# Lecture 2&3
{ ROS2 }

★ **ros2 bag**
- → file generated *.db3
- → Storage id: sqlite3

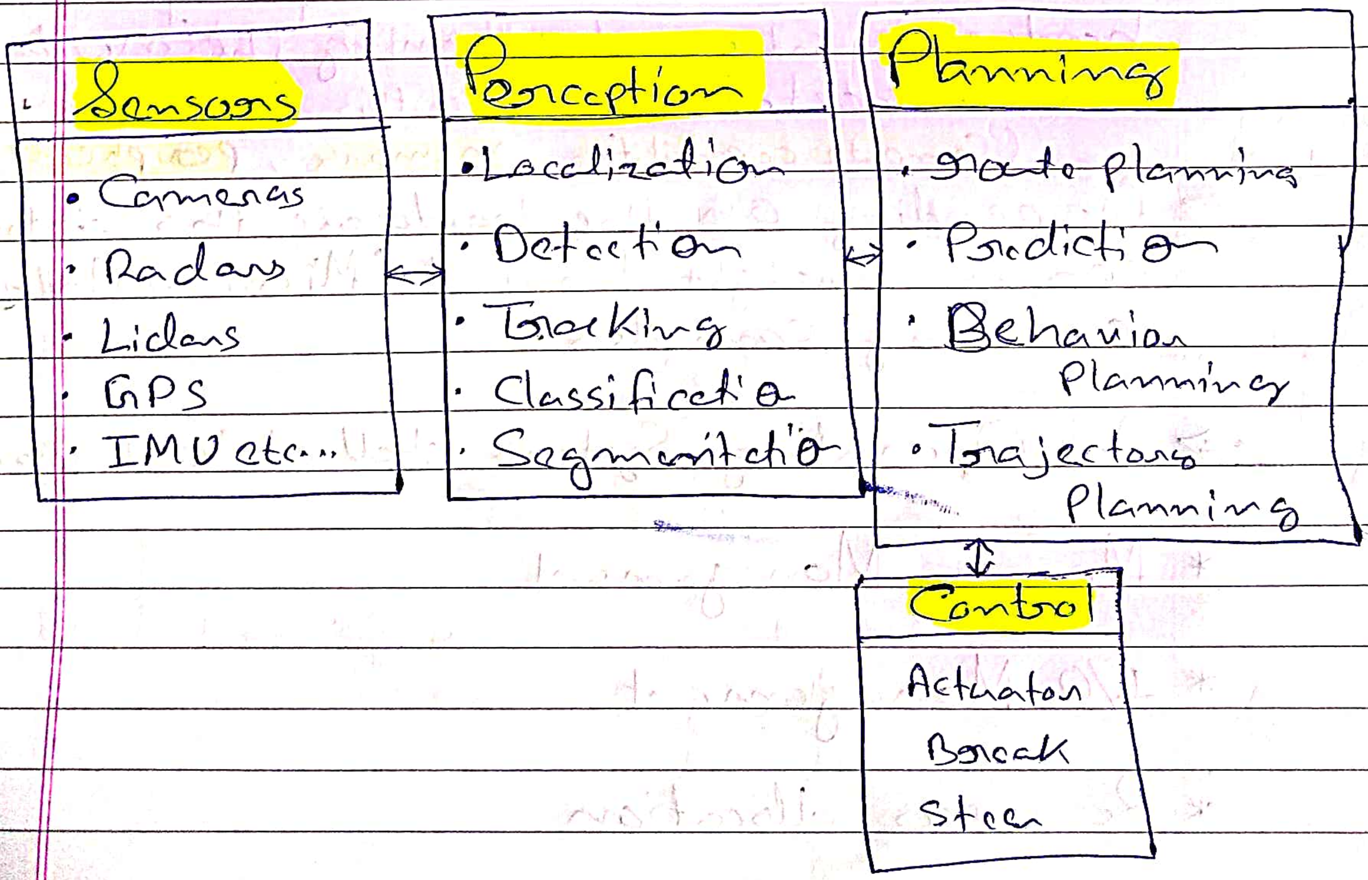
Lecture 4 (Platform HW, RTOS, DDS)

{ Part 1: ECU and RTOS }

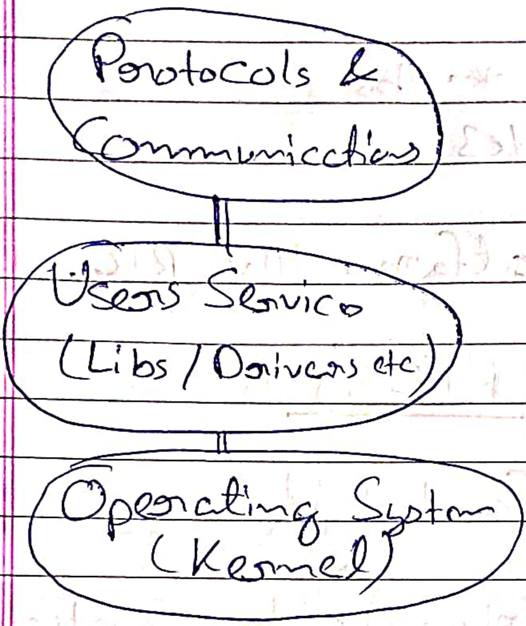
ECU ⇒ Electronic Control Unit

ADAS ⇒ Advanced Driver Assistant System

AD ⇒ Autonomous Driving



★ Supporting Softwares for ECU


Protocols & Communications
Users Service (Libs / Drivers etc)
Operating System (Kernel)

POSIX / DDS / ROS etc...

Provides utilities & Customizations for ECU

Enable Hardware / Abstract multiple hardware

★ Operating System

⇒ For the user to take benefit of the hardware when developing, he needs an abstracted access to this hardware.
(Compute capabilities, memory, peripherals)

⇒ Depending on the hardware this abstraction can be straight forward (Micro-controller) or very complex.

⇒ An Operating System shall at least provide:

* Memory Management

* I/O Management

* Resources allocation

* Error detection and handling

* A Kernel for:
  - Task management:
    - Context switch scheduling
    - Communication
    - Synchronization
  - Interrupt management

★ <u>Real-Time Operating System</u>

⇒ Embedded system require predictable behavior:
  - Deterministic
  - Hard Real-Time
  - Guarantee of time for task completion
  - Highly responsive to external events.

(RTOS)

<u>Deterministic</u>

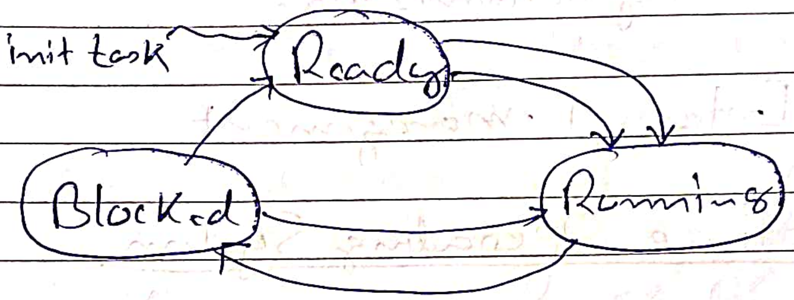
⇒ The time interval between input event & Output event must be predictable.
  - The system always respond with a specified laps of time.

Date
Page

★ Tasks and Tasks Priorities

⇒ In RTOS, you will have many tasks to run, all time sensitive.

⇒ To enable hierarchy of tasks, you can rely on Task State and Priority.



★ Scheduler

⇒ The tasks to be executed within the RTOS must be carefully selected and sequenced.

⇒ Several scheduling algorithms, main ones are:

- Co-operative
- Round-Robin
- Pre-emptive ✓

39:00

Date
Page

★ Interrupts

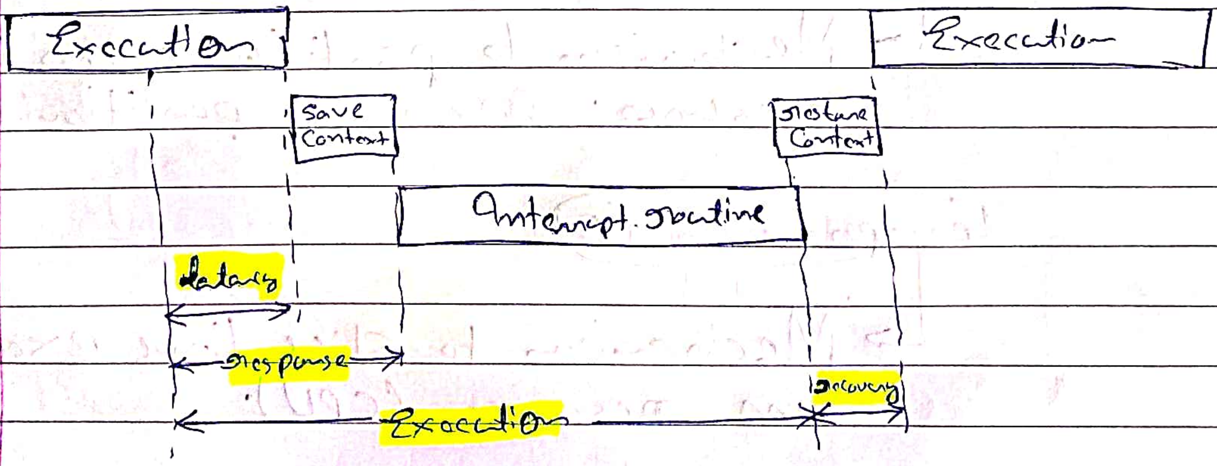
⇒ An interrupt breaks the sequence of Operations:

→ External Interrupts: generated by a peripherals

→ Internal Interrupts: Special instruction in a program or exception

⇒ When an interrupt occurs:

→ Suspend execution of task
→ Save the Context
→ Set the PC to start address of interrupt handler routine
→ Process the interrupt handler
→ Restore the Context.



Date
Page

★ Types of Kernels

→ Micro Kernel
{User services & Kernel services are not in the same address space}

→ Monolithic Kernel
{User service and Kernel services are in the same address space.}

★ Spatial and Temporal Isolation

⇒ An ECU and even CPU are becomming more & more complex by being heterogeneous.
↳ Management of such system requires sometimes more than one RTOS.

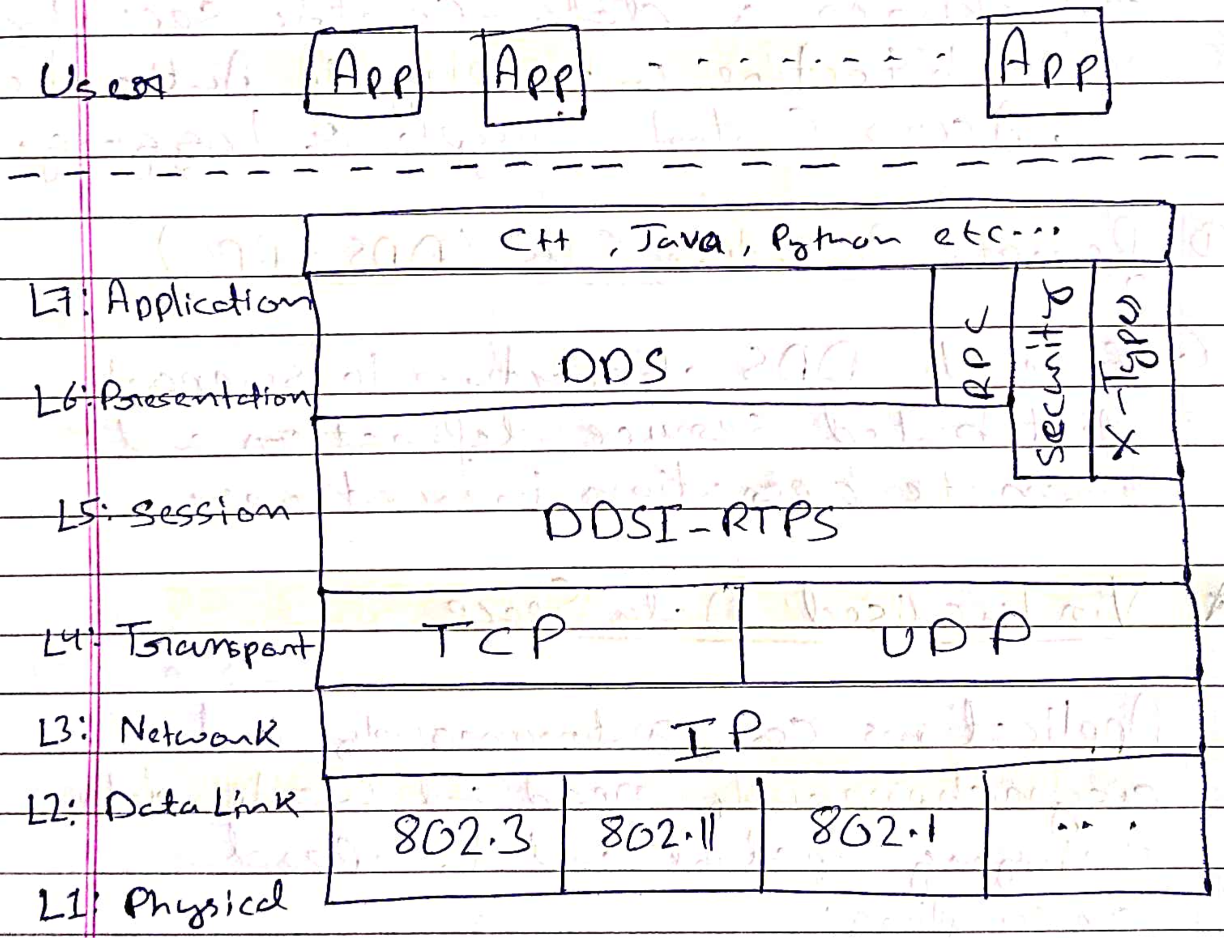
Spatial Isolation
↳ Mechanism to partition access to resources: Memory partitioning.

Temporal Isolation
↳ Mechanism to slice time execution on resources (CPU).

{Part 2: DDS}

"DDS is a standard technology for ubiquitous interoperable, secure, platform independent, and real-time data sharing across network connected devices"

DDS {Data Distribution Service}



① DDS

↳ Defines a high level API for Programming language, OS and architecture independent data sharing.

Date
Page

② X-Types

↳ Extends the DDS type system from nominal to structural, thus providing very good support for evolutions and forward compatibility.

③ DDS - Security

↳ Defines a data-centric security architecture with plugable Authentication, Access Control, Crypto & Logging.

④ Remote Procedure Calls (DDS-RPC)

↳ Extends DDS abstractions to support distributed service defination and remote operation invocations.

★ Virtualised Data Space

Applications can autonomously and asynchronously read and write data enjoying spatial and temporal decoupling.

★ Topic

⇒ A Topic is defined by means of a
<name, type, qos>

Date
Page

★ Dynamic Discovery

⇒ Built-In dynamic discovery isolates applications from network topology and connectivity details.

⇒ No single point of failure or bottleneck.

Cyclone DDS

↳ Eclipse Cyclone DDS was born with the ambition of developing the best DDS implementation ever.

★ DDS Entities

⇒ DDS provides three different entities to control where and what data is read/written.

⇒ DomainParticipant, Publisher and Subscriber relate to the "where".

⇒ DataReader and DataWriter relates to the "what".

⇒ DDS QoS policies control at a large extent to the "how" data is shared.