

# **PROGETTO SISTEMI OPERATIVI A.A 2018/19**

Piazzesi Niccolò Matricola:6335623  
`niccolo.piazzesi@stud.unifi.it`

Bernabei Pietro Matricola:6291312  
`pietro.bernabei@stud.unifi.it`

Hong Massimo Matricola:6365472  
`massimo.hong@stud.unifi.it`

September 11, 2019

# Contents

|   |          |
|---|----------|
| <b>COMPILAZIONE ED ESECUZIONE . . . . .</b> | <b>1</b> |
| DESCRIZIONE AD ALTO LIVELLO . . . . .       | 2        |
| CARATTERISTICHE PRINCIPALI . . . . .        | 3        |
| ESECUZIONE . . . . .                        | 4        |

# COMPILAZIONE ED ESECUZIONE

Il progetto è composto dai seguenti file:

- CARTELLA INCLUDE/
  - ecu.h
  - azioni.h
  - creazione.h
  - attuatori.h
  - log.h
  - SocketConnection.h
  - sensori.h
- CARTELLA SRC/
  - input.c
  - ecu.c
  - azioni.c
  - creazione.c
  - attuatori.c
  - log.c
  - SocketConnection.c
  - sensori.c
  - output.c
- CARTELLA INPUT/
  - frontCamera.data
  - randomARTIFICIALE.binary
  - urandomARTIFICIALE.binary
- avvio.sh
- Makefile

Assicurarsi che i file siano tutti presenti e organizzati nella gerarchia corretta .

Per poter compilare e successivamente eseguire il progetto seguire questi passaggi:

1. Estrarre il progetto dalla zip allegata in una cartella qualsiasi
2. Aprire un terminale all'interno della cartella in cui è stato estratto (oppure spostarsi all'interno della cartella da un terminale già aperto)
3. eseguire il comando `make all` (oppure soltanto `make`) per avviare la compilazione
4. Al termine della compilazione, il progetto può essere avviato attraverso lo script `avvio.sh` presente nella cartella. Per eseguirlo scrivere **`bash avvio.sh NORMALE`** (oppure **`ARTIFICIALE`**) nel terminale e premere Invio

## DESCRIZIONE AD ALTO LIVELLO

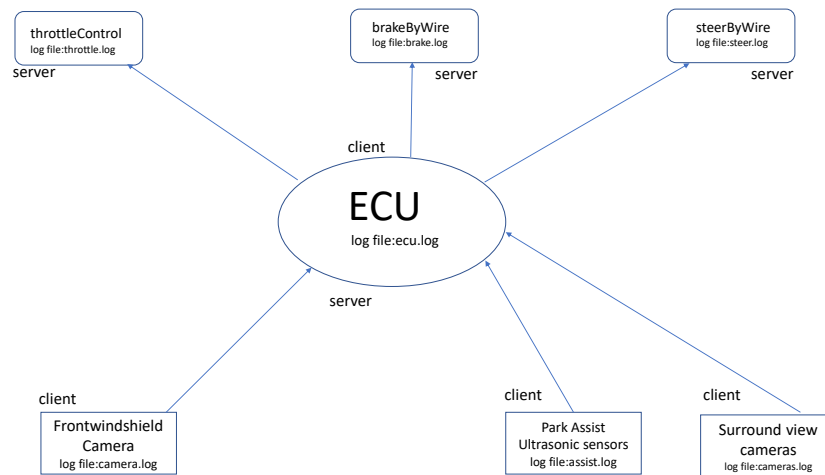


Figure 1: schema dell'implementazione

La struttura fondamentale utilizzata è l'architettura client-server. Durante l'esecuzione il sensore Front Windshield Camera raccoglie i dati, si connette

alla ECU,e glieli invia. L'ECU elabora i dati raccolti,si connette a uno degli attuatori,e invia il comando appropriato. Durante il parcheggio,i due sensori Park Assist Ultrasonic Sensors e Surround View Cameras si comportano come Front Windshield Camera. La differenza principale è che,in questo caso,la ECU elabora direttamente i dati senza inviarli agli attuatori.La ECU si comporta quindi da server per i sensori e da client per gli attuatori.

## **CARATTERISTICHE PRINCIPALI**

### **SISTEMI OBIETTIVO**

Per sviluppare il programma sono state utilizzate le seguenti distribuzioni Linux:

- Ubuntu 18.04 LTS
- Mx Linux 18.03 (Debian 9 stable)

Tutte le macchine utilizzate hanno un'architettura x86 a 64 bit

### **GESTIONE PROCESSI**

Il programma è suddiviso in 5 gruppi di processi:

- HMI-Input
- ECU
- Sensori
- Attuatori
- HMI-output

All'avvio vengono creati soltanto i due processi della HMI. Quando il processo di input riceve il comando INIZIO crea la ECU. La ECU è responsabile di creare tutti gli altri processi utili durante l'esecuzione. In particolare crea Front Windshield Camera e i tre attuatori(throttleControl,brakeByWire,steerByWire).Ogni attuatori è

composto da due processi:il padre aspetta le richieste della ECU mentre il figlio logga NO ACTION ogni secondo durante l'attesa.Quando HMI-Input riceve il comando PARCHEGGIO lo invia alla ECU.Quest'ultima,dopo aver fatto fermare la macchina,uccide tutti i processi figli e crea i due sensori di parcheggio(Front Windshield Camera e Surround View Cameras). Alla fine della procedura di parcheggio la ECU uccide i due sensori e termina,portando il programma al completamento.

## **COMUNICAZIONE TRA PROCESSI**

La maggior parte della comunicazione avviene attraverso lo scambio di messaggi tra client e server. Vengono però utilizzati anche i segnali,per gestire determinate situazioni. Ad esempio quando Front Windshield legge PERICOLO,invia un segnale alla ECU che,sempre attraverso i segnali,comunica l'arresto dell'auto e ferma Front Windshield fino a quando non arriva il comando di ripartire.

## **ESECUZIONE**

Per mostrare un'esecuzione tipo abbiamo usato un sottinsieme di dati tale da fare eseguire tutti i comandi possibili al programma.

**50**  
**70**  
**50**  
**DESTRA**  
**SINISTRA**  
**PERICOLO**  
**50**  
**50**

Figure 2: I dati letti

```
MACCHINA ACCESA  
INCREMENTO 50  
INCREMENTO 20  
FRENO 20  
DESTRA  
SINISTRA  
PERICOLO,MACCHINA FERMATA  
INCREMENTO 50  
LA MACCHINA PROCEDE ALLA VELOCITÀ DI 50 km/h  
PARCHEGGIO
```

Figure 3: I comandi eseguiti

COMANDO: INCREMENTO 50

```
AUMENTO 5  
AUMENTO 5  
AUMENTO 5  
AUMENTO 5  
AUMENTO 5  
AUMENTO 5  
AUMENTO 5  
AUMENTO 5  
AUMENTO 5  
AUMENTO 5  
AUMENTO 5  
AUMENTO 5  
AUMENTO 5
```

Figure 4: throttle.log

COMANDO: INCREMENTO 20

```
AUMENTO 5  
AUMENTO 5  
AUMENTO 5  
AUMENTO 5  
NO ACTION  
NO ACTION  
NO ACTION  
NO ACTION  
NO ACTION  
NO ACTION
```

Figure 5: throttle.log

COMANDO: FRENO 20

```
DECREMENTO 5  
DECREMENTO 5  
DECREMENTO 5  
DECREMENTO 5  
NO ACTION  
NO ACTION  
NO ACTION  
NO ACTION  
NO ACTION  
NO ACTION
```

Figure 6: brake.log

COMANDO: DESTRA

```
STO GIRANDO A DESTRA  
STO GIRANDO A DESTRA  
STO GIRANDO A DESTRA  
STO GIRANDO A DESTRA  
NO ACTION  
NO ACTION  
NO ACTION  
NO ACTION  
NO ACTION  
NO ACTION
```

Figure 7: steer.log

COMANDO: SINISTRA

```
STO GIRANDO A DESTRA  
STO GIRANDO A DESTRA  
STO GIRANDO A DESTRA  
STO GIRANDO A DESTRA  
NO ACTION  
NO ACTION  
NO ACTION  
NO ACTION  
NO ACTION  
NO ACTION
```

Figure 8: steer.log

COMANDO:PERICOLO

```
NO ACTION  
ARRESTO AUTO  
NO ACTION
```

Figure 9: brake.log



```
macchina fermata,scrivere INIZIO per rimettere in moto  
INIZIO  
Macchina di nuovo in moto,premere INVIO due volte per tornare all'interfaccia di comando  
  
digitare PARCHEGGIO per fermare la macchina I
```

Figure 10: L'interfaccia di input

All'arrivo del comando di parcheggio si avviano i sensori appositi che loggano i dati letti nei propri file

**95 4 51 59**

Figure 11: Una riga di assist.log