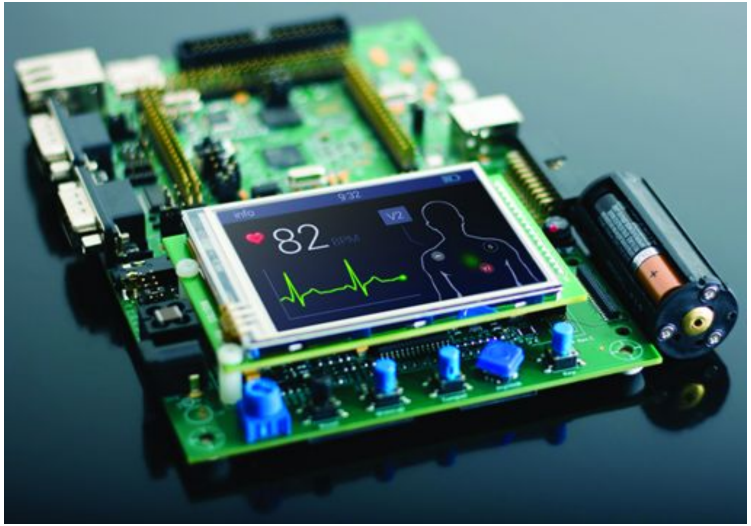
Embedded Software

CS 145/145L



Caio Batista de Melo

Announcements (2022-05-12)

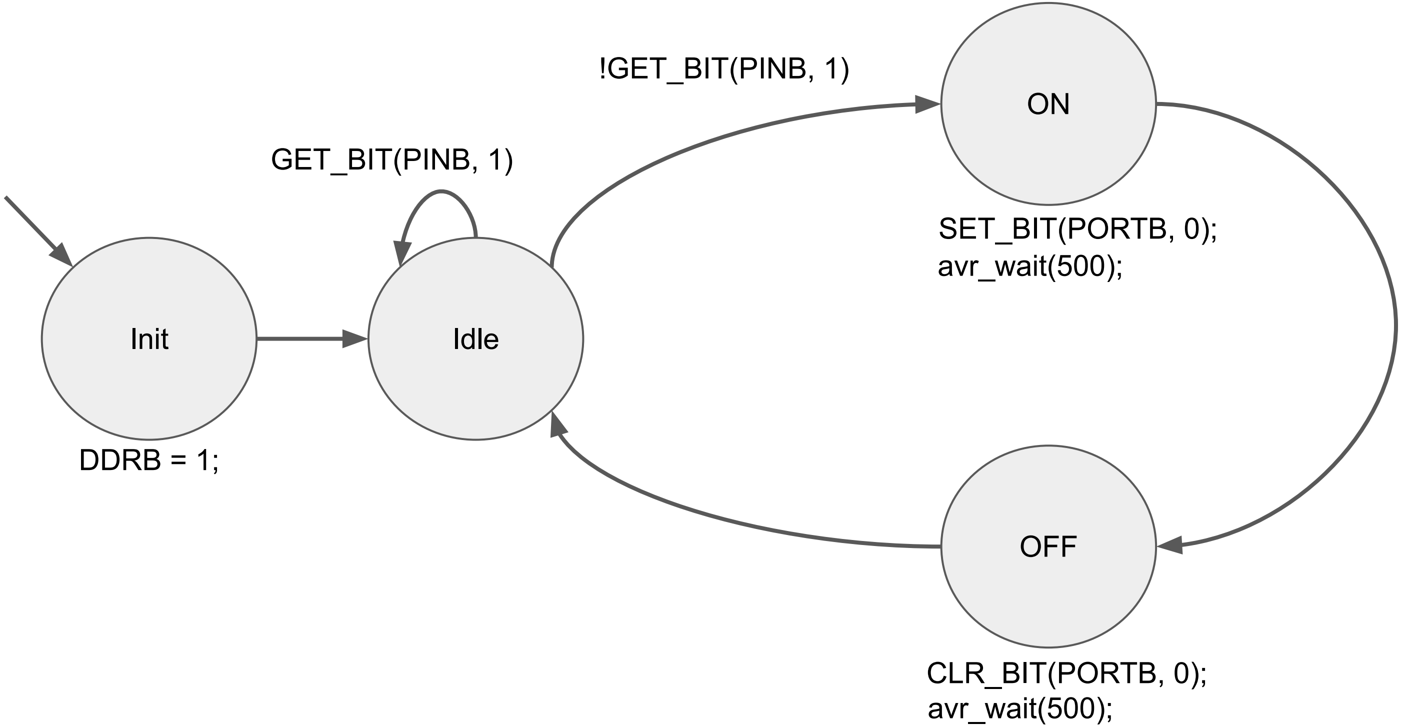
- We’ll try to have two guest lecturers this quarter!
- First one will be next Thursday (2022-05-19)
    - Talk about real-time systems
    - Active researcher – recently published in that field!
- Second one (tentatively) will be on the Tuesday following that (2022-05-24)
    - Industry speaker
    - TBD

Agenda

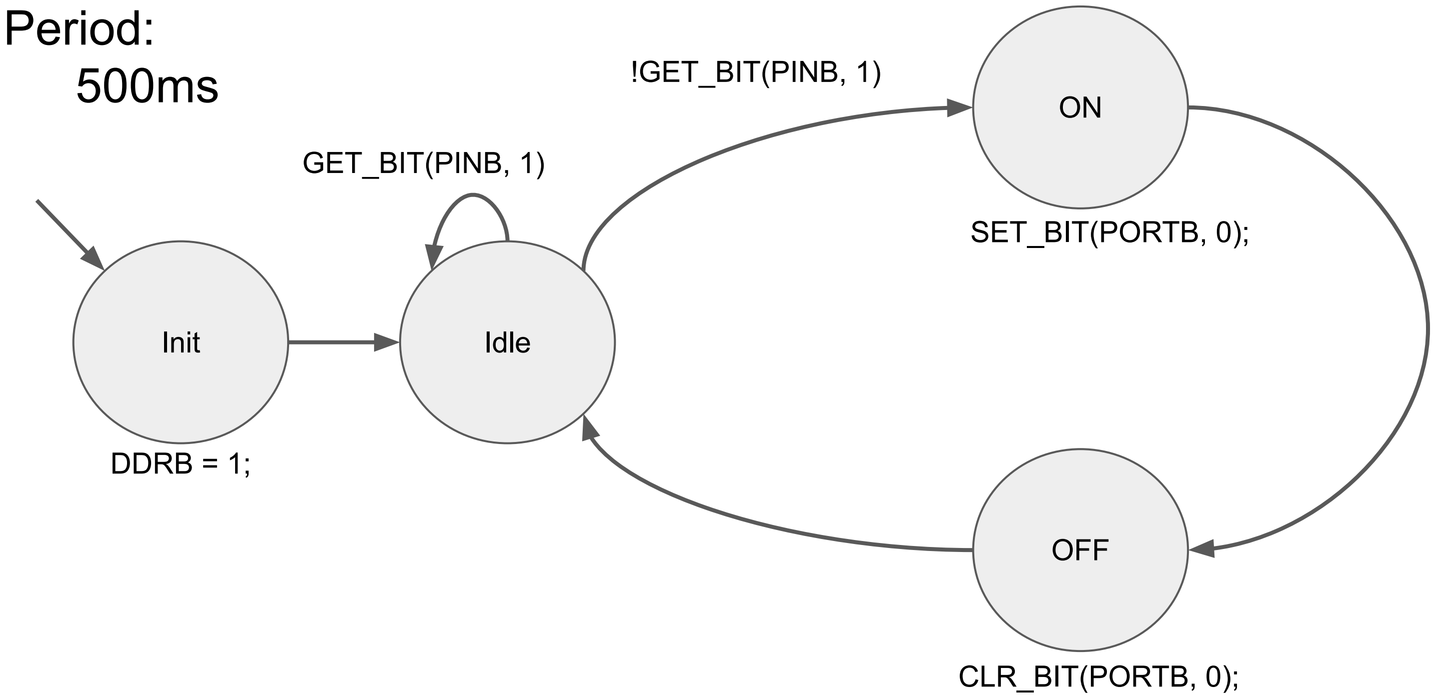
- Recap
- Simple Cooperative Scheduler
- Problems
  - Timer Overrun
  - Utilization
  - Worst-Case Execution Time (WCET)
- Preemptive Scheduler
  - Scheduling States
  - Priorities
- Examples

Recap

State Machine

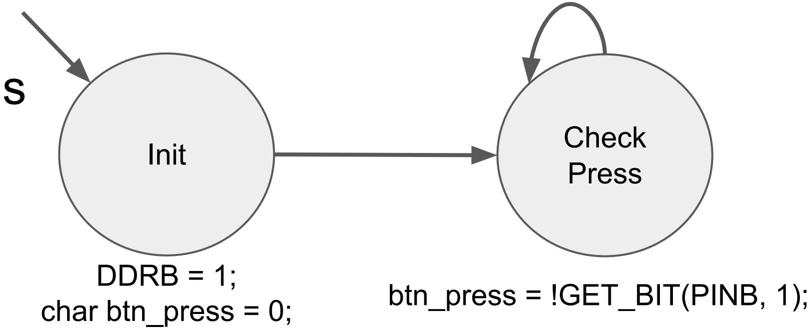


Synchronous State Machine

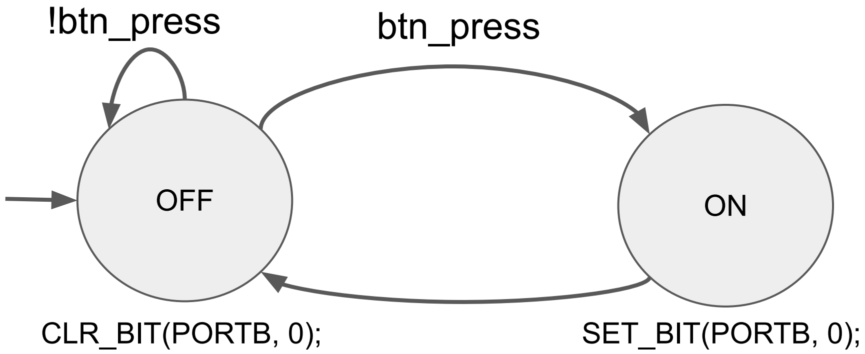


Concurrent Synchronous State Machines

Period:
10ms



Period:
500ms



Sharing Time Over Tasks

Sharing Time

- You can have multiple things going on at once
  - Checking for input
  - Blinking a light
  - Playing a note
- Without multi-processing (i.e., if you can only run one task at a time), how can you do everything?
- Run each thing for a little bit
  - Share the processor’s time with all your tasks

Simple Cooperative Scheduler

Task Structure

```c
typedef struct task {
   int state;                   // Task's current state
   unsigned long period;        // Task period
   unsigned long elapsedTime;   // Time elapsed since last task tick
   int (*TickFct)(int);         // Task tick function
} task;
```

synchSM task from zybooks

Cooperative Scheduler

```
// For each task, call task tick function if task's period is up
for (i=0; i < tasksNum; i++) {
   if (tasks[i].elapsedTime >= tasks[i].period){
      // Task is ready to tick, so call its tick function
      tasks[i].state = tasks[i].TickFct(tasks[i].state);
      tasks[i].elapsedTime = 0; // Reset the elapsed time
   }
   tasks[i].elapsedTime += tasksPeriodGCD;
}
```

Scheduler from zybooks

Add Interrupt Handling

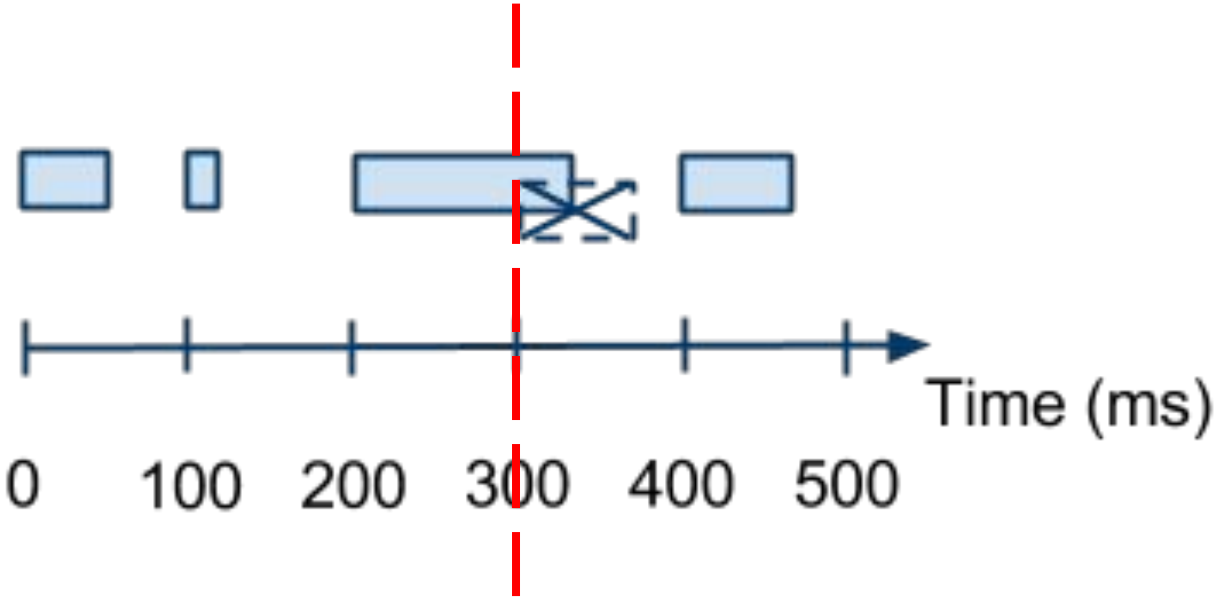
- Set interrupts to happen whenever you want to check for new tasks
  - Probably related to tasks' periods

- Whenever your interrupt happens, you go through all your tasks in the ISR

- Assuming all tasks finish quickly, this should allow everything to execute according to their periods.

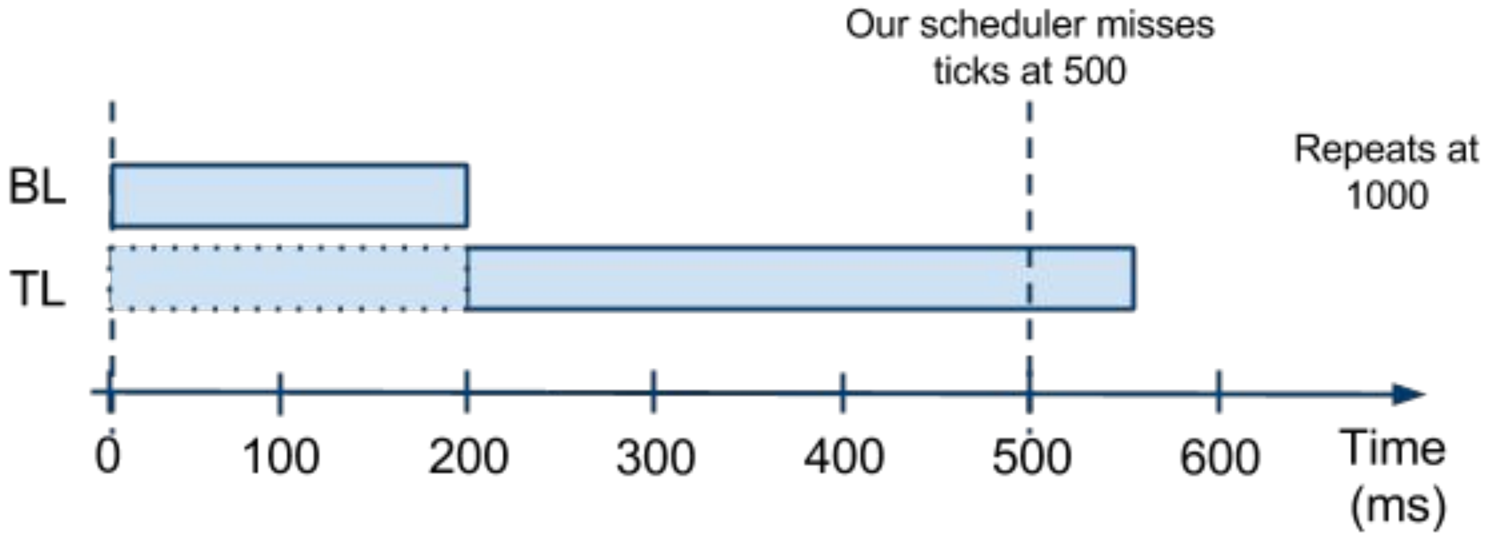
Problems

Timer Overrun



The tasks take longer to complete their states than their period.

Utilization



How much of the available time we're using.
In this example, for 500ms periods, we use 550 / 500 = 110%

Worst-Case Execution Time (WCET)

CountThree   Period: 500 ms

unsigned char cnt;

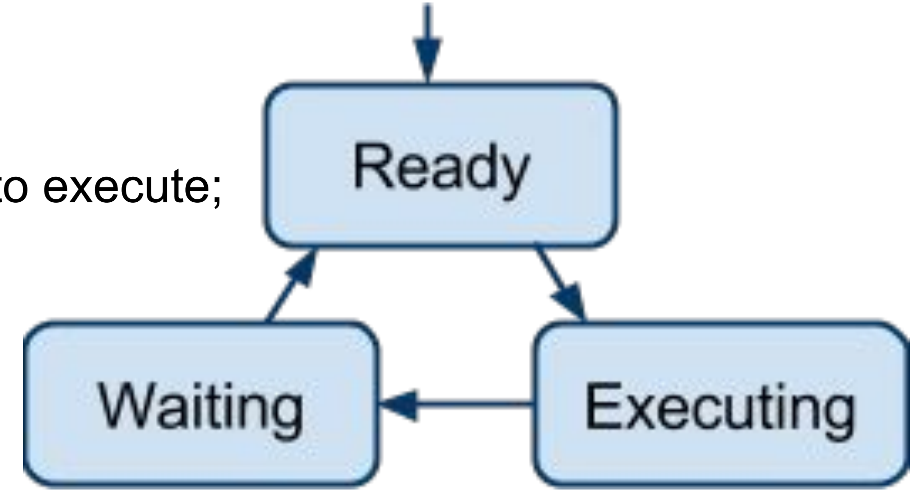
	Estimated assembly instructions
cnt = 0;	3
cnt = cnt + A0;	+ 3
cnt = cnt + A1;	+ 3
cnt = cnt + A2;	+ 3
cnt = cnt + A3;	+ 3
B1 = (cnt >= 3);	+ 3 + 2 Total: 20

WCET is used a lot in real time systems!
For example, what is the maximum time to process a video frame;
Or how long it takes to finish an ADC conversion (page 206 of manual).

Preemptive Scheduler

Scheduling States

- Tasks goes through these 3 states;
- A scheduler picks one of the ready tasks to execute;
- What makes a task stay in waiting state?
- What if there are many *ready* tasks?



Prioritization

- Different tasks might have different levels of importance
- Your scheduler should try to execute the most important ones first
- To achieve this, it should be able to stop a task mid-execution

Priorities: Example

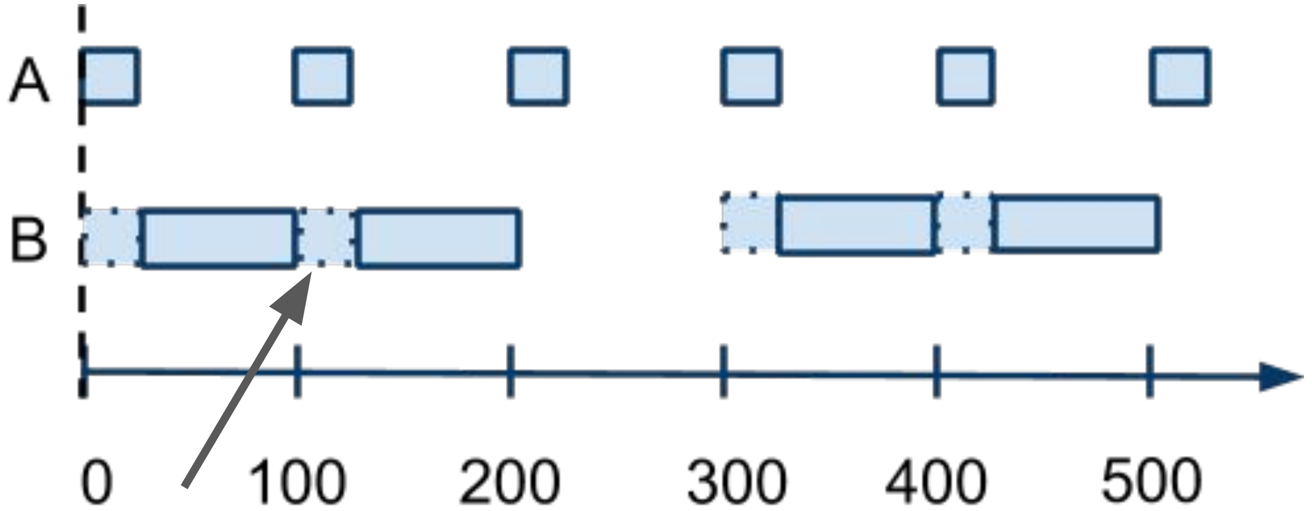
Multiple tasks in a car

- What's more important?
  1. Blinking a turning signal before changing lanes
  2. Braking when a collision is imminent
  3. Changing the radio station

- You probably rank them [2] > [1] >> [3]
- Need to make sure that [2] executes whenever it needs!

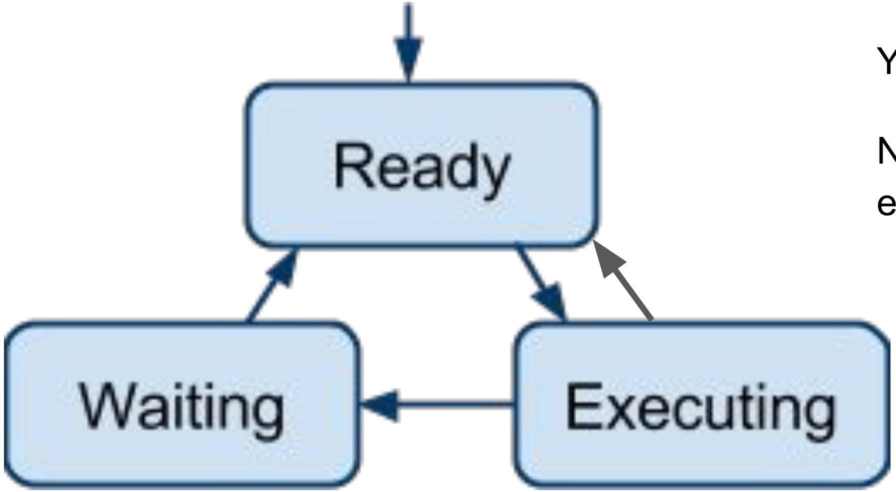
Preemptive Scheduler

Chooses the most important task from the ready pool.



B stops executing so A can execute!
Where does B start executing from after the pause?

States for a Preemptive Scheduler



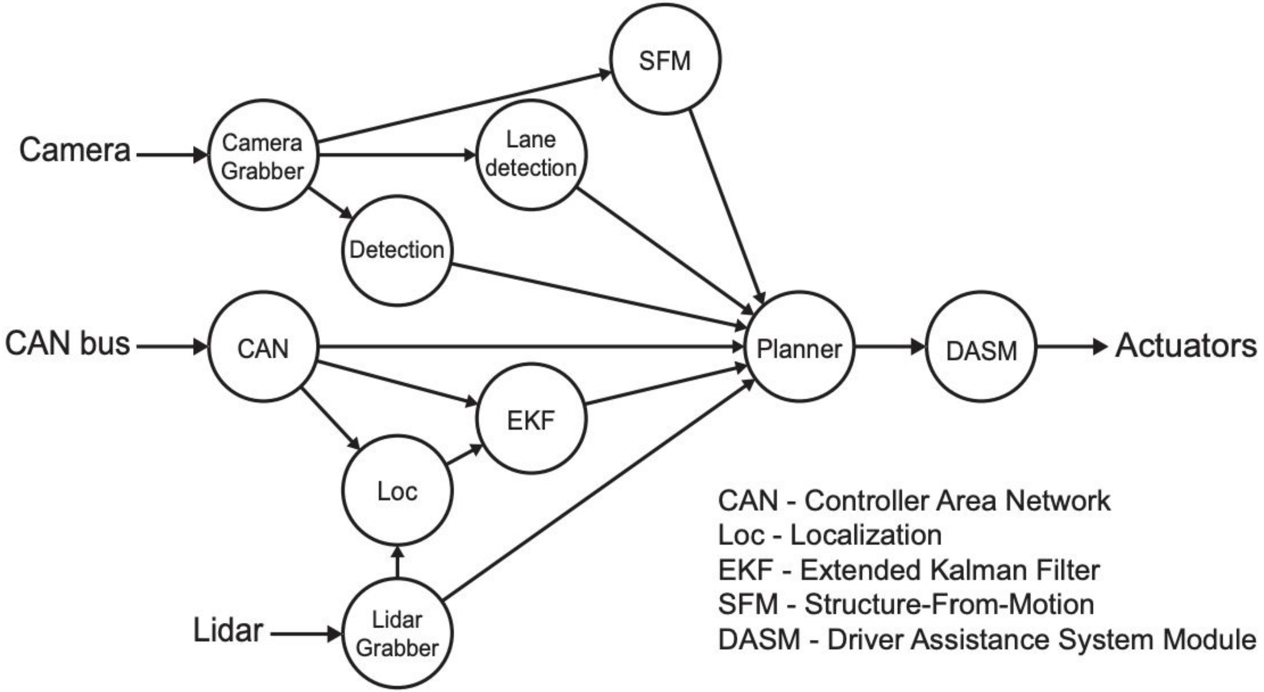
Anything changes?

Yes!

New transition from executing to ready

Examples

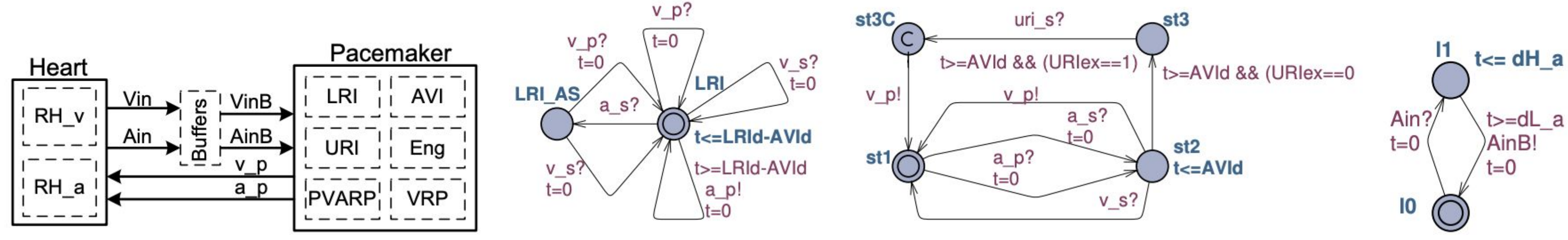
Real Time Systems (Autonomous Vehicles)



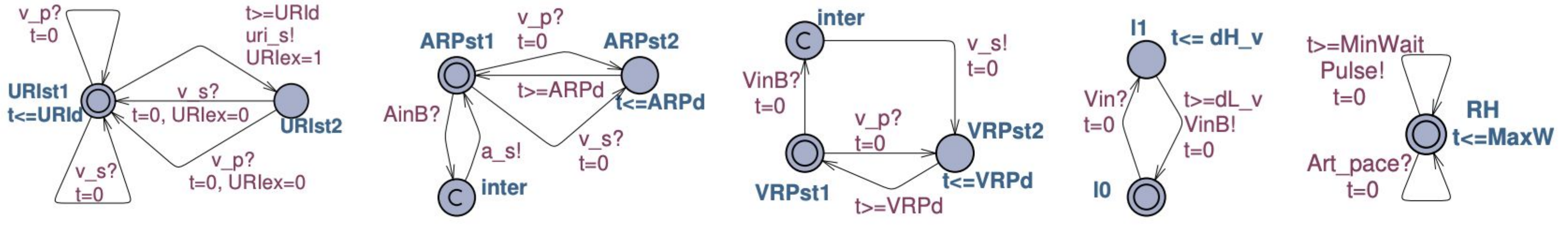
CAN - Controller Area Network
Loc - Localization
EKF - Extended Kalman Filter
SFM - Structure-From-Motion
DASM - Driver Assistance System Module

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9470238

Real Time Systems (Pacemaker)



(a) Interaction between the pacemaker and heart (b) LRI component (c) AVI component (d) Atrial Buffer



(e) URI component (f) PVARP component (g) VRP component (h) Vent. Buffer (i) Random Heart

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6200049

Real Time Operating Systems

- RTEMS: https://github.com/RTEMS/rtems
  - Can run on simulators: https://devel.rtems.org/wiki/Developer/Simulators/gem5
- uC/OS: https://github.com/weston-embedded/uC-OS2
- RTOSes on Raspberry Pi:
  - https://www.cse.unr.edu/~fredh/papers/conf/190-rartosotrp/paper.pdf
  - https://github.com/PicoCPP/RPI-pico-FreeRTOS
  - https://pebblebay.com/raspberry-pi-embedded/

See you next time :)

Q & A