Real-Time Systems

Exam Preparation

Abstract

The following exercises are a compilation from the exams Real-Time Systems summer semester 2017 and 2020. In addition, they contain exercises that were given by the lecturer as hints for possible examination tasks, but were not part of the exams in this form. Exercises from the 2017 and 2020 exams can be identified by the points to be achieved for them. The correctness of all exercises is not guaranteed and it is recommended to check everything again by yourself.

Exercise 1: (11 Points)

a) Name the two most important requirements for real-time systems. (2P)

1. The functional requirements of the system must be met.

2. A response must be provided within a specified time period. (Must meet deadlines)

b) Name the three types of hardness of real-time systems and make a short explanation. Give an example of each. (3P)

1. *Hard real-time systems* must meet their deadlines. Not meeting deadlines leads to total failure. (example: nuclear reactor control system)

2. *Soft real-time systems* can miss their deadlines, but this can cause the result to may be unusable. (example: weather stations)

3. *Firm real-time systems* can exceed their deadlines, but this will always lead to an unusable result. (example: video conferencing system)

c) Specify the remaining criteria (not those of b) to classify real-time systems. (3P)

1. They can be classified into *time-driven* or *event-driven* real-time systems.

2. They can be classified into *distributed* and *centralized* real-time systems.

3. They can be classified as *interactive* or *autonomous* controlled real-time systems.

4. They can be classified based on reliablity and fault tolerance. Reliable means Soft- and Hardware must work even after a long time. Fault tolerance mean in case it's not reliable mechanisms must exist to still make it reliable.

5. They can be classified in hierarchical or independent real-time systems. Hierachical work accorss multiple layer.

6. They can be classified as periodic, aperiodic, sporadic real-time systems.

7. They can be classified based on cyclic or asychronous scheduling

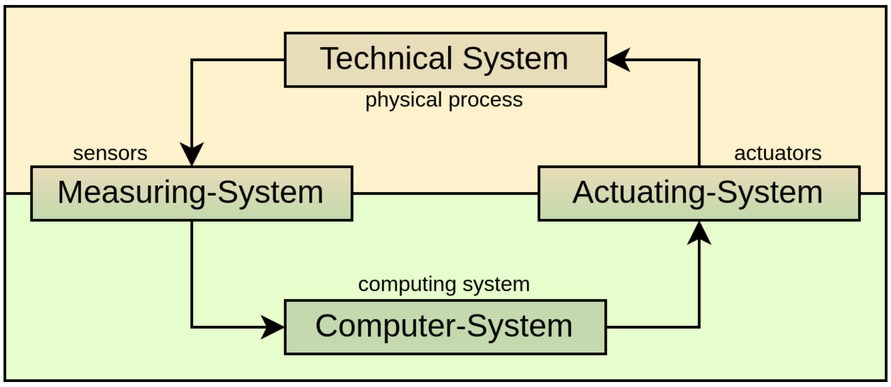
c) Name and explain all possible tasks of a real-time system.

1. *Periodic tasks:* A task that repeats after a fixed time interval.

2. *Sporadic tasks:* A task that recurs at random times, with a minimum time interval between two tasks.

3. *Aperiodic tasks:* can occur at random instants, where the minimum distance between two consecutive tasks can be 0.

d) Explain and draw the idealized model of real-time systems.



The technical system is the system to be controlled, e.g. a drone. A measuring system uses sensors to measure data from the technical system. It converts the physical processes of the technical system into electrical signals and forwards them to the computer system. This processes the electrical signals coming from the sensor. After processing, the electrical signals are forwarded to the actuator system, which converts the electrical signals into physical processes. This is returned to the technical system, which is controlled accordingly.

e) Explain how a timer works.

1. The timer is initalized.
2. The duration is set.
3. The start event is triggered.
4. The timer alarm is decreasing.
5. When the time is <= 0 the timer rings

f) Explain how a stopwatch works.

In the case of a stopwatch, nothing has to be set beforehand. A stopwatch starts at the reference point 00:00:00s. The operation of a stopwatch is done in 4 steps.

1. The trigger leads to the start of the stopwatch.
2. The stopwatch counts up and displays the relative time difference to the time of triggering.
3. The trigger is released again and the stopwatch stops.
4. The stopwatch displays the time difference between the stop and start processes.

g) Explain how a stopwatch with laps works.

A stopwatch with lap times works similar to a stopwatch with additional functions. Between start and stop time there is an additional event called lap times. An additional timestamp is created for the calculation of a lap. The timestamp is a relative time and is the difference between two aboslute subsequent timestamps.

h) Name additional requirements of a real-time system

- Determinability and predictability
- Safety-critical
- Reliability

i) Requirements for real-time-clocks

- High resolution
- High accuracy (low jitter)
- Smooth time adjustment

j) Name an advantage and possible issues of concurrency

- *Advantage:* Multiple processes can be evenly distributed too all cores or systems within a network
- *Possible issues:* deadlocks, livelocks, race confition

Exercise 2: Earliest Deadline First (Take your time and work carefully)

Schedule taskset $T^k = (\Delta t_{exec}{}^k, \Delta t_{dead}{}^k)$: $T^1 = (1,3)$; $T^2 = (2,7)$; $T^3 = (3,11)$ for slice 0..26

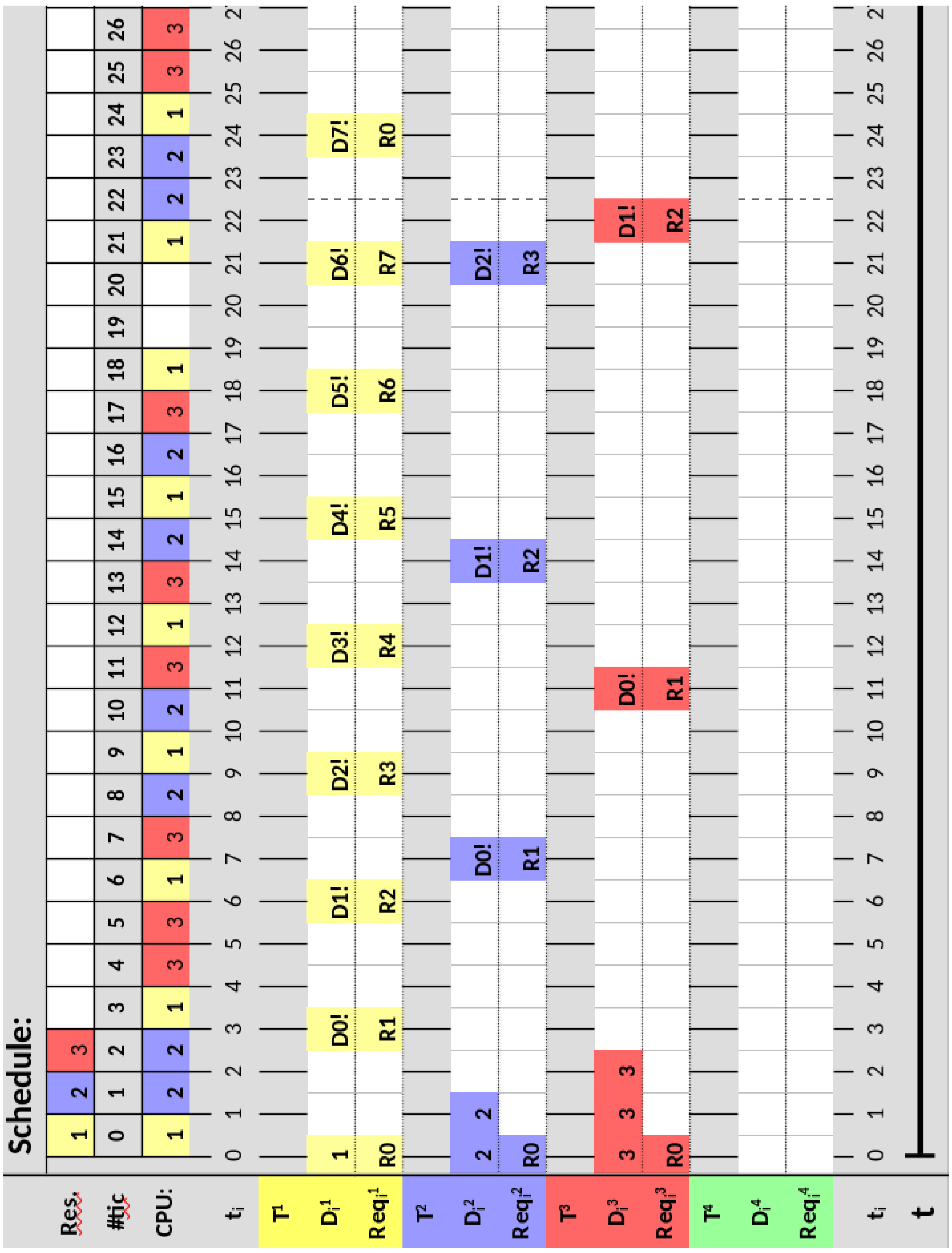
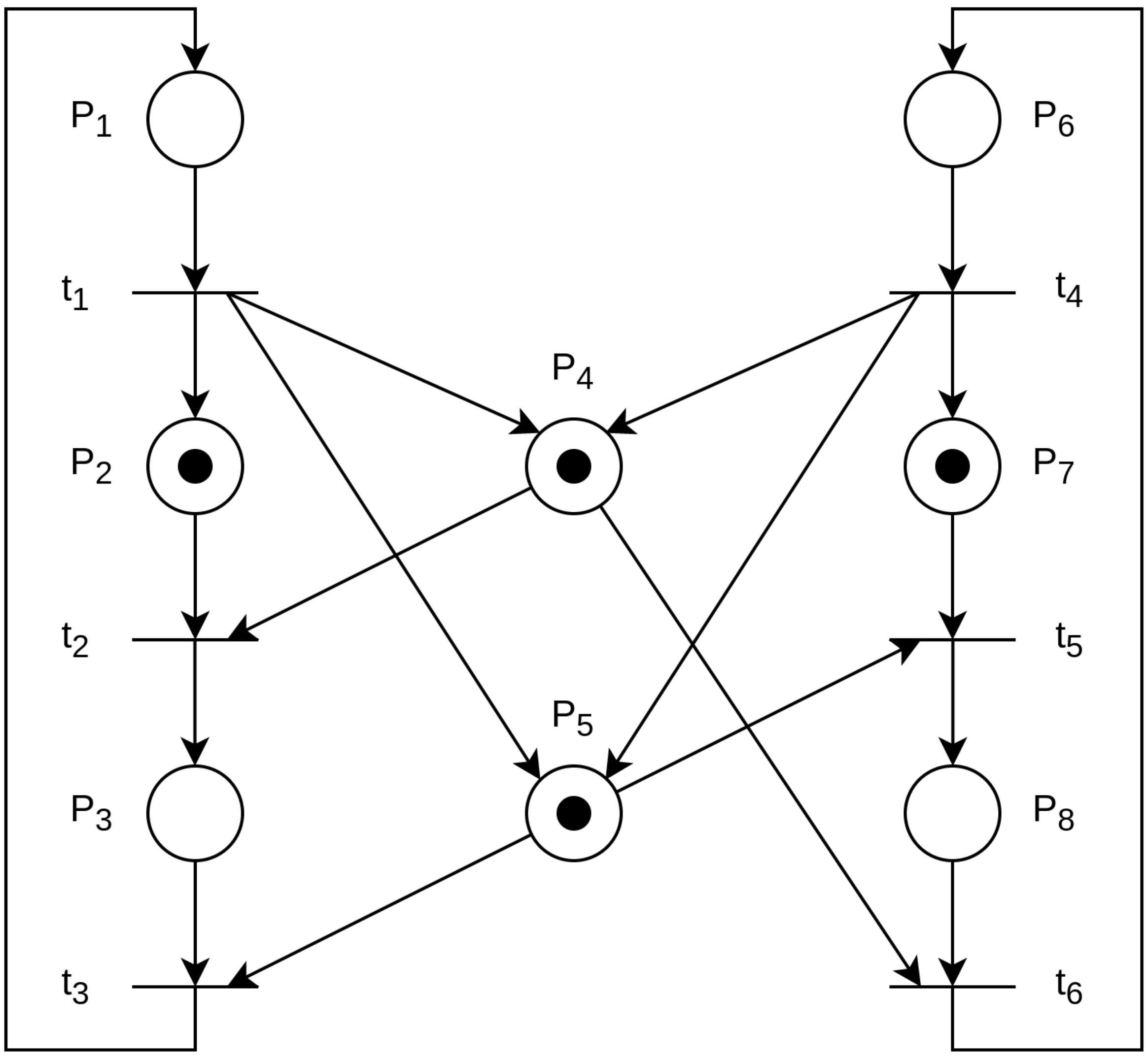


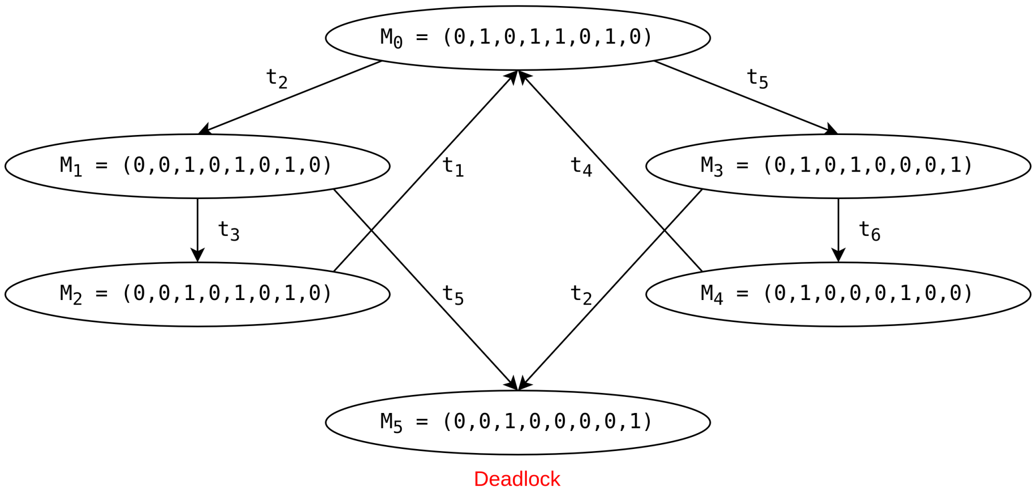
Figure 1: EDF (Slice 0..26)

Exercise 3: (10 Points)

Please analyse the Petri net on the next page:



Please prepare a detailed reachability graph with all places! (4P)
(You may use the back side of the paper, if necessary)



Please verify or falsify the following properties!
Underline the result and give a short argument for your decision.

The net is alive/ is not alive (0,66P)

because (1,33P): Thus a deadlock exist (after t_2, t_5 or after t_5, t_2) it's not alive.

The net is safe/ is not safe (0,66P)

because (1,33P): The petri net is safe because it's 1-bounded.

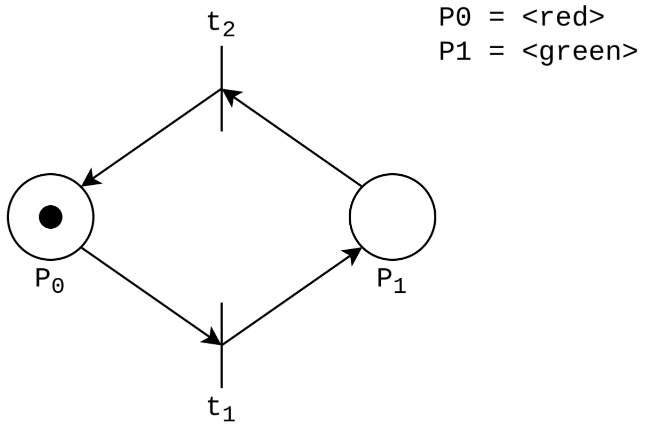
All places are reachable/ some places are not reachable (0,66P)

because (1,33P): Every place can be reaches

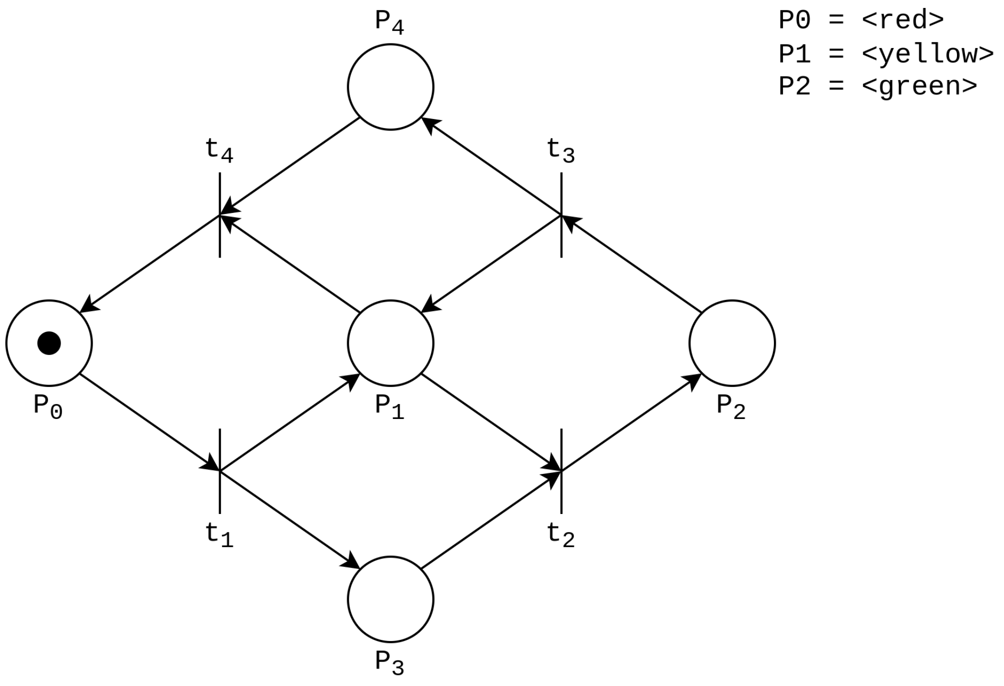
Exercise 4: (8 Points) Petri-Net for traffic lights (do not care about times of the traffic lights)

(Hint: No inhibitors-net, no timed Petri-net, no colored Petri-net)

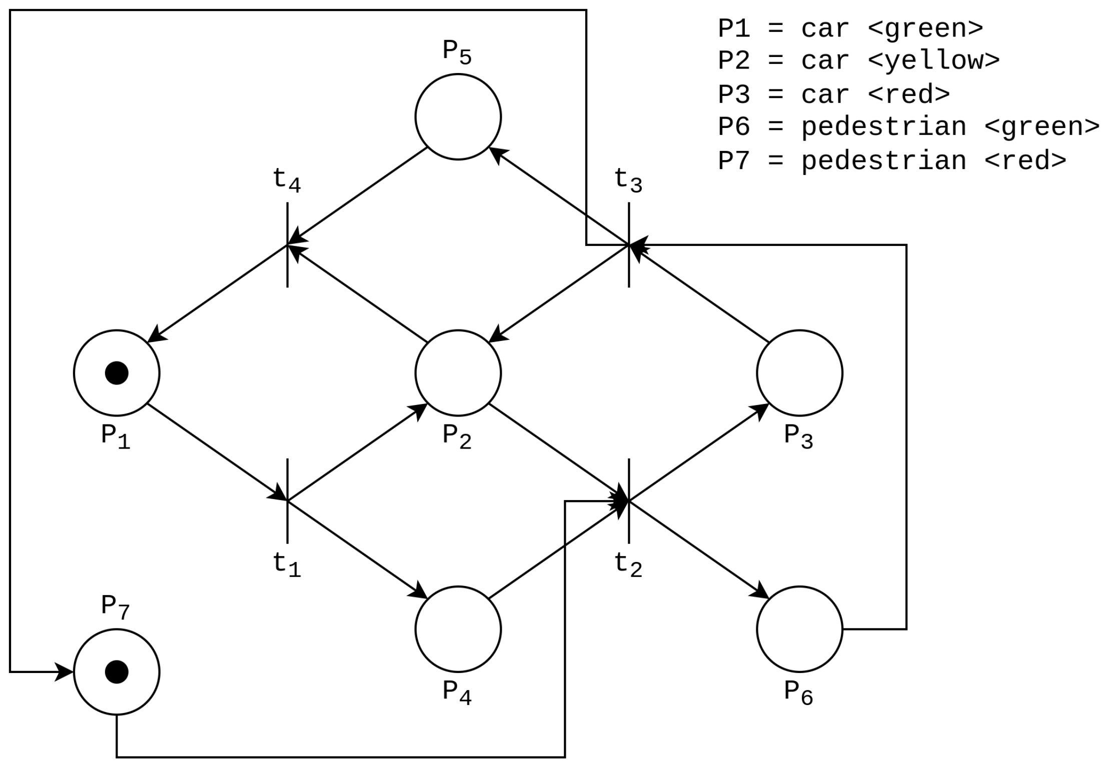
a) Please model a Petri-Net that represents the different statuses of traffic lights for pedestrians. It has two states <red> and <green> (2P)



b) Please model a Petri-Net that represents the different statuses of traffic lights for automotive traffic.
It has three states: <red>, <yellow> and <green> (3P) The switching sequence is <red> to <yellow> to <green> to <yellow> to <red>. There is one color only at a time.



c) Combine the two Petri-Nets of a) and b) in that way that pedestrians have <green> only when the automotive traffic has <red> (make them mutual exclusive). (3P)



Exercise 5: Bicycle (3 Points)

The rate of the foot pedal driven by the driver is between 0 and 120 tunes per minute.

The gear ("Gangschaltung") is 52 teeth (front) to 13 teeth (back) = 4:1. That means the wheels of the bicycle turn 4 times faster than the foot pedal.

The rate of heart-beat of the driver is between 60 and 180 beats per minute.

Your task is to design a bicycle computer that measures the rates of the:

1. foot pedal, 2. back wheel, 3. the heart beat of the driver.

Please calculate the minimum period (in seconds) for the three parameters:

$$\begin{aligned}
\min(\Delta \mathrm{t_{per}}^{\mathrm{foodpedal}}) &= \frac{1}{\mathrm{rate}} = \frac{1}{120 : 60 \text{ s}} = 0,5 \text{ s} \\
\min(\Delta \mathrm{t_{per}}^{\mathrm{wheel}}) &= \frac{1}{(120 \cdot 4) : 60 \text{ s}} = \frac{1}{8} \text{ s} = 0,124 \text{ s} \\
\min(\Delta \mathrm{t_{per}}^{\mathrm{heartbeat}}) &= \frac{1}{180 : 60 \text{ s}} = 0,32 \text{ s}
\end{aligned}$$

Exercise 6a: (in relation to No 5. You can solve 6 without 5)

If you use units of 1/24 seconds you will get the following paramters.

$\min(\Delta \mathrm{t}_{\mathrm{per}}{}^{\mathrm{foodpedal}}) = 12$

$\min(\Delta \mathrm{t}_{\mathrm{per}}{}^{\mathrm{wheel}}) = 3$

$\min(\Delta \mathrm{t}_{\mathrm{per}}{}^{\mathrm{heartbeat}}) = 8$

The execution times for the computer are given:

$\min(\Delta \mathrm{t}_{\mathrm{exec}}{}^{\mathrm{foodpedal}}) = 2$

$\min(\Delta \mathrm{t}_{\mathrm{exec}}{}^{\mathrm{wheel}}) = 1$

$\min(\Delta \mathrm{t}_{\mathrm{exec}}{}^{\mathrm{heartbeat}}) = 3$

This leads to $\mathrm{T}^{\mathrm{i}} = (\Delta \mathrm{t}_{\mathrm{exec}}{}^{\mathrm{i}}, \Delta \mathrm{p}^{\mathrm{i}})$

$\mathrm{T}^{\mathrm{footpedal}} = (2, 12)$; $\mathrm{T}^{\mathrm{wheel}} = (1, 3)$ $\mathrm{T}^{\mathrm{heartbeat}} = (3, 8)$

Please calculate the load of the CPU for all three tasks and in total (2P)

(you need to write down the complete calculation, and not the result only)

What does this mean for the schedulability (Yes/No/Perhaps) any why) (2P)

Load of Tasks: $U_i = \dfrac{\Delta t_{i,\text{exec}}}{D_i}$

Systemload: $U = \sum_{i=1}^{n} U_i = \sum_{i=1}^{n} \dfrac{t_{i,\text{exec}}}{p_i}$, where n is the number of periodic tasks T

Load $U = 1$ means the processor never idles

Load $U > 1$ there is no feasible schedule

Load $U < 1$ means that this test does not exclude that a feasible schedule may exist

Schedulability Test: $U \leq n \cdot (2^{\frac{1}{n}} - 1)$

Load of Task $\text{T}^{\text{footpedal}}$: $U_1 = \dfrac{2}{12} = \dfrac{1}{6}$

Load of Task T^{wheel}: $U_2 = \dfrac{1}{3}$

Load of Task $\text{T}^{\text{heartbeat}}$: $U_3 = \dfrac{3}{8}$

$n = 3$, because number of period tasks T is 3 ($\text{T}^{\text{footpedal}}$, T^{wheel}, $\text{T}^{\text{heartbeat}}$)

$$\begin{aligned}\text{Systemload: } U &= \sum_{i=1}^{3} U_1 + U_2 + U_3 \\ &= \sum_{i=1}^{3} \frac{1}{6} + \frac{1}{3} + \frac{3}{8} = \frac{7}{8} = \underline{\underline{0{,}875}} \rightarrow \text{feasible schedule may exist.}\end{aligned}$$

Schedulability Test: $U = 0,875 \nleq 3 \cdot (2^{\frac{1}{3}} - 1) = 0,779 \rightarrow$ feasible schedule may exist

Exercise 6b: refering to Exercise 6a

Please assign priorities to the three tasks (a high number should be a high priority) (2P)

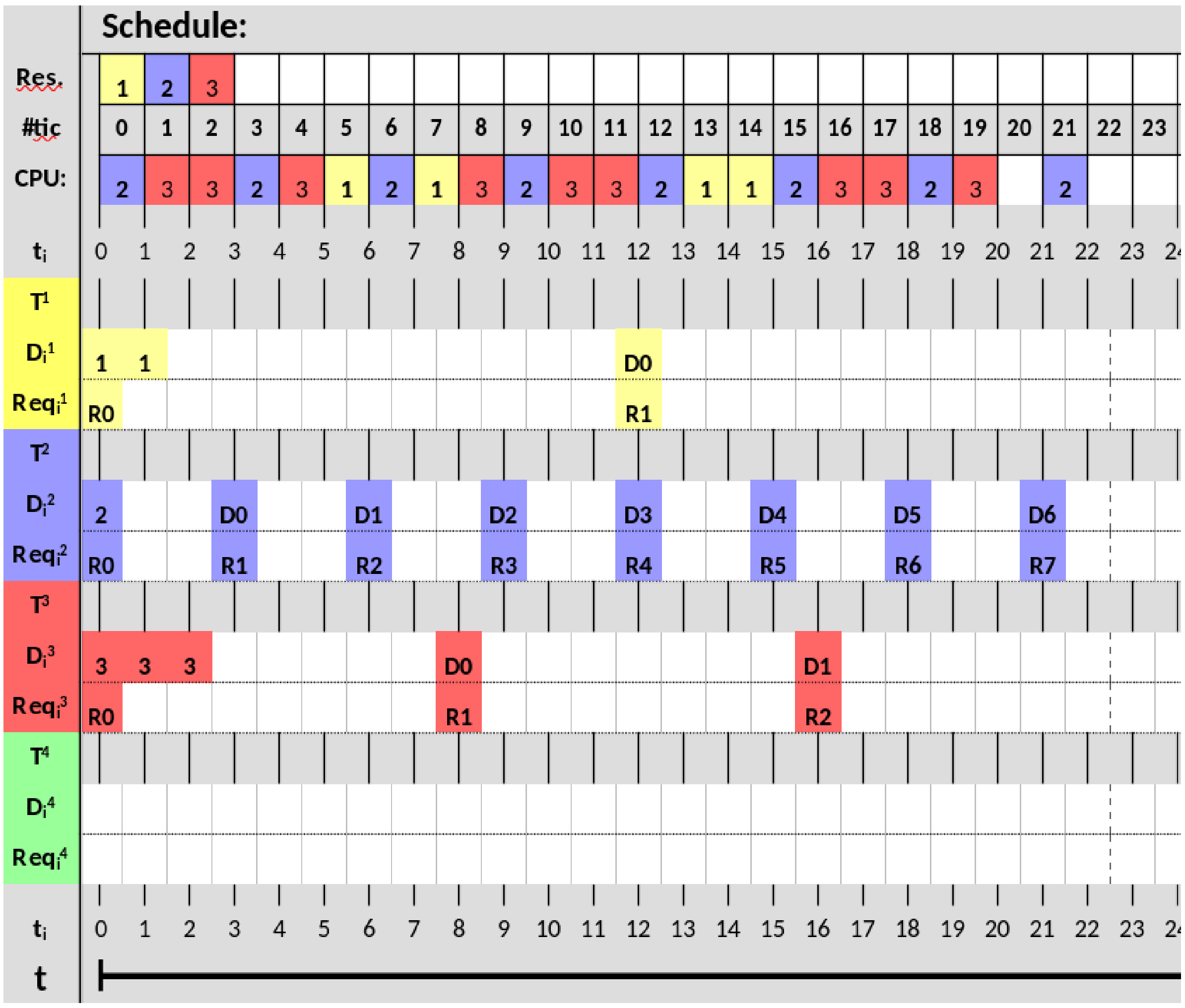
$$\min(\text{proprity}^{\text{footpedal}}) = 1$$

$$\min(\text{proprity}^{\text{wheel}}) = 3$$

$$\min(\text{proprity}^{\text{heartbeat}}) = 2$$

Schedule the problem on the solution sheet with RMS, interruptible tasks.

Fill in A or B or C or nothing in row CPU for slices 0..23 (6P)



Res.	1	2	3																					
#tic	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
CPU:	2	3	3	2	3	1	2	1	3	2	3	3	2	1	1	2	3	3	2	3		2		
t_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
T^1																								
D_i^1	1	1											D0											
Req_i^1	R0												R1											
T^2																								
D_i^2	2			D0			D1			D2			D3			D4			D5			D6		
Req_i^2	R0			R1			R2			R3			R4			R5			R6			R7		
T^3																								
D_i^3	3	3	3						D0								D1							
Req_i^3	R0								R1								R2							
T^4																								
D_i^4																								
Req_i^4																								
t_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Figure 2: RMS (Slice 0..23)

Exercise 7: (4 Points)

You shoud use a microcontroller (e.g STR9) and a 7-segment-display. Given the specification of a 7-segment-display connected to a microcontroller-ports.

The seven segment display is connected to the following ports:

– segment g at GPIO 4.0

– segment f at GPIO 4.1

– segment e at GPIO 4.2

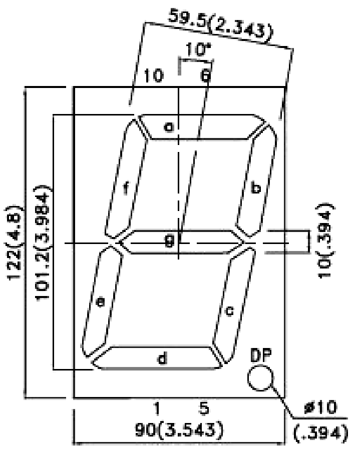
– segment d at GPIO 4.3

– segment c at GPIO 4.4

– segment b at GPIO 4.5

– segment a at GPIO 4.6

The segments of this board are active low.
You can set all bits at one time by: `GPIO4 -> DATA[255<<2] = BITARRAY`, e.g to display a 1: `1100 1111 = 0xCF -> GPIO4 -> DATA[255<<2] = 0xCF`



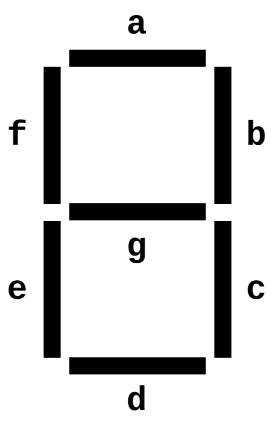
What hexadecimal number you need to store to display a 4.

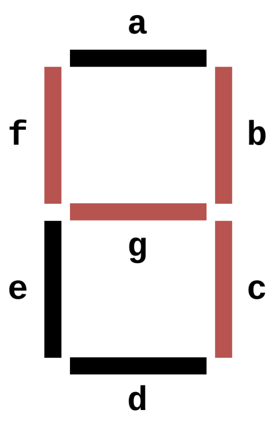
`GPIO4 -> DATA[255<<2] = 0x`_____.

What hexadecimal number you need to store to display a 2.

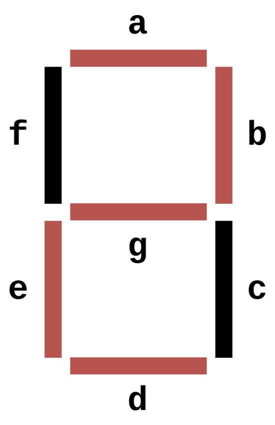
`GPIO4 -> DATA[255<<2] = 0x`_____.

Active Low → `1 = OFF`, `0 = ON`





7-segment display when
4 is displayed



7-segment display when
2 is displayed

What hexadecimal number you need to store to display a 4.

	a	b	c	d	e	f	g
1	1	0	0	1	1	0	0
	OFF	ON	ON	OFF	OFF	ON	ON

`1100 1100 = CC`

`GPIO4 -> DATA[255<<2] = 0xCC`.

What hexadecimal number you need to store to display a 2.

	a	b	c	d	e	f	g
1	0	0	1	0	0	1	0
	ON	ON	OFF	ON	ON	OFF	ON

`1001 0010 = 92`

`GPIO4 -> DATA[255<<2] = 0x92`.

Exercise 8a: (4 Points)

Name at least 8 different functional units that are integrated on a microcontroller development chip.

1. Microprozessor
2. RAM
3. Memory
4. Serial bus interface
5. Input/output ports
6. Programmable read only memory

Exercise 8b: (extra 2 Points)

What is a minor page fault? Explain it one or two senteces (1,5P)

A page fault is an exception type triggered by computer hardware when a running program accesses a memory page that is not currently mapped by the memory management unit (MMU) in the virtual address space of a process. If the page is loaded into memory at the time the error occurs, but is not marked in the memory management units that are loaded into memory, it is called a minor or soft page fault. No I/O operations are required

Why are minor page faults unwanted in real time systems and are prevented by `mlockall(MCL_CURRENT | MCL FUTURE)`? (0,5P)

Exercise 9: Explain what active high and active low is.

- *Active Low:* A signal is "active low", i.e. the signal performs its function when its logic level is 0.

- *Active High:* A signal is "active high", i.e. the signal performs its function when its logic level is 1.

Exercise 10: Explain what the Waiting Time Δt_{wait} is.

The waiting time Δt_{wait} is the time difference between the point in time where the executing of an job starts (t_S) and the point in time where the job was requested (t_{Req}).

Exercise 11: Explain what the Execution Time Δt_{exec} is.

The execution time Δt_{exec} is the duration in which a job is executed, i.e. the difference between the completion time (t_C) and the point in time when the job was started (t_S).

Exercise 12: Explain what the Response Time Δt_{resp} is.

The execution time Δt_{resp} is the difference between the completation time (t_C) of an request and the point in time where the job was requested (t_{Req}).

Exercise 13: Explain what the Slack Time Δt_{slack} is.

The slack time Δt_{slack} is the time difference between the absolute deadline (t_D) of a job and the point in time where the job completed (t_C).

Exercise 14: Explain what the Tardiness Δt_{tard} is.

Tardiness is a measure of the delay in the execution of certain job. It is therefore the difference between the point in time of completion of a job (t_C) and the actual deadline of a job (t_D).

Exercise 15: Explain where information about every process in the computer is stored.

A *Process Control Block (PCB)* stores information about every process in the computer.

Exercise 16: When is the Process Control Block created and when it is removed?

When a process is created (initialized) the operating system creates a process control block. When the process terminates the process control block is removed.

Exercise 17: Where is the Process Control Block stored?

The process control block is stored in a protected memory area of the operating system kernel.

Exercise 18: What information does a Process Control Block store?

- *Process State:* Specifies the process state, i.e. new, ready, running, waiting, etc.

- *Process Counter:* Contains the address of the next instruction to be executed in the process.

- *CPU Register:* Specifies the registers that are used by the process. These may include accumulators, stack, pointers, working registers, instruction pointer, etc.

- *CPU Scheduling Information:* It contains the process priority, pointers to the standby queue and device queue, and the device queue and scheduling information.

- *Accounting Information:* The information includes the total CPU time used, the actual time used, the process number, etc.

- *I/O Status Information:* It contains a list of I/O devices assigned to a process and a list of files opened by the process on the hard disk. A file can be open for either read or write.

- *Memory-Mangement Information:* It contains the addresse of the page tables or the segment tables. It also contains the value of the base registers, limit registers, etc.

Exercise 19: Explain what an interrupt is and how it works.

An interrupt is an event, called trigger, that stops a currently running program, starts another program, the so called interrupt service routine (ISR), returns to the interrupted program and resumes exactly where it was interrupted.

Exercise 20: Name and explain the types of interrupts.

- *Software Interrupts:* Is the interrupt that is generated by any internal system of the computer. It has highest priority among all interrupts.

- *Hardware Interrupts:* Is an interrupt generated from an external device or hardware. It has lowest priority than software interrupts.

Exercise 21: Explain what a quantization error is.

A quantization error is the difference between the analog signal and the closest available digital value in each sample

Exercise 22: Explain what the sample theorem is.

The sampling theorem states that a signal can be reproduced exactly if it is sampled at a frequency F, where F is greater than twice the maximum frequency.

Exercise 23: Explain what anti-aliasing is.