

Relazione progetto SO

compilazione

per la compilazione del progetto seguire le seguenti azioni:

1. Decomprimere il progetto.
2. Nella cartella decompressa di nome ADAS saranno presenti i file sorgenti, il pdf della relazione, i file data e un file log per registrare i messaggi di debug. Aprire un terminale e digitare “make -f ADAS.make all” che compila i file c, generando i file oggetto e gli eseguibili.
3. Digitare quindi “make -f ADAS.make install”. Questo comando dovrebbe eliminare i file oggetto, spostare i file eseguibili nella cartella bin appena creata, spostare i file data nella cartella data e il file di log nella cartella log.
4. A questo punto il progetto è stato installato in maniera corretta ed'è pronto per essere eseguito. Per disinstallare il programma basta usare la funzionalità clean del makefile.

Esecuzione

per l'esecuzione del software seguire i seguenti punti:

1. Aprire un terminale. È consigliato seguire su un secondo terminale la crescita del file di log per i messaggi di debug (“debuglog.txt”), usando il comando tail -f. Infatti, nel caso in cui l'ECU rimane inattivo e non stampa nessun messaggio sul terminale di output, viene mostrato il motivo nel file di log.
2. Posizionarsi all'interno della cartella bin con il comando “cd bin/”.
3. Eseguire HMIinput con un argomento a piacere: i possibili argomenti sono “NORMALE” o “ARTIFICIALE”. Se viene inserito un argomento illegale allora l'esecuzione viene impostato in modalità normale e viene stampata una stringa che notifica del fatto. Una volta messo in esecuzione HMIinput apre un nuovo terminale dove mette ad eseguire HMIoutput il quale stampa esclusivamente i comandi che la Central ECU invia ai vari altri moduli che compongono il sistema.
4. A questo punto il programma si sarà avviato in modalità specificata e avrà effettuato l'inizializzazione dei processi. HMIinput chiederà quindi l'inserimento della stringa “INIZIO” per poter iniziare la navigazione e sbloccare l'ECU, che inizia quindi a gestire le richieste da front windshield camera e a mandare output sul secondo terminale (quello che sta seguendo output_terminal.txt).
5. Se viene inserita una stringa diversa da “INIZIO” non si avrà l'avvio dell'ECU e verrà stampato di nuovo la richiesta.
6. Una volta inserito il comando di inizio a HMIinput, gli unici comandi che il terminale di input continuerà ad accettare saranno “ARRESTO” e “PARCHEGGIO”.

Informazioni sul SO

OS Name: Ubuntu 22.04.3 LTS

OS Type: 64-bit

GNOME Version: 42.9

Elementi facoltativi realizzati

#	Elemento facoltativo	Realizzato (SI/NO)	Descrizione dell'implementazione con indicazione del metodo/i
1	Ad ogni accelerazione, c'è una possibilità di 10^{-5} che l'acceleratore fallisca. In tal caso, il componente throttle control invia un <u>segnale</u> alla Central ECU per evidenziare tale evento, e la Central ECU avvia la procedura di ARRESTO	SI	throttle_control prima di effettuare un'accelerazione, dopo aver ricevuto il segnale dalla Central ECU, chiama il metodo generate() che restituisce un numero in base all'ora attuale. Si calcola poi il modulo 100000 di questo numero e lo si confronta con 0: se è diverso da 0 allora l'accelerazione viene effettuata in maniera normale altrimenti avviene un fallimento. In caso di fallimento viene mandato un segnale SIGQUIT al padre del processo, cioè all'ECU che la gestisce in maniera appropriata e avvia la procedura di ARRESTO
2	Componente "forward facing radar"	NO	-
3	Quando si attiva l'interazione con park assist, la Central ECU sospende (o rimuove) tutti i sensori e attuatori, tranne park assist e surround view cameras.	SI	Se ECUoutput riceve questo messaggio da front windshield camera allora il processo figlio che era stato creato per gestire la richiesta chiama la funzione di parking, il quale termina i processi dedicati alla gestione dei sensori e degli attuatori dopo aver interagito con brake_by_wire per portare la velocità a zero.
4	Il componente Park assist non è generato all'avvio del Sistema, ma creato dalla Central ECU al bisogno	SI	Sia nel caso in cui ECUoutput riceve il comando di parcheggio dal terminale sia nel caso lo riceva da front windshield camera viene sempre chiamata la funzione parking() che al momento opportuno esegue park_assist
5	Se il componente surround view cameras è implementato, park assist trasmette a Central ECU anche i byte ricevuti da surround view cameras	SI	Park_assist riceve i dati da surround_view_cameras e li invia alla Central ECU in una modalità simile a surround_view_cameras.
6	Componente "surround view cameras"	SI	Al suo lancio park_assist crea anche il processo surround_view_cameras, che a seconda della modalità va a leggere 8 byte o da /dev/uRandom o da uRandomARTIFICIALE.binary. Prima di inviare questi 8 byte di dati vengono inviati 2 byte per avvertire park_assist se in seguito ho 8 byte di dati significativi o no.
7	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	SI	Quando ECUoutput riceve da ECUInput il segnale SIGUSR2 avvia in maniera automatica la procedura di parcheggio e quindi chiude tutti i processi figli creati per la gestione delle richieste inviando un segnale di SIGABRT al gruppo di processi a cui apparteneva.

	avviare le procedure di parcheggio		
8	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 di totale terminazione dell’esecuzione	SI	Una volta ricevuto il segnale di fallimento acceleratore sottoforma di una SIGQUIT, ECUoutput invia a HMIoutput una stringa per evidenziare il fallimento dell’acceleratore, imposta la velocità a 0 e manda il segnale di terminazione a tutti i processi del proprio gruppo

Progettazione e implementazione

il progetto consiste in 9 moduli: HMIinput, HMIoutput, ECUinput, ECUoutput, park_assist, front_windshield_camera, steer_by_wire, brake_by_wire e throttle_control. Il programma viene avviato a partire da HMIinput, che inizializza prima ECUoutput e poi HMIoutput. ECUoutput fa eseguire il resto dei moduli e tra questi park_assist è avviato solo al bisogno. HMIinput ha il compito di raccogliere i comandi dal prompt di comandi e inviarli al resto del programma, oltre a quello di inizializzare il programma. HMIoutput invece, esiste con il solo compito di stampare i messaggi dell’ECU sul secondo terminale. HMIoutput termina quando l’ECUoutput chiude il pipe attraverso cui comunicano, mentre HMIinput termina solo se gli viene mandato il segnale di terminazione software da qualche altro modulo del programma.

ECUoutput è il primo processo ad essere creato dopo HMIinput e costituisce la parte di logica più importante per il programma. In ECUoutput vengono creati tutti i pipe con nome che verranno poi usati nella comunicazione con i vari moduli figli e viene anche implementato un server concorrente: con un ciclo while(true) il modulo accetta richieste di connessione da parte di front_windshield_camera e li gestisce in maniera concorrente quando possibile (solo quando si cerca di effettuare un giro a destra o a sinistra dopo aver avuto una richiesta sulla velocità). Per poter gestire meglio i segnali di parcheggio e arresto, ECUoutput, prima di entrare nel ciclo while, cambia il proprio gid in maniera che i processi figli creati dopo possano essere poste in un gruppo diverso rispetto agli altri processi e quindi rendere possibile la manovra di terminarli tutti insieme senza influenzare i processi degli attuatori o sensori. Per implementare la comunicazione con front_windshield_camera in maniera che, ad esempio l’accelerazione e la decelerazione dell’auto non si sovrappongano, il processo figlio, prima di terminare, invia attraverso il socket un messaggio di “OK” che sblocca front_windshield_camera che era rimasto in attesa con un read.

In ECUoutput sono installati anche una serie di handler per i vari segnali che può ricevere durante il suo funzionamento. Tra queste è interessante parkHandler, che gestisce la procedura di parcheggio in caso di input da terminale: condivide la funzione parking con gli eventuali processi generati dal server in caso di richiesta da front_windshield_camera di parcheggio. Essenzialmente quando l’handler entra in azione la prima cosa che fa è quella di terminare tutti i processi in esecuzione che gestiscono le connessioni con front_windshield_camera usando il trucco che abbiamo visto prima, e quindi procede a chiamare la funzione parking più volte finché l’auto non viene parcheggiato con successo. Parking è una funzione abbastanza complessa che implementa la procedura del parcheggio: all’inizio effettua una serie di freni interagendo con brake_by_wire e una volta che si raggiunge una velocità uguale a zero rimuove i processi associati agli attuatori e sensori per poi avviare la park_assist e ricevere da essa byte che analizza per stabilire il successo o no dell’operazione. Per stabilire se surround_view_cameras ha mandato i dati o no, ECUoutput fa prima una lettura di 2 byte e poi procede con la lettura dei 8 successivi. Quindi resituisce alla fine un valore booleano al chiamante.

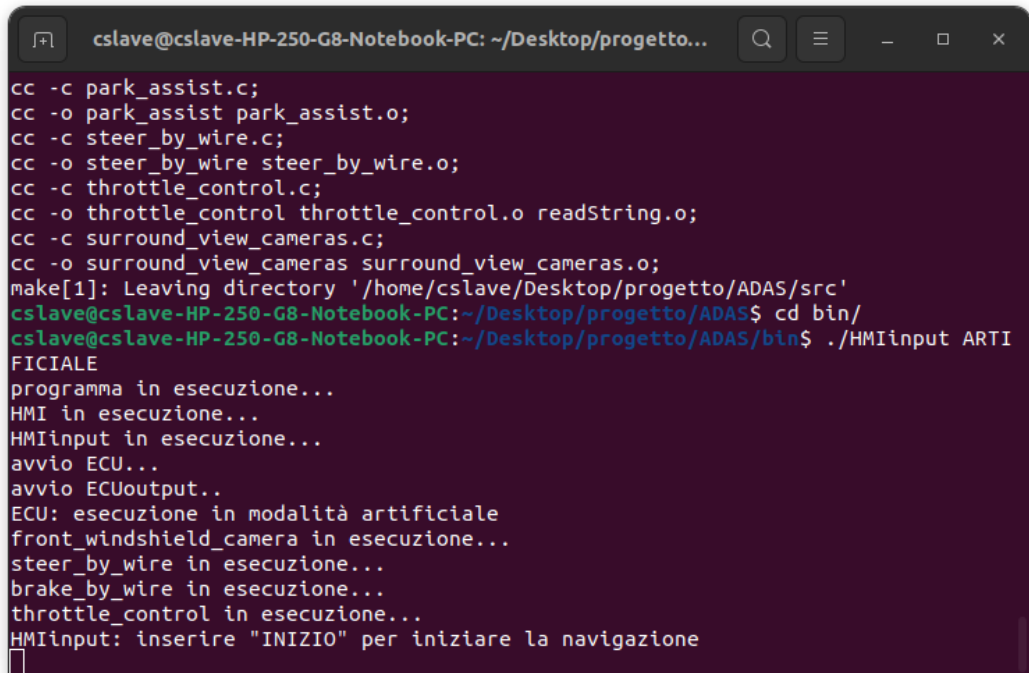
ECUinput riceve i messaggi da HMIinput e li gestisce mandando un segnale opportuno a ECUoutput. ECUinput termina la sua esecuzione solo se riceve un segnale di terminazione, stesso per ECUoutput il quale però ha molti più tipi di segnali che hanno questo effetto. Park-assist è il modulo che si occupa della procedura di parcheggio. In particolare, una volta avviata, legge da /dev/urandom o urandomARTIFICIALE.binary (a seconda della modalità di funzionamento) e invia in un ciclo di 30 secondi 8 byte di dati al secondo a ECUoutput il quale usa questi dati per stabilire se il parcheggio è avvenuto con successo o meno. Prima di entrare nel ciclo for 30 secondi park_assist avvia surround_view_cameras e inizia a ricevere i byte da esso: precedentemente a ogni lettura nel pipe va a verificare se surround_view_cameras ha in effetti inviato i dati. Questo controllo è effettuato andando prima a leggere il messaggio preventivo mandato da surround_view_cameras proprio allo scopo. Ricevuto questi ulteriori byte park_assist li invia insieme ai 8 precedenti a ECUoutput.

Front_windshield_camera legge da frontCamera.data stringhe che ECUoutput usa per stabilire l'operazione da svolgere. Per inviare queste stringhe front_windshield_camera si deve connettersi attraverso il suo socket lato client con il socket lato server di ECUoutput. Dopo aver inviato il messaggio, prima di chiudere il modulo controlla se deve o no attendere il completamento di ECUoutput prima di inviare la richiesta successiva, se sì allora si sospende facendo una read sul socket di comunicazione. Steer_by_wire è il modulo usato per controllare lo sterzo: fa uso di un pipe con nome non bloccante per comunicare con ECUoutput. Ogni secondo legge da questo pipe e se questa è vuota siccome è non bloccante non rimane bloccato e quindi può continuare la sua esecuzione e scrivere nel steer.log, come richiesto, la stringa "NO ACTION". Se invece il pipe non è vuoto allora controlla se la stringa è il comando di giro a destra o giro a sinistra e registra conseguentemente l'azione corrispondente nel log. Nella comunicazione tra ECUoutput e steer_by_wire viene anche fatto uso di un pipe bloccante come semaforo: per avere giri di 4 secondi ECUoutput invia 4 richieste di giro a steer_by_wire e si blocca in 4 successive read su semaforo. Solo quando saranno completate le 4 richieste da parte di steer_by_wire, che alla fine di ogni ciclo fa una write sullo stesso semaforo, ECUoutput completa l'operazione del girare.

Park_assist conclude una volta che finisce di eseguire il ciclo for di 30 secondi, front_windshield_camera termina o perché ha finito di leggere il file o perché gli è stato inviato un segnale di terminazione software, mentre steer_by_wire termina o perché ECUoutput ha chiuso il suo lato del pipe o perché ha ricevuto un segnale di terminazione software. Brake_by_wire e throttle_control controllano uno il freno e l'altro l'acceleratore e sono fatti in maniera molto simile: entrambi leggono continuamente dal loro pipe con cui comunicano con ECUoutput e quando ricevono un messaggio effettuano l'accelerazione o il freno del veicolo. In entrambi i casi però la variabile globale della velocità, rappresentata attraverso un valore in un file, è decrementata o incrementata dall'ECUoutput e non dai moduli stessi. In particolare throttle_control implementa anche la procedura di fallimento dell'acceleratore. Entrambi o terminano perché il pipe è stato chiuso in lato scrittura da ECUoutput o perché hanno ricevuto un segnale di arresto. Surround_view_cameras viene avviato da park_assist e conclude la sua esecuzione solo ricevendo un segnale di terminazione di esecuzione dallo stesso. Durante la sua esecuzione surround_view_cameras legge al secondo, o da urandom o da urandomArtificiale (a seconda della modalità di esecuzione), 8 byte e li invia a park_assist che li invia a sua volta a ECUoutput dopo aver registrato i valori nel log.

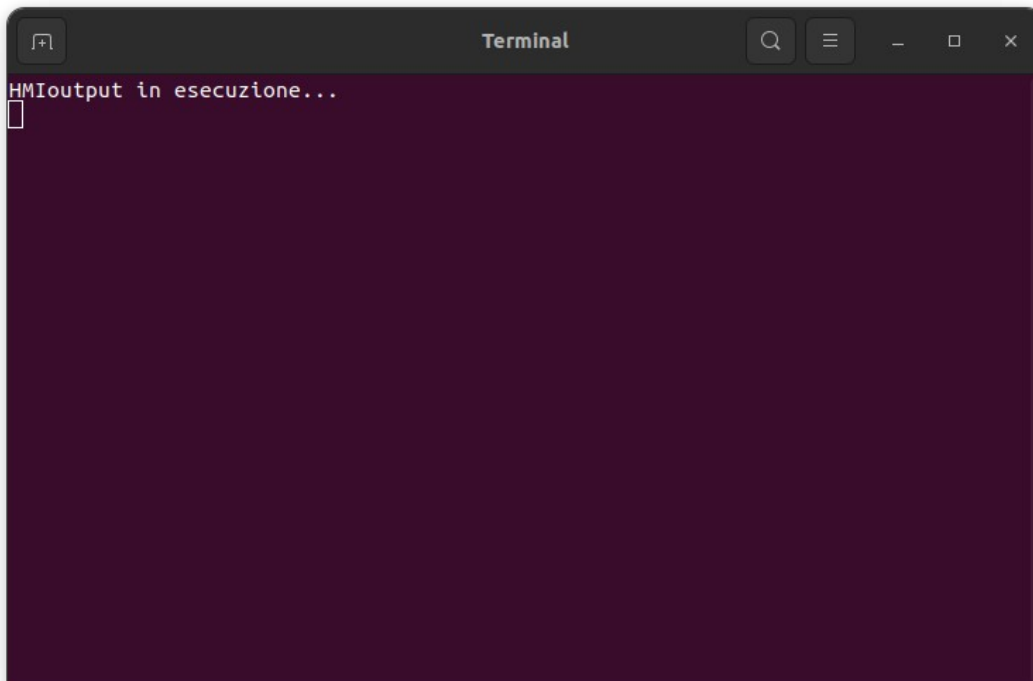
Esempio Esecuzione

1. esecuzione a conclusione naturale del programma in modalità artificiale:



```
cslave@cslave-HP-250-G8-Notebook-PC: ~/Desktop/progetto...
cc -c park_assist.c;
cc -o park_assist park_assist.o;
cc -c steer_by_wire.c;
cc -o steer_by_wire steer_by_wire.o;
cc -c throttle_control.c;
cc -o throttle_control throttle_control.o readString.o;
cc -c surround_view_cameras.c;
cc -o surround_view_cameras surround_view_cameras.o;
make[1]: Leaving directory '/home/cslave/Desktop/progetto/ADAS/src'
cslave@cslave-HP-250-G8-Notebook-PC:~/Desktop/progetto/ADAS$ cd bin/
cslave@cslave-HP-250-G8-Notebook-PC:~/Desktop/progetto/ADAS/bin$ ./HMIinput ARTI
FICIALE
programma in esecuzione...
HMI in esecuzione...
HMIinput in esecuzione...
avvio ECU...
avvio ECUoutput..
ECU: esecuzione in modalità artificiale
front_windshield_camera in esecuzione...
steer_by_wire in esecuzione...
brake_by_wire in esecuzione...
throttle_control in esecuzione...
HMIinput: inserire "INIZIO" per iniziare la navigazione
█
```

Sul terminale di input all'avvio del programma vengono stampati le seguenti stringhe che indicano quali componenti sono in esecuzione, e la modalità di esecuzione. Una volta che si inserisce INIZIO e si digita invio parte la navigazione.



```
Terminal
HMIoutput in esecuzione...
█
```

HMIinput lancia un nuovo terminale e mette in esecuzione HMIoutput.

```
Terminal
HMIoutput in esecuzione...
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5

```

Terminale che mostra l'output dell'ECU: si può osservare che dopo un primo momento di accelerazione non viene più stampato nulla. Questo è dovuto al fatto che in frontCamera.data ha molte velocità uguali di seguito e ECU non ha bisogno di accelerare o decelerare per un lungo periodo di tempo.

```
cslave@cslave-HP-250-G8-Notebook-PC: ~/Desktop/progetto...
ECU: operazione richiesta completata
ECU: server in ascolto degli input dei componenti...
f_w_c: avviato tentativo di connessione all'ECU...
messaggio inviato all'ECU: 70
f_w_c: collegamento con l'ECU avvenuto con successo
f_w_c: in attesa del completamento dell'operazione richiesta...
ECU: collegamento avvenuto con successo, richiesta in elaborazione...
ECU: ricevuto messaggio "70"
ECU: velocità attuale auto 70
ECU: l'auto è già alla velocità desiderata
ECU: velocità desiderata raggiunta. velocità=70
ECU: operazione richiesta completata
ECU: server in ascolto degli input dei componenti...
f_w_c: avviato tentativo di connessione all'ECU...
messaggio inviato all'ECU: 70
f_w_c: collegamento con l'ECU avvenuto con successo
ECU: collegamento avvenuto con successo, richiesta in elaborazione...
f_w_c: in attesa del completamento dell'operazione richiesta...
ECU: ricevuto messaggio "70"
ECU: velocità attuale auto 70
ECU: l'auto è già alla velocità desiderata
ECU: velocità desiderata raggiunta. velocità=70
ECU: operazione richiesta completata

```

Tuttavia possiamo cmq vedere sul log di debug le azioni di controllo che effettua la ECU.

Andiamo a vedere cosa succede quando ECUoutput riceve una richiesta di PARCHEGGIO da front_windshield_camera:

```
cslave@cslave-HP-250-G8-Notebook-PC: ~/Desktop/progetto...
steer_by_wire in esecuzione...
IZIO
HMIinput: inserire "INIZIO" per iniziare la navigazione
ECU: ricevuto segnale "INIZIO"
avvio ECUinput...
^Ccslave@cslave-HP-250-G8-Notebook-PC:~/Desktop/progetto/ADAS/bin$ ./HMIinput AR
TIFICIALE
programma in esecuzione...
HMI in esecuzione...
HMIinput in esecuzione...
avvio ECU...
HMIoutput in esecuzione...
avvio ECUoutput..
ECU: esecuzione in modalità artificiale
brake_by_wire in esecuzione...
front_windshield_camera in esecuzione...
steer_by_wire in esecuzione...
throttle_control in esecuzione...
HMIinput: inserire "INIZIO" per iniziare la navigazione
INIZIO
ECU: ricevuto segnale "INIZIO"
avvio ECUinput...
park_assist in esecuzione...
```

Quando ECUoutput riceve il messaggio di parcheggio da front_windshield_camera viene avviato park_assist:

```
cslave@cslave-HP-250-G8-Notebook-PC: ~/Desktop/progetto...
FRENO 5
FRENO 5
tail: output_terminal.txt: file truncated
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
INCREMENTO 5
FRENO 5
FRENO 5
FRENO 5
FRENO 5
FRENO 5
FRENO 5
FRENO 5
FRENO 5
FRENO 5
FRENO 5
```

si registra una serie di di comandi “FRENO 5” per portare a zero la velocità del veicolo prima che park_assist venga avviato per inviare dati all’ECU.

```
cslave@cslave-HP-250-G8-Notebook-PC: ~/Desktop/progetto...
f_w_c: in attesa del completamento dell'operazione richiesta...
ECU: collegamento avvenuto con successo, richiesta in elaborazione...
ECU: ricevuto messaggio "PARCHEGGIO"
ECU: tentativo di parcheggio...
ECU: server in ascolto degli input dei componenti...
brake_by_wire ha concluso la sua esecuzione
throttle_control ha concluso la sua esecuzione
steer_by_wire ha concluso la sua esecuzione
front_windshield_camera ha concluso la sua esecuzione
park_assist in esecuzione...
p_a: aperta comunicazione con ECU
surround_view_camera in esecuzione...
p_a: aperta comunicazione con s_v_c
ECU: valori trasmessi da p_a 56847 24840 41806 58574
ECU: valori trasmessi da s_v_c 56847 24840 41806 58574
ECU: valori trasmessi da p_a 8129 42100 48597 34461
ECU: valori trasmessi da s_v_c 8129 42100 48597 34461
ECU: valori trasmessi da p_a 54915 46541 40604 12137
ECU: valori trasmessi da s_v_c 54915 46541 40604 12137
ECU: valori trasmessi da p_a 42476 24442 710 42332
ECU: valori trasmessi da s_v_c 42476 24442 710 42332
ECU: valori trasmessi da p_a 54723 11093 53772 40959
ECU: valori trasmessi da s_v_c 54723 11093 53772 40959
```

Sul terminale che mostra il contenuto di debuglog.txt possiamo osservare i numeri che vengono mandati da park_assist a ECUoutput e la conclusione del programma in seguito.

Alla conclusione del programma il secondo terminale viene chiuso automaticamente e sul terminale di partenza vengono lasciati i seguenti messaggi.

```
cslave@cslave-HP-250-G8-Notebook-PC: ~/Desktop/progetto...
avvio ECUoutput..
ECU: esecuzione in modalità artificiale
front_windshield_camera in esecuzione...
steer_by_wire in esecuzione...
brake_by_wire in esecuzione...
throttle_control in esecuzione...
HMIinput: inserire "INIZIO" per iniziare la navigazione
INIZIO
ECU: ricevuto segnale "INIZIO"
avvio ECUinput...
ARRESTO
throttle_control ha concluso la sua esecuzione
steer_by_wire ha concluso la sua esecuzione
brake_by_wire ha concluso la sua esecuzione
front_windshield_camera ha concluso la sua esecuzione
park_assist in esecuzione...
surround_view_camera in esecuzione...
surround_view_cameras ha concluso la sua esecuzione
park_assist ha concluso la sua esecuzione
ECUinput ha concluso la sua esecuzione
ECUoutput ha concluso la sua esecuzione
HMIinput ha concluso la sua esecuzione
il programma ha concluso la sua esecuzione
cslave@cslave-HP-250-G8-Notebook-PC:~/Desktop/progetto/ADAS/bin$
```