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

Full Name: Naum Doka **e-mail:** nandodoka09@gmail.com

Linux Distribution: Ubuntu 22.04.3

HMI: costituisce il modulo responsabile della gestione dell'interazione con l'utente, gestendo l'input e l'output attraverso un terminale. Essa interpreta i comandi forniti dall'utente, trasmettendoli all'ECU mediante l'utilizzo di una pipe con nome per la comunicazione asincrona tra i componenti. Utilizza le seguenti funzioni:

- ***openOutputTerminal():*** apre un terminale e mostra l'output all'utente usando il comando 'system' (gnome-terminal) insieme a un percorso specificato della pipe con nome(FIFO). Apre la pipe con nome per la scrittura. Inoltre, imposta i componenti e i file necessari all'ECU.
- ***writeToOutputTerminal():*** verifica se la pipe con nome è disponibile per la scrittura. Scrive il 'message' fornito nella FIFO per visualizzarlo nel terminale di uscita.
- ***closeOutputTerminal():*** controlla se la FIFO è aperta per la scrittura e, nel caso affermativo, la chiude, garantendo una corretta gestione delle risorse e della comunicazione tra l'HMI e l'ECU.
- ***initHMI():*** assegna il percorso specificato (/tmp/myfifo) all'attributo 'fifoPath' della struttura HMI. Successivamente, crea e apre una pipe con nome utilizzando 'mkfifo()' per stabilire il canale di comunicazione. Chiama ***openOutputTerminal()*** per l'apertura del terminale di output, configurando così l'HMI per l'interazione con l'utente.
- ***HMILoop():*** rappresenta il ciclo principale dell'interfaccia. Utilizza 'select()' per controllare l'input da stdin (standard input) o un timeout di 1 secondo. Legge e converte l'input da stdin in lettere maiuscole, inviando i comandi al terminale di output chiamando ***writeToOutputTerminal()*** . Restituisce valori in base ai comandi ricevuti (1 per START, 2 per PARK, 3 per STOP) o 0 in caso di timeout.

steerByWire: riceve il comando di sterzata DESTRA o SINISTRA dalla Central ECU. Gestisce le informazioni di logging relative alle azioni di sterzata . Utilizza le seguenti funzioni:

- ***initSteer():*** apre o crea il file "steer.log" in modalità di scrittura. Gestisce eventuali errori di apertura del file. Scrive "Start" nel file per segnalare l'avvio della funzionalità di sterzata. Infine, procede alla chiusura del file in modo appropriato.
- ***right():*** apre il file "steer.log" in modalità append per aggiungere contenuto alla fine. Gestisce eventuali errori di apertura del file. Registra una log entry che indica un'azione di sterzata a destra. Procede alla chiusura del file in modo appropriato.
- ***left():*** apre il file "steer.log" in modalità append per aggiungere contenuto alla fine. Gestisce eventuali errori di apertura del file. Registra una log entry che indica un'azione di sterzata a sinistra. Procede alla chiusura del file in modo appropriato.

L'ECU interagisce con il modulo 'steerByWire' tramite la chiamata delle sue funzioni per l'inizializzazione e il controllo delle azioni di sterzata. Quindi, la comunicazione avviene tramite chiamate di funzione e operazioni su file.

throttleControl: riceve comando di accelerazione dalla Central ECU. Gestisce il logging delle azioni di controllo dell'acceleratore. Utilizza le seguenti funzioni:

- ***initThrottle():*** apre o crea il file "throttle.log" in modalità di scrittura. Gestisce eventuali errori di apertura del file. Scrive "Start" nel file per segnalare l'avvio della componente throttleControl. Successivamente, si occupa della chiusura del file in modo appropriato.
- ***increase5():*** all'interno di questa funzione, viene dichiarato un buffer denominato 'dateTimeStr' utilizzato per memorizzare la data e l'ora formattate. Recupera l'ora

corrente e la formatta utilizzando le funzioni ‘time()’ e ‘localtime()’. Successivamente, formatta la data e l'ora in una stringa leggibile usando ‘strftime()’. Apre il file "throttle.log" in modalità append per aggiungere contenuto alla fine. Gestisce eventuali errori di apertura del file. Scrive nel file il timestamp formattato insieme a un messaggio che indica l'azione di aumento del throttle. Procede alla chiusura del file in modo appropriato.

L'ECU interagisce con il modulo ‘throttleControl’ mediante l'invocazione delle sue funzioni per l'inizializzazione e il controllo dell'acceleratore. Quindi, la comunicazione avviene tramite chiamate di funzione e operazioni su file.

brakeByWire: riceve dalla Central ECU il comando di decelerazione o un segnale di pericolo. Gestisce il logging delle azioni relative al freno. Utilizza le seguenti funzioni:

- ***initBrake():*** apre o crea il file "brake.log" in modalità di scrittura. Gestisce eventuali errori di apertura del file. Scrive "Start" nel file per segnalare l'avvio della componente brakeByWire. Procede alla chiusura del file in modo appropriato.
- ***brake5***(): all'interno di questa funzione, viene dichiarato un buffer denominato ‘dateTimeStr’ utilizzato per memorizzare la data e l'ora formattate. Recupera l'ora corrente e la formatta utilizzando le funzioni ‘time()’ e ‘localtime()’. Successivamente, formatta la data e l'ora in una stringa leggibile usando ‘strftime()’. Apre il file "brake.log" in modalità append per aggiungere contenuto alla fine. Gestisce eventuali errori di apertura del file. Scrive nel file il timestamp formattato insieme a un messaggio che indica l'azione del freno. Procede alla chiusura del file in modo appropriato.
- ***stop():*** all'interno di questa funzione, viene dichiarato un buffer ‘dateTimeStr’ per memorizzare la data e l'ora formattate. Recupera l'ora corrente e la formatta utilizzando le funzioni ‘time()’ e ‘localtime()’. Formatta la data e l'ora in una stringa leggibile usando ‘strftime()’. Apre il file "brake.log" in modalità append per aggiungere contenuto alla fine. Controlla se l'apertura del file fallisce e in tal caso restituisce 1. Scrive nel file il timestamp formattato insieme a un messaggio che indica l'azione di arresto. Procede alla chiusura del file in modo appropriato.

ECU interagisce con il modulo ‘brakeByWire’ indirettamente attraverso l'uso di meccanismi di logging e operazioni su file. L'ECU registra le azioni relativi al freno nel file "brake.log".
Il modulo ‘brakeByWire’ monitora questo file di log, interpreta le azioni "loggate" ed esegue le azioni corrispondenti.

frontWindshieldCamera: legge dati e li invia alla Central ECU. Utilizza le seguenti funzioni:

- ***initFrontCameraComponent():*** apre o crea il file "camera.log" in modalità di scrittura. Gestisce eventuali errori di apertura del file. Scrive "Start" nel file per segnalare l'avvio della componente frontWindshieldCamera. Procede alla chiusura del file in modo appropriato.
- ***readLineFromFile():*** prende come argomento un puntatore a un FILE per leggere una riga da quel file. Utilizza un buffer statico per memorizzare la riga letta. Controlla se durante l'operazione di lettura del file si verifica la fine del file o un errore e in tal caso restituisce NULL. Apre il file "camera.log" in modalità append per aggiungere contenuto alla fine. Gestisce eventuali errori di apertura del file. Scrive la riga letta nel file "camera.log". Chiude il file "camera.log". Restituisce il puntatore alla riga letta.

La funzione ***readLineFromFile()*** implica che l'ECU legge i dati o i comandi relativi alla front windshield camera dal file "frontCamera.data" e registra questi dati nel file "camera.log".

forwardFacingRadar: acquisice dati da sorgenti diverse in base alla modalità di esecuzione correcnte e li invia alla ECU. Utilizza le seguenti funzioni:

- ***initforwardFacingRadar():*** apre o crea il file "radar.log" in modalità di scrittura e vi scrive "Start", segnalando l'inizio delle operazioni del forward facing radar.
- ***readForwardFacingRadar():*** il comportamento di questa funzione varia in base al valore dell'opzione specificata:
  se l'opzione == 1: Apre '/dev/urandom' per leggere random bytes. Legge 8 byte di dati e li mette nel buffer. Apre "radar.log" in modalità append. Formatta e aggiunge i byte casuali in formato esadecimale in "radar.log". Chiude i file (/dev/urandom e "radar.log").
  se l'opzione != 1: Legge 8 byte di dati dal file "urandomARTIFICIALE.binary". Apre "radar.log" in modalità append. Formatta e aggiunge i byte letti in formato esadecimale in "radar.log". Infine, procede alla chiusura dei file di input e output in modo appropriato.

L'ECU interagisce con il modulo 'forwardFacingRadar' mediante l'invocazione delle sue funzioni per ricevere i dati letti. Tale comunicazione avviene in modo indiretto attraverso le chiamate di funzione specifiche e le operazioni su file.

parkAssist: agisce solo su richiesta della Central ECU, quando riceve un comando di attivazione da quest'ultima. Utilizza le seguenti funzioni:

- ***initParkAssist():*** apre o crea il file "park assist.log" in modalità di scrittura. Controlla se la creazione/apertura del file è avvenuta con successo. In caso contrario, stampa un messaggio di errore e restituisce 0. Scrive "Start" nel file "park assist.log" per indicare l'avvio della componente Park Assist. Chiude il file in modo appropriato dopo la scrittura.
- ***readParkAssist():*** apre il file "/dev/urandom" o "urandomARTIFICIALE.binary" in base al valore del parametro 'option'. Utilizza /dev/urandom se option == 1 o il file urandomARTIFICIALE.binary, altrimenti. Legge 1 singolo byte dal file aperto e viene ripetuta 8 volte nell'ECU per simulare la lettura di 8 byte. I byte letti vengono combinati con i dati ottenuti da ***readSurroundViews()*** per formare un output di tipo uint16_t. I byte combinati vengono registrati nel file "park assist.log" in formato esadecimale.
  Infine, la funzione restituisce l'output uint16_t combinato.

Durante la sequenza di parcheggio (parking_seconds == 30), l'ECU interagisce attivamente con il modulo 'Park Assist' richiamando ripetutamente la funzione readParkAssist() in un ciclo. Questa azione è finalizzata una volta ricevuti i dati del Park Assist. Questi dati vengono poi elaborati dall'ECU per monitorare l'ambiente circostante ed eseguire le manovre di parcheggio.

surroundViewsCamera: agisce solo quando Park Assist è attivo. Legge i dati e li invia al Park Assist. Utilizza le seguenti funzioni:

- ***initSurroundViews():*** crea/apre un file denominato "park assist.log" in modalità di scrittura. Se l'apertura del file non avviene, viene visualizzato un messaggio di errore. Scrive "Start" nel file, che rappresenta l'avvio della componente surroundViewsCamera. Chiude il file in modo appropriato.
- ***readSurroundViews():*** Legge 1 singolo byte di dati da diverse sorgenti, in base all'opzione fornita. Viene chiamata 8 volte nell'ECU per simulare la lettura di 8 bytes:

Se opzione == 1, la funzione legge i dati dal file “/dev/urandom”;
Se opzione != 1, la funzione legge i dati dal file "urandomARTIFICIALE.binary";
In seguito alla lettura dei dati da una delle due sorgenti, restituisce il byte di dati letto dall'origine specificata.

La “communicazione” tra il modulo ‘surroundViewsCamera’ e il modulo ‘parkAssist’ avviene in modo indiretto. Il modulo ‘parkAssist’ chiama ***readSurroundViews()***, per ottenere i dati letti da ‘surroundViewsCamera’. Successivamente, combina questi dati con quelli letti dal modulo stesso tramite readParkAssist(), formando un output di tipo uint16_t, che viene infine inviato all’ECU .

main: il punto di ingresso principale del programma, utilizzato per scegliere la modalità di esecuzione**.** Dentro la funzione main(), viene chiamata la funzione ***run()***, implementata nel file ‘ECU.c’. Il parametro passato a questa funzione indica la modalità di esecuzione del progetto. Valore 1 indica l’esecuzione NORMALE, valore 0 indica l’esecuzione ARTIFICIALE. Restituisce un valore intero che rappresenta il codice di uscita del programma.

ECU: utilizza le seguenti funzioni ***run()***, ***isInList()*** *e* ***logECU()***. La funzione ***run()*** apre due file ‘frontCamera.data’ in modalità di lettura per leggere i dati dei sensori e ‘ECU.log’ in modalità di scrittura per scrivere e inserire ogni comando inviato dall’ECU in questo log file. Gestisce gli errori relativi all'apertura dei file.
L'avvio del sistema avviene attraverso l'inizializzazione dell'HMI tramite l'invocazione della funzione ***initHMI()***, con la trasmissione di un puntatore alla struttura HMI. Inoltre, si procede con l'impostazione delle variabili correlate agli stati e ai timer di sistema, nonché con l'inizializzazione delle variabili atte al controllo della velocità, della sterzata e dei parametri di parcheggio.
Il modulo entra in un ciclo infinito (while(1)), interagendo in modo continuo con l'HMI attraverso la funzione ***HMILoop()***, ricevendo input da quest'ultimo. L'ECU risponde ai comandi provenienti dall'HMI (START, PARK, STOP), eseguendo azioni differenti in base a tali comandi utilizzando una struttura switch-case. Tuttavia, l'esecuzione non inizia fintanto che non viene ricevuto il comando START. All’interno della switch-case, si verificano i seguenti stati:

- Stato di avvio (caso 1): Tale stato implica l'inizializzazione dei componenti quali throttleControl (mediante l'invocazione di ***initThrottle()***), brakeByWire (tramite ***initBrake()***), frontCamera (tramite ***initFrontCameraComponent()***), steerByWire (tramite ***initSteer()***), forwardFacingRadar (tramite ***initForwardFacingRadar()***) e parkAssist (tramite ***initParkAssist()***).
- Stato di parcheggio (caso 2): Questo stato comporta l'attivazione del modulo parkAssist (tramite ***initParkAssist()***) e la gestione della transizione verso la modalità di parcheggio. Stato di arresto (caso 3): Questo stato arresta il sistema e azzera la velocità(invocando ***stop()***).

Successivamente al ricevimento del comando 'START', l'ECU inizia a leggere i dati provenienti dal Forward Facing Radar (attraverso l’invocazione di ***readForwardFacingRadar()***) e dalla Front Windshield Camera (attraverso l’invocazione di ***readFromLine()***), agendo in base ai dati ottenuti mediante l'invio di comandi ai vari componenti del sistema.
Durante la modalità di parcheggio, l'ECU riceve dati dal Park Assist (attraverso l’invocazione di ***readParkAssist()***). Il Park Assist, a sua volta, invia all'ECU i dati ottenuti insieme ai dati acquisiti dalla componente Surround View Cameras. La Central ECU registra tutti i comandi

inviati nel file di log denominato 'ECU.log' tramite la funzione ***logECU()***. Inoltre, tali comandi sono scritti in una pipe con nome utilizzata anche dall'HMI attraverso la chiamata a ***writeToOutputTerminal()***, per essere visualizzati a schermo. La funzione ***isInList()*** è impiegata per verificare la presenza di valori specifici in un elenco predeterminato. Quest'ultima funzione è utilizzata per la gestione della situazione in cui, nel caso la Central ECU non riceva, per un intervallo di 30 secondi, nessuno dei seguenti valori: i) 0x172A, ii) 0xD693, iii) 0x0000, iv) 0xBDD8, v) 0xFAEE, vi) 0x4300, l'auto viene considerata parcheggiata e la missione terminata. Infine, l'ECU provvede la chiusura dei file e la pulizia delle risorse al momento della terminazione del programma.

Compilazione ed Esecuzione:

Il progetto è eseguibile anche come utente non privilegiato, ovvero senza i privilegi di root.

```
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$ gcc -c ECU.c -o ECU.o
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$ gcc -c HMI.c -o HMI.o
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$ gcc -c brakeByWire.c -o brakeByWire.o
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$ gcc -c forwardFacingRadar.c -o forwardFacingRadar.o
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$ gcc -c frontCameraComponent.c -o frontCameraComponent.o
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$ gcc -c main.c -o main.o
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$ gcc -c parkAssist.c -o parkAssist.o
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$ gcc -c steerByWire.c -o steerByWire.o
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$ gcc -c surroundViewsCameras.c -o surrouncViewsCameras.o
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$ gcc -c throttleControl.c -o throttleControl.o
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$ gcc ECU.o HMI.o brakeByWire.o forwardFacingRadar.o frontCameraComponent.o main.o parkAssist.o steerByWire.o
 surroundViewsCameras.o throttleControl.o -o progetto
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$ sudo ./progetto
start
park
stop
park
albeats@AlBeats:/mnt/c/Users/nando/Desktop/ADAS$
```