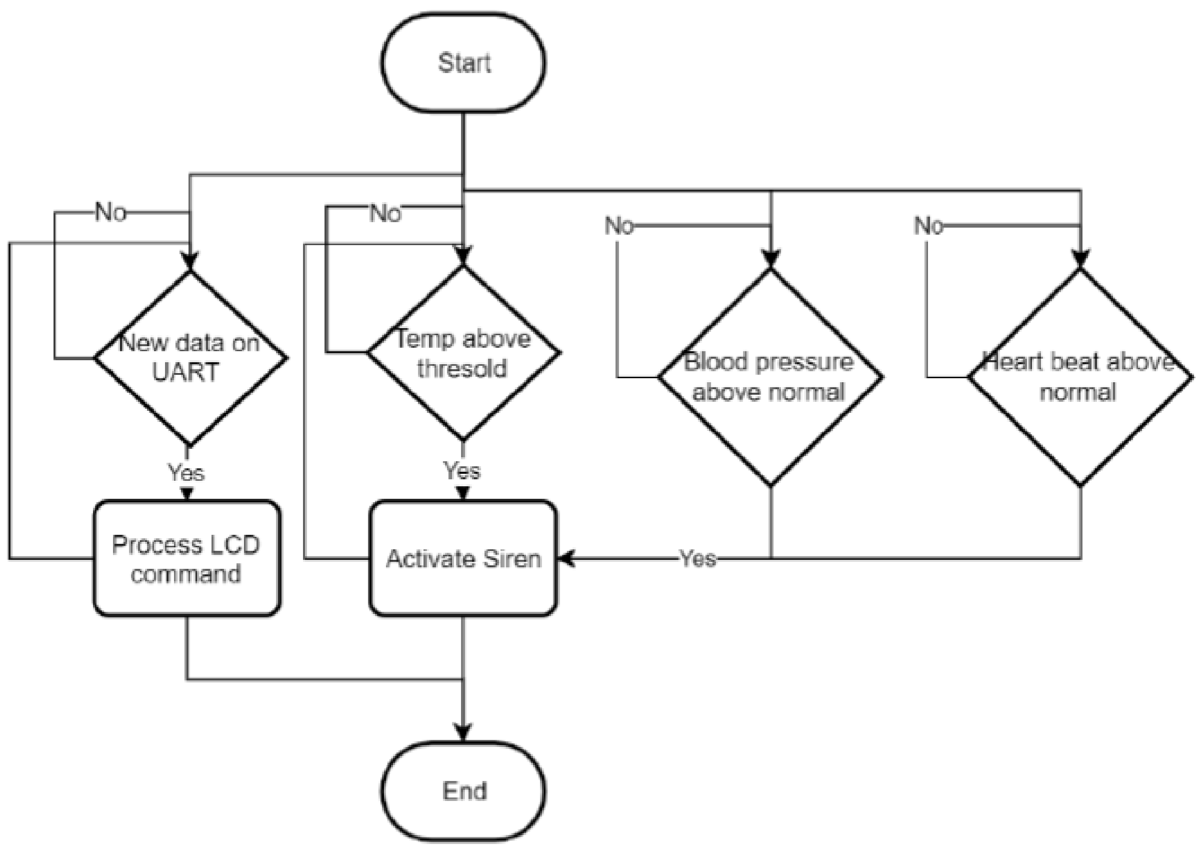
Cairo, 2021

- Contents:

- Problem Statement and system diagram:

Task: Design a healthcare system using RTOS with the following requirements:

- A touch LCD as input that can control the system and give commands. Every LCD command is represented in 4 bytes. LCD is connected to the micro-controller through UART with speed 9600 bps [Bit per second]. (Reading 4 bytes and processing the command takes 2 ms)
- Blood pressure sensor with new data every 25ms. (Reading the sensor and processing its data takes 3 ms)
- Heart beat detector with new data every 100ms. (Reading the sensor and processing its data takes 1.5 ms)
- Temperature sensor with new data every 10ms. (Reading the sensor and processing its data takes 2.5 ms)
- Alert siren. (Activate or Deactivate the siren takes 1 ms)



- Breaking down tasks:

Five tasks, they are:

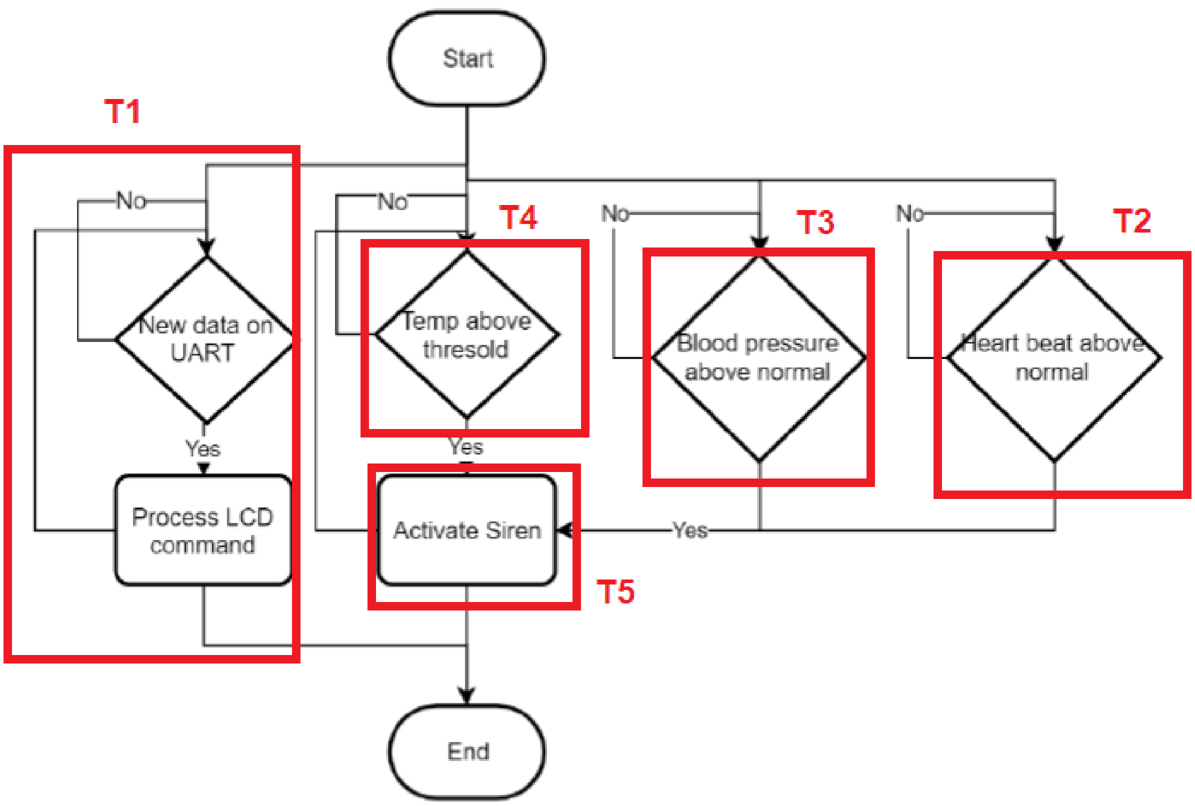
T1: UART/LCD

T2: Heart Beat Sensor

T3: Blood Pressure Detector

T4: Temperature Sensor

T5: Alert Siren



- Finding task parameters (Priority - Periodicity - Deadline):

T1: UART/LCD (P = D = 20 ms, E = 2 ms)

T2: Heart Beat Sensor (P = D = 100 ms, E = 1.5 ms)

T3: Blood Pressure Detector (P = D = 25 ms, E = 3 ms)

T4: Temperature Sensor (P = D = 10 ms, E = 2.5 ms)

T5: Alert Siren (P = D = 10 ms, E = 1 ms)

Assumptions:

- UART/LCD task runs every 20 ms to ensure debouncing and high-precision input
- Alert Siren task runs every 10 ms as it has to immediately respond to any enabling or disabling as fast as the temperature sensing.
- Deadline is the same as periodicity.

- Calculating system tick rate:

Tick rate = Highest common factor of all periodicities in the system

H.C.F. (20, 100, 25, 10, 10) = 5 ms

20 = 5 x 2 x 2
100 = 5 x 5 x 2 x 2 (Using prime factorization)
25 = 5 x 5
10 = 5 x 2

- Calculating hyper period:

Hyperperiod = Lowest common multiple of all periodicities in the system

L.C.M. (20, 100, 25, 10, 10) = 100 ms

20 = 5 x 2 x 2
100 = 5 x 5 x 2 x 2
25 = 5 x 5
10 = 5 x 2

LCM = 5 x 5 x 2 x 2 = 100
(Multiply max occurrence of prime factors)

- Calculating CPU load:

$$\text{CPU load} = \frac{\sum E_i \times \frac{H}{P_i}}{H} = \sum_{i=1}^{5} \frac{E_i}{P_i}$$

$$\therefore \text{CPU load} = \frac{2}{20} + \frac{1.5}{100} + \frac{3}{25} + \frac{2.5}{10} + \frac{1}{10}$$

$$= 0.1 + 0.015 + 0.12 + 0.25 + 0.1$$

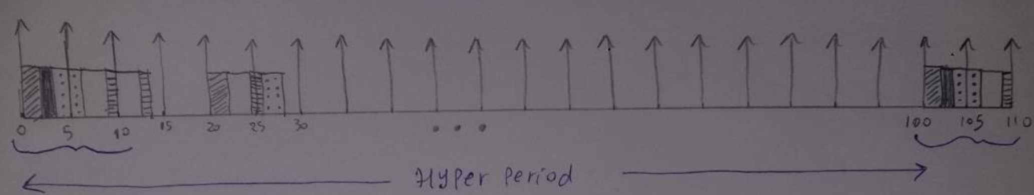
$$= 0.585 \times 100\,\% = \boxed{58.5\ \%}$$

(Analytically system is healthy)

- Drawing system manually and analyzing schedulability:

Finding schedulability

Case 1: No priority & No preemption (Tasks are running in order $T_1 \rightarrow T_5$)



$\because \sum E_i = 10$ and system is non-preemptive
↳ which is the least periodicity
$\because$ HyperPeriod on graph matches the analytical value
→ Tasks will not miss their deadlines & system is schedulable

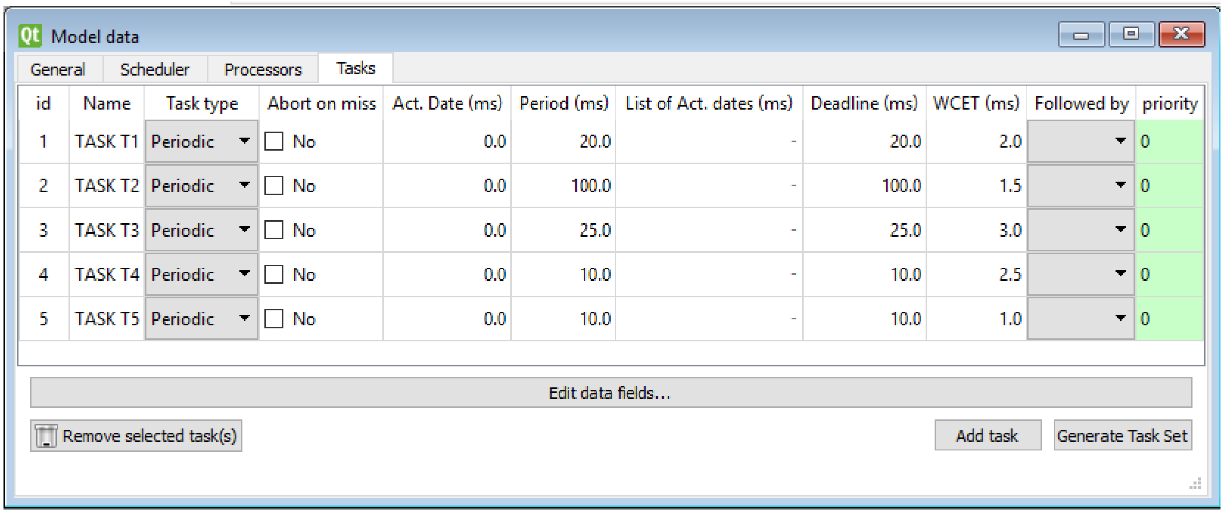
Case 2: Priority & Preemption (Drawn on simso since hand drawing is more complex)

Same results → system is healthy
→ Tasks will not miss their deadlines
→ system is schedulable

Are they really the same? Let's check more details

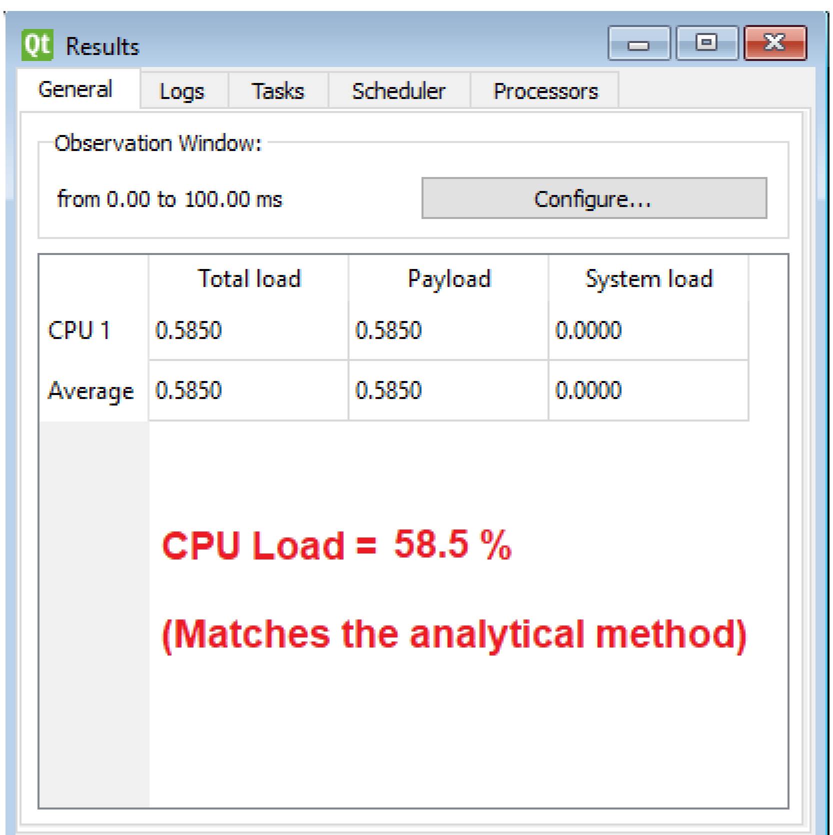
- Modeling the system on simso and verifying schedulability:

Case 1: No priority



id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)	Followed by	priority
1	TASK T1	Periodic	No	0.0	20.0	-	20.0	2.0		0
2	TASK T2	Periodic	No	0.0	100.0	-	100.0	1.5		0
3	TASK T3	Periodic	No	0.0	25.0	-	25.0	3.0		0
4	TASK T4	Periodic	No	0.0	10.0	-	10.0	2.5		0
5	TASK T5	Periodic	No	0.0	10.0	-	10.0	1.0		0

Result:

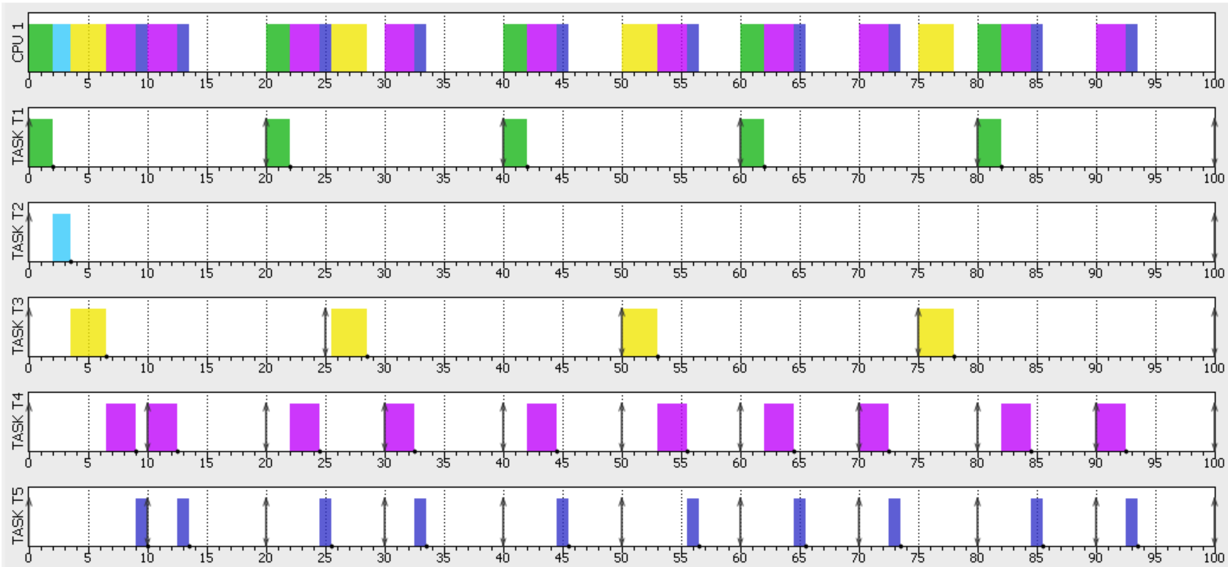


	Total load	Payload	System load
CPU 1	0.5850	0.5850	0.0000
Average	0.5850	0.5850	0.0000

CPU Load = 58.5 %

(Matches the analytical method)

Gantt chart:



Case 2: Multiple priority levels based on periodicity

Model data

General | Scheduler | Processors | Tasks

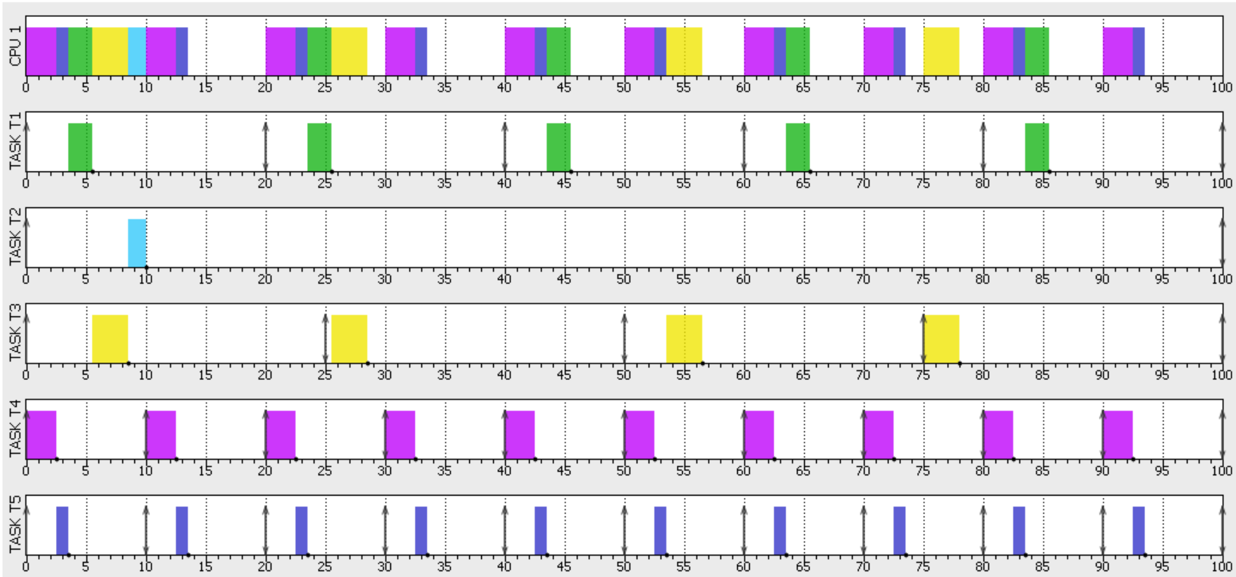
id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)	Followed by	priority
1	TASK T1	Periodic	No	0.0	20.0	-	20.0	2.0		2
2	TASK T2	Periodic	No	0.0	100.0	-	100.0	1.5		0
3	TASK T3	Periodic	No	0.0	25.0	-	25.0	3.0		1
4	TASK T4	Periodic	No	0.0	10.0	-	10.0	2.5		3
5	TASK T5	Periodic	No	0.0	10.0	-	10.0	1.0		3

Edit data fields...

Remove selected task(s) | Add task | Generate Task Set

Result: the same regardless the case

Gantt chart:



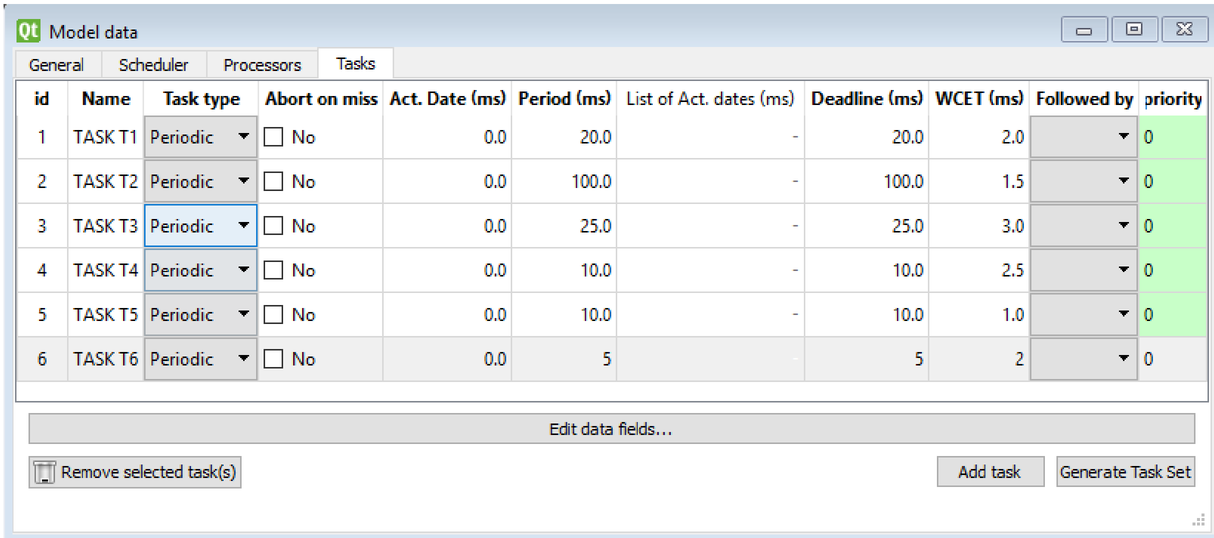
Analytical results are verified!

- Introducing new tasks:

Introducing a new task to the system to load the system:

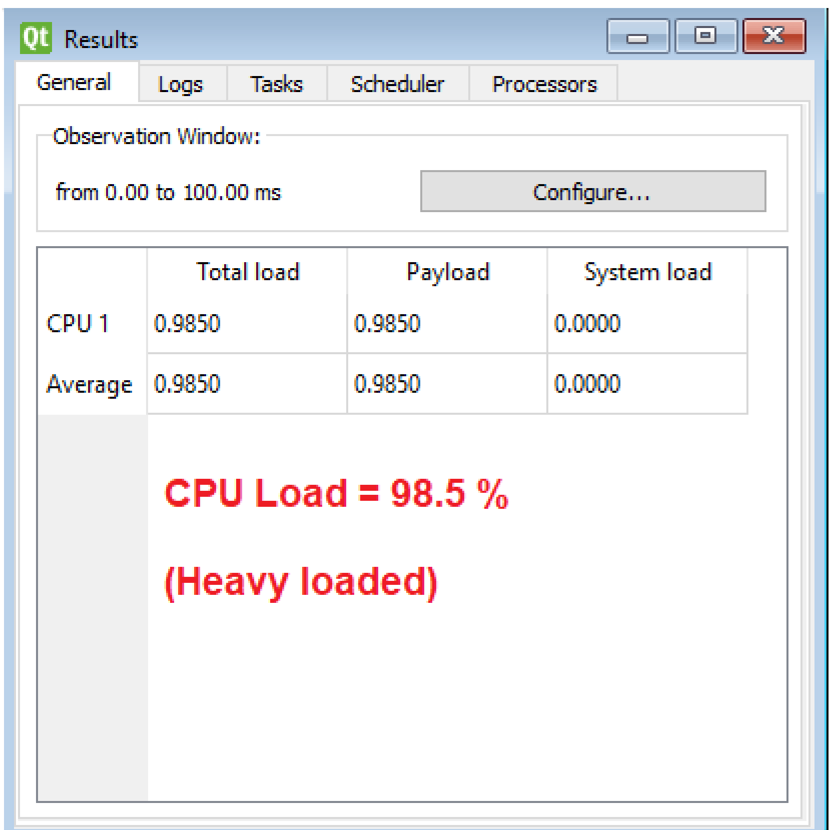
T6 (P = D = 5 ms, E = 2 ms)

Case 1: No priority



id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)	Followed by	priority
1	TASK T1	Periodic	No	0.0	20.0	-	20.0	2.0		0
2	TASK T2	Periodic	No	0.0	100.0	-	100.0	1.5		0
3	TASK T3	Periodic	No	0.0	25.0	-	25.0	3.0		0
4	TASK T4	Periodic	No	0.0	10.0	-	10.0	2.5		0
5	TASK T5	Periodic	No	0.0	10.0	-	10.0	1.0		0
6	TASK T6	Periodic	No	0.0	5		5	2		0

Result:

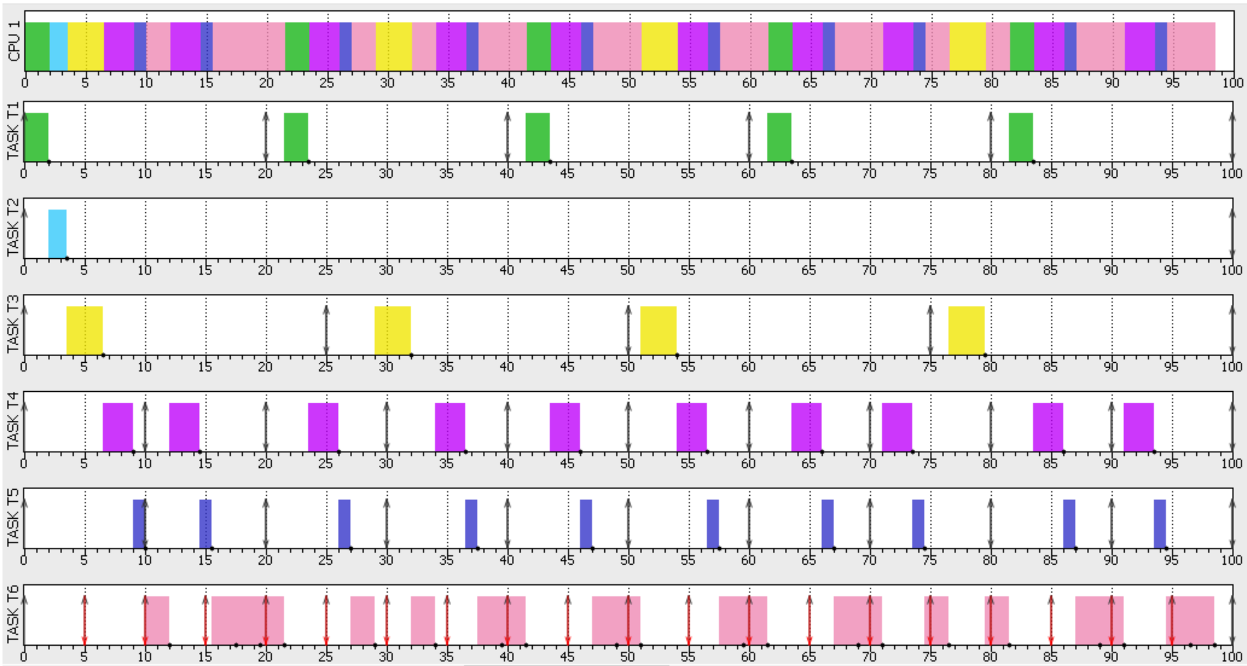


	Total load	Payload	System load
CPU 1	0.9850	0.9850	0.0000
Average	0.9850	0.9850	0.0000

CPU Load = 98.5 %

(Heavy loaded)

Gantt chart:



Case 2: Multiple priority levels based on periodicity

Model data

General | Scheduler | Processors | Tasks

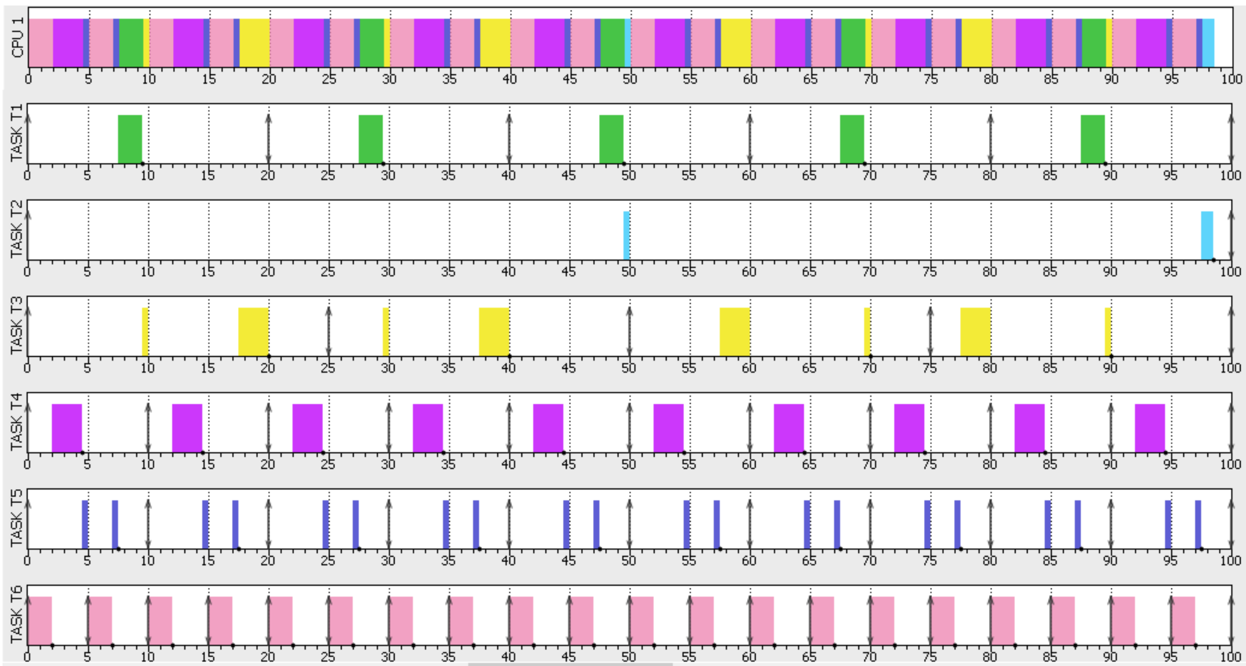
id	Name	Task type	Abort on miss	Act. Date (ms)	Period (ms)	List of Act. dates (ms)	Deadline (ms)	WCET (ms)	Followed by	priority
1	TASK T1	Periodic	No	0.0	20.0	-	20.0	2.0		2
2	TASK T2	Periodic	No	0.0	100.0	-	100.0	1.5		0
3	TASK T3	Periodic	No	0.0	25.0	-	25.0	3.0		1
4	TASK T4	Periodic	No	0.0	10.0	-	10.0	2.5		3
5	TASK T5	Periodic	No	0.0	10.0	-	10.0	1.0		3
6	TASK T6	Periodic	No	0.0	5	-	5	2		4

Edit data fields...

Remove selected task(s) | Add task | Generate Task Set

Result: The same regardless the case

Gantt chart:



- Final comments:

To conclude these experiments:

1) Our system is healthy and schedulable regardless the usage of priority.

2) When further loading the system, it's wise to start adding priority to the system so the more frequent tasks don't miss their deadline.

3) Overall, preemptive system is more preferable if the system is eligible for upgrading (by adding more functionalities to the system).