Relazione

Progetto A.A. 2022/2023
ADAS made trivial: rappresentazione ispirata alle interazioni in un sistema di guida autonoma

Diciotti Matteo	Manucci Agostino	Montes Anacona Àlvaro
7072181	7084379	7117731

10 agosto 2023

Titolo Relazione progetto A.A. 2022/2023 di Basi di Dati e Sistemi Informativi

Autori Lista degli autori ordinata per numero di matricola

Matricola	Cognome	Nome	e-mail
7072181	Diciotti	Matteo	matteo.diciotti@stud.unifi.it
7084379	Mannucci	Agostino	agostino.mannucci@stud.unifi.it

Obiettivo Obiettivo del progetto è la realizzazione, per fini accademici, di una base di dati a partire dall'identificazione di una richiesta realmente proponibile, seguita dalla fase di progettazione concettuale cui dovrà succedere la fase di progettazione logica e quindi la realizzazione in MySQL.

Introduzione al progetto La realizzazione di una base di dati prevede che sia definita rigorosamente una richiesta, pilastro attorno cui si sviluppa il database passando da alcune importanti fasi: una fase di studio, concentrata principalmente nella fase di *progettazione concettuale*, una fase di analisi, suddivisa tra la fase di *progettazione concettuale* e la fase di *progettazione logica*, ed infine una fase di sviluppo, concentrata nella fase di *progettazione logica* e nella realizzazione in linguaggio *MySQL*.
La creazione della base proposta parte dalla scrittura di una richiesta completa e precisa, affinché fosse plausibile che la fase preliminare, comprendente il confronto col richiedente, fosse già avvenuta proficuamente avendo quindi predefinito le specifiche necessarie ed i dettagli importanti per il cliente. Nella progettazione siamo quindi partiti da una richiesta che non dovesse essere modificata ulteriormente ma che potesse considerarsi definitiva e quindi immediatamente appetibile per le fasi di progettazione suddette, bacino nozionistico del corso.
La stesura della relazione segue a grandi linee l'ordine con cui sono state affrontate le varie fasi di progettazione e realizzazione: *analisi della richiesta*, *analisi dei termini*, *traduzione del testo* e *creazione del glossario dei termini*, dopodiché è stato *creato il modello ER* (questa prima parte rappresenta la progettazione concettuale). Per quanto concerne la fase di progettazione logica sono state eseguite le scelte implementative durante le procedure di *eliminazione delle gerarchie*, *eliminazione degli attributi multi-valore* e nella *scelta degli identificatori principali*, giungendo infine al *modello ER ristrutturato*. Da questo è stato quindi definito passo-passo lo schema logico, fino a giungere allo

schema logico completo a pagina **??**.
La fase di realizzazione in MySQL viene affrontata nella presente relazione suddividendo lo script in sezioni tematiche, ovvero in cinque parti che rappresentano i meccanismi focali dell'implementazione:

- Creazione e popolamento delle tabelle
- Interrogazioni
- Procedure e funzioni
- Viste
- Trigger

Richiesta

Una società che organizza tornei di calcio a 5 e a 7 è presente nel territorio toscano con tornei attivi nella città di Firenze e nelle zone limitrofe.

Di questi tornei la società desidera mantenere informazioni relative alle fasi di gioco, alle squadre partecipanti, ai giocatori che compongono le formazioni e alle partite.
I tornei sono identificati tramite un codice e sono caratterizzati da un nome, da un'edizione, dalla tipologia di gioco (calcio a 5 o a 7) e dalla categoria di genere (maschile, femminile o mista).
Ogni torneo può essere suddiviso in più fasi di gioco (tipicamente da 1 a 3), le quali si distinguono per la modalità di organizzazione degli scontri tra le squadre (a gironi o ad eliminazione diretta) e che sono caratterizzate dal nome della fase e da un numero rappresentante la quantità di scontri tra due squadre nella fase. Alle fasi possono corrispondere uno o più insiemi di squadre partecipanti e un gruppo di giornate di gioco, che rappresentano i turni della fase.
Gli insiemi di squadre, identificati dalla fase a cui appartengono e dal nome dell'insieme, raggruppano le formazioni partecipanti a quella fase.
Le giornate di gioco calendarizzano le partite di una fase e possiedono su queste i vincoli imposti alla modalità di organizzazione degli scontri (a gironi o ad eliminazione diretta). Le giornate sono identificate dalla fase a cui appartengono e dal numero della giornata.
In particolare è rilevante per ogni partita conoscere la squadra di casa e la squadra ospite, la data di gioco, il campo, l'arbitro che dirige la gara e al punteggio finale.
Per ogni partita la società ha interesse a mantenere le statistiche sui giocatori che hanno effettuato azioni rilevanti in quella partita (gol fatti, assist effettuati, espulsioni, ammonizioni), mentre le squadre si distinguono per il nome, la tipologia di calcio a cui giocano e il genere dei giocatori che la compongono.
La società richiede inoltre che sia mantenuta l'informazione sul campo di casa il quale è contraddistinto dagli altri campi attraverso l'indirizzo. I campi possiedono comunque un nome proprio e un recapito. Le persone tesserate alla società si suddividono in due tipologie: i giocatori e gli arbitri i quali condividono un numero di tessera univoco per ogni tesserato.
Di tutti gli iscritti si conoscono nome e cognome, data di nascita e genere, mentre per i giocatori si conosce, qualora partecipino a qualche torneo, anche la squadra a con la quale gareggiano ed il relativo numero di maglia.

Progettazione Concettuale

Analisi della richiesta

Si procede all'analisi della richiesta evidenziando:

- in *corsivo* i termini ambigui o imprecisi;
- sottolineati i termini con i quali si esprimono concetti diversi;
- in **grassetto** i termini concettualmente equivalenti, distinguendo successivamente le equivalenze.

Una società che organizza tornei di calcio a 5 e a 7 è presente nel territorio toscano con tornei attivi nella città di Firenze e nelle zone limitrofe.

Di questi tornei la società desidera mantenere informazioni relative alle fasi di gioco, alle **squadre** partecipanti, ai giocatori che compongono le **formazioni** e alle **partite**.
I tornei sono identificati tramite un codice e sono caratterizzati da un nome, da un'*edizione*, dalla tipologia di gioco (calcio a 5 o a 7) e dalla categoria di genere (maschile, femminile o mista).
Ogni torneo può essere suddiviso in più fasi di gioco (tipicamente da 1 a 3), le quali si distinguono per la modalità di organizzazione degli **scontri** tra le squadre (a gironi o ad eliminazione diretta) e che sono caratterizzate dal nome della fase e da un numero rappresentante la quantità di scontri tra due squadre nella fase. Alle fasi possono corrispondere uno o più insiemi di squadre partecipanti e un gruppo di giornate di gioco, che rappresentano i turni della fase.
Gli insiemi di squadre, identificati dalla fase a cui appartengono e dal nome dell'insieme, raggruppano le formazioni partecipanti a quella fase.
Le giornate di gioco calendarizzano le partite di una fase e possiedono su queste i vincoli imposti alla modalità di organizzazione degli scontri (a gironi o ad eliminazione diretta). Le giornate sono identificate dalla fase a cui appartengono e dal numero della giornata.
In particolare è rilevante per ogni partita conoscere la squadra di casa e la squadra ospite, la data di gioco, il campo, l'arbitro che dirige la **gara** e al *punteggio* finale.
Per ogni partita la società ha interesse a mantenere le statistiche sui giocatori che hanno effettuato azioni rilevanti in quella partita (gol fatti, assist effettuati, espulsioni, ammonizioni), mentre le squadre si distinguono per il nome, la tipologia di calcio a cui giocano e il genere dei giocatori che la compongono. La società richiede inoltre che sia mantenuta l'informazione sul campo di casa il quale è contraddistinto dagli altri campi attraverso l'*indirizzo*. I campi possiedono comunque un nome proprio e un recapito. Le **persone tesserate** alla società si suddividono in due tipologie: i giocatori e gli arbitri i quali condividono un numero di tessera univoco per ogni **tesserato**.
Di tutti gli **iscritti** si conoscono nome e cognome, data di nascita e genere, mentre per i giocatori si conosce, qualora **partecipino** a qualche torneo, anche la squadra a con la quale **gareggiano** ed il relativo numero di maglia.

Analisi dei termini

Termini ambigui o imprecisi

- *edizione*: si riferisce al numero in cui è stato ripetuto lo stesso torneo e si indica con un numero intero positivo diverso da zero, per semplicità diremo "numero di edizione";
- *punteggio*: si riferisce ai gol segnati dalla squadra di casa e quelli segnati dalla squadra ospite;
- *indirizzo*: si riferisce all'indicazione del comune, della via in cui è situato e del numero civico;
- *recapito*: si riferisce al numero di telefono dei gestori del campo sportivo.

Termini concettualmente plurimi

- partecipante: il temine è utilizzato per indicare una squadra che partecipa ad un torneo ("squadra partecipante" o "formazione partecipante") sia per indicare un giocatore membro di una squadra che partecipa ad un torneo ("giocatore partecipante");
- nome: il termine è utilizzato per indicare il nome del torneo, il nomi di una fase del torneo, il nome di un insieme di squadre, il nome di una squadra, il nome di un campo e i nome di un tesserato.
- tipologia: il termine sta ad indicare la tipologia di calcio giocato che un torneo prevede ("tipologia del torneo"), la tipologia di calcio che una squadra gioca ("tipologia di calcio della squadra") ed infine la tipologia di tesseramento di una persona alla società, se si iscrive come giocatore o come arbitro ("tipologia di tesseramento");

- genere: il temine è utilizzato per indicare il genere dei tesserati ("genere dei tesserati"), il genere dei giocatori di una squadra ("genere dei giocatori della squadra") ed la composizione di genere dei giocatori ammessi a partecipare ad un torneo ("categoria di genere");
- numero: il termine è utilizzato per indicare il numero rappresentante la quantità di scontri tra due squadre in una fase ("numero di scontri"), il numero che identifica una giornata dio gioco ("Numero della giornata") il numero di tessera di un iscritto ("numero di tessera") e il numero di maglia di un giocatore all'interno di una squadra ("numero di maglia");
- data: il termine è utilizzato per indicare il giorno e l'orario in cui si disputa una partita ("data di gioco") e per indicare la data di nascita di un tesserato ("data di nascita");
- campo: il termine è utilizzato per indicare un campo gestito dalla società organizzatrice dei tornei ("campo"), il campo di casa selezionato da una squadra come preferenza ("campo di casa") ed il campo in cui si svolge una partita ("campo d gioco").

Termini concettualmente equivalenti

Termine	Descrizione	Sinonimi
squadra	Insieme di giocatori che unitamente possono partecipare ad un torneo	formazione
partita	Evento nel quale due squadre si affrontano e il cui punteggio determina un unico vincitore oppure un pareggio	scontro, gara
tesserato	Persona iscritta alla società che gestisce i tornei	persona tesserata, iscritto

Testo tradotto

Si evidenziano i **concetti principali** e i **termini ad essi correlati**:

Una società che organizza **tornei** di calcio a 5 e a 7 è presente nel territorio toscano con tornei attivi nella città di Firenze e nelle zone limitrofe.

Di questi tornei la società desidera mantenere informazioni relative alle fasi di gioco, alle **squadre** partecipanti, ai **giocatori** che compongono le squadre e alle **partite**.
I tornei sono identificati tramite un **codice** e sono caratterizzati da un **nome del torneo**, dal **numero dell'edizione**, dalla **tipologia di gioco** (calcio a 5 o a 7) e dalla **categoria di genere del torneo** (maschile, femminile o mista).
Ogni torneo può essere suddiviso in più fasi di gioco (tipicamente da 1 a 3), le quali si distinguono per la **modalità** di organizzazione delle partite tra le squadre (a **gironi** o ad **eliminazione diretta**) e che sono caratterizzate dal **nome della fase** e dal **numero di scontri**, rappresentante la quantità di partite tra due squadre nella fase. Alle fasi possono corrispondere uno o più **insiemi di squadre** partecipanti e un gruppo di **giornate** di gioco, che rappresentano i turni della fase.
Gli insiemi di squadre, identificati dalla fase a cui appartengono e dal **nome dell'insieme**, raggruppano le squadre partecipanti a quella fase.
Le giornate di gioco calendarizzano le partite di una fase e impongono su queste i vincoli dettati dalla modalità di organizzazione degli scontri (a gironi o ad eliminazione diretta). Le giornate sono identificate dalla fase a cui appartengono e dal **numero della giornata**.
In particolare è rilevante per ogni partita conoscere la **squadra di casa** e la **squadra ospite**, la **data di gioco**, il **campo di gioco**, l'**arbitro** che dirige la partita e i **gol seganti dalla squadra di casa** e i **gol segnati dalla squadra ospite**.
Per ogni partita la società ha interesse a mantenere le **statistiche** sui giocatori che hanno effettuato azioni rilevanti in quella partita (**gol** fatti, **assist** effettuati, **espulsioni**, **ammonizioni**), mentre le squadre si distinguono per il **nome della squadra**, la **tipologia di calcio** a cui gioca la squadra, il **genere dei giocatori** che la compongono e i **colori** della divisa della squadra.
La società richiede inoltre che, per ogni squadra, sia mantenuta l'informazione sul **campo di casa** il quale è contraddistinto dagli altri campi attraverso l'indicazione del **comune**, della **via** e del **numero civico** in cui è situato. I **campi** possiedono comunque un **nome** del campo e un **numero di telefono** dei gestori del campo.
I **tesserati** alla società si suddividono in due tipologie di tesseramento: i **giocatori** e gli **arbitri** i quali condividono un **numero di tessera** univoco per ogni tesserato.
Di tutti i tesserati si conoscono il **nome** e **cognome** del tesserato, **data di nascita** e **genere** del tesserato, mentre per i giocatori si conosce, qualora siano membri di una squadra che partecipa a qualche torneo, anche la **squadra** con la quale partecipa ed il relativo **numero di maglia**.

Glossario dei termini

Nome	Descrizione	Termini relativi	Collegamenti
Torneo	Manifestazione sportiva organizzata dalla società gestrice	Codice del torneo, nome del torneo, numero dell'edizione, tipologia di gioco, categoria di genere del torneo	Fase
Fase	Parte del torneo che determina la modalità di organizzazione delle partite	Nome della fase, numero di scontri	Insieme di squadre, Giornate
Insieme di squadre	Gruppo di squadre che si affrontano nelle giornate della fase	Nome dell'insieme	Fase, Squadra
Giornata	Insieme di partite fra squadre della stessa fase	Numero della giornata	Fase, Partita
Partita	Evento calcistico nel quale si affrontano due squadre	Data di gioco (data e orario), Punteggio (gol casa, gol ospite)	Giornata, Squadra casa, Squadra ospite, Arbitro, Campo
Squadra	Insieme di giocatori che unitamente possono partecipare ad un torneo	Nome, tipologia di calcio della squadra, genere dei giocatori della squadra, campo di casa, colori	Insieme di squadre, Partita, Giocatore, Campo
Tesserato	Persona iscritta alla società che gestisce i tornei	Nome del tesserato, cognome, data di nascita, genere, numero di tessera	Squadra, Partita, Giocatore, Arbitro
Campo	Luogo in cui si disputa una partita	Nome del campo, indirizzo (comune, via, numero civico), recapito telefonico	Partita, Squadra

Modello Entità-Relazione

Si mostra alla pagina seguente il modello entità-relazione derivante dalla richiesta e dall'analisi svolta. Per la costruzione dello schema concettuale è stata adottata una strategia principalmente a inside-out, ma complessivamente mista, partendo dalla raffinazione dello schema Torneo (strategia bottom-up), procedendo successivamente a macchia d'olio verso Fase, Insieme di squadre e Squadre, a cui sono stasti associati successivamente gli attributi (top-down parziale), dopodiché è stato definito il concetto di giornata e quello di partita, il concetto di campo ed infine è stato creato, raffinato e collegato il concetto di tesserato come generalizzazione dei concetti Giocatore e Arbitro.

Elementi facoltativi La tabella a pagina 7 mostra gli elementi facoltativi implementati seguiti da una breve descrizione dell'implementazione.

#	Elemento facoltativo	Realizzato (SI/NO)	Descrizione dell'implementazione con indicazione del metodo/i principale/i
1	Ad ogni accelerazione, c'è una probabilità di 10^{-5} che l'acceleratore fallisca. In tal caso, il componente *throttle control* invia un segnale alla *central-ECU* per evidenziare tale evento, e la central-ECU avvia la procedura di *ARRESTO*	SI	Metodo: *throttle-control* $\rightarrow$ *throttle_failed()*. Il metodo, tramite la funzione `rand()` della `libc` ottiene un numero aleatorio il cui modulo per 10000 simula una probabilità del 1 su 10^5 se eguagliato a 0[a]
2	Componente *forward facing radar*	SI	Sorgente *bytes-sensors.c*. Il processo esegue un ciclo infinito di letture, invii e scritture su file di log.
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	Nella *central-ECU* vengono segnalati con un `SIGKILL` tutti i processi attuatori e sensori esistenti prima di procedere all'inizializzazione del parcheggio.
4	Il componente *park assist* non è generato all'avvio del sistema, ma creato dalla *central-ECU* al bisogno	SI	La *central-ECU* esegue la `fork` per la creazione di *park-assist* in *park_assist_init()*, il quale inizializza il processo prima di entrare nel ciclo di parcheggio.
5	Se il componente *surround view cameras* è implementato, *park assist* trasmette a *central-ECU* anche i byte ricevuti da *surround view cameras*	SI	Nel ciclo principale di *park-assit* si esegue una `read` da *cameras.pipe* (non bloccante) e una `write` dei dati ricevuti sulla socket *assist.sock*
6	Componente *surround view cameras*	SI	Sorgente *bytes-sensors.c*. Vedi facoltativo 2 - *forward facing radar*
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 avviare le procedure di parcheggio	SI	Nella *central-ECU*, nel ciclo principale, dopo la lettura da hmi-input, *kill(-processes_groups.actuators_group, SIGKILL)*. Si esegue la segnalazione di chiusura immediata dei processi *steer* e *throttle* (brake rimane per il ciclo di frenata).
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	Nella *central-ECU*, una volta ricevuto il segnale di *fallimento accelerazione*, gestito tramite *ECU_signal_handler* si esegue la procedura di arresto, stampa la stringa di terminazione e si conclude l'esecuzione dei processi.

[a] La probabilità non risulta esattamente 10^{-5} dato l'intervallo di valori producibili da `rand()`, ma la differenza risulta trascurabile ai fini del progetto.

Descrizione architettura sistema

Nella seguente sezione viene presentata l'architettura del progetto e le scelte implementative prese, cercando di descrivere i motivi che hanno portato alle singole decisioni prese.

Implementazione componenti Segue una tabella nella quale si mostrano quali sorgenti implementano le componenti

Componente	Sorgente
Human-Machine Interface	hmi-input.c hmi-output.c
steer-by-wire	steer-by-wire.c
throttle control	throttle-control.c
brake-by-wire	brake-by-wire.c
front windshield camera	windshield-camera.c
forward facing radar	bytes-sensors.c
park assist	park-assist.c
surround view cameras	bytes-sensors.c
central-ECU	central-ECU.c

Gerarchia del programma La gerarchia del progetto è stata ottenuta dalla richiesta cercando di massimizzare la semplicità. Questa rappresenta lo scheletro del progetto e definisce quindi il flusso di lavoro dello stesso. In particolare si può notare in figura **??** che esistono due soli processi che inizializzano dei figli, ovvero l'unita di controllo centrale *central-ECU*, che rappresenta anche il programma genitore di tutto il sistema, e *park-assist*, che inizializza il suo unico figlio *surround-view cameras*.
È stato scelto di rendere l'unità di controllo la componente genitore di tutto il sistema dato che è in collegamento con la quasi totalità dei processi agenti, cosicché il sistema fosse inizializzato e gestito nel flusso di questa. La soluzione sembrava essere coerente con la strutturazione dei sistemi ADAS reali.
Nella *central-ECU* vengono quindi inizializzati le pipe, vengono eseguite varie `fork` per la creazione e la connessione in lettura/scrittura ai pipe degli attuatori, delle due componenti relative alla *Human-Machine Interface* ed infine i due sensori *front windshield camera* e *forward facing radar*. L'unico componente figlio della *central-ECU* non immediatamente inizializzato rimane *park-assist*[1] il quale verrà inizializzato (comprensivo della socket che utilizzerà per comunicare con la *central-ECU*) e messo in esecuzione dalla stessa unità di controllo quando questa riceverà un comando di *PARCHEGGIO* dall'interfaccia oppure dal sensore *windshield camera*.
Con le stesse motivazioni è stato deciso di rendere la *central-ECU* "server" nella connessione socket tra questa e *park assist*. In questo modo sarà sempre la central-ECU a gestire il flusso di lavoro e a dettare i tempi del sistema. All'avvio della procedura di parcheggio, successivamente al ciclo di rallentamento, la *central-ECU* avvierà la componente *park-assist*, la quale si connetterà alla socket e inizializzerà *surround-view cameras*. Terminata l'inizializzazione, l'unita di controllo invierà un messaggio ("INIZIO") tramite la socket a *park assist*, la quale leggerà per 30 iterazioni (intervallate da `sleep(1)`) i dati dalla sorgente binaria e i dati ricevuti dalla componente figlia tramite la pipe di comunicazione e li invierà alla *central-ECU* che li scandirà alla ricerca di particolari pattern[2]. Se non venissero evidenziate congruenze allora la central-ECU informerà tramite un messaggio ("CONTINUA") *park assist* perché possa

[1] La decisione di non inizializzare il componente immediatamente deriva dalla richiesta, ovvero dall'elemento facoltativo numero 4. Vedi la tabella di pagina 7.

[2] Per conoscere i pattern binari (espressi in codifica esadecimale) prendere visione del paragrafo 2, sotto-paragrafo *Componente central-ECU* della richiesta *Allegato_01.pdf*

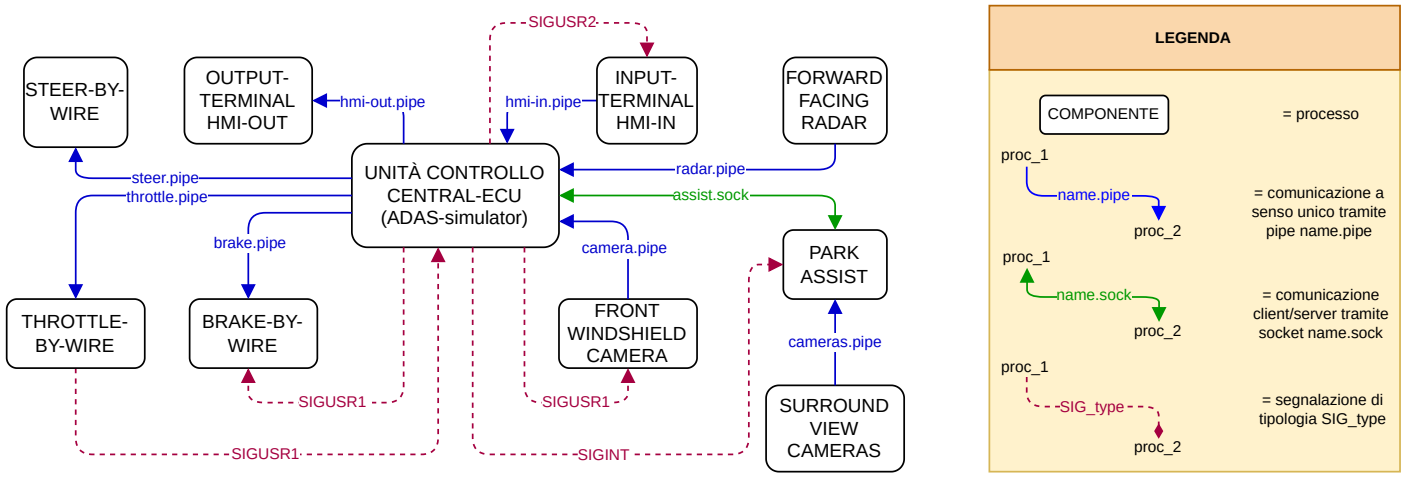

Figura 1: In figura sono rappresentati i percorsi comunicativi inseriti nel sistema: sono presenti 8 pipe, 1 socket e 12 di segnalazioni, 5 rappresentate e 7 assenti. Sono infatti stati esclusi i percorsi delle segnalazioni `SIGKILL` per alleggerire il grafico.

proseguire nell'iterazione successiva. Se così non fosse allora verrà posto sulla socket un messaggio di riavvio ("RIAVVIO"), affinché il parcheggio inizi nuovamente il ciclo delle 30 iterazioni. Concluse le 30 iterazioni senza "insuccessi" allora l'esecuzione del parcheggio risulterà conclusa e il processo terminerà.

IPC nel sistema In figura 1, a pagina 9, si mostra una schematizzazione molto stilizzata della rete comunicativa del sistema. La struttura di comunicazione maggiormente sfruttata all'interno del sistema è la **pipe** la quale implementa 8 canali di comunicazione su 9 (segnali esclusi). Il motivo della scelta delle **pipe** a discapito di altri metodi risiede nel fatto che le comunicazioni sono sostanzialmente unidirezionali (sensori $\rightarrow$ unità di controllo, unità di controllo $\rightarrow$ attuatori, con le due evidenti eccezioni di *park assist* e di *surround-view cameras*). La scelta ha permesso di implementare un sistema relativamente semplice, con un'unica **socket**, struttura più complessa da implementare.
Il protocollo di gestione dei canali risulta unico per tutte le tipologie di canali e per tutti i processi: il processo padre, l'unico processo che comunica con i propri processi figli, produce ed inizializza correttamente il file `.pipe` o `.sock` (rappresentante socket `UNIX`) nella directory *tmp* (dopo aver avuto l'accortezza di eliminare creazioni pendenti da vecchie esecuzioni tramite un `unlink`), il processo figlio si connette con l'adeguata procedura, differente tra pipe e socket, al canale di comunicazione nella fase inizializzazione del processo. Se la connessione non dovesse riuscire (se la `openat` o la `connect` dovessero generare un errore) il figlio terminerebbe con codice di errore `EXIT_FAILURE` e stamperebbe il codice tramite `perror(3)` nel file di log adibito a *stderr*: `./log/errors.log`
Per l'implementazione del parcheggio il canale di comunicazione è stato strutturato sotto forma di una socket per semplificare la comunicazione tra i due processi e perché la gestione del contenuto della socket fosse meno soggetto a problemi di concorrenza. Ogni volta che la *central-ECU* riceve dalla socket uno dei pattern non ammissibili, che simulano una situazione di parcheggio non accettabile, il protocollo prevede che invii un messaggio di riavvio a *park assist*. Il protocollo prevede che ad ogni ciclo *park assist* legga dalla pipe non bloccante *cameras.pipe* i byte da inviare alla *central-ECU* e che li inoltri. Se, eseguito l'invio, riceve dalla *central-ECU* un messaggio che indica la necessità di riavvio, allora non fa altro che resettare il contatore di iterazioni e riprendere il ciclo.

Gestione dei log I log rappresentano, nel sistema implementato, la simulazione delle componenti reali, ovvero mostrano le azioni eseguite dalle varie componenti.
Per ogni componente, esclusa la *Human-Machine Interface*, è presente un file di log specifico nel quale vengono scritte le azioni eseguite. Gli inserimenti nel file di log seguono le richieste specificate nella richiesta del progetto *Allegato_1.pdf*. Perché le shell di input e di output risultassero pulite da ogni

eventuale segnalazione diretta allo *stderr* è stato adottato il file `./log/errors.log` come soluzione per inserirvi tutti i messaggi dirette a questo.

Funzioni condivise Durante lo sviluppo del sistema sono risultate utili la definizione di alcune funzioni di utilità generale per il sistema e di alcune macros, anch'esse sfruttate in più contesti. Tutto ciò è stato inserito all'interno del sorgente `service-functions.c`, il cui header risulta quindi `service-functions.h`.

Esempi di esecuzione

Nella seguente sezione sono mostrati 2 esempi di funzionamento: il primo mostra un funzionamento in modalità ARTIFICIALE con comando dato da input "INIZIO" e alcuni "ARRESTO", nel secondo si esegue in modalità NORMALE direttamente il parcheggio con un terminale selezionato. All'interno dello `.zip` è stata inserita una cartella *Esempi di funzionamento* nella quale sono stati inseriti i file di log risultanti dalle esecuzioni presentate.

Esempio: ARTIFICIALE standard Il presente esempio è teso a mostrare un'esecuzione tipica del programma. Il programma è stato eseguito tramite il comando `./ADAS-simulator ARTIFICIALE`. Non appena il programma si è inizializzato è stato inserito il comando "INIZIO". Il programma si è avviato correttamente iniziando ad inserire sul terminale di output i comandi impartiti da parte della *central-ECU* alle rispettive componenti. Sono stati inseriti, durante l'esecuzione, quattro comandi di "ARRESTO". Il programma ha eseguito correttamente la procedura di arresto, bloccando istantaneamente l'auto e ripartendo la corsa istantaneamente. La procedura di parcheggio, ultimo comando da parte della *windshield camera*, è terminata dopo 30 iterazioni (30 secondi).

Esempio: NORMALE terminale personalizzato Il presente esempio è teso a mostrare un'esecuzione del programma relativamente atipica e potenzialmente problematica, un'esecuzione nella quale l'utente inserisce inizialmente una stringa non accettabile, poi il comando "ARRESTO", sebbene il mezzo non sia ancora in movimento ed infine il comando "PARCHEGGIO". Il programma è stato eseguito tramite il comando `./ADAS-simulator NORMALE -term xfce4-terminal`. Non appena il programma si è inizializzato è stato inserito come comando "42", che è risultato non accettabile, poi "ARRESTO", anch'esso non accettabile, ed infine "parcheggio" che ha dato il via alla procedura di parcheggio, terminata con successo dopo un ciclo di iterazioni. È stata infatti implementata una funzione per il confronto tra stringhe case-insensitive per i comandi dalla hmi-input.

Indice