ADAS made trivial: rappresentazione ispirata alle interazioni in un sistema di guida autonoma

Andrea Ceccarelli
andrea.ceccarelli@unifi.it

Progetto per il corso di Sistemi Operativi
Anno Accademico 2022-2023

NOTA

La lettura di queste slides è fortemente raccomandata, ma HA SOLO SCOPO ESPLICATIVO del progetto.

La specifica del progetto è presente sul sito del corso.

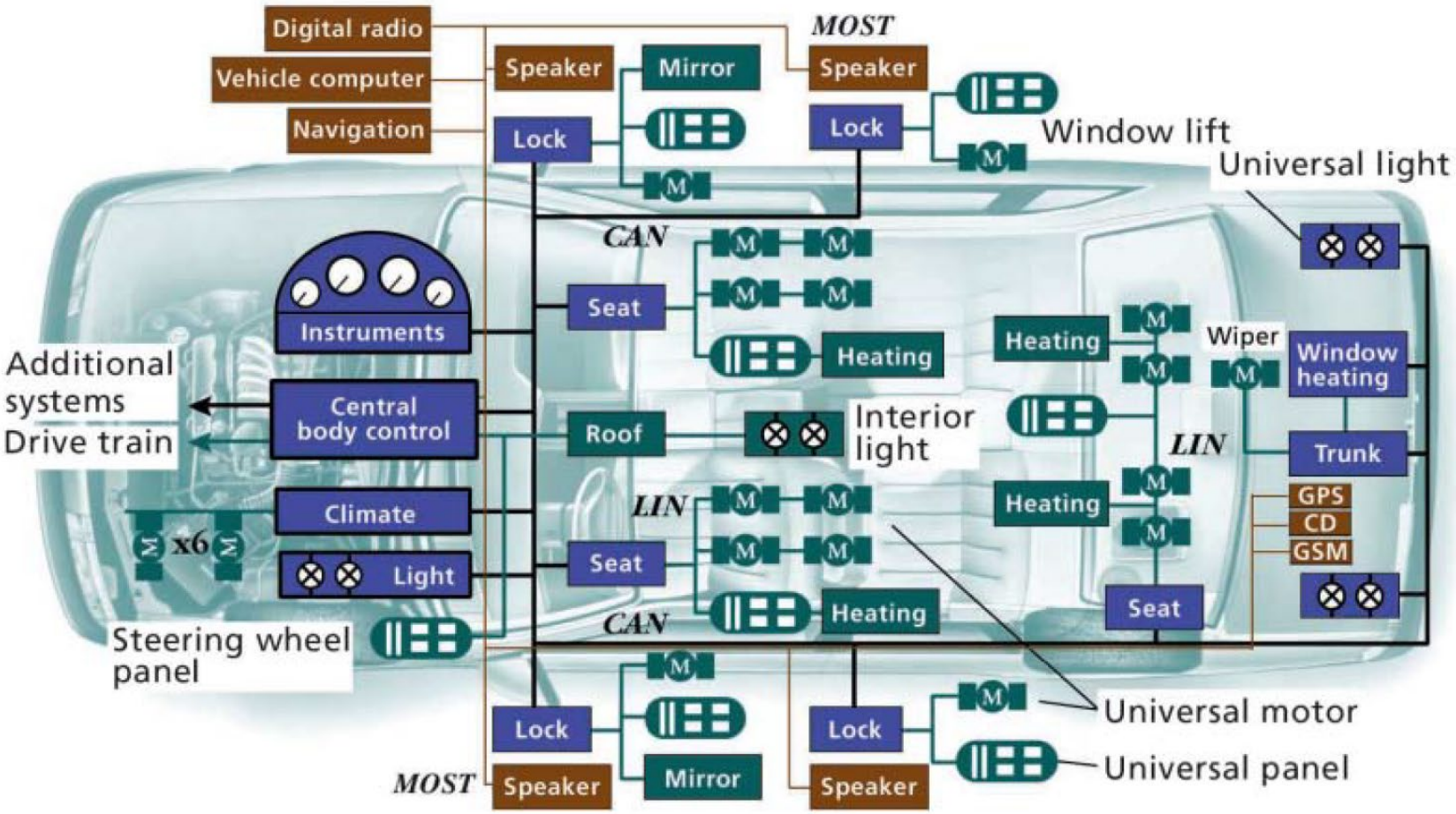
Un po' di background sul tema: SAE J3016 levels of automation [2]

SOCIETY OF AUTOMOTIVE ENGINEERS (SAE) AUTOMATION LEVELS

Full Automation

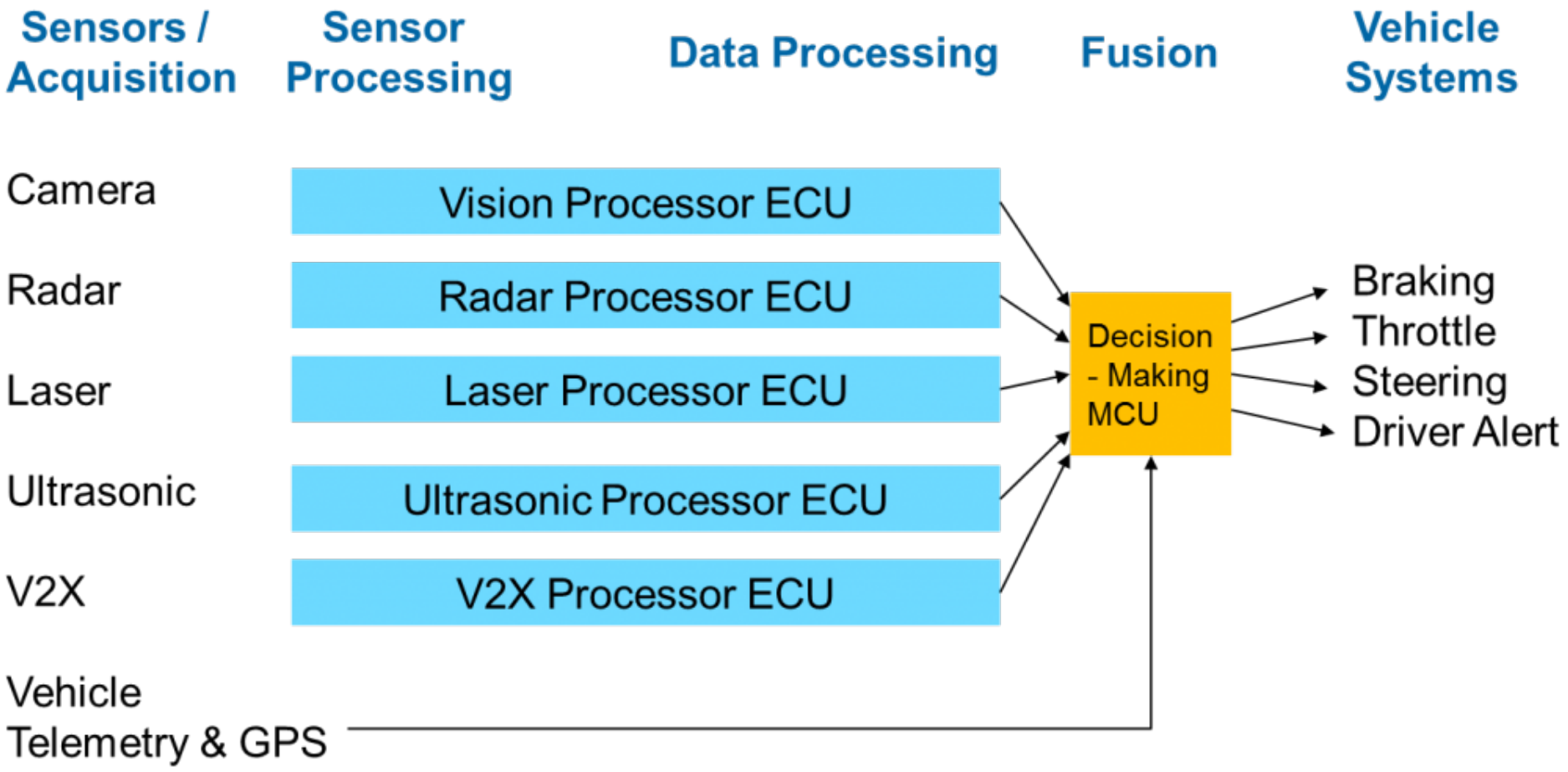
0	1	2	3	4	5
No Automation	**Driver Assistance**	**Partial Automation**	**Conditional Automation**	**High Automation**	**Full Automation**
Zero autonomy; the driver performs all driving tasks.	Vehicle is controlled by the driver, but some driving assist features may be included in the vehicle design.	Vehicle has combined automated functions, like acceleration and steering, but the driver must remain engaged with the driving task and monitor the environment at all times.	Driver is a necessity, but is not required to monitor the environment. The driver must be ready to take control of the vehicle at all times with notice.	The vehicle is capable of performing all driving functions under certain conditions. The driver may have the option to control the vehicle.	The vehicle is capable of performing all driving functions under all conditions. The driver may have the option to control the vehicle.

Un po' di background sul tema: alcuni componenti che possiamo trovare in un'auto[1]

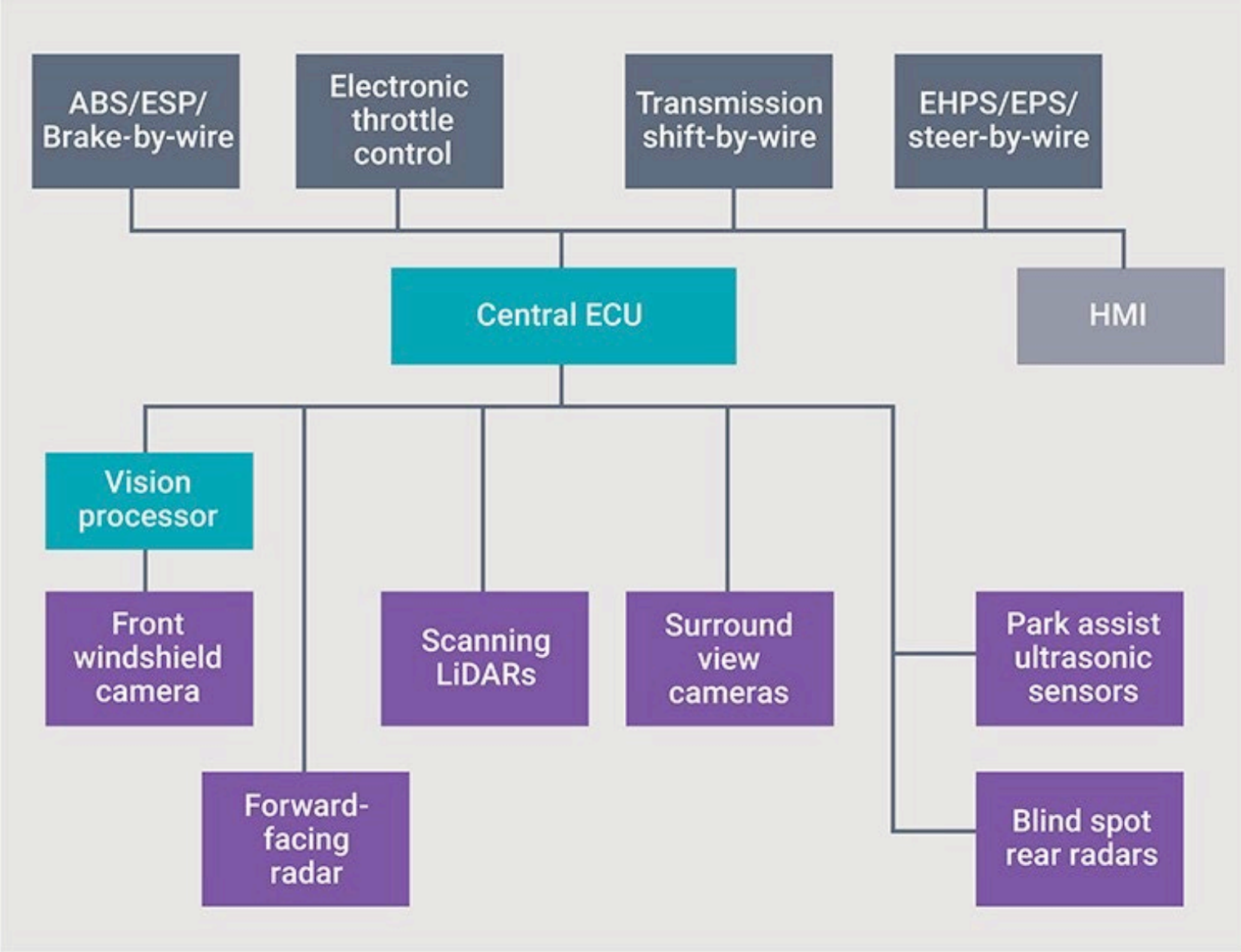


CAN Controller area network
GPS Global Positioning System
GSM Global System for Mobile Communications
LIN Local interconnect network
MOST Media-oriented systems transport

Un po' di background sul tema: elementi base per ADAS [3] (autonomous driving assistance systems)



Un po' più nello specifico: la vista architetturale che considereremo [4]



Alcune nozioni e interpretazioni dell'architettura

Central ECU: Unità di Controllo → ce ne possono essere decine, ma in questa vista architetturale la consideriamo come «elemento centrale» di elaborazione

Vision processor: acceleratore dedicato al riconoscimento di immagini

HMI: Human-Machine Interface

Blocchi grigio scuro: attuatori

- ABS/ESP brake by wire
- Electronic throttle control
- Transmission shift-by-wire
- EHPS/EPS steer-by-wire: electrohydraulic power steering pump/electric-power steering

Blocchi viola: sistemi di acquisizione dati (sensori)

Riferimenti (provenienza delle immagini)

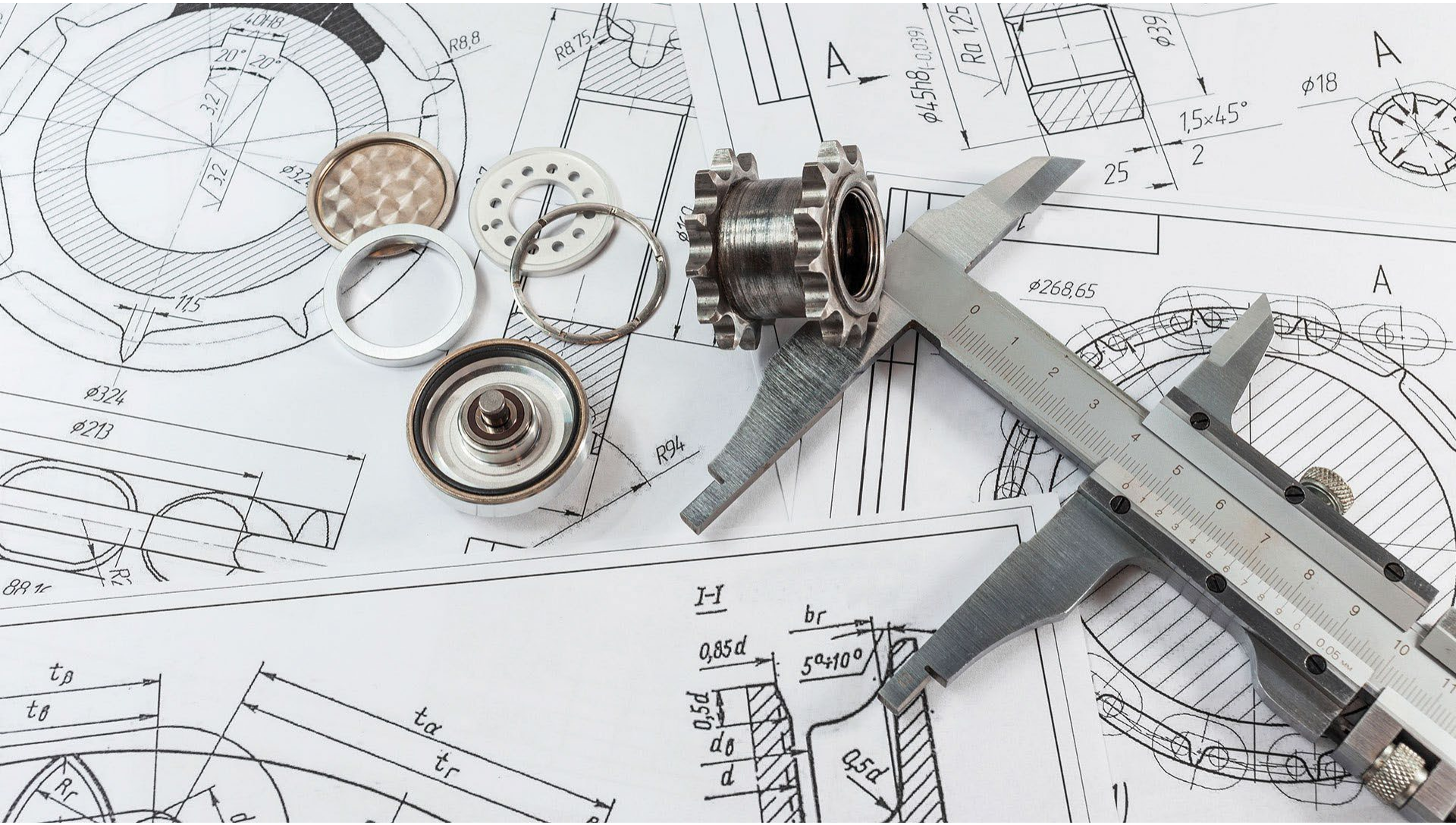
[1] A. M. Vegni, M. Biagi and R. Cusani, "Smart Vehicles, Technologies and Main Applications in Vehicular Ad hoc Networks", InTechOpen, 2013.

[2] Automated Vehicles for Safety, https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[3] Complex Trends and Challenges in Designing ADAS Systems https://www.embedded-vision.com/industry-analysis/technical-articles/2014/09/30/complex-trends-and-challenges-designing-adas-systems

[4] Managing the evolving architecture of integrated ADAS controllers https://www.techdesignforums.com/practice/technique/managing-the-evolving-architecture-of-integrated-adas-controllers/

Presentazione del progetto



Obiettivo

Obiettivo del progetto è costruire una architettura, ovviamente **estremamente stilizzata e rivisitata (e difforme)**, per sistemi ADAS, descrivendo possibili interazioni e alcuni comportamenti tra componenti in scenari specifici.

Il seguito delle slides entrano nei dettagli della semplificazione richiesta.

Richieste implementative

Se non diversamente specificato (tipicamente, nel testo, con il termine «facoltativo»), le seguenti richieste implementative sono da intendersi prescrittive, equivalenti alla parola MUST secondo l'RFC 2119.

MUST This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

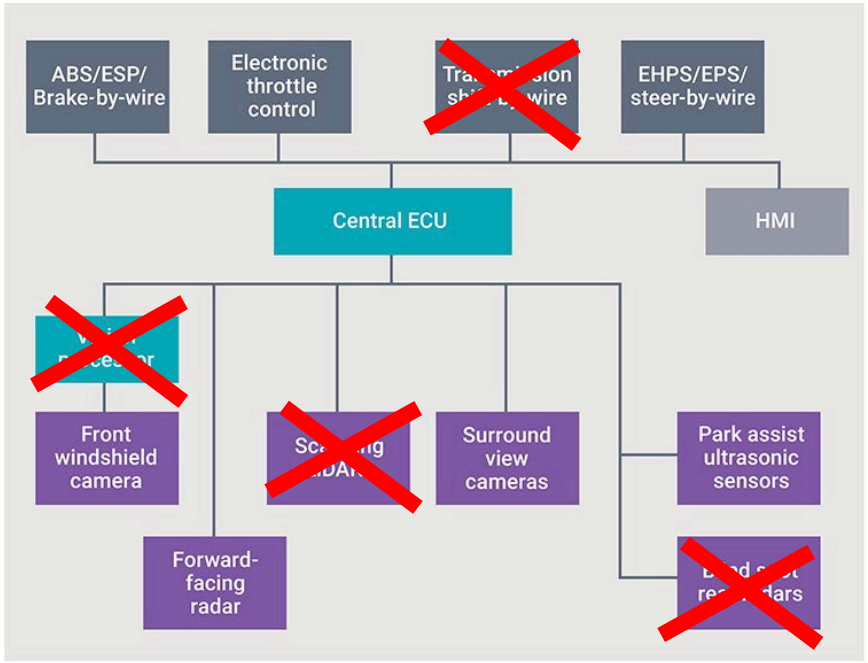
Il sistema richiesto

Per semplicità, alcuni elementi sono esclusi dallo schema architetturale.

Ciascun componente dello schema architetturale è rappresentato da **almeno un processo**.

- No **pthread**

I processi hanno un comportamento, e interagiscono, secondo le specifiche descritte in queste slides.



Source: Ian Riches, Strategy Analytics ©Strategy Analytics (2014)

Funzionalità dei componenti e loro rappresentazione: Interfaccia HMI

Human-Machine Interface: mostra l'output a video, secondo quanto comunicato dalla *Central ECU*, e invia input dell'utente alla Central ECU

- È realizzato con due terminali. Un terminale è sfruttato per mostrare in output i valori ricevuti dalla Central ECU, l'altro terminale è invece dedicato alla raccolta e trasmissione degli input alla Central ECU.
- Come output, il processo stampa a schermo tutte le azioni comandate dalla Central ECU.
- Gli input possibili, da digitare su terminale, sono:
  - `INIZIO`: avvia il veicolo verso la sua destinazione
  - `PARCHEGGIO`: si attiva la procedura di parcheggio, che conclude il percorso.
  - `ARRESTO`: si attiva la procedura di arresto

Funzionalità dei componenti e loro rappresentazione: ATTUATORI

steer-by-wire: riceve il comando di sterzata DESTRA o SINISTRA dalla Central ECU. Non fallisce mai.

E' un processo che riceve il comando DESTRA o SINISTRA dalla ECU, ed esegue la sterzata in 4 secondi (ovvero, l'azione di "sterzare" dura 4 secondi).

Non necessita di inviare ACK.

Ogni secondo, stampa nel file di log `steer.log`: "no action", "sto girando a destra", "sto girando a sinistra", sulla base dell'azione in corso.

Nota: NON arrivano 2 comandi di sterzata a distanza inferiore a 4 secondi.

Funzionalità dei componenti e loro rappresentazione: ATTUATORI

throttle control: riceve comando di accelerazione dalla Central ECU, nel formato "`INCREMENTO 5`", dove 5 indica l'aumento di velocità. Alla ricezione di "`INCREMENTO 5`", la velocità aumenta di 5 km/h.

- La velocità è aumentata di 5 KM/H per ogni messaggio. Conseguentemente, scrive "`INCREMENTO 5`" nel file di log `throttle.log`
- Quando riceve un messaggio, stampa nel file di log `throttle.log`: "TIMESTAMP: INCREMENTO 5"
- Non necessita di inviare ACK.
- **Facoltativo**. Ad ogni accelerazione, c'è una probabilità di 10^{-5} che l'acceleratore si rompa. Se questa si verifica, *throttle control* invia un **SEGNALE** alla Central ECU per evidenziare tale evento, che avvia la procedura di ARRESTO.

Funzionalità dei componenti e loro rappresentazione: ATTUATORI

brake-by-wire: riceve dalla Central ECU il comando di decelerazione o un segnale di pericolo. Non fallisce mai.

- E' un processo che riceve una indicazione di riduzione della velocità dalla Central ECU, ed esegue.
- Non necessita di inviare ACK.
- Riceve un messaggio di tipo FRENO 5, dove 5 indica la velocità da diminuire. Similmente all'accelerazione, questo equivale a scrivere "FRENO 5" nel file di log `brake.log`
- Se riceve il **SEGNALE** di ARRESTO dalla Central ECU, arresta l'auto (azione istantanea). Questo equivale a scrivere "arresto auto" nel file di log `brake.log`

Funzionalità dei componenti e loro rappresentazione: SENSORI

Front windshield camera: è un processo che legge dati e li invia alla Central ECU

- I dati sono letti (e inviati) 1 volta ogni al secondo
- Ciascuna lettura corrisponde a leggere una riga dal file `frontCamera.data`
- I dati inviati sono loggati in *camera.log*

Facoltativo. *Forward facing radar*: è un processo che legge i dati e li invia alla Central ECU

- I dati sono letti iterativamente 1 volta al secondo
- Ciascuna lettura corrisponde a *provare a* leggere 8 bytes da /dev/urandom
- Se riesce a leggere 8 byte, questi sono trasmessi alla Central ECU. I dati inviati sono loggati nel file *radar.log*.
- Altrimenti (se legge meno di 8 byte), riprova a leggere all'iterazione successiva

Funzionalità dei componenti e loro rappresentazione: SENSORI

Park assist: è un processo che è agisce **solo su richiesta** della Central ECU. Quando riceve un comando di attivazione dalla Central ECU:

- Per 30 secondi, 1 volta al secondo, legge 8 byte da /dev/urandom, li invia alla Central ECU, scrive su file di log.
- **Facoltativo**: quando si attiva l'interazione con Park Assist, la Central ECU sospende o uccide tutti gli altri processi sensore e attuatori tranne Surround view cameras.
- **Facoltativo**: Park Assist è creato dalla Central ECU al bisogno, e non all'avvio del sistema.
- **Facoltativo**. Se il componente surround view cameras è implementato, park assist trasmette a Central ECU anche i byte ricevuti da surround view cameras.

Facoltativo. *Surround view cameras*: è un processo che agisce solo quando *Park assist* è attivo. Legge i dati e li invia a Park Assist:

- Finchè Park Assist è Attivo, 1 volta al secondo, legge 8 byte da /dev/urandom, li invia a park assist, e logga i dati inviati.
- **Suggerimento**: Surround view cameras può essere un figlio di Park Assist, creato e terminato al bisogno

Funzionalità dei componenti e loro rappresentazione: CONTROLLO - 1

Central ECU: All'avvio del sistema, la Central ECU imposta la velocità attuale a 0, e ignora qualsiasi informazione ricevuta dai sensori, finchè riceve dall'HMI la stringa "INIZIO". Questo indica l'avvio del tragitto.

A questo punto, la Central ECU legge continuamente dati dai sensori. Central ECU utilizza i dati ricevuti nel modo seguente:

- Per ogni dato ricevuto da *Front windshield camera*, opera le seguenti azioni:
  - Se è un un numero, il numero indica la velocità desiderata del veicolo. Questo è confrontata con la velocità attuale, e la Central ECU comunica con i componenti «throttle control», o «brake-by-wire» per rispettivamente accelerare o decelerare. **Opera con ciclo 1 secondo**, ovvero ogni secondo manda il comando necessario di INCREMENTO 5 o FRENO 5, fino a raggiungere la velocità desiderata.
  - La sterzata verso destra o sinistra, se vi sono i termini rispettivamente DESTRA o SINISTRA. Il comando è inviato al componente «steer-by-wire».
  - Il termine PERICOLO indica invece l'invio di un segnale di ARRESTO al componente «brake-by-wire».
  - Il termine PARCHEGGIO avvia la procedura di parcheggio (slide successiva).

Funzionalità dei componenti e loro rappresentazione: CONTROLLO - 2

In seguito all'esecuzione di un segnale di pericolo, la velocità è zero. Per re-iniziare, è necessario che l'utente digiti nuovamente INIZIO.

Se il comando ricevuto dalla HMI è invece "ARRESTO", la Central ECU invia un segnale di arresto al componente brake-by-wire e imposta la velocità a 0.

Nota: azzerata la velocità, il sistema però riparte subito, ovvero i vari sensori continuano a mandare dati che saranno quindi elaborati.

Funzionalità dei componenti e loro rappresentazione: CONTROLLO - 3

Se il comando ricevuto dalla HMI è invece "PARCHEGGIO", la Central ECU avvia la procedura di parcheggio. Central ECU:

- ignora i messaggi da front windshield camera, forward facing camera e HMI (ovvero, non si cura più dei dati ricevuti dai canali di input a meno di quelli dedicati al parcheggio).
- invia, con ciclo di 1 secondo, richieste di FRENO 5 a brake-by-wire fino al raggiungimento di velocità 0.
- quando la velocità raggiunge 0, attiva Park assist ultrasonic sensors e Surround view cameras.

Se la Central ECU non riceve da nessuno dei due, per 30 secondi, uno dei valori: i) 0x172A, ii) 0xD693, iii) 0x0000, iv) 0xBDD8, v) 0xFAEE, vi) 0x4300, l'auto è parcheggiata e la missione termina.

Altrimenti, la Central ECU ri-avvia la procedura di PARCHEGGIO.

Nota: Durante la fase di PARCHEGGIO, si ignorano eventuali dati provenienti da sensori diversi da quelli specificati

Facoltativo. Il comando di PARCHEGGIO potrebbe arrivare mentre i vari attuatori stanno eseguendo ulteriori comandi (accelerare o sterzare). I vari attuatori interrompono le loro azioni, per avviare le procedure di parcheggio.

Funzionalità dei componenti e loro rappresentazione: CONTROLLO - 3

Tutti i comandi inviati dalla Central ECU sono inseriti in un file di log `ECU.log` e stampati a video tramite la HMI.

Facoltativo. Se la Central ECU riceve il segnale di fallimento accelerazione da throttle control, imposta la velocità a 0 e invia all'output della HMI un messaggio opportuno (terminazione esecuzione).

Modalità di avvio

Si richiedono due modalità di esecuzione:

NORMALE: il progetto è lanciato in esecuzione come specificato sopra, leggendo dati da /dev/random e /dev/urandom.

ARTIFICIALE: il progetto è lanciato in esecuzione leggendo

- Invece che da /dev/urandom, dal file urandomARTIFICIALE.binary
- Invece che da /dev/random, dal file randomARTIFICIALE.binary

Le due modalità sono selezionate dal parametro NORMALE o ARTIFICIALE passato come argomento al main (ovvero, argv[1]).

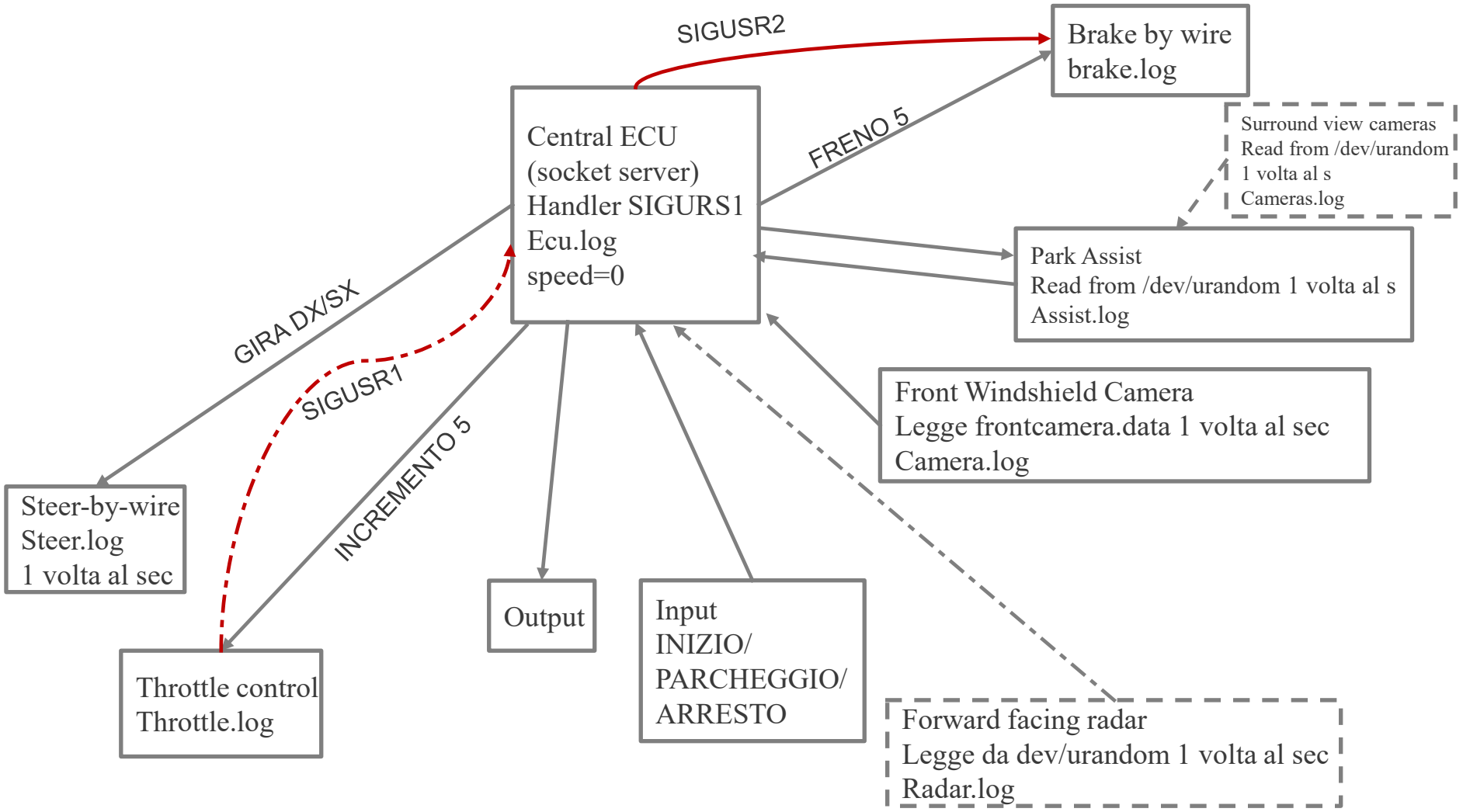
Modalità di terminazione

L'esecuzione termina quando l'utente digita l'input PARCHEGGIO, e la procedura di parcheggio conclude con successo, oppure quando si riceve un segnale di fallimento da throttle control.

Schema e riassunto delle osservazioni effettuate in aula

NON VINCOLANTE

PER LA CORRETTA REALIZZAZIONE DEL PROGETTO



Elementi tratteggiati: facoltativi
In rosso: segnali
La freccia indica la direzione dei dati. Esempio: Input effettua una write (o send), Central ECU effettua una read (o receive).

Osservazioni in aula - 1

L'implementazione più facile è ***probabilmente*** considerare la Central ECU un server socket iterativo. I vari sensori, attuatori ed HMI sono client.

- Con l'eccezione di *Output*, attuatori e sensori sono figli di Central ECU. Output può essere un eseguibile lanciato da una seconda shell.
- In breve, Central ECU effettua prima le read dai sensori, e quindi tutte le write verso gli attuatori
- Da considerare che il messaggio da INPUT può arrivare in qualsiasi momento. Central ECU dovrà quindi prevedere una «read» corrispondente. Vari modi:
  1. Usare un segnale per gestire la read tramite un handler
  2. Effettuare una read non bloccante su Central ECU
  3. Input può, con periodo 1 secondo, mandare un messaggio vuoto oppure il messaggio dato dall'utente

Osservazioni in aula - 2

Alcuni tra gli approcci alternativi discussi in classe:

- Organizzare la central ECU in 2 processi: un processo dedicato alle letture, ed uno alle scritture. Lo scambio di informazioni tra i 2 processi può essere effettuata tramite canale unidirezionale, file condiviso, socket, …
- Creare un server Central ECU concorrente, definendo "coppie" di processi per la gestione delle varie comunicazioni
- Avere tutte socket non-bloccanti, per risolvere possibili problemi di sincronia (read in attesa di dati non disponibili)

Osservazioni in aula - 3

Avvio del progetto

- L'esecuzione è su due terminali: uno per inviare input, uno per mostrare l'output
- Si può implementare in vari modi, ad esempio (soluzione più facile) avere il progetto diviso in 2 eseguibili da lanciare dai 2 terminali:
  - TERMINALE 1: ./input ARTIFICIALE
  - TERMINALE 2: ./output ARTIFICIALE

Osservazioni in aula - 4

/dev/urandom è una sorgente di byte, **NON** sono caratteri ASCII

- Ad esempio, se si stampa a video quanto letto da /dev/urandom come se fossero caratteri (%c), non si visualizza niente di "sensato"

→ **%x ; %X print value in hexadecimal format**

→ Stesso ragionamento per la scrittura su file di log

→Provare ad esempio a leggere da /dev/urandom con hexdump-c oppure hexdump -x

La frase

non riceve da nessuno dei due, per 30 secondi, uno dei valori: i) 0x172A, ii) 0xD693, …

vuol dire che si deve cercare, all'interno dei dati ricevuti, i valori identificati. Ad esempio:

0x12B3A172A9CC5D98