UNIVERSITÀ DEGLI STUDI DI PISA

DIPARTIMENTO DI INGEGNERIA DELL' INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

NEVIO LONGO

PROGETTAZIONE E SVILUPPO DI UN SISTEMA DI VALUTAZIONE DELLO STILE DI GUIDA IN AMBIENTE EMBEDDED E MOBILE

TESI DI LAUREA

Relatore: Chiar.mo Prof. Luciano Lenzini

Correlatore: Chiar.mo Prof. Enzo Mingozzi

Tutors Aziendali: Ing. Luca D'onofrio

Ing. Alessandro Rossi

ANNO ACCADEMICO 2012-2013

Indice generale

Indice acronimi

OBD	On-board diagnostics
RMP	Revolutions per minute
IVI	In-Vehicle Infotainment

Indice delle illustrazioni

Indice delle tabelle

1. INTRODUZIONE

In un'era in cui l'utilizzo intensivo dell'automobile è ormai comune abitudine, progettare un sistema che renda il suo utilizzo sicuro ed efficiente dal punto di vista del consumo del carburante, diventa un problema fortemente sentito ed affrontato in letteratura per gli effetti che ha nella quotidiana vita di un automobilista.

Alla luce della sempre più grande affluenza di dispositivi tecnologici nell'ambito automotive[1], l'aggiunta di un sistema di valutazione dello stile di guida risulta naturalmente integrabile nel computer di bordo, di cui ormai tutte le auto di una certo segmento sono dotati, e costituente una componente ormai indispensabile nell'era tecnologica in cui viviamo.

Il lavoro di tesi sviluppato si prefigge lo scopo di analizzare, mediante opportune tecniche, lo stile di guida differenziando in esso una componente relativa al *Fuel Saving*[2], di seguito utilizzato per indicare il carburante utilizzato e relativo all'aspetto di *Safety Driving*[3], ovvero sicurezza dello stile di guida.

Relativamente allo studio affrontato si pone una maggiore attenzione alla costruzione di uno strumento facilmente espandibile al fine di aggiungere nel tempo componenti che migliorino l'analisi effettuata.

1.1 PROBLEMA

Indipendentemente dalle capacità tecnologiche del veicolo, l'analisi dello stile di guida è quasi sempre rivolto al controllo delle emissioni inquinanti, consumo carburante e condizioni di pericolo, direttamente legate al comportamento del guidatore, nello specifico analizzabile mediante un set di

1 *Automotive*: Scienza che si occupa dello sviluppo di prodotti per mezzi in movimento, quali automobili, motocicli, rimorchi, etc..
2 *Fuel Saving:* insieme di comportamenti per ridurre il consumo del carburante
3 *Safety Driving*: insieme di comportamenti per uno stile di guida sicuro

variabili che ne rappresentano il comportamento.

Nel contesto del lavoro svolto la componente di controllo delle emissioni inquinanti non viene affrontato, non si esclude comunque la possibilità di raccogliere tale informazioni, in quanto come successivamente esposto si pensa come naturale e futura evoluzione del progetto una raccolta on-line di tutta una serie di dati successivamente consultabili dall'utente stesso o terze parti autorizzate.

1.2. VALUTAZIONE DELLO STILE DI GUIDA

Manifestare al guidatore il corretto stile di guida motivandolo a seguire una determinata propensione ad una guida “safe” risulta lo scopo principale, seppur in contrasto spesso con le necessità, e le tempistiche a cui è soggetto il guidatore(per esempio, se il guidatore necessità di arrivare prima in un determinato luogo, esso incrementerà la velocità andando in contrasto con lo scope del “Fuel Saving ” e “Safety Driving”.

2. STATO DELL'ARTE IN DRIVING STYLE EVALUATION

Diversi sono gli approcci in letteratura in merito al *Driving Style Evaluation,* in tutti viene rivolta particolare attenzione al set di attributi che definiscono il set di parametri necessari per la valutazione dello stile di guida.

I parametri principalmente considerati sono la velocità, RPM, accelerazione, frenata(decelerazione sopra una certa soglia), uso marce, posizione delle leve dell'acceleratore e del freno.

Tale set in molti casi viene ampliato e variato a secondo delle esigenze. Kuhler [1] introduce un set di dieci variabili. Tali variabili sono state studiate in laboratorio per analizzare il consumo e le emissioni del carburante. Altri autori come André [2] e Fomunung [3] incrementano il numero di

variabili sostituendone alcune per gli specifici scopi prefissati.

Diverso risulta anche l'approccio valutativo e il modello di classificazione utilizzato qualora si implementi la logica fuzzy[4], ovvero la definizione di alcuni costrutti attraverso i quali definire una scelta, delle reti neurali di Elman[5-6], quale approccio dei lavori di Augustynowicz [7] Augustynowicz and Bartecki[8], l'ultimo dei quali si basa sulla classificazione del guidatore in relazione al tempo di copertura di un determinato tracciato e intensività nell'uso dell'acceleratore secondo una scala: -1(Mild),0(Neutral),1(Active).

Ulteriore metodo di analisi è rappresentato dalla Hierarchical Cluster Analysis(HCA) ed Principal Component Analysis (PCA) [9].

Il Cluster Analysis (CA) è un metodo con apprendimento unsupervised, ovvero non basato sulla conoscenza dell'attributo classe, ed è una metodologia statistica usata per caratterizzare i singoli oggetti nel gruppo con significato comune(omogenei).

Tipicamente questo metodo viene usato quando non è possibile conoscere a priori il numero di gruppi, quindi dinamicamente viene svolta una divisione in gruppi.

L'identificazione dei gruppi si basa sul concetto di minimizzare la differenza tra elementi dello stesso gruppo, massimizzando quella tra oggetti di gruppi diversi.

La Hierarchical Cluster Analysis (HCA) viene definita a partire da punti individuali rappresentanti un cluster (approccio agglomerative) successivamente fusi fino all'ottenimento di un unico cluster basandosi sul Ward’s method[10] e distanza euclidea.

Comparato con altri metodi gerarchici, esso usa un variance approch per valutare la distanza tra cluster, ed è comunemente considerato molto efficiente nella creazione dei cluster.

Algorithm 1 - HCA - basic Hierarchical Clustering Algorithm

Compute the proximity matrix.

repeat

Merge the closest two clusters.

Update proximity matrix.

until only one cluster remains.

Principal Component Analysis è un metodo statistico per organizzare grandi array di dati.

Trasforma un numero di variabili possibilmente correlate in un (piccolo) insieme di variabili non correlate chiamato Principal Component. Il primo principal component rappresenta gran parte della variabilità possibile dei dati, e ogni successivo component al massimo rappresenta la parte di variabilità residua. I principal components sono calcolati dalla matrice delle correlazioni tra le variabili, estraendo autovalori (la quantità di varianza rappresentata da ogni componente) ed il loro peso(come le variabili sono correlate con la componente principale). Questa analisi cerca di tracciare e organizzare queste variabili in un spazio lower-dimension, dove più elementi strettamente correlati sono tracciate vicini l'uno all'altro rispetto ad elementi meno correlati.

Algorithm 2 - PCA - Principal Component Analysis

1: Organize the data as a $m * n$ matrix, where m is the number of measurement types ans n is the number of trials.

2: Substract off the mean for each measurement type or row in the matrix.

3: Calculate the covariance matrix.

4: Calculate the eigenvectors and eigenvalues of the covariance matrix.

5: Rearrange the eigenvectors and eigenvalues in order of decreasing eigenvalue.

Un ulteriore metodo di analisi è basato sul Gaussian mixture model (GMM[11]), strettamente legato al comportamento di guida rispetto all'uso del freno ed acceleratore nel quale viene analizzata la distribuzione delle operazioni sui pedali non elaborate o le caratteristiche dello spettro a seguito di una analisi delle caratteristiche spettrali sui segnali provenienti dai pedali.

L'ultimo approccio relativo alla classificazione della driving style evaluation considerato è relativo alla definizione di alberi decisionali o reti bayesiane, particolare attenzione soprattutto al primo insieme mediante attuazione della classificazione Random Forest[12] sviluppata da Leo Breiman nel 1991.

Tale scelta merita particolare attenzione, come descritto in [13], in quanto rispetto ad altri algoritmi decisionali quali J48 e le Naïve Bayes, presenta caratteristiche di performance e accuratezza migliori.

Prima di relazionare tali algoritmi, poniamo particolare attenzione a cosa rappresenti un albero di decisione.

Gli alberi di decisione costituiscono il modo più semplice di classificare degli "oggetti" in un numero finito di classi. Essi vengono costruiti suddividendo ripetutamente i record in sottoinsiemi omogenei rispetto alla variabile risposta. La suddivisione produce una gerarchia ad albero, dove i sottoinsiemi (di record) vengono chiamati nodi e, quelli finali, foglie.

In particolare, i nodi sono etichettati con il nome degli attributi, gli archi (i rami dell'albero) sono etichettati con i possibili valori dell'attributo sovrastante, mentre le foglie dell'albero sono etichettate con le differenti modalità dell'attributo classe le quali descrivono le classi di appartenenza. Un oggetto è classificato seguendo un percorso lungo l'albero che porti dalla radice a una foglia. I percorsi sono rappresentati dai rami dell'albero che forniscono una serie di regole.

Ritornando al confronto valutativo tra gli algoritmi precedentemente esposti brevemente si può analizzare come nonostante l'algoritmo *J48* solitamente usato nel contesto di modelli decisionali,

presenti ottimi risultati nel contesto della classificazione, soffre di un sensibile consumo di risorse computazionali indotte sul dispositivo ospitante durante la sua esecuzione.

Diversamente *Naïve Bayes* rappresenta tra i tre algoritmi descritti quello di più facile implementazione e seppur molto efficiente nell'uso delle risorse dei dispositivi mobili, presenta i drawbacks di raggiungere un'accuratezza più bassa rispetto agli altri, oltre al fatto che tutti gli attributi sono considerati indipendenti tra loro(affermazione non vera nel contesto della valutazione dello stile di guida).

Random Forest risulta essere il più efficiente tra i tre, oltre a raggiungere una accuratezza maggiore. Un drawback a cui è sottoposto è la tendenza a minimizzare l'error rate di classificazione che tende a privilegiare la predizione di classi maggioritarie rispetto a classi minoritarie, pur mantenendo un livello di superiorità difficilmente colmato dal J48.

Nel contesto dello studio realizzato, un approccio di classificazione legato a metodi decisionali e reti bayesiane, si lega ad una scelta progettuale di sfruttare il framework Weka[14], mediante il quale si rende non più necessaria una strumentazione hardware avanzata, ma piuttosto le comuni capacità di un sistema di elaborazione presente in un automobile o dispositivo mobile al quale vengono aggiuntive e le caratteristiche ormai comuni, quali sensori GPS, accelerometri, giroscopi, connessione dati per l'aggregazione dei dati rilevati con dati contestuali al mondo esterno.

Tale approccio basato su alberi decisionali è stato preferito ai metodi precedentemente esposti perché più indirizzato verso lo scopo di definire un prodotto che note a priori le classi definite, permettesse integrando la logica su un dispositivo mobile di realizzare un classificatore con un carico computazionale ridotto, caratteristica molto importante nel mondo mobile.

Inoltre per quanto detto si è scelto un classificatore Random Forest per motivi di performance precedentemente esposti , successivamente verificati e tratti da [14].

	Time (seconds)	Correctly Classified Instances (%)
Random Forest	8,22	83,1731
J48	7,156	71,1538
Naïve Bayes	3,457	66,8269

Illustrazione 1: classificazione Artemisia dataset Sonar

	Time (seconds)	Correctly Classified Instances (%)
Random Forest	41,519	97,2
J48	20,608	81,0667
Naïve Bayes	14,070	95,2

Illustrazione 2: classificazione Artemisia dataset Segment

	Time (seconds)	Correctly Classified Instances (%)
Random Forest	24,3531	92,9722
J48	78,84	90,776
Naïve Bayes	3,672	92,6794

Illustrazione 3: classificazione Artemisia dataset Soybean

	Time (seconds)	Correctly Classified Instances (%)
Random Forest	257	94,9359
J48	299	92,6103
Naïve Bayes	48,4111	79,6131

Illustrazione 4: classificazione Artemisia dataset Spambase

rispettivamente tratte da prove sperimentali su determinati datasets [15].

	Attributes	Instances
Sonar	61	208
Segment	20	1500
SoyBean	36	683
Spambase	58	4601

Illustrazione 5: datasets Artemisia

3. CLASSIFICAZIONE

Il problema della classificazione di uno stile di guida si fonda sulla capacità di costruire un modello conforme al fenomeno analizzato, in grado non solo di rappresentarlo in maniera completa e accurata, ma soprattutto nel contesto applicativo di interesse di associare a tutte le rilevazioni una ben nota classe di appartenenza del dominio descritto.

La definizione delle possibili classi di appartenenza rappresenta una fase preliminare dello studio del contesto applicativo del sistema da analizzare nel quale viene individuato un attributo target detto anche "classe" che rappresenta la caratteristica di base del modello che descrive pienamente la così detta classe di appartenenza, ed una serie di attributi secondari che influenzano il valore dell'attributo classe.

Nel contesto di classificazione, risulta necessario in prima analisi, individuare tutta una serie di caratteristiche descriventi il sistema da analizzare, istruire il modello di classificazione mediante informazioni completamente descritte e ben note, ovvero un Training set, e successivamente valutare l'accuratezza del modello descritto mediante l'analisi dei risultati ottenuti.

In tal modo mediante un confronto tra più modelli descrittivi è possibile verificare quale meglio descriva la realtà dei fenomeni del dominio di analisi, e capace quindi di meglio dedurre la classificazione di nuovi fenomeni.

3.1 ATTRIBUTI DEL MODELLO

Le caratteristiche che descrivono il dominio possono rappresentare valori discreti o continui a seconda se siano o meno limitati entro un set di valori possibili, o se sia possibile definire una relazione di distanza ed ordinarietà tra gli elementi.

Analogamente anche l'attributo "classe" può essere un valore continuo o discreto, e in relazione a

questa sua caratteristica lo studio di classificazione prenderà il nome di Classificazione se rappresenta un valore discreto oppure Regressione se continuo.

Vige una certa relazione Y=c(X) tra il set di attributi "X" e l'attributo "Y" di classe, ed il modello di classificazione di prefigge proprio lo scopo di trovare la miglior rappresentazione h() come ipotesi di c().

La fase di valutazione del classificatore può essere sviluppata in diversi modi tra i quali:

Training set: il classificatore è valutato in base alla correttezza della predizione effettuata sulle istanze usate per produrre il modello;

Test set supplementare: il classificatore è valutato grazie a un test set supplementare appositamente caricato da file;

Cross-validation: i record vengono suddivisi in un certo numero di folds . Ogni fold "a turno" funge da test set (la parte restante dei dati funge da Training set) e infine si calcola il mean square error.

Percentage split o Holdout: il classificatore è valutato in base alla correttezza della predizione effettuata su una % di dati tenuti da parte appositamente per il testing, prelevati casualmente dal Training set.

3.2 CARATTERISTICHE DEL MODELLO

Il modello da definire deve presentare una serie di caratteristiche che ne definiscono la completezza ed affidabilità, ovvero:

capacità descrittiva: misura dell'interpretabilità della rappresentazione del modello e la conoscenza sui dati che esprime.
Tale caratteristica decresce all'aumentare della dimensione del modello (es: numero dei nodi dell'albero, numero regole, etc...).

capacità predittiva: *accuratezza* nella predizione della classe degli esempi(percentuale di esempi la cui classe predetta coincide con la classe reale).

errore di classificazione (1-accuratezza) ovvero la percentuale di esempi la cui classe predetta non coincide con la classe reale;

È possibile definire tre tipi di accuratezza in relazione al set a cui viene applicato e in relazione alla definizione del set c(x) e h(x) precedentemente definiti:

True Error(ideale) : misura il numero di volte in cui c(x)≠h(x) sull'intera popolazione X;

Rappresenta la probabilità di errare la classificazione di un elemento random.

$$err(h) = P[c(x) \neq h(x)]$$

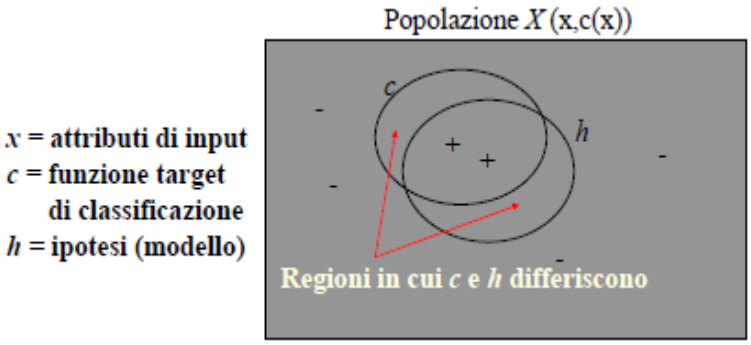


Illustrazione 6: Accuratezza Classificazione

Training Error : misura il numero di volte in cui c(x)≠h(x) nel training set;

Sample Error : misura il numero di volte in cui c(x)≠h(x) su un test set indipendente;

3.2.1 MATRICE DI CONFUSIONE E KAPPA DI COHEN

La **matrice di confusione** rappresenta uno strumento tabellare in grado di far emergere il grado di affidabilità del modello prescelto , presentando l'insieme di predizioni corrette o meno.

Name	Gender	Height	Actual	Pred.
Kristina	F	1.6m	Short	Medium
Jim	M	2m	Tall	Medium
Maggie	F	1.9m	Medium	Tall
Martha	F	1.88m	Medium	Tall
Stephanie	F	1.7m	Short	Medium
Bob	M	1.85m	Medium	Medium
Kathy	F	1.6m	Short	Medium
Dave	M	1.7m	Short	Medium
Worth	M	2.2m	Tall	Tall
Steven	M	2.1m	Tall	Tall
Debbie	F	1.8m	Medium	Medium
Todd	M	1.95m	Medium	Medium
Kim	F	1.9m	Medium	Tall
Amy	F	1.8m	Medium	Medium
Wynette	F	1.75m	Medium	Medium

- colonne: classi predette
- righe: classi effettive
- cella (**i**,**j**): numero di istanze di classe **i** che il modello assegna alla classe **j**
- Le predizioni corrette sono sulla diagonale, mentre quelle scorrette (*misclassificazioni*) sono fuori dalla diagonale

Actual Membership	Assignment Short	Medium	Tall
Short	0	4	0
Medium	0	5	3
Tall	0	1	2

Illustrazione 7: Matrice di confusione

definiremo a questo punto nel contesto della classificazione rispetto ad una classe:

True Positive(TP) →elementi di classe C classificati correttamente come elementi di C

True Negative(TN) →elementi di classe diversa da C non classificati come elementi di C

False Positive(FP) →elementi di una classe diversa da C classificati come elementi di C(ERR)

False Negative(FN) →elementi di classe C assegnati a classi diverse da C (ERR)

Actual Membership	Assignment Short	Medium	Tall
Short	0	4	0 (FP)
Medium	0	5	3 (FP)
Tall	0 (FN)	1 (FN)	2 (TP)

Illustrazione 8: Matrice di confusione e sue componenti

Tali indicatori di correttezza sono contenuti in forma tabellare nella Matrice di confusione.

Un ottimo indicatore di accuratezza del modello usato è il **Kappa di Cohen,** il quale è un coefficiente statistico che rappresenta il grado di accuratezza e affidabilità in una classificazione statistica effettuata da due giudici; è un indice di concordanza calcolato in base al rapporto tra l'accordo in eccesso e l'accordo massimo ottenibile.

Questo valore deve il suo nome allo scienziato Jacob Cohen. Attraverso la matrice di confusione è possibile valutare questo parametro:

dove  è data dalla somma della prima diagonale della matrice e rappresenta la proporzione dei giudizi concordanti tra i giudici. Mentre  è il prodotto dei totali positivi sommato a quelli negativi e rappresenta la proporzione dei giudizi concordanti casualmente.

Se  , allora la statistica rappresenta il caso ottimo. Infatti  .

Esistono diversi "gradi di concordanza", in base ai quali possiamo definire se Kappa di Cohen è scarso o ottimo:

- se k assume valori compresi tra 0-0,4, allora la concordanza è scarsa;
- se k assume valori compresi tra 0,4-0,6, allora la concordanza è discreta;
- se k assume valori compresi tra 0,6-0,8, la concordanza è buona;
- se k assume valori compresi tra 0,8-1, la concordanza è ottima.

		Valori attuali		
		p	***n***	**totale**
Valori predetti	***p'***	Veri positivi	Falsi positivi	P'
	n'	Falsi negativi	Veri negativi	N'
	totale	P	N	

Tabella 1: Kappa di Cohen in Matrice di confusione

3.3 MODELLI DI CLASSIFICAZIONE

I vari modelli di classificazione differiscono per il formalismo utilizzato per rappresentare la funzione di classificazione.

Alcuni modelli di classificazione sono i seguenti:

Basati su esempi : si basano su un concetto di somiglianza tra i dati del training set e i dati da classificare (es. Nearest neighbor)

Matematici : sfruttano funzioni matematiche attraverso i cui parametri viene definita la classificazione(es. Reti Neurali Artificiali, SVM)

Statistici : basandosi su distribuzioni probabilistiche, viene definita la probabilità di appartenenza alle varie classi(es. Naïve Bayes);

Logici : la funzione di classificazione è espressa mediante condizioni logiche sui valori degli attributi (es. Alberi e Regole di Decisione);

Concentreremo l'attenzione verso un modello di classificazione Logico perché meglio si presta, come largamente dimostrato nello stato dell'arte,e successivamente nel lavoro di tesi, per la modellazione del *Driving Style Evaluation*, in cui i parametri estraibili dalla centralina opportunamente trattati ben si prestano ad un tipo di analisi di condizioni logiche che li relazionano,

seppur viene comunque presa.

Analizzando la struttura dell'albero decisionale, ogni nodo rappresenta un attributo A(n) del set degli attributi del dominio, ed ogni arco uscente da tale nodo ne definisce le relazioni rispetto ad un set di possibili valori di A.

ogni foglia è etichettata con il valore atteso della classe per gli oggetti descritti dal cammino che collega la foglia alla radice.

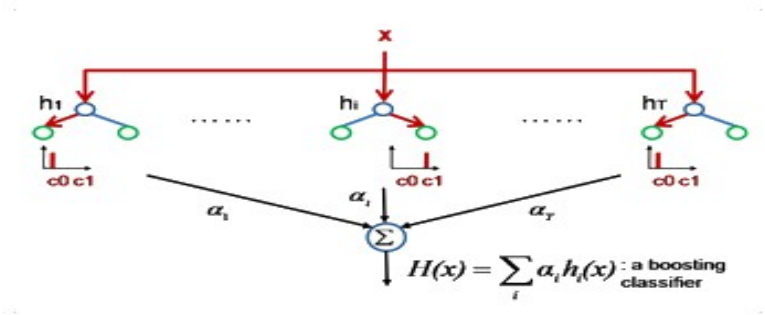


Illustrazione 9: Albero decisionale 1

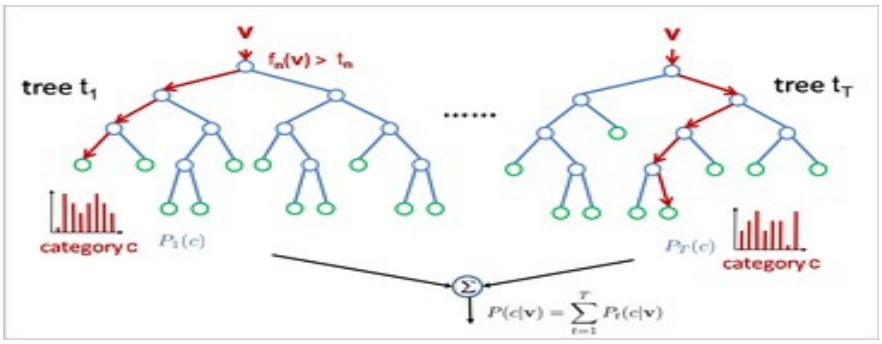


Illustrazione 10: Albero decisionale 1

Un albero decisionale equivale quindi ad una serie di regole logiche mutuamente esclusive (una per ogni foglia) mediante la quale classificare un oggetto.

4. PROGETTAZIONE E MODELLO ARCHITETTURALE

Il progetto si accosta particolarmente al lavoro sviluppato da “Artemisa”, ma non basa il lavoro sul dispositivo mobile, in quanto è vero che accentrando il tutto su un dispositivo mobile si rende il progetto indipendente dall'auto, ma il principale scopo è quello di definire analogamente ai molti lavori citati un sistema di valutazione che sia espandibile sia su una scheda embedded che rappresenterebbe il computer di bordo dell'automobile sia sul dispositivo mobile, mantenendo quanto più possibile l'indipendenza degli stessi .

Tale scelta comunque non preclude la possibilità di sfruttare il dispositivo come componente integrante del sistema in grado si aggiungere, qualora non presenti nella dotazione hardware dell'automobile una serie di informazione quali gps, accelerometro e condizioni meteo della strada molto utili per affinare il modello di analisi dello stile di guida.

I macro-moduli che rappresentano l'architettura generale del sistema sono:

- On_Car
- On_Database
- On_Board
- On_Phone

Che rispettivamente identificano il veicolo dal quale vengono prelevati i dati da analizzare, il sistema di memorizzazione dei dati prelevati e classificati, il sistema caricato sulla scheda di sviluppo e infine il modulo caricato sullo smartphone.

Quest'ultimi due di particolare interesse in quanto definiscono i moduli progettati nel lavoro di tesi effettuato.

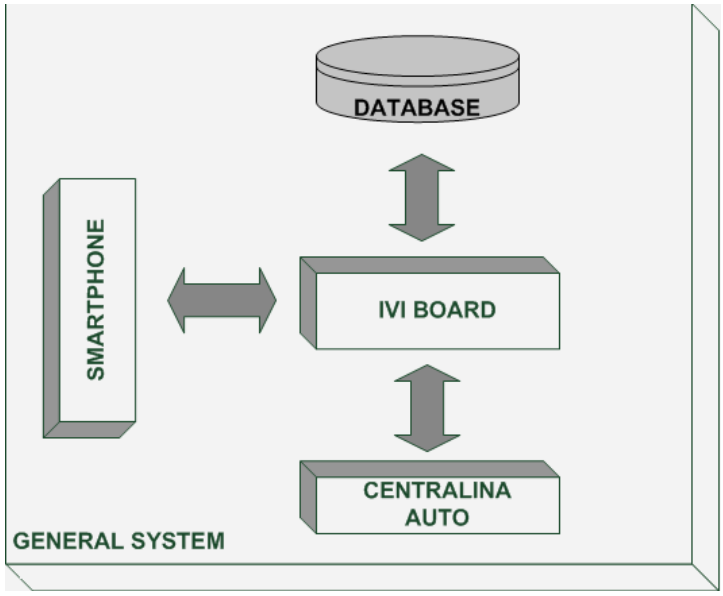


Illustrazione 11: Sistema generale

Ogni modulo rappresenta quindi un'entità del sistema generale necessario per definire l'architettura di un sistema di rilevazione automobilistica e valutazione dello stile di guida.

Ognuno è ben definito e ben strutturato allo scopo di definire quanto più possibile un modulo assestante , ben progettato e di facile manutenzione, mediante uno sviluppo quanto più possibile modulare.

Entrando nello specifico Il sotto-modulo IVI BOARD che rappresenta l'unità contenente le logiche operative di accesso all'interfaccia OBD, e capace di operare tutte le logiche di memorizzazione, trasferimento e classificazione dei dati campionati. È così definito:

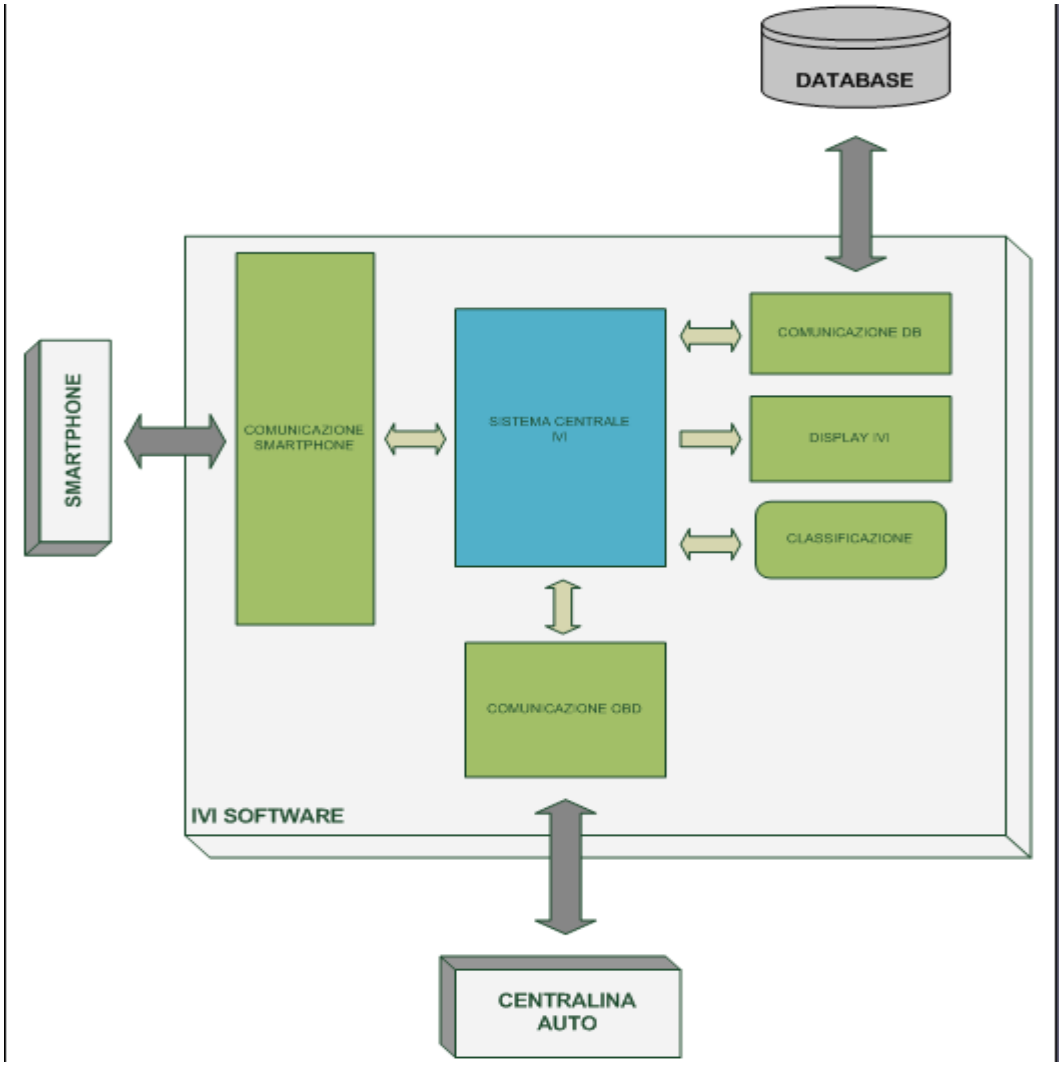


Illustrazione 12: IVI Software

All'interno del modulo è possibile discriminare una componente centrale di management delle operazione , strettamente connesso ai sotto-moduli di interrogazione dell' OBD (COMUNICAZIONE OBD), connessione verso un possibile smartphone (COMUNICAZIONE SMARTPHONE), memorizzazione su database centralizzato (COMUNICAZIONE DB) e i sotto moduli di classificazione e display, rispettivamente CLASSIFICAZIONE e DISPLAY IVI.

Dettagliando ogni sotto-modulo risulta chiaro cosa ogni unità software debba fornire e implementare delle funzionalità richieste, utilizzate da altri sotto-moduli per implementare i propri servizi.

Il sotto-modulo SISTEMA CENTRALE IVI presenta al suo interno una serie di routine mediante le quali vengono definite le logiche di interrogazione dell'interfaccia OBD, memorizzazione persistente dei dati in un DB e connessione con il dispositivo mobile appoggiandosi a moduli specifici che realizzano direttamente quanto analizzato.

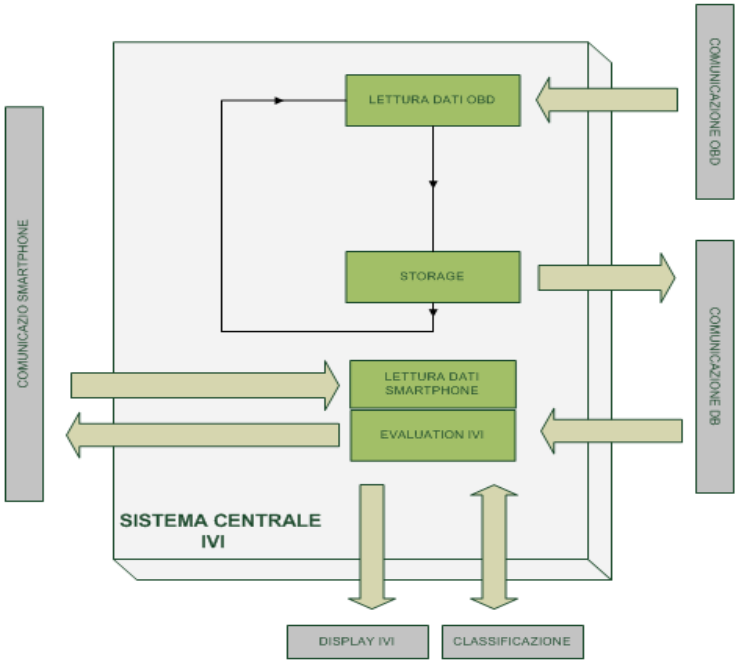


Illustrazione 13: Sistema Centrale IVI

Tali routine si interessano quindi nel definire lo scheletro del sistema per definire le letture dei dati dall'interfaccia OBD mediante il modulo COMUNICAZIONE OBD, definire le logiche di memorizzazione e sincronizzazione delle info da memorizzare o leggere ai fini del trasferimento verso un terminale mobile attraverso i moduli esterni rappresentati da COMUNICAZIONE

SMARTPHONE e COMUNICAZIONE DB,

Il modulo COMUNICAZIONE OBD risulta interessato nelle interazioni con il sistema ON_Car che mediante la libreria e i cavi peak USB-CAN e CAN-OBD2, permettono di interrogare l'automobile per campionare i dati di interesse, integrando nel sotto-modulo MANAGER ELABORAZIONE le logiche di filtraggio dei dati non utili e nel sotto-modulo MANGER OBD-DATA la strutturazione dei dati utili in un formato trasferibile tra moduli.

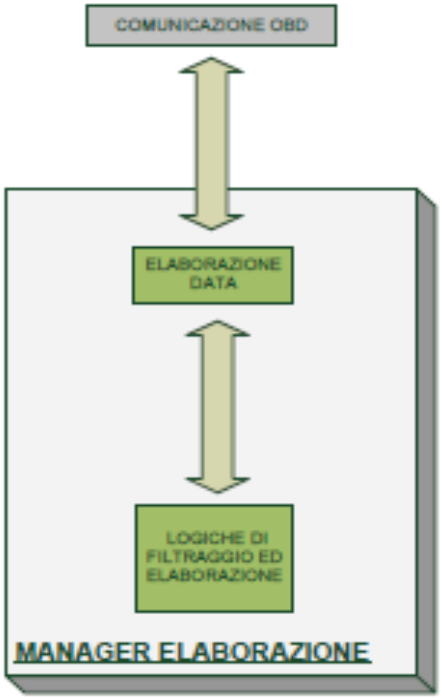


Illustrazione 14: Manager Elaborazione

Punto centrale del lavoro di tesi è lo sviluppo di un modulo che relativamente ai dati letti definisca un logica di classificazione dei dati successivamente utilizzata per produrre dati trasferibili al modulo di display o dispositivi mobili connessi.

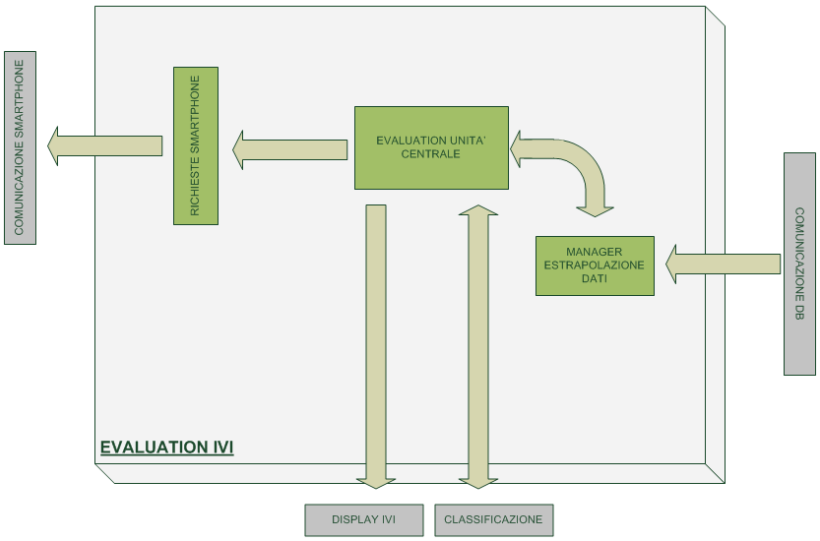


Illustrazione 15: Evaluation Genivi

il modulo di EVALUATION IVI contiene al suo interno sia un modulo legato alla libreria di classificazione contenente il modello di classificazione, sia le specifiche routine di discriminazione logica della classificazione.

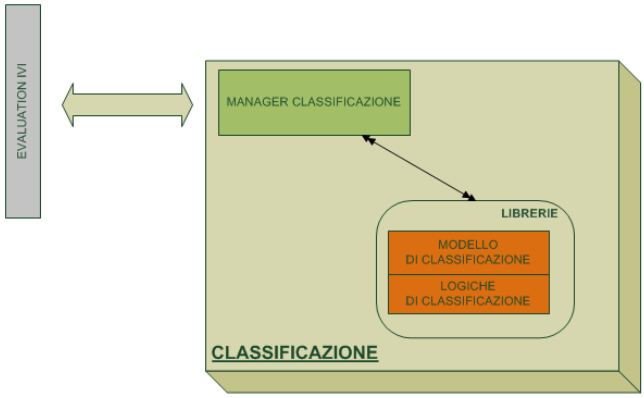


Illustrazione 16: Classificazione

I restanti moduli presenti permettono rispettivamente la memorizzazione dei dati su database, mediante opportune strutture e componenti che rendano l'intero sistema indipendente dai tempi necessari a memorizzare le informazioni, e dal modulo di presentazione dei dati su una interfaccia grafica.

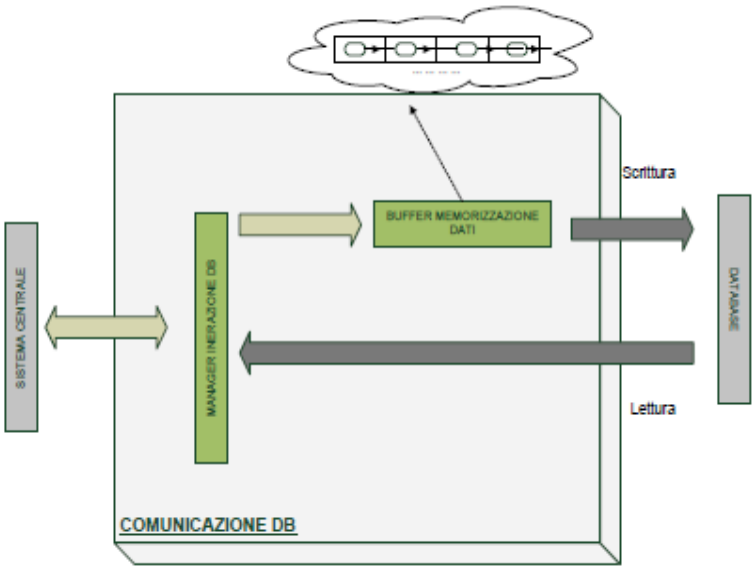


Illustrazione 17: Comunicazione DB

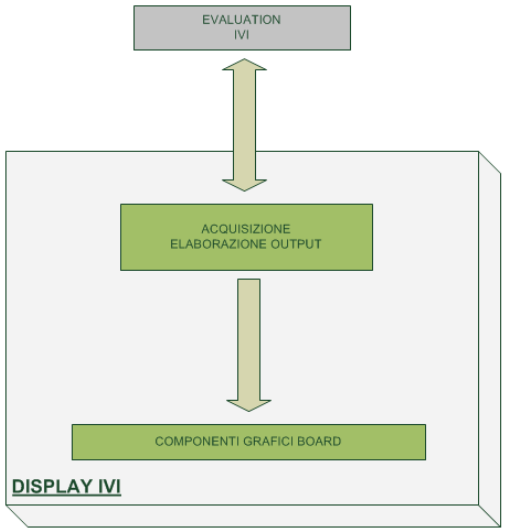


Illustrazione 18: Display IVI

Nell'ottica di poter dialogare con dispositivi mobile quali smartphone, tablet e computer , si correda l'architettura del sistema con un modulo che capace di definire connessioni mediante bluetooth riesca a condividere i dati e la classificazione ricavata con dispositivi esterni.

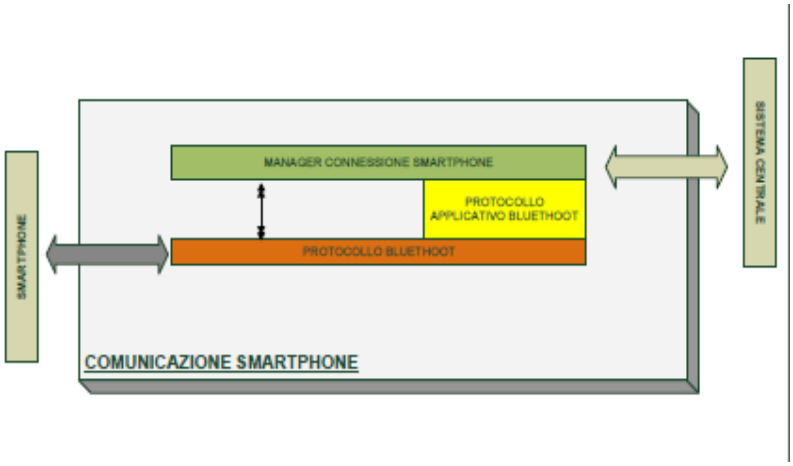


Illustrazione 19: Comunicazione Smartphone

Passando al modello architetturale del SISTEMA SMARTPHONE, parte della logica architetturale definita per l' IVI SYSTEM viene replicata nel modello smartphone quale la strutturazione presente un sistema centrale di coadiuvo dei vari sotto moduli del macro-modulo corrispondente.

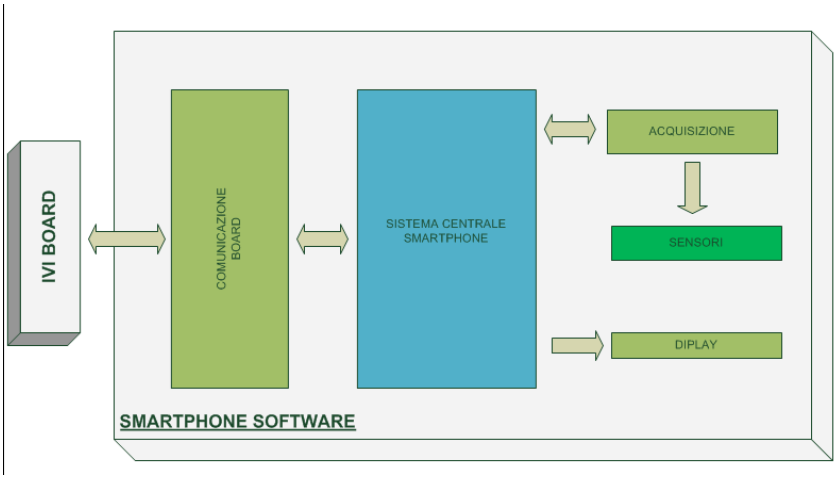


Illustrazione 20: Smartphone Software

Analogamente a quanto descritto per il modello della IVI board è presente un sotto modulo di connessione verso l' IVI system che presenta la caratteristica di definire tutte le logiche di connessione e gestione dati da trasmettere e ricevere.

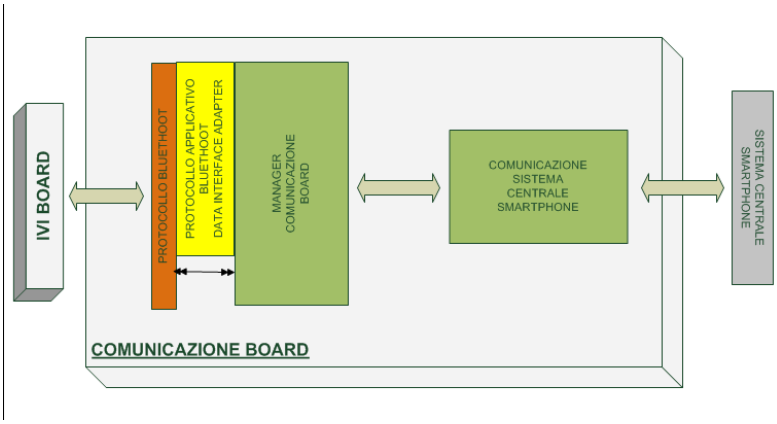


Illustrazione 21: Comunicazione OBD

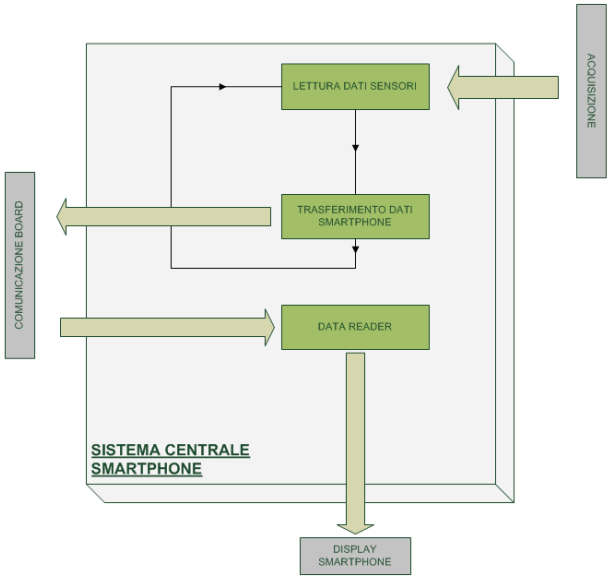


Illustrazione 22: Sistema Centrale Smartphone

Infine un modulo di interfaccia si occupa di tutta la definizione del look-out dei dati su dispositivi mobile.

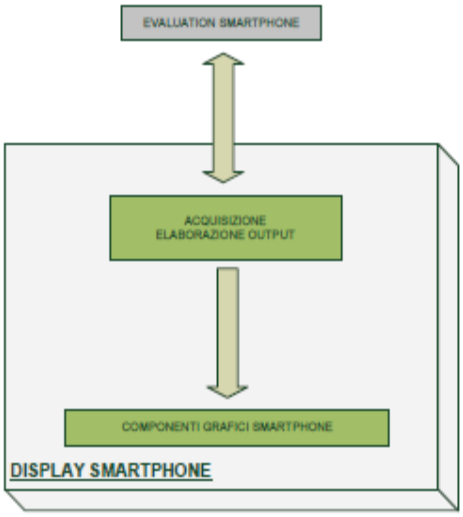
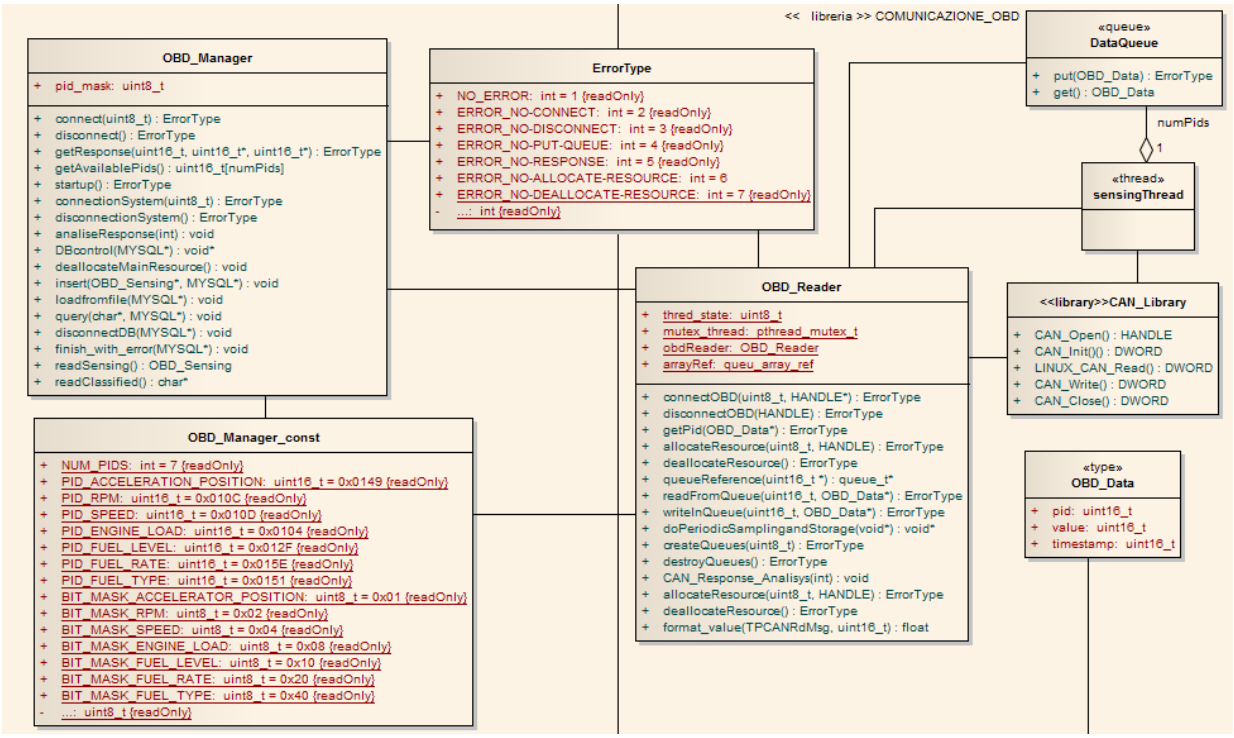


Illustrazione 23: Display Smartphone

Di seguito alcuni approfondimenti sulla definizione UML del modulo On_Board ed alcuni activity e sequence diagram relativi alla connessione, disconnessione e richiesta dati al sistema On_Car.



Dallo snapshoot si evince che sono stati progettati essenzialmente due moduli , uno di interfaccia verso il modulo On_Car, modulo di più basso livello ed uno di interfaccia verso l'esterno e capace di trasferire tra moduli ed esternamente i sensing effettuati, rispettivamente i moduli OBD_Reader e OBD_Manager .

In aggiunta ai suddetti moduli sono presenti una serie di classi aggiuntive atte a definire constanti, strutture dati e rappresentanti librerie esterne utilizzate.

connection activity diagram

Parte cruciale dell'intero sistema è la fase di connessione e richiesta dati all'interfaccia OBD, per questo si è scelto di dettagliare maggiormente questa parte.

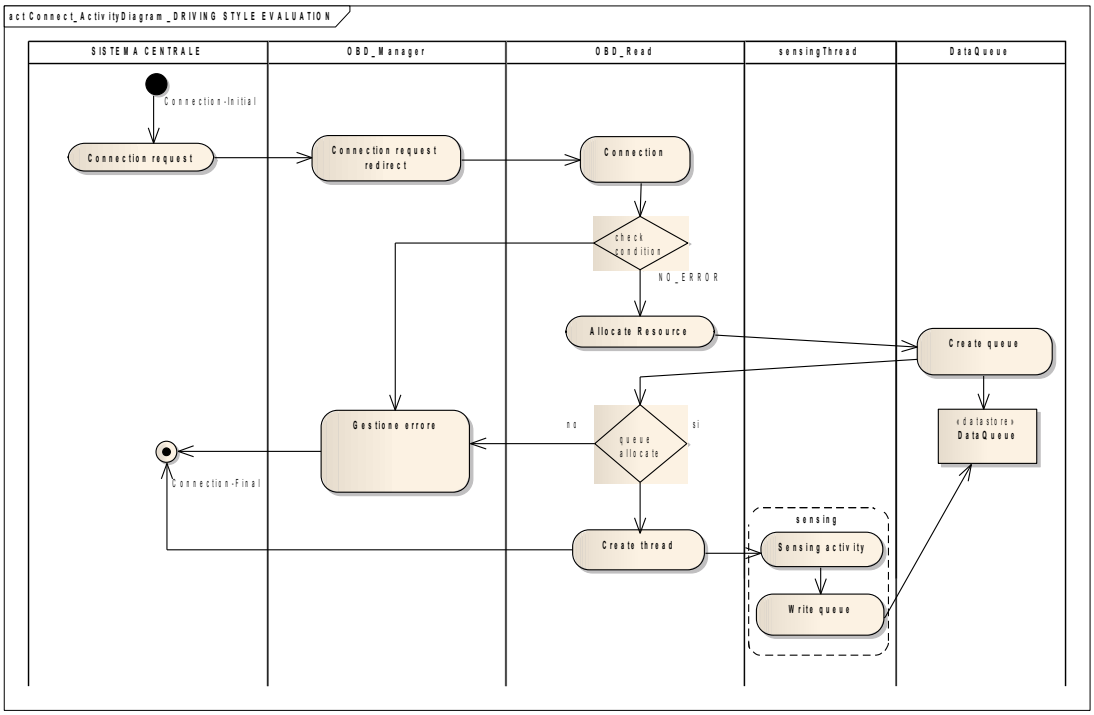


Illustrazione 25: Activity Diagram Connection IVI Software

La fase di connessione viene direttamente effettuata durante l'avvio del sistema e in particolare richiamato mediante la routine *startup()* del modulo `OBD_Manager`, in cui oltre a quanto detto vine anche gestita la connessione verso il database MySQl dedicato alla memorizzazione dei dati da

classificare e classificati.

```
ErrorType startup()
{
	const char* host     = "localhost";       // host
	const char* database = "prova";     // database
	const char* db_user  = "root";      // nome utente
	const char* db_pass  = "root";      // password
	ErrorType e;
	DWORD timestamp,timeReference;
	float value;
	float data[NUM_PIDS];

	uint8_t pids[] = { PID_ACCELERATION_POSITION, PID_RPM,
	                   PID_SPEED,PID_ENGINE_LOAD, PID_FUEL_LEVEL };

	if(queue_init(&db_queue,DIM_QUEUE))//Inizializzazione coda fallita.
	{
		return Error_cannotCreateQueue;
	}
	else
	{
		do
		{
			e=connectionSystem(BIT_MASK_WITH_FUEL_LEVEL);

			if(e!=No_Error)
			{
				printf("errore %d",e);
			}
			usleep(10);
		}
		while(e!=No_Error);

		connInit = mysql_init(NULL);
		if (connInit == NULL)
		{
			printf("errore init\n");
			deallocateMainResource();
		}
		else
		{
			connReal = mysql_real_connect(connInit, /*connection handler*/
			                              host, /* host */
			                              db_user, /* user name */
			                              db_pass, /* password */
			                              database, /* database */
			                              0, /* porta */
			                              NULL, /* socket */
			                              0); /* flags */
			if(connReal==NULL)
			{
				printf("Error %u: %s \t", mysql_errno(connReal),
				                                  mysql_error(connReal));
				printf("ERRORE NEL CONNECT\n");
				deallocateMainResource();
				return Error_MysqlConnection;
```

```
                }
                else
                {
                      th_database =pthread_create(&(gestoreDB),NULL,DBcontrol,
                                                  (void*) connReal);
                      if(th_database!=0)
                      {
                            deallocateMainResource();
                            return Error_cantCreateThread;
                      }
                      int i;
                      .
                      RICHIESTA DATI
                      .
                }
          }
      }
      return No_Error;
}
```

il controllo dell'effettiva connessione viene effettuato mediante le routine *queueInit(...) e connectionSystem()*.

```
bool queue_init(queue_t *queue, uint16_t capacity)
{
      queue->buffer = (queue_node_t *) malloc(sizeof(queue_node_t) * capacity);

      if (queue->buffer == NULL )
            return true;
      queue->capacity = capacity;
      queue->size = 0;
      queue->in = 0;
      queue->out = 0;
      pthread_mutex_init(&(queue->mutex),NULL);
      pthread_cond_init(&(queue->cond_full),NULL);
      pthread_cond_init(&(queue->cond_empty),NULL);

      return false;
}
```

In tale routine vengono create le opportune code richieste (in relazione ai pids di interesse che nella versione attuale ricopre tutti i pids utilizzati nel sistema di valutazione), verificando l'effettiva capacità di allocare le risorse necessarie.

```
ErrorType connectionSystem(uint8_t mask)
{
```

```
        pid_mask=mask;
        ErrorType error=connectOBD(mask, &handle);
        analyseResponse(error);
        return  error;
}
```

con tale routine invece viene re-inoltrata la richiesta di connessione al modulo OBD_Reader direttamente connesso all'interfaccia OBD mediante la libreria peak® per linux.

disconnect activity diagram

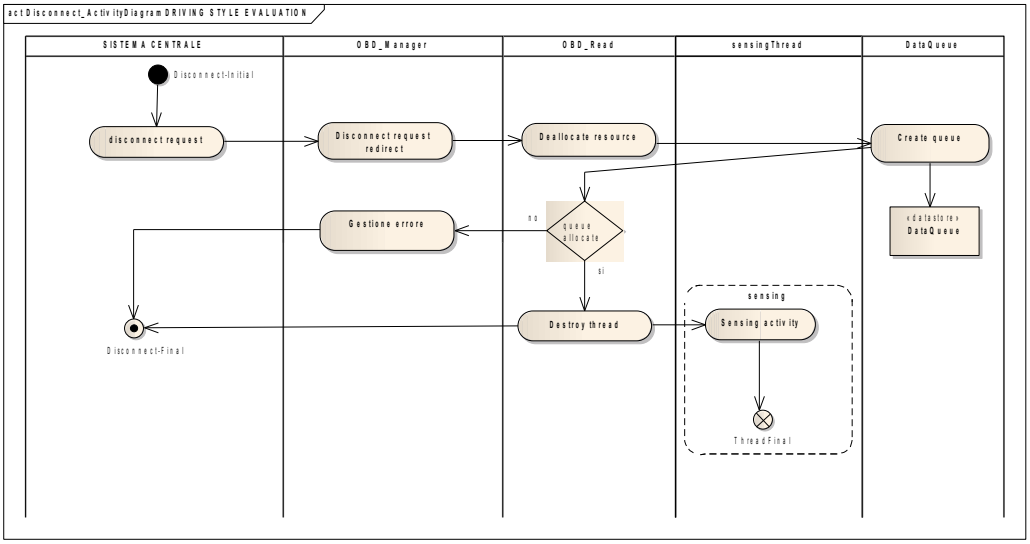


Illustrazione 26: Activity Diagram disconnection IVI Software

Lo scenario di disconnessione rappresenta la fase di fine connessione oppure la situazione in cui lungo l'iter d inizializzazione o esecuzione un problema richiesta la de-allocazione di risorse e chiusura del canale comunicativo.

Quest'ultimo rappresenta lo scenario realmente implementato in quanto nella logica del sistema dopo l'avvenuta connessione non è necessaria una disconnessione se non per un problema legato alla reale perdita di connessione o errore riscontrato.

```
void deallocateMainResource()
{
```

```
	ErrorType e;
	if(connInit==NULL)
	{
		queue_remove(&db_queue);
	}
	else if(connReal==NULL)
	{
		queue_remove(&db_queue);
		disconnectDB(connReal);
	}
	else if(th_database!=0)
	{
		queue_remove(&db_queue);
		disconnectDB(connReal);
	}
	else if (sensing == NULL)
	{
		queue_remove(&db_queue);
		pthread_cancel(gestoreDB);
		disconnectDB(connReal);
	}
	e=disconnectionSystem();
	if(e!=No_Error)
		printf("\nerrore disconnect\n");
	else
		printf("\nclosing canconnection...");
}
```

La routine di de-allocazione delle risorse in seguito ad un errore, verifica l'entità del problema ed in relazione a quest'ultimo rilascia opportunamente tutte le risorse fino a quel momento riservate.

Nello specifico quindi controlla la presenza di una connessione attiva verso l'interfaccia OBD mediante la variabile *connInit,* connessione verso il database mediante la variabile *connReal* e terminati i thread di gestione del database e della lettura dall' OBD richiamando le routine *disconnectDB()* e *disconnectOBD()*.

```
void disconnectDB(MYSQL *conn)
{
	printf("closing Database...\n");
	mysql_close(conn);
}


ErrorType disconnectionSystem()
{
		ErrorType error=disconnectOBD(handle);
		analyseResponse(error);
		return  error;
}
```

mediante tali routine viene chiusa la connessione al database e richiamata la funzione di

OBD_Reader per rilasciare le risorse per la connessione all' OBD rispettivamente.

Response request Activity Diagram

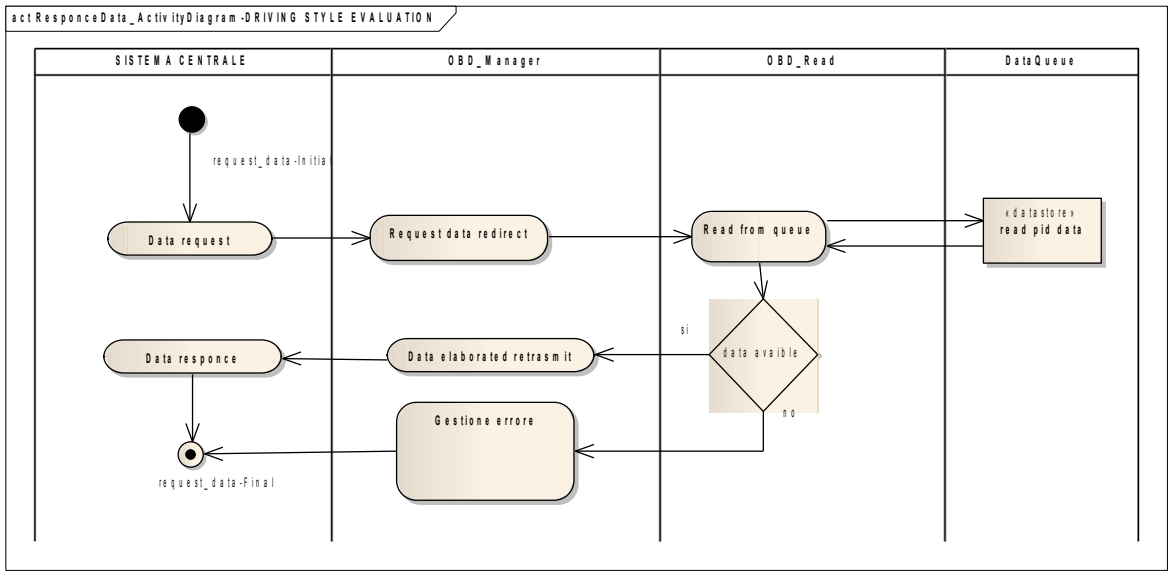


Illustrazione 27: Activity Diagram Response request IVI Software

Come detto in precedenza una volta effettuata la connessione verso l'interfaccia OBD un opportuno thread si occupa di richiedere periodicamente i dati necessari e una volta costruito il dato di interesse, ovvero una struttura dati contenente tutti i dati campionati vengono memorizzati sul database per essere successivamente letti.

La struttura dati definita è la seguente:

```
typedef struct OBD_Sensing
{
	/**
	 * pid's byte
	 */
	float rpm;
	float acc_position;
	float speed;
	float engine_load;
	float fuel_level;
	float fuel_rate;
	DWORD timestamp;

}OBD_Sensing;
```

Tale struttura tiene traccia di tutte le info necessarie alla classificazione, ed è condivisa sia dal

modulo On_Board relativo alla lettura da OBD , sia al modulo di lettura integrato nel modulo Qt di interfaccia utente.

Il seguente codice definisce le logiche di richiesta dati:

```
.
.
.
while(true)
{
	for(i=0;i<NUM_PIDS;i++)
	{
		e=getResponse(pids[i],&value,&timestamp);
		data[i]=value;
		if(i==0)timeReference=timestamp;
		if(e!=No_Error)
		{
			printf("Errore prelevamento dati\n");
		}
		else
		{
			printf("main %x %f %u\n\n",pids[i],data[i],timestamp);
		}
	}
	sensing = (OBD_Sensing*) malloc(sizeof(OBD_Sensing) * 1);
	if (sensing != NULL)
	{
		sensing->rpm = data[RPM_LOC];
		sensing->acc_position = data[ACCELERATOR_POSITION_LOC];
		sensing->speed = data[SPEED_LOC];
		sensing->engine_load = data[ENGINE_LOAD_LOC];
		sensing->fuel_level = data[FUEL_LEVEL_LOC];
		sensing->timestamp =timeReference+((timestamp-
							timeReference)/NUM_PIDS);
		printf("a[%f] r[%f] s[%f] e[%f] f[%f] t[%u]\n",
		sensing->acc_position,
		sensing->rpm,
		sensing->speed,
		sensing->engine_load,
		sensing->fuel_level,
		sensing->timestamp);
		node.message = sensing;
		queue_enqueue(&db_queue, &node);
	}
	else
	{
		printf("errore malloc\n");
		deallocateMainResource();
		return Error_MallocCompromised;
	}
}
```

operando su un ciclo infinito, all'atto dell'avvio del sistema vengono sequenzialmente richiesti tutti i parametri.

Una volta effettuata la completa iterazione viene confezionato un oggetto OBD_Sensing, memorizzato nell'opportuna coda mediante la routine *queue_enque(..)*;

Connect and Setting Sequence Diagram

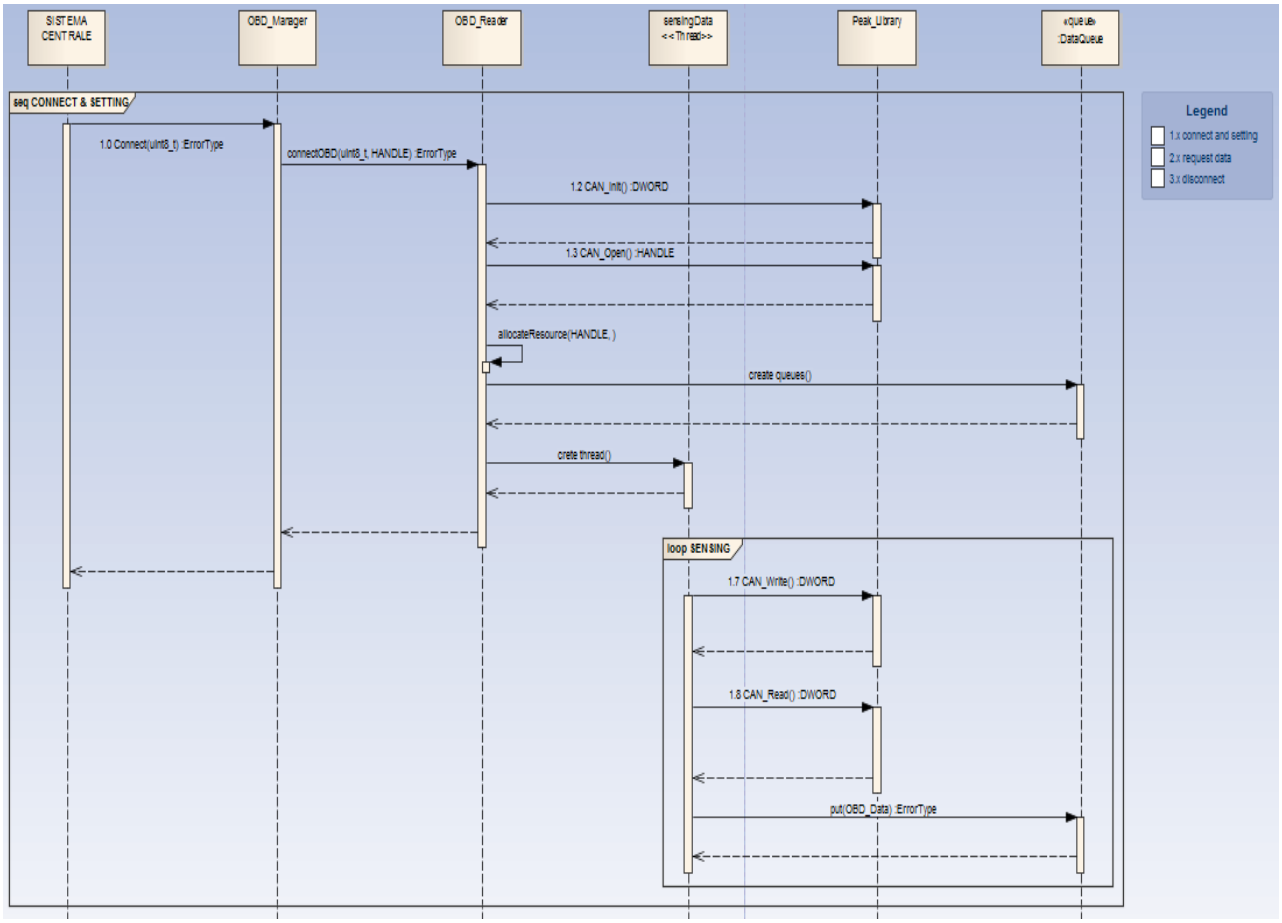


Illustrazione 28: Sequence Diagram Connect and Setting IVI Software

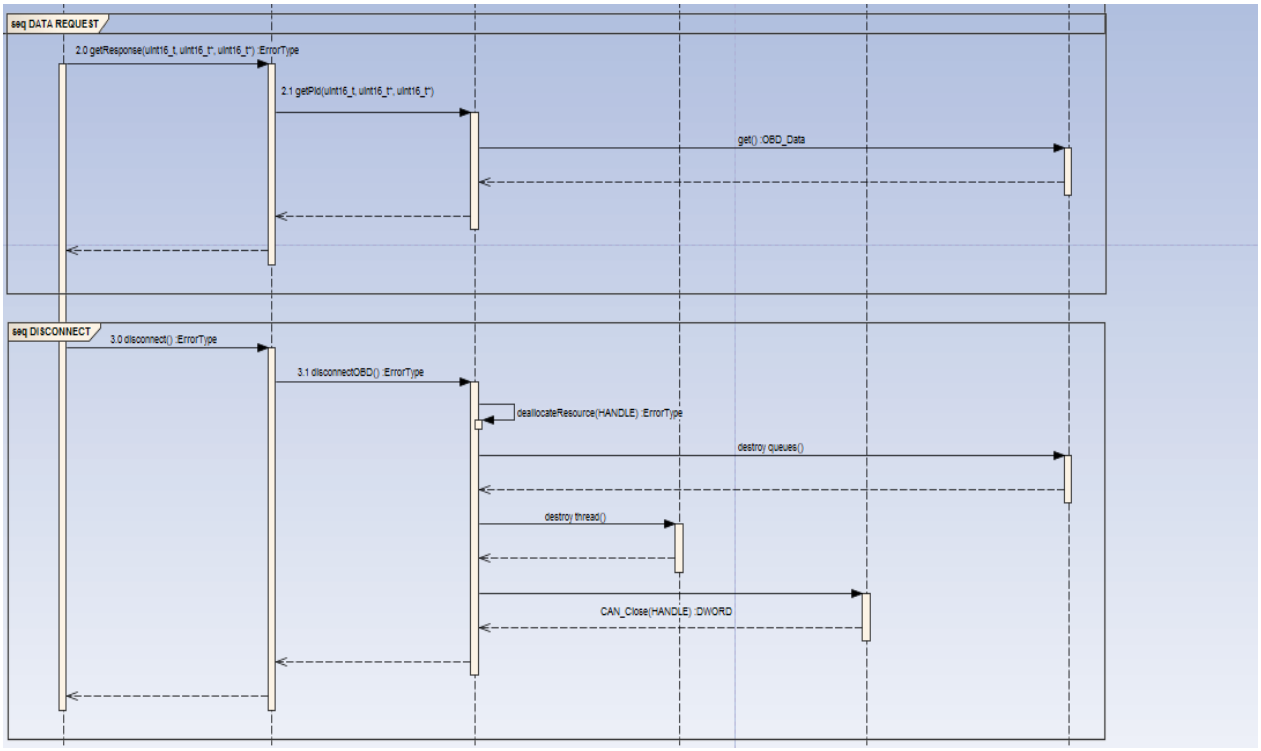


Illustrazione 29: Sequence Diagram Request and DIsconnect IVI Software

4.1 SISTEMA IVI

Lo scenario in background del sistema On_Board è l'integrazione dello stesso nella piattaforma del consorzio Genivi.

GENIVI® è una alleanza industriale senza scopo di lucro creata per lo sviluppo di una serie di strumenti adottati su piattaforma open-source per l'impiego dell' In-Vehicle Infotainment (IVI), ovvero intrattenimento sul veicolo. Ciò comprende sia i sistemi multimediali comuni come impianti audio video sia sistemi di controllo della centralina del veicolo in generale.

L'alleanza mira ad allineare i requisiti, fornire implementazioni di riferimento, offrire programmi di certificazione, per favorire l'incremento di una comunità di sviluppo IVI open source.

Tale lavoro traduce in cicli di sviluppo più brevi, più rapido time-to-market, e la riduzione dei costi

per le aziende che sviluppano apparecchiature e software IVI.

4.2 SISTEMA OBD

La principale fonte dati del progetto sviluppato è rappresentata dalla centralina OBD, mediante la quale vengono prelevate regolarmente le seguenti informazioni:

- RPM, il numero dei giri del motore molto utili nel verificare se l'auto viaggia a bassi o alti regimi, ed in relazione al rapporto usato valutare il consumo di carburante.
- Carico dell'automobile, in relazione ad altri parametri costruiti il carico dell'auto influisce sulla valutazione dello stile di guida e quindi sul consumo.
- Velocità del veicolo.
- Posizione del pedale dell'acceleratore, mediante il quale valutare se il guidatore tende a mettere sotto stress il sistema con frequenti accelerazioni.

Altri invece seppur disponibili in linea generale su un qualunque veicolo non sono stati realmente prelevati dall'auto ma simulati.

- Livello carburante e Consumo Medio e quindi tutta una serie di relazioni sul consumo.
- Tipo carburante, in relazione al tipo di alimentazione risultano diverse alcune soglie di analisi quali gli RPM ottimali(si suppone una alimentazione a benzina).

4.3 PROTOCOLLO CAN ED OBD CONNECTOR

Il protocollo di campo utilizzato è il CAN, le più recenti macchine sono ormai provviste oltre che di una interfaccia OBD2 utile a monitorare tutta una serie di info di diagnostica e controllo, di una vera e propria rete interna che interconnette tutte le centraline operative interne all'auto e dedicate a determinati aspetti di controllo, a cui è possibile accedere direttamente mediante un accesso al Bus CAN, oppure mediante il connettore OBD2 quando provvisto dei pin specifici(CAN_HIGH e CAN_LOW).

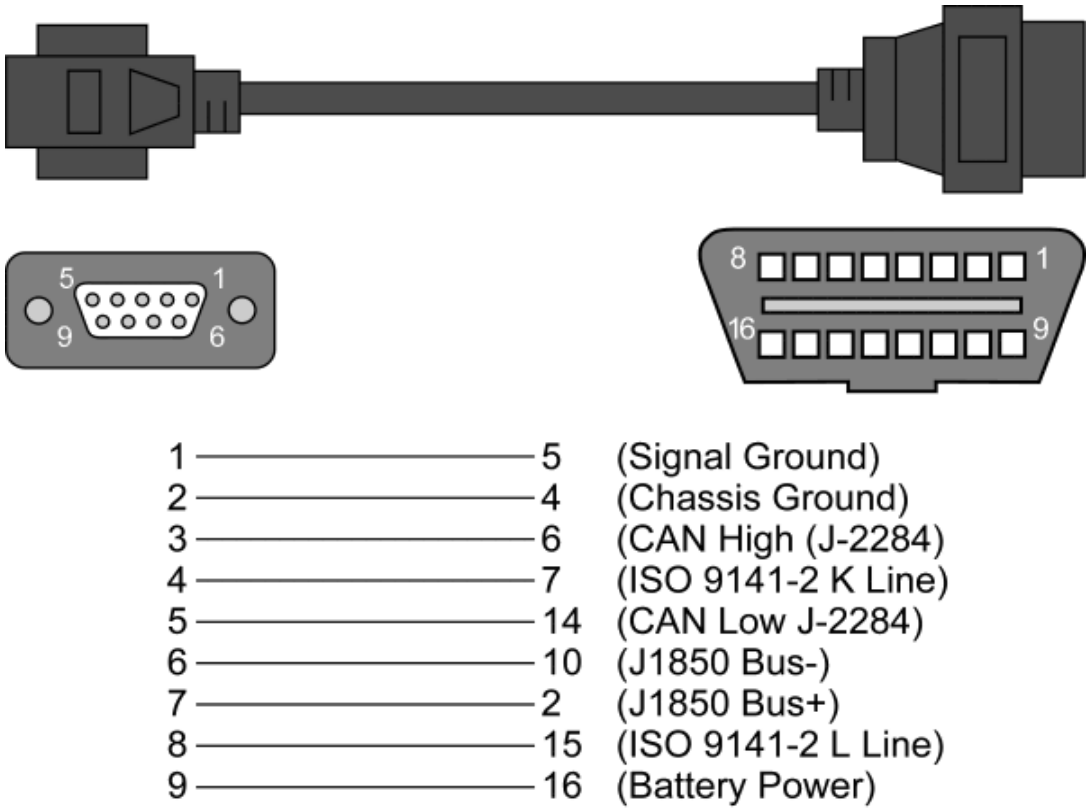


Illustrazione 30: OBD Cable

Mediante l'ausilio di dongle CAN-USB è stato infatti possibile estrarre i valori numerici relativi ai parametri di interesse, successivamente elaborati.

Successiva alla fase di acquisizione nell'auto i dati sono stati memorizzati su una serie di file ed utilizzati per simulare una vera campagna di guida e quindi valutazione del sistema di evaluation creato e della guida sottoposta al sistema stesso di controllo.

4.3.1 CAN PROTOCOL

Il **Controller Area Network**, noto anche come **CAN-BUS**, è uno standard seriale per bus di campo (principalmente in ambiente automotive), di tipo multicast, introdotto negli anni ottanta dalla Robert Bosch GmbH, per collegare diverse unità di controllo elettronico (ECU). Il CAN è stato espressamente progettato per funzionare senza problemi anche in ambienti fortemente disturbati

dalla presenza di onde elettromagnetiche e può utilizzare come mezzo trasmissivo una linea a differenza di potenziale bilanciata come la RS-485. L'immunità ai disturbi EMC può essere ulteriormente aumentata utilizzando cavi di tipo twisted pair (*doppino intrecciato*).

Sebbene inizialmente applicata in ambito automotive, come bus per autoveicoli, attualmente è usata in molte applicazioni industriali di tipo embedded, dove è richiesto un alto livello di immunità ai disturbi. Il bit rate può raggiungere 1 Mbit/s per reti lunghe meno di 40 m. Velocità inferiori consentono di raggiungere distanze maggiori (ad es. 125 kbit/s per 500 m). Il protocollo di comunicazione del CAN è standardizzato come ISO 11898-1 (2003). Questo standard descrive principalmente lo strato (*layer*) di scambio dati (*data link layer*), composto dallo strato sottostante (*sublayer*) "logico" (*Logical Link Control, LLC*) e dallo strato sottostante del *Media Access Control, (MAC)* e da alcuni aspetti dello strato "fisico" (*physical layer*) descritto dal modello ISO/OSI (*ISO/OSI Reference Model*). I protocolli di tutti gli altri layer sono lasciati alla libera scelta del progettista della rete. [16]

Nell'ambito della comunicazione i frame rappresentano l'effettiva unità di trasmissione dei dati. I messaggi possono avere due formati:

- *Base frame format*: con 11 bit di identificazione.
- *Extended frame format*: con 29 bit di identificazione.

Lo standard CAN *deve* obbligatoriamente riconoscere il formato *base frame* e *può* opzionalmente riconoscere il formato *extended frame format* (che, tuttavia, deve essere tollerato).

Il CAN *base* permette 2^{11} = 2048 tipi di messaggi diversi, ma da specifiche Bosch se ne possono usare solo 2031. Nella versione *extended* si possono avere fino a 2^{29} = 536 870 912 tipi di messaggi.

Formato del Base frame

Il formato del base frame ha la seguente struttura:

Nome del campo	Lunghezza (numero di bit)	Funzione
Start-of-frame	1	Indica l'avvio della sequenza di trasmissione
Identificatore	11	Identificatore (unico) dei dati
Richiesta remota di trasmissione (RTR)	1	Deve essere un bit dominante
Bit aggiuntivo di identificazione (IDE)	1	Deve essere un bit dominante
Bit riservato (r0)	1	Riservato
Codice di lunghezza dati (DLC)	4	Numero di byte per codificare ciascun dato (0-8 byte)
Campo dati	0-8 byte	Dati da trasmettere (la lunghezza è specificata dal campo DLC)
CRC	15	Controllo di parità a ridondanza
delimitatore CRC	1	Deve essere un bit recessivo
Slot ACK	1	Il trasmettitore invia un bit recessivo e ogni ricevitore può confermare la ricezione con un bit dominante
Delimitatore ACK	1	Deve essere un bit recessivo
End-of-frame (EOF)	7	Devono essere bit recessivi

Tabella 2: CAN frame Base

Un vincolo imposto dal campo dell'identificatore è che i primi 7 bit non possono essere tutti recessivi.

Formato dell'Extended frame

Il formato dell'Extended Frame ha la seguente struttura:

Nome del campo	Lunghezza (numero di bit)	Funzione
Start-of-frame	1	Indica l'avvio della sequenza di trasmissione
Identificatore A	11	Prima parte dell'identificatore (unico) dei dati
Richiesta remota sostitutiva (SRR)	1	Deve essere un bit recessivo
Bit aggiuntivo di identificazione (IDE)	1	Deve essere un bit recessivo
Identificatore B	18	Seconda parte dell'identificatore (unico) dei dati
Richiesta remota di trasmissione (RTR)	1	Deve essere un bit dominante
Bit riservati (r1 & r0)	2	Riservati
Codice di lunghezza dati (DLC)	4	Numero di byte del dato (0-8 byte)
Campo dati	0-8 byte	Dati da trasmettere (lunghezza specificata dal campo DLC)
CRC	15	Controllo di parità a ridondanza
Delimitatore CRC	1	Deve essere un bit recessivo
Slot ACK	1	Il trasmettitore invia un bit recessivo e ogni ricevitore può confermare la ricezione con un bit dominante
Delimitatore ACK	1	Deve essere un bit recessivo
End-of-frame (EOF)	7	Deve essere un bit recessivo

Tabella 3: CAN frame extended

Tutte le richieste inoltrate vengono quindi incapsulate su di una frame CAN al cui interno viene imposta la seguente struttura:

```
typedef struct
{
  DWORD ID;             // 11/29 bit code standard or extended
  BYTE  MSGTYPE;        // bits of MSGTYPE_*
  BYTE  LEN;            // count of data bytes (0..8)
  BYTE  DATA[8];        // data bytes, up to 8
} TPcanMsg;             // for Pcan_WRITE_MSG
```

nello specifico il campo DATA[8] è così definito secondo lo standard OBD2:

byte 0 → number of byte plus the first (byte 1,2)

byte 1 → code of request or responce (0x01 or 0x41)

byte 2 → pid code

4.3.2 USB-OBD CABLES

I cavi utilizzati nel contesto del lavoro di analisi sono, come già esposto una coppia di dongle usb-CAN e un adattatore CAN-OBD2 (da adesso in generale CAN-OBD) della casa produttrice Peak, di cui sono state utilizzate le libreria di accesso al CAN.

PEAK USB-CAN

I dongle USB-CAN fanno riferimento al modello ipeh-002021 della Peak, mediante tali cavi è possibile inoltrare le richieste CAN dal proprio modulo software, collegato mediante USB.

Illustrazione 31: Peak USB-Dongle

Specifiche

- Adattatore per connessione USB (USB 1.1, e compatibile con USB 2.0)
- alimentazione mediante USB
- Bit rates di 1 Mbit/s
- risoluzione del Timestamp circa 42 µs
- compatibile con le specifiche CAN 2.0A (11-bit ID) and 2.0B (29-bit ID)
- connessione CAN bus via D-Sub, 9-pin (secondo specifica CiA® 102)
- NXP SJA1000 CAN controller, 16 MHz clock frequency
- NXP PCA82C251 CAN transceiver
- 5-Volts supply to the CAN connection CAN be connected through a solder jumper,

  e.g. for external bus converter
- ampio range operativo di temperatura tra -40 to 85 °C (-40 to 185 °F)

opzionalmente:

- isolamento galvanico del CAN connection sopra i 500 V

schema pin D-Sub

Pin	Pin assignment
1	Non connesso / optional +5V
2	CAN-L
3	GND
4	Non connesso
5	Non connesso
6	GND
7	CAN-H
8	Non connesso
9	Non connesso / optional +5V

Tabella 4: CAN pin

PEAK CAN-OBD

Il connettore CAN-OBD permette invece di interfacciare il CAN-BUS con l'interfaccia OBD dell'auto, instaurando una connessione necessaria per il trasferimento dati in entrambe le direzioni, impacchettando le opportune richieste secondo standard OBD in frame CAN.

Illustrazione 33: Peak CAN-OBD cable

Illustrazione 32: Can Pins

Specifiche

- connessione CAN bus via D-Sub, 9-pin (secondo specifica CiA® 102)
- connettore OBD-2 – asegnati per il CAN sui pin:
    - Pin 6: CAN High (J-2284)
    - Pin 14: CAN Low (J-2284)
- Lunghezza 1.0 m
- senza resistore di terminazione
- tutti gli ODB-2 pins sono montati all'atto del plugin e possono essere assegnati quando desiderato

Scopo dell'alimentazione

- Pcan-Cable OBD-2

4.3.2 STRUTTURA MESSAGGI E PID

La modalità con la quale vengono inoltrate le richieste verso l'automobile segue una specifica formattazione, ovvero basandosi sul trasferimento di una frame CAN, permette di incapsulare al suo interno una struttura che risulti conforme allo standard OBD esistente.

Quest'ultimo standard definisce la struttura secondo la quale definire un interrogazione e contiene al suo interno informazioni relative all'operative Code di richiesta, ovvero i dati attualmente in circolo(Code = 01) oppure dati precedentemente congelati per una richiesta successiva al momento di loro creazione (Code = 02) , il pid da richiede, cioè l'identificativo dell'attributo di interesse

```
typedef struct
{
  DWORD ID;             // 11/29 bit code
  BYTE  MSGTYPE;        // bits of MSGTYPE_*
  BYTE  LEN;            // count of data bytes (0..8)
  BYTE  DATA[8];        // data bytes, up to 8
} TPcanMsg;             // for Pcan_WRITE_MSG

typedef struct
{
  TPcanMsg Msg;         // the above message
  DWORD    dwTime;      // a timestamp in msec, read only
  WORD     wUsec;       // remainder in micro-seconds
} TPcanRdMsg;           // for Pcan_READ_MSG
```

ID 4byte	MSGTYPE 1byte	LEN 1byte	DATA 8byte								Dwtime 4byte	WuSEC 2byte
			B0	B1	B2	B3	B4	B5	B6	B7		

Add bytes	Code request	Pid request					

Dove il primo byte identifica i bytes successivi a quello nella frame CAN, seguito dal code precedentemente esposto e dal pid richiesto.

Secondo tale schema ogni richiesta risulta così basata:

Code	Pid	Byte	Description	Range			Data Contruction
01	0C	2	Engine RPM	0	16,383.75	rpm	((A*256)+B)/4
01	04	1	Calculated engine load value	0	100	%	A*100/255
01	0D	1	Vehicle speed	0	255		km/h A
01	20	4	PIDs supported [21 - 40]				Bit encoded [A7..D0] == [PID $21..PID $40]
01	2F	1	Fuel Level Input	0	100	%	A*100/255
01	40	4	PIDs supported [41 - 60]				Bit encoded [A7..D0] == [PID $41..PID $60]
	49	1	Accelerator pedal position D	0	100	%	A*100/255
	4A	1	Accelerator pedal position E	0	100	%	A*100/255
	4B	1	Accelerator pedal position F	0	100	%	A*100/255
01	03	2	Fuel system status				
01	5E	2	Engine fuel rate	0	3212.75	L/h	((A*256)+B)*0.05

PIDs Supported specifica nel seguente modo se un pid è o meno disponibile:

```
              B   E   1   F   A   8   1     3

         ---- ---- ---- ---- ---- ---- ----  ----------

supported?  1011 1110  0001   1111  1010 1000 0001  0   0   1   1

PID num     1234 5678 9ABC DEF.  ....      ....     ....  1D 1E 1F 20
```

ex. 0101, 0103, 0104, 0105, 0106, 0107, 010C, 010D, 010E, 010F 011E, 011F

0 = not supported

1 = supported

01	51	1	Fuel Type	From fuel type table see below

Mode 1 PID $51 ritorna un valore da una lista di enumerates corrispondente al tipo di alimentazione del veicolo. Il tipo di alimentazione è contenuto in un byte nel seguente modo:

01	Gasoline
02	Methanol
03	Ethanol
04	Diesel
05	LPG
06	CNG
07	Propane
08	Electric
09	Bifuel running Gasoline
0A	Bifuel running Methanol
0B	Bifuel running Ethanol
0C	Bifuel running LPG
0D	Bifuel running CNG
0E	Bifuel running Prop
0F	Bifuel running Electricity
10	Bifuel mixed gas/electric
11	Hybrid gasoline
12	Hybrid Ethanol
13	Hybrid Diesel
14	Hybrid Electric
15	Hybrid Mixed fuel
16	Hybrid Regenerative

4.4 SISTEMA MOBILE

Il sistema mobile sviluppato è stato progettato per la piattaforma Android e permette di presentare una serie di informazioni già presenti nell'interfaccia grafica del dispositivo IVI on-board, quali i dati prelevati, la valutazione sia dal punto di vista del consumo che della sicurezza, integrando anche il grafico in tempo reale dell'andamento della velocità e dei giri del motore.

Il sistema mobile definisce quindi in prima istanza un sistema di ripresentazione delle info collezionate dal sistema IVI sul dispositivo mobile, non è comunque da escludere la possibilità che il dispositivo stesso diventi un classificatore.

Tale variazione sarebbe facilmente implementabile trasferendo non le tuple classificate ma quelle campionate dall'auto ed integrando i moduli di classificazione al software mobile, la scelta renderebbe il dispositivo indipendente dalla presenza o meno di un sistema intelligente in auto, supponendo solo la presenza di una unità di trasferimento, probabilmente bluetooth, dall'OBD allo smartphone.

4.4.1 PROTOCOLLO APPLICATIVO BLUETOOTH

Il protocollo utilizzato per la connessione tra il sistema di acquisizione e il sistema mobile è stato, come già citato, il bluetooth e nello specifico sfruttando le api BlueCove[17] lato IVI, e le api Android lato mobile è stato realizzato un semplice protocollo dati per rendere le informazioni facilmente estraibili.

```
data type ->  acc:rpm:speed:engine_load:f_level:f_rate:f_type:evaluate:timestamp
```

4.4.2 PARAMETRI AGGIUNTIVI

In relazione all'utilizzo di un sistema smartphone sono stati aggiunti come valori utilizzabili al fine

della classificazione dati presenti solo sullo smartphone, ovvero la pendenza a cui è sottoposto il veicolo, la propria posizione gps e la condizione atmosferica.

Tali informazioni sono molto utili in quanto permettono di definire dei parametri correttivi alla classificazione svolta.

La conoscenza della pendenza permette di definire il tipo di strada, e relativamente alla sua pendenza essere più o meno tollerabili rispetto alla valutazione effettuata.

Conoscendo la posizione GPS è possibile introdurre un sistema di controllo relativo al tipo di strada e alle condizioni di traffico e quindi implementare logiche sia di consiglio sull'itinerario che si sta seguendo sia relativo alle velocità permesse su quel tipo di tratto stradale.

Infine la condizione atmosferica permette di meglio definire una logica di sicurezza di guida in presenza i condizioni avverse come scarsa visibilità o pioggia.

4.5 RICONOSCIMENTO E METODOLOGIA DI CLASSIFICAZIONE ED ANALISI DELLO STILE DI GUIDA

Il riconoscimento dello stile di guida viene definito principalmente a partire dal modello di classificazione riprodotto e dipendente dal training set costruito ed imposto.
Nel caso specifico un modello basato su albero decisionale.

Tale modello si fonda , come già detto, sulla costruzione di un file di training al cui interno sono contenuti una serie di elementi completamente classificati, secondo i soli parametri prelevati dall'interfaccia OBD e non dai parametri del telefonino.

L'assenza di tali parametri nella definizione del file di training è legato alla possibile assenza di tale dispositivo nell'auto, ed in uno scenario di indipendenza del sistema On-Board ed On-Phone, comporterebbe l'impossibilità di definire dei valori per tali attributi al fine della classificazione.

La struttura del file di training è quindi il seguente:

@relation OBD_sens ← definisce il nome della relazione di classificazione da associare tra training e classification file

@attribute accelerator real ← Lista degli attributi considerati

@attribute rpm real

@attribute speed real

@attribute engine real

@attribute class {well_secure, normal_secure, bad_secure, well_insecure, normal_insecure, bad_insecure}

↑ Attributo di classe (discriminatorio)

@data ← dati classificati

10,1500,40,10,well_insecure

125.4,2300,60,15,normal_secure

9.9,1278,35,10,well_insecure

9.3,1400,44,10,well_insecure

25.1,4200,98,20,bad_secure

.

.

.

La scelta di sfruttare come parametri per la classificazione i dati di accelerator position, rpm, speed ed engine load è legato al fatto che tali attributi sono necessari per una analisi esaustiva del “gear-recommended”, ovvero del sistema che permetta di definire una politica di consiglio nell'utilizzo delle marce.

Aggiungere il parametro di fuel consuption sarebbe stato di difficile integrazione per il modello di classificazione in quanto legato a molti eventi esterni quali eventi di tempistiche a cui l'utente è sottoposto oppure a condizioni come pendenza e condizione atmosferica e di traffico, al momento non completamente esaminate se non nella componente di pendenza della strada percorsa e condizione della strada, lato mobile.

Si presume quindi che in una seconda fase di miglioramento, tali dati già disponibili, potranno essere integrati con la componente di tipo strada e condizione di traffico per includere anche tale parametro nelle dinamiche di classificazione.

Il risultato della classificazione è scomponibile in due fattori, il primo legato al consumo di carburante, rispettivamente presentato all'utente sotto forma di rating bar o indicazione semaforica nel sistema On-Phone e nel sistema On-Board, mentre la seconda componente discrimina una condizione di guida o meno sicura.

In entrambi i casi si suppone la possibilità di effettuare diverse campagne simulative allo scopo di raccogliere informazioni sufficienti a definire le logiche di Fuel Saving e Safety Driving.

4.5.1 ACQUISIZIONI ON_CAR

Tutte le acquisizioni sono state effettuate su di un tracciato di pochi chilometri in condizioni di poco traffico con fondo bagnato e non .

Guida prudente:

- velocità entro i 50Km/h.
- Rpm <=2500;
- uso dell'acceleratore e freno molto uniforme (nessun accelerazione improvvisa o frenate brusche)
- marce inserite sequenzialmente.

Guida sportiva:

- velocità senza limiti.
- Rpm senza limiti;

- uso dell'acceleratore e freno molto brusco (accelerazione improvvisa, frenate brusche ripetute, e ritardate in taluni casi)
- marce inserite sia sequenzialmente che sfruttando la marcia come freno motore .

Tracciato rettilineo e provvisto di curve come da immagine.
Guida in salita (sollecitazione engine_load e verifica dato con pendenza variabile)
Completa escursione marce
Completa escursione acceleratore
Completa escursione RPM.
Parcheggio.

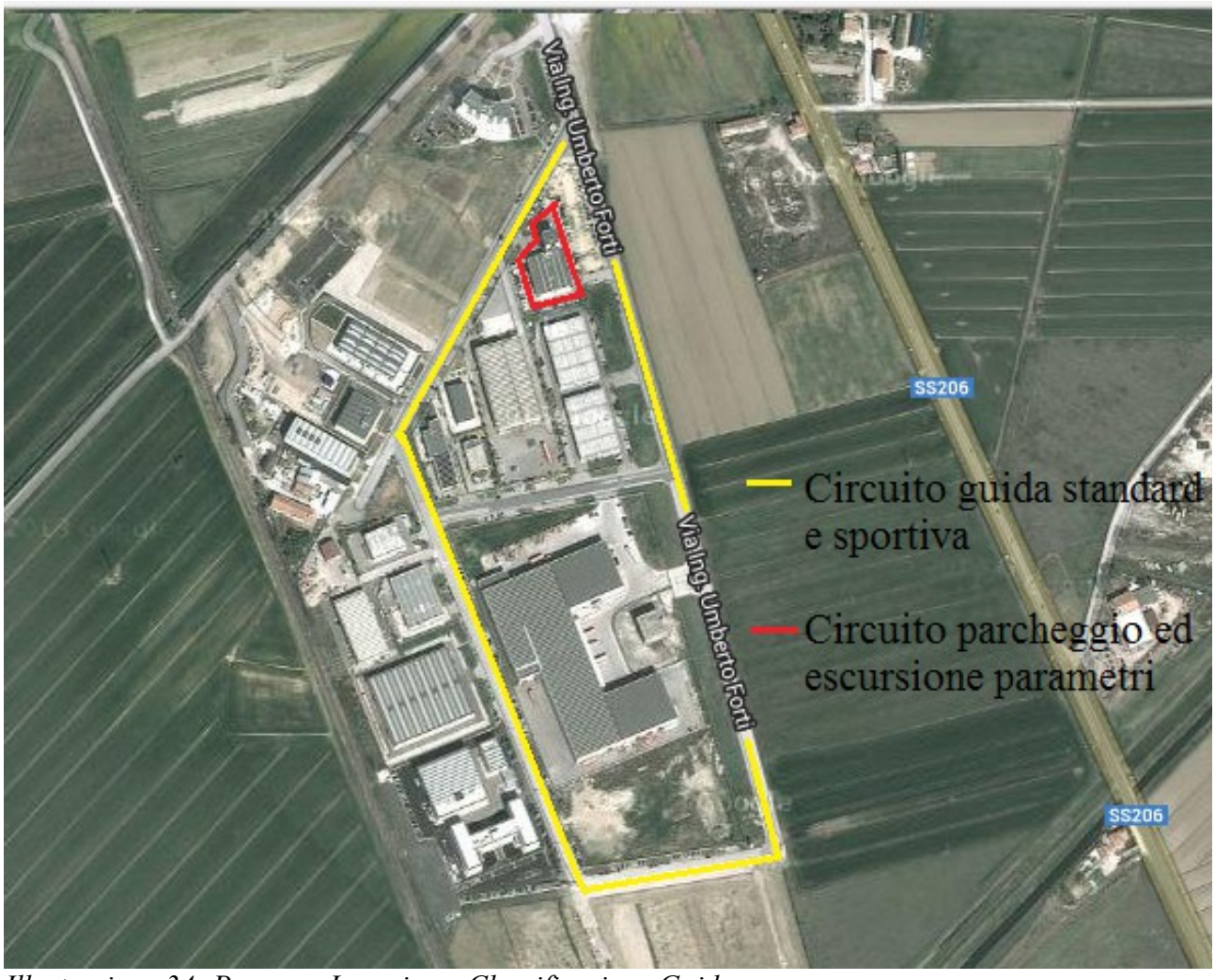


Illustrazione 34: Percorso Learning e Classificazione Guida

4.6 MEMORIZZAZIONE PERSISTENTE SU DATABASE

Tutti i sampling effettuati convogliano verso una memorizzazione persistente del dato in una banca dati gestita mediante api MySQL.

Tale scelta si pone alla base la volontà di rendere i dati campionati disponibili anche a posteriori alla luce di possibili analisi future, quali stime sullo stile di guida entro certi intervalli temporali anche lunghi, o per semplici interpolazioni sui dati raccolti.

4.6.1 STRUTTURA DATI DATABASE

sensing_table

Field	Type	Null	Key	Default	Extra
idsensing	mediumint(9)	NO	PRI	NULL	auto_increment
timestamp	timestamp	NO		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP
accelerator_position	double	YES		NULL	
rpm	int(11)	YES		NULL	
speed	int(11)	YES		NULL	
engine_load	double	YES		NULL	
fuel_level	double	YES		NULL	
fuel_rate	double	YES		NULL	
fuel_type	varchar(30)	YES		NULL	

classified_table

Field	Type	Null	Key	Default	Extra
idsensing	mediumint(9)	NO	PRI	NULL	auto_increment
timestamp_start	timestamp	NO		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP
timestamp_end	timestamp	NO		0000-00-00 00:00:00	
accelerator_position	double	YES		NULL	

rpm	int(11)	YES		NULL		
speed	int(11)	YES		NULL		
engine_load	double	YES		NULL		
fuel_level	double	YES		NULL		
fuel_rate	double	YES		NULL		
fuel_type	varchar(30)	YES		NULL		
evaluate	varchar(30)	NO		NULL		

5. SOFTWARE

Il sistema realizzato consta essenzialmente di due componenti, una relativa al sistema IVI-side ed integrato nell'ambiente automotive(SOFTWARE ON-BOARD), e quello smartphone-side risiedente sul terminale mobile di consultazione(SOFTWARE ON_PHONE).

5.1 SOFTWARE ON-BOARD

Il software risiedente sulla scheda IVI è così definito:

- ComunicazioneOBDStorage
- QTMultimedia
- ClassifierDataTransfer
- TestcanTraffic
- TrainingClassifier

5.1.1 STRUTTURA PROGETTUALE

Il sistema On_Board è analizzabile aggregando tutti i moduli in tre macro moduli, il primo macro modulo è costituito dai moduli *ComunicazioneOBDStorage* e *QTMultimedia* ed essenzialmente si occupa delle comunicazioni verso l'interfaccia OBD e dell'interfaccia grafica verso l'utente, il

secondo dal modulo *ClassifierDataTransfer,* il quale definisce la classificazione dei dati analizzati e connessione con dispositivi mobile, mentre l'ultimo costituito dai moduli *TestcanTraffic* e *TrainingClassifier* che rappresentano rispettivamente i moduli interessati alla simulazione del sistema CAN in assenza di una veicolo e costruzione del modello di classificazione mediante analisi e classificazione dei dati di training acquisiti dall'auto.

5.1.2 COMUNICAZIONE OBDSTORAGE E QTMULTIMEDIA

Il modulo principale del progetto sviluppato sicuramente è quello relativo al sensing dei dati dall'interfaccia OBD.

A partire dal modulo ComunicazioneOBDStorage, vengono estese le capacità di un sistema, denominato appunto Intecs GENIVI Multimedia con la possibilità di interagire con un veicolo dal quale prelevare informazioni prima simulate in maniera statica.

Il sistema di Infotainment comprende oltre al sistema di valutazione dello stile di guida , una serie di specifiche aggiuntive ben correlate al mondo automotive.

Il progetto Intecs GENIVI Multimedia si occupa dello sviluppo di un'applicazione multimediale per la piattaforma GENIVI.

Questa piattaforma utilizza un sistema operativo Linux e librerie con licenza open source.

Sotto la schermata principale dell'applicazione ***Intecs GENIVI Multimedia .***

Illustrazione 35: IVI Software GUI

L'applicazione è divisa in diversi moduli che sono elencati di seguito:

- *Radio*
- *Speedometer*
- *Bluetooth*
- *Weather Widget*
- *Notes*
- *Virtual keyboard*
- *Browser con supporto HTML5*
- *Maps*
- *Calendar*
- *Calculator*

- *Removable Devices*
- *Wifi*
- *Info*

In particolare la parte interessata dal lavoro svolto è l'interfaccia Speedmeter, nella quale è stata in parte rivista al logica aggiungendo a quanto precedentemente sviluppato, le componenti di classificazione sia dal punto di vista del *Fuel Saving* che *Safety Driving,* consiglio nell'uso della marcia adeguata e visualizzazione dei parametri di velocità e giri del motore non più staticamente emulati , ma prelevati dal sistema di guida del veicolo .

SPEEDMETER

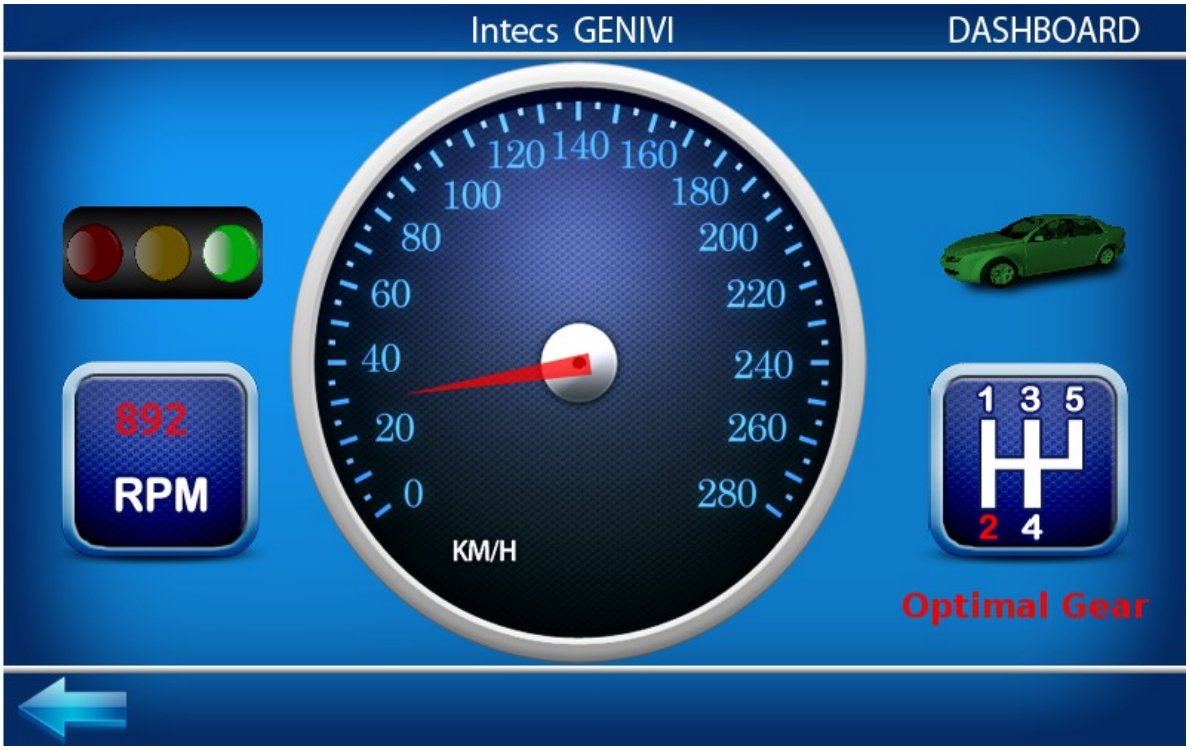


Illustrazione 36: SpeedMeter GUI

La finestra della componente Speedometer ha le seguenti funzionalità:

- visualizzazione di un tachimetro con misurazione della velocità in KM/H n real-time tramite interrogazione del sistema OBD-CAN .
- visualizzazione dei giri motore (RPM) in real-time tramite interrogazione del sistema OBD-CAN.
- presentazione visiva del cambio delle marce consigliate in relazione ai valori di engine_load, accelerator position, speed and rpm, mediante un apposito algoritmo.
- Visualizzazione grafica dello stato di sicurezza della guida (automobile con sfondo rosso o verde).
- Indicatore della qualità dello stile di guida inerente al consumo di carburante.

Libreria e API usate:

- **QtGUI** - librerie grafiche Qt per la GUI, qt.nokia.com/products/
- **Peak_Library –** libreria di interfacciamento verso i dispositivi CAN, www.peak-system.com/fileadmin/media/linux
- MySQL_library- libreria per l'interazione con i db mysql, http://dev.mysql.com/downloads/connector/
- **DLT** - le API della libreria libDLTCore, www.intecs.it/prodotti_dettagli.asp?ID_Prodotto=13

Relativamente al modulo Speedmeter, gran parte della logica è inserita nella routine di aggiornamento dell'interfaccia grafica, come di seguito specificato:

```
void SpeedoMeterWindow::paintEvent(QPaintEvent *)

{
  .
    .
    char query[100];
    snprintf(query,100,"%s","select*from sensing
                            order by idsensing desc limit 1");
    MYSQL_RES *result;
    MYSQL_ROW row;
    MYSQL_FIELD *field;
    if (mysql_query(connReal,query)) //into outfile '/var/lib/mysql/prova/dati.txt'
    {
        finish_with_error(connReal);
    }

    result = mysql_store_result(connReal);

    if (result == NULL)
    {
        finish_with_error(connReal);
    }
    while ((row = mysql_fetch_row(result)))
    {
        sens =(OBD_Sensing*)malloc(sizeof(OBD_Sensing)*1);
        sens->timestamp=atol(row[1]);
        sens->acc_position=atof(row[2]);
        sens->rpm=atof(row[3]);
        sens->speed=atof(row[4]);
        sens->engine_load=atof(row[5]);
        sens->fuel_level=atof(row[6]);
        sens->fuel_rate=atof(row[7]);
    }
    mysql_free_result(result);
    snprintf(query,100,"%s","select evaluate from classified");
    if (mysql_query(connReal,query))
    {
        finish_with_error(connReal);
    }

    result = mysql_store_result(connReal);

    if (result == NULL) {
        finish_with_error(connReal);
    }

    int num_fields = mysql_num_fields(result);

    int i;
    while ((row = mysql_fetch_row(result))) {
        for (i = 0; i < num_fields; i++)
        {
            evaluation=row[i];
        }
    }
    mysql_free_result(result);
```

In prima fase quindi vengono prelevati i dati dalle tabelle mysql Sensing e Classified, allo scopo di instanziare gli opportuni oggetti sensing ed evaluation necessari per produrre un risultato video con i dati dei parametri valutativi e la classificazione stessa.

```
int instantSpeed ;
if(sens!=NULL)
// SpeedoMeter Painter
{
    painter.rotate(1.0*(sens->speed-130));
    instantSpeed = sens->speed;  // >= 0 km/h
}

painter.drawConvexPolygon(minutehand,3);
painter.restore();                          // save speedometer painter, in
                                               order to draw with the gear
                                               painter

// Using the same QPainter, draw the gear transitions
painter.setPen(Qt::NoPen);
painter.translate(625,231);         // Translate to the top of the gear
                                       image

const QRect gearRect(0,0,136,127);  // pixel size of gear1.png
```

Successivamente in relazione ai dati letti viene calcolata la marcia consigliata.

Ciò permette al guidatore di effettuare una pre-valutazione del proprio stile di guida, in quanto la presenza di un rapporto diverso da quello indicato lo predispone in uno stato di attenzione sullo stile che sta adoperando spingendolo possibilmente ad un uso più corretto del veicolo.

Tale specifica arricchisce quindi l'utilità del sistema in quanto oltre a produrre una valutazione sul Fuel Saving e Safety Driving, permette di dare una indicazione di quale potrebbe essere uno dei parametri di guida adoperati in maniera non adatta come le marce, la cui comprensione possibilmente risulta meno immediata della velocità, o uso del pedale, di immediata comprensione, o anche del rumore dei giri del motore che seppur correlati possono non essere un chiaro ed univoco campanello di allarme di un errato uso delle marce.

```
OBD_gear gearOBD;
gearOBD.gearShift[ACCELERATOR_POSITION_LOC].data=sens->acc_position;
gearOBD.gearShift[RPM_LOC].data=sens->rpm;
gearOBD.gearShift[SPEED_LOC].data=sens->speed;
```

```
gearOBD.gearShift[ENGINE_LOAD_LOC].data=sens->engine_load;
int CG;
int RG=recommendedGear(gearOBD,&CG);
switch (RG)
{
    case 1:
    {
       painter.drawImage(gearRect, g1);
       break;
    }
    case 2:
    {
        painter.drawImage(gearRect, g2);
        break;
    }
    case 3:
    {
        painter.drawImage(gearRect, g3);
        break;
    }
    case 4:
    {
        painter.drawImage(gearRect, g4);
        break;
    }
    case 5:
    {
        painter.drawImage(gearRect, g5);
        break;
    }
    default:
    {
        painter.drawImage(gearRect, g2);
        break;
    }

}

//fuel evaluate image
const QRect evaluateRect(-585,-100,135,60);

//secure evaluate image
const QRect secureRect(0,-100,135,60);
painter.drawImage(secureRect,ag);

if(evaluation!="")
{
    if(evaluation=="bad_secure")
    {
        painter.drawImage(evaluateRect,fsr);
        painter.drawImage(secureRect,ag);
    }
    else
    {
        if(evaluation=="normal_secure")
        {
            painter.drawImage(evaluateRect,fsy);
```

```
                painter.drawImage(secureRect,ag);
            }
            else
            {
                if(evaluation=="well_secure")
                {
                    painter.drawImage(evaluateRect,fsg);
                    painter.drawImage(secureRect,ag);
                }
                else
                {
                    if(evaluation=="well_insecure")
                    {
                        painter.drawImage(evaluateRect,fsg);
                        painter.drawImage(secureRect,ar);
                    }
                    else
                    {
                        if(evaluation=="normal_insecure")
                        {
                            painter.drawImage(evaluateRect,fsy);
                            painter.drawImage(secureRect,ar);
                        }
                        else
                        {
                            if(evaluation=="bad_insecure")
                            {
                                painter.drawImage(evaluateRect,fsr);
                                painter.drawImage(secureRect,ar);
                            }
                        }
                    }
                }

              }
        }
    }

    painter.setPen(speedArrowColor);

    painter.save();

    if(sens!=NULL)
    {
        RPMvalue = sens->rpm;
        QString strRPM = QString("%1").arg( RPMvalue );
        QFont font = painter.font();
        font.setBold(true);
        font.setPointSize(22);
        painter.setFont(font);
        painter.translate(-650,-80);
        painter.drawText(QRect(100,100,100,100),strRPM);
    }

    painter.restore();
}
```

Relativamente da quanto prodotto a video da tale modulo, evince che il carico del motore non è mostrato seppur componente inclusa tra gli attributi di classificazione.

Tale scelta deriva dalla necessità di mantenere quanto più chiara e concisa la produzione a video dei dati estrapolati.

Il carico del motore seppur indispensabile nel processo di classificazione risulta molto stazionario nella sua evoluzione e fortemente influenzato da strade percorse con discreta pendenza, quindi non necessario come componente primario da visualizzare.

5.1.3 STRUTTURA PROGETTUALE INTERAZIONE SISTEMA ON_CAR

Il sistema viene quindi progettato tenendo in considerazione che tutte le componenti di interfaccia verso il Car_side fanno uso di librerie specifiche per l'interazione con i prodotti dongle della Peak®, i livelli più esterni sfruttano tali informazioni per estrarre delle caratteristiche e per presentarli agli altri componenti del sistema interni o esterni, ovvero componenti dello stesso sistema o meno.

In aggiunta per rendere l'analisi veloce ed indipendente dalla presenza di una macchina reale è stato, come già detto, implementato un modulo di Testing che leggendo una serie di dati da un file testuale le trasmette periodicamente sul CAN.

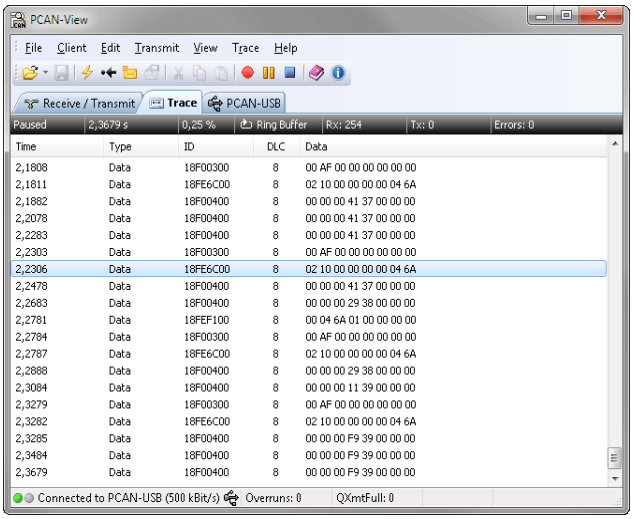
L'insieme delle informazioni prelevate dall'auto durante la campagna di acquisizione è composta sia dai dati richiesti che da informazioni non utili ai fini della classificazione.

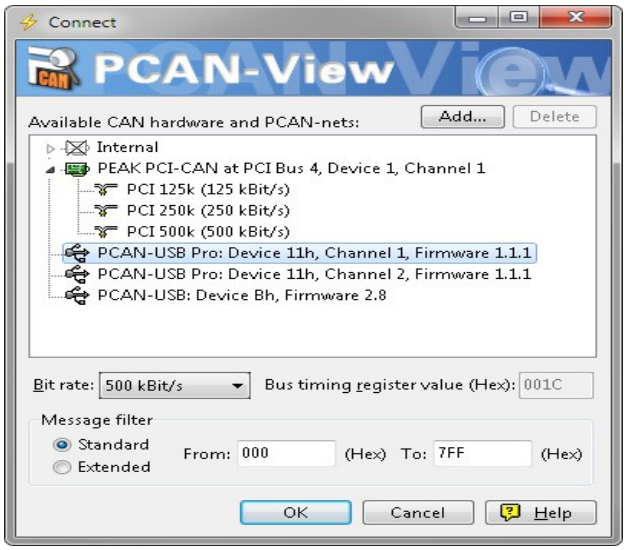
Mentre i dati relativi a reali richieste vengono opportunamente trasmesse sul CAN solo in presenza di richieste da parte del modulo di lettura, implementando così la reale sequenza di dati sul CAN relativi a effettive richieste del dato, le info non utili vengono trasmesse sempre senza richieste.

APPLICAZIONE DI SIMULAZIONE DATI CAN-OBD

Le simulazioni effettuate sul campo sono state realizzate mediane un opportuno software della peak, che instaurata una connessione tra l'auto ed il pc mediante dongle USB-CAN (ed adattatore CAN-OBD per l'interfaccia dell'auto) permette di registrare i dati di interesse memorizzandoli su un file di track.

Il software in questione è il PeakView.





Tale software Pcan-View, disponibile su piattaforma windows, è un semplice monitor CAN, per visionare,ricevere e mandare traffico CAN.I massaggi possono essere trasmessi manualmente o periodicamente con un certo bitrate non oltre 1 Mbit/s. Gli errori del bus di comunicaizione o di overflow sono controllati e posti a video durante il processo di comunicazione. Una importante funzione è il “Trace” dei dati ovvero memorizzazione di tutto il traffico trasmesso e ricevuto

durante una determinata sessione di comunicazione, ciò molto utile al fine di mantenere memoria di richieste e risposte per la realizzazione di un tester.

Successiva alla fase di prelievo dei dati di testing successivamente analizzata, si è reso necessario aggiungere un ulteriore step prima dell'utilizzo di tali dati nel progetto, al fine di formattare correttamente i dati prelevati (considerando sia i dati utili che quelli spuri viaggianti sul CAN), formattandoli opportunamente prima della lettura di simulazione.

La routine interessata alla definizione delle logiche di filtraggio e formattazione e contenuta nel modulo TestcanTraffic è *preparatore()*:

```
void preparatore()
{
	preparatorfile = fopen("preparator.txt","r");
	filein = fopen("filein.txt","a");
	if(preparatorfile== NULL || filein ==NULL)
	{
		printf("error_ FILE NOT FOUND BLOCK OF CODE \n");
		exit(0);
	}

	resread=fscanf(preparatorfile,"%x %f\n",&id,&t);
	printf(" pid %x value %f",id,t);
	bool a=false,r=false,s=false,e=false;
	while(resread!=EOF)
	{
		switch(id)
		{
			case 73:
			{
				if(!(a) & !(r) & !(s) & !(e))
				{
					a=true;
					fprintf(filein,"%f,",t);
				}
				break;
			}
			case 12:
			{
				if(a & !(r))
				{
					r=true;
					fprintf(filein,"%f,",t);
				}
				break;
			}
			case 13:
			{
```

```
                        if(a & r & !(s))
                        {
                              s=true;
                              fprintf(filein,"%f,",t);
                        }
                        break;
                  }
                  case 4:
                  {
                        if(a & r & s & !(e))
                        {
                              e=true;
                              fprintf(filein,"%f",t);
                        }
                        break;
                  }
            }
            if(a & r & s & e)
            {
                  a=false;r=false;s=false;e=false;
                  fprintf(filein,"\n");
            }
            resread=fscanf(preparatorfile,"%x %f\n",&id,&t);
            printf(" pid %x value %f",id,t);

            }
}
```

aperto in lettura il file contenente i dati spuri dell'applicativo PeakcanView vengono opportunamente letti i dati di interesse, ovvero posizione dell'acceleratore, giri del motore,velocità e carico del motore, precedentemente filtrati da tutti i dati letti dall'auto, in modo da ricostruire un file in cui tali dati risultano ordinati.

Tale necessità nasce dall'esigenza di costruire un file formattato per realizzare il Training set, ricordiamo che tale set costituisce la base di analisi per il lavoro di classificazione in quanto derivante da una fase di campionamento sperimentale guidato corrisponde ad una classificazione ben nota e specificata dal modulo Training_classifier, il cui codice in parte è di seguito riportato:

```
.
.
BufferedReader bi= new BufferedReader(new FileReader("filein.txt"));
BufferedWriter bo= new BufferedWriter(new FileWriter("fileout.txt",true));
.
.
training_valutate()
{
      String[] array= {};
      String read,write="";
      Float acc,rpm,speed,carico_motore;
```

```
        try{
              read=bi.readLine();
              while(leggo dal file)
              {
                    array = read.split(",");
                    read=bi.readLine();
                    for(int i = 0 ;i< array.length; i ++)
                    {
                          System.out.println(array[i]stile+"\t");
                          write+=(i<=array.length-1)?array[i]+",":
                                                      array[i];
                    }
                    acc=Float.parseFloat(array[0]);
                    rpm=Float.parseFloat(array[1]);
                    speed=Float.parseFloat(array[2]);
                    carico_motore=Float.parseFloat(array[3]);
                    if(acc)>33||rpm>2000||speed>50||carico_motore)> 10   )

                          if(acc)>66||rpm>2500||speed>90||carico_motore)>20)
                          {
                                if(rpm>3000 || speed>90)
                                //3000rpm e pedale quasi tutto
                                inclinato sono indici di insecure   driving
                                      write+="bad_insecure";
                                else
                                      write+="bad_secure";
                          }
                          else
                          {
                                if(rpm>3000 || speed>90)
                                //3000rpm e pedale quasi tutto
                                inclinato sono indici di insecure   driving
                                      write+="normal_insecure";
                                else
                                      write+="normal_secure";
                          }
                    else
                    {
                          if(rpm>3000 || speed>90)
                          //3000rpm e pedale quasi tutto
                          inclinato sono indici di insecure   driving
                                write+="well_insecure";
                          else
                                write+="well_secure";
                    }
                    bo.append(write+"\n");
                    System.out.println("\n");
                    write="";
              }
              bi.close();
              bo.close();
        }catch(Exception e){}
  }
  .
  .
```

Dal codice ben si evince come la costruzione di una classificazione preliminare sia basata su una opportuna guida prescelta e da una logica a soglia che comunemente viene usata per valutare sia la sicurezza che l'uso dell'auto.

In particolare si fa riferimento alle velocità standard in prossimità di centri urbani e strade extraurbane secondarie 50 Km/h in centri urbani e 90Km/h in strade di collegamento tra paesi e città che non siano autostrade o strade extraurbane principali, ove i limiti sono rispettivamente 130 Km/h e 110Km/h previa diversa prescrizione).

Tale limite verrebbe successivamente rivisto in relazione ad una geo-localizzazione del veicolo sul tipo di strada, operando al momento in uno scenario urbano ed extraurbano a esclusione di autostrade, ove lo stile di guida risulta più legato a non superare il solo limite imposto, potendo operare una guida molto più fluida e Fuel Saving.

Relazionando anche gli altri parametri operativi si sceglie di dividere l'escursione del pedale dell'acceleratore in tre settori, strettamente legato ad un numero di classi di valutazione pari a tre stati (guida ottimale, normale e sconsigliata, classificazione alla componente del Fuel Saving), ed analogamente a giri e carico del motore si utilizzano dati sperimentali in cui si verifica che un cambio marce molto rilassato viene ottenuto per veicoli a benzina(come quello in analisi) tra 2000 e 2500 rpm, definendo quindi un margine di 500 rpm dal quale si ottengono le soglie :

rpm < 2000	guida ottimale
2000 < rpm < 2500	guida normale
3000 < rpm	guida sconsigliata

Analogamente il carico del motore vien relazionato alle soglie 10% e 20%:

carico < 10%	guida ottimale
10% < carico < 20%	guida normale
20% < carico	guida sconsigliata

relativamente alla classificazione relativa al Safety Driving vengono presi in considerazione solo velocità e giri del motore , tralasciando il carico del motore e la pressione del pedale

dell'acceleratore in quanto la componente di quest'ultimo che può provocare situazioni di pericolo è strettamente legata alla velocità e all' rpm.

Si valutano entrambi, e nello specifico anche l'rpm oltre alla velocità che è chiaramente legata ad un fattore di safety in quanto è possibile da quest'ultimo estrarre un fattore di errato uso dei giri che può provocare danni al motore o altri parti meccaniche dell'auto senza avere dirette connessioni con la velocità del veicolo. P.e. Una situazione in cui l'auto ha una media velocità ma un numero di giri particolarmente alto rende il veicolo meno pronto a repentini cambi di marcia , velocità o frenate brusche in condizione di ostacoli o situazioni in cui l'intervento del guidatore si rende necessaria per evitare situazioni di pericolo.

Definito il file di interesse e verificata la corretta apertura dello stesso, viene opportunamente formattata la lettura dei dati prelevati dal file di preparazione derivante dalle simulazioni fatte:

```
sensingfile = fopen("parking_drive.txt","r");

if(sensingfile== NULL)
{
      printf("error_ FILE NOT FOUND BLOCK OF CODE \n");
      exit(0);
}
.
.

resread=fscanf(sensingfile,"%s%f%s%x%d%x%x%x%x%x%x%x ,&n,&t,&tr,&id,&nb,&b[0],
                                  &b[1],&b[2],&b[3],&b[4],&b[5],&b[6],&b[7]);
```

Di seguito si realizza la creazione del dato relativo al fuel level, dato di interesse ma non reso disponibile dalla centralina dell'auto a disposizione.

```
.
.
      int j;
      while(true)
      {
            if(j==10)j=0; //artificio per leggere con equiprobabilità i livelli
                           di fuel simulato;
            for(i=j+1;i<11;i++)
            {
                  pMsgRdBuff=exponential_fuel_value(i);
                  _pMsgRdBuff=(TPcanRdMsg *) malloc(sizeof(TPcanRdMsg) * 1);
                  _response=queueTestReference(pMsgRdBuff.Msg.DATA[2]);
```

```
                _pMsgRdBuff->Msg=pMsgRdBuff.Msg;
                _pMsgRdBuff->dwTime=pMsgRdBuff.dwTime;
                _pMsgRdBuff->wUsec=pMsgRdBuff.wUsec;
                node.message=_pMsgRdBuff;
                queue_enqueue(_response,&node);
        }
.
.
```

Una particolarità del modulo è la capacità di discriminare i dati di interesse da quelli relativi a traffico CAN non richiesto.

Nel contesto del traffico di non interesse, tra una richiesta di pid e quella successiva il sistema riceve anche informazioni che al fine del progetto possiamo chiaramente scartare ovvero immettere sul CAN senza che il sistema ne risulti interessato.

Analogamente tra i dati sono presenti anche le richieste pid effettuate e le relative risposte, mentre le richieste vengono scartate e non reimmesse sul CAN perché corrispondenti a dinamiche direttamente imposte dal modulo di richiesta, le risposte vengono conservate su una specifica coda diversa da quella dei dati non richiesti in quanto immesse sul CAN solo al ricevimento di una richiesta per lo specifico pid.

Ovvero viene mantenuta la correlazione tra permanenza di un dato sul CAN relativo ad un pid di interesse solo a seguito di una richiesta dello stesso, come nella realtà non simulata avverrebbe.

Il sistema presenta una serie di controlli per cui viene manifestata l'assenza di tale dato nel sensing fino ad ora esaminato o ricerca di un pid non relativamente registrato tra quelli di interesse mediante la routine *queueTestReference()*, continuando quindi la sottomissione di richieste dei pid successivi, di seguito il codice relativo.

```
.
.
//Control if is a 7DF, 7E8 id message of request or response,
  otherwise is a sniffing message,
//     7)        31.2  Rx         07E8  8  04 41 0C 0C 32 00 00 00

if(id==0x7DF || id==0x7E8)
{
      if(id==0x7E8)
```

```
	{
		pMsgRdBuff.Msg.DATA[0] =(BYTE) b[0];
		pMsgRdBuff.Msg.DATA[1] =(BYTE) b[1];
		//Change in relation to the pid selected.
		pMsgRdBuff.Msg.DATA[2] =(BYTE) b[2];
		pMsgRdBuff.Msg.DATA[3] =(BYTE) b[3];
		pMsgRdBuff.Msg.DATA[4] =(BYTE) b[4];
		pMsgRdBuff.Msg.DATA[5] =(BYTE) b[5];
		pMsgRdBuff.Msg.DATA[6] =(BYTE) b[6];
		pMsgRdBuff.Msg.DATA[7] =(BYTE) b[7];
		pMsgRdBuff.Msg.ID=(DWORD)id;
		pMsgRdBuff.Msg.LEN=8;
		pMsgRdBuff.Msg.MSGTYPE=MSGTYPE_STANDARD;
		gettimeofday(&times, 0);
		timestamp = (times.tv_sec);
		pMsgRdBuff.dwTime=timestamp;
		pMsgRdBuff.wUsec=0;

		resp.pid=(uint16_t)b[2];
		resp.data=format_value(pMsgRdBuff,b[2]);

		resp.timestamp=timestamp;
		_pMsgRdBuff=(TPcanRdMsg *)malloc(sizeof(TPcanRdMsg)*1);
		_response=queueTestReference(pMsgRdBuff.Msg.DATA[2]);

		_pMsgRdBuff->Msg=pMsgRdBuff.Msg;
		_pMsgRdBuff->dwTime=pMsgRdBuff.dwTime;
		_pMsgRdBuff->wUsec=pMsgRdBuff.wUsec;

		node.message=_pMsgRdBuff;
		queue_enqueue(_response,&node);

	}
}
else
{
	pMsgRdBuff.Msg.DATA[0] =(BYTE) b[0];
	pMsgRdBuff.Msg.DATA[1] =(BYTE) b[1];
	pMsgRdBuff.Msg.DATA[2] =(BYTE) b[2];
	pMsgRdBuff.Msg.DATA[3] =(BYTE) b[3];
	pMsgRdBuff.Msg.DATA[4] =(BYTE) b[4];
	pMsgRdBuff.Msg.DATA[5] =(BYTE) b[5];
	pMsgRdBuff.Msg.DATA[6] =(BYTE) b[6];
	pMsgRdBuff.Msg.DATA[7] =(BYTE) b[7];
	pMsgRdBuff.Msg.ID=(DWORD)id;
	pMsgRdBuff.Msg.LEN=8;
	pMsgRdBuff.Msg.MSGTYPE=MSGTYPE_STANDARD;
	gettimeofday(&times, 0);
	timestamp = (times.tv_sec);
	pMsgRdBuff.dwTime=timestamp;
	pMsgRdBuff.wUsec=0;
	sniff.pid=(uint16_t)b[2];
	sniff.data=format_value(pMsgRdBuff,b[2]);
	sniff.timestamp=timestamp;
	pMsgRdBuff=(TPcanRdMsg *) malloc(sizeof(TPcanRdMsg) * 1);
	pMsgRdBuff->Msg=pMsgRdBuff.Msg;
	_pMsgRdBuff->dwTime=pMsgRdBuff.dwTime;
	_pMsgRdBuff->wUsec=pMsgRdBuff.wUsec;
```

```
                node.message=_pMsgRdBuff;
                queue_enqueue(_sniffing,&node);
                sleep(1);

        }
        resread=fscanf(sensingfile,"%s%f%s%x%d%x%x%x%x%x%x%x%x\n",&n,&t,&tr,
                &id,&nb,&b[0],&b[1],&b[2],&b[3],&b[4],&b[5],&b[6],&b[7]);

        if(resread==EOF )
        {
                i++;
                int res= fseek(sensingfile, 0, SEEK_SET);
        }
        j++;
}
fclose(sensingfile);
```

Si evince dal codice che la stessa funzione effettua la formattazione del dato relativamente a quanto specificato per ogni pid, leggendo ciclicamente lo stesso file, ogni dato infatti secondo lo standard OBD è definito su uno o due bytes e presenta come specificato in precedenza una specifica ricostruzione del proprio valore.

```
float format_value(TPcanRdMsg pMsgRdBuff, uint16_t pid)
{

        OBD_Data obdData;
        switch (pid)
        {
                case REQUEST_PID_ACCELERATION_POSITION:
                {
                        obdData.data = (float) pMsgRdBuff.Msg.DATA[3] * 100 / 255;
                        break;
                }
                case REQUEST_PID_RPM:
                {
                        obdData.data = (float) ((pMsgRdBuff.Msg.DATA[3] * 256) +
                                                pMsgRdBuff.Msg.DATA[4]) / 4;
                        break;
                }
                case REQUEST_PID_SPEED:
                {
                        obdData.data = (float) pMsgRdBuff.Msg.DATA[3];
                        break;
                }
                case REQUEST_PID_ENGINE_LOAD:
                {
                        obdData.data = (float) pMsgRdBuff.Msg.DATA[3] * 100 / 255;
                        break;
                }
                case REQUEST_PID_FUEL_LEVEL:
                {
                        obdData.data = (float) pMsgRdBuff.Msg.DATA[3] * 100 / 255;
```

```
                  break;
            }
            case REQUEST_PID_FUEL_RATE:
            {

                  obdData.data = (float) ((pMsgRdBuff.Msg.DATA[3] * 256)+
                                          pMsgRdBuff.Msg.DATA[4]) * 0.05;
                  break;
            }
            case REQUEST_PID_FUEL_TYPE:
            {
                  obdData.data = (float) (pMsgRdBuff.Msg.DATA[3] * 256
                                          | pMsgRdBuff.Msg.DATA[4]);
                  break;
            }
            default:
            {
                  obdData.data = (int) (pMsgRdBuff.Msg.DATA[3]|
                                          pMsgRdBuff.Msg.DATA[4]);
                  break;
            }
      }
      return obdData.data;
}
```

5.1.4 LIBRERIE

In relazione al sistema sviluppato diverse librerie esterne sono state utilizzate al fine di rendere possibile l'interazione con i database, con i prodotti dongle della peak, e per l'interazione mediante il bluetooth.

LIBRERIA PEAK-CAN

La libreria peak-CAN utilizzata fa riferimento ad un ambiente linux, e permette in tale sistema di interagire con prodotti usb-CAN.

Tale libreria necessita per un corretto funzionamento la presenza di opportuni driver per il riconoscimento dei dispositivi.

L'istallazione di tali driver in ambiente linux segue il seguente procedimento:

- download dei driver peak-linux-driver-X.Y.tar.gz dal sito

  http://www.peak-system.com/fileadmin/media/linux/

- installazione di una serie di reference:

- installazione libpopt-dev: sudo apt-get install libpopt-dev

- installazione g++: sudo apt-get install g++

(necessario solo per eseguire il test tool transmittest)

- installazione

  - unpack dei driver: tar -xzf peak-linux-driver-X.Y.tar.gz

  - accesso alla cartella cd peak-linux-driver- X.Y

  - make clean

  - make NET=NO

  - sudo make install

  - sudo modprobe pcan

- controllare l'effettiva corretta installazione

  cat /proc/pcan

```
*------------ PEAK-Systems CAN interfaces (http://www.peak-system.com) -------------
*-------------------------- Release_20110912_n (7.4.0) ----------------------
*---------------- [mod] [isa] [pci] [dng] [par] [usb] [pcc] -----------------
*--------------------- 1 interfaces @ major 250 found -----------------------
*n -type- ndev --base-- irq --btr- --read-- --write- --irqs-- -errors- status
  32 usb -NA- 30520000 255 0x001c 00000000 00000000 00000000 00000000 0x0000
```

la presenza del comando make NET=NO serve per effettuare una installazione di tipo chardev piuttosto che netdev e riscontrabile nell'esecuzione del comando “cat” notando la voce NA in corrispondenza del campo ndev.

Per la natura dello stesso driver e dei riferimenti in esso definiti all'atto dell'installazione risulta necessaria una re-installazione degli stessi in presenza di aggiornamenti del kernel mediante i

seguenti comandi:

sudo rmmod pcan

cd peak-linux-driver- X.Y

make clean

make NET=NO

sudo make install

cd /driver

sudo insmod pcan.ko

cat /proc/pcan

and -NA- should be displayed in the netdev-column

INTERAZIONI CAN MEDIANTE LIBRERIA

Tutte le interazioni col CAN sono state effettuate mediante l'opportuna libreria specificata, in assenza della quale si sarebbe reso necessario trovare un meccanismo di interazione con i dongle utilizzati, perfettamente definito utilizzando le suddette librerie appartenenti alla stessa azienda produttrice dei cavi e connettori utilizzati

CONNESSIONE ED INIZIALIZZAZIONE

```
DWORD CAN_Init(HANDLE hHandle, WORD wBTR0BTR1, int ncanMsgType);
```

il parametro HANDLE hHandle, identifica l'attributo mediante il quale è possibile fare riferimento alla specifica in-stanza di connessione, ed utilizzato per tutte le interazioni col CAN.

Tale parametro risulta derivante dall'apertura di una connessione mediante l'invocazione della funzione `HANDLE LINUX_CAN_Open(const char *szDeviceName, int nFlag);`

nel caso specifico definito obdReader una istanza della struct OBD_Reader

```
typedef struct OBD_Reader
{
      HANDLE handle;           /**< Handle for CAN connection */
      ErrorType error;         /**< ErrorType as return value of function */
      pthread_t sensingThread;/**< Thread of OBD sensing. */
}OBD_Reader;
```

viene richiamata come di seguito:

```
obdReader.handle = LINUX_CAN_Open("/dev/pcan32", O_RDWR);
```

specificando il descrittore del dongle CAN-USB e la modalità O_RDWR relativamente ai parametri di apertura della connessione.

WORD wBTR0BTR1 identifica invece il baud rate di connessione con il CAN, dopo diverse prove sperimentali si è verificato che il baud rate da utilizzare è 500Kbps , specificato mediante la constante CAN_BAUD_500K della libreria.

Infine il parametro **int** ncanMsgType, specifica il tipo di frame CAN da utilizzare. Nel caso in questione si è usata una frame standard specificata dalla constante CAN_INIT_TYPE_ST.

Il valore di ritorno identifica il codice della connessione, qualora fosse diverso da “0” identifica una connessione correttamente inizializzata altrimenti manifesta un errore.

SCRITTURA E LETTURA

Rispettivamente scrittura e lettura sul CAN si riferiscono al trasferimento di richieste e lettura delle relative risposte e/o dati in transito sul bus.

Le richieste vengono effettuate mediante la funzione CAN_Write la cui signature risulta definita come di seguito:

```
DWORD CAN_Write(HANDLE hHandle, TPCANMsg* pMsgBuff);
```

specificando come parametri l'handle relativo alla connessione e la struttura dati contenente la

frame CAN e in precedenza descritto.

Analogamente mediante la lettura definita mediante la funzione così descritta:

```
DWORD LINUX_CAN_Read(HANDLE hHandle, TPCANRdMsg* pMsgBuff);
```

è possibile leggere dal CAN la relativa risposta.

Tale funzione risulta bloccante quindi una volta trasmessa una richiesta ed invocato la lettura da CAN, il sistema attende un qualunque dato prima di riprendere l'esecuzione.

Relativamente ai valori di ritorno di tali funzioni è stata predisposta una routine che presa in ingresso il risultato dell'invocazione ritorna in uscita la presenza o meno di un errore, specificandone quando possibile la natura.

La routine in questione è CAN_Response_Analisys(int code), la quale è così definita:

```
void CAN_Response_Analisys(int code)
{
	switch (code)
	{
		case No_Error:
		{
			obdReader.error = No_Error;
			#ifdef DEBUG_MODE
			//printf("no errore \n");
			#endif
			break;
		}
		case EBADF:
		{
			obdReader.error = Error_WrongParameters;
			//error in passed parameters
			#ifdef DEBUG_MODE
			printf("wrong parameters \n");
			#endif
			break;
		}
		default:
		{
			obdReader.error = General_Error;
			//error from list of errno-base.h
			#ifdef DEBUG_MODE
			printf("general error \n");
			#endif
			break;
		}
	}
}
```

LIBRERIA MYSQL

Permette di interfacciarsi verso il database, attraverso l'installazione dei connector mysql.

Il sistema On_Board sfrutta la libreria mysql e mediante quest'ultima viene definito l'accesso, l'interrogazione e scrittura delle tabelle definite.

Dal sito [1] è possibile scaricare i connector linux_oriented e successivamente averla installata con i seguenti comandi:

```
shell> su root

shell> rpm -ivh mysql-connector-odbc-5.2.2.i386.rpm
```

se il driver esiste effettuare un update con i seguenti comandi:

```
shell> su root
shell> rpm -Uvh mysql-connector-odbc-5.2.2.i386.rpm
```

Una volta installati i driver è possibile scaricare la libreria dall' indirizzo [2] e linkarla al proprio progetto.

Relativamente al codice tale libreria si è rea necessaria per aprire una connessione verso il database di interesse e per la memorizzazione ed estrazioni dei dati mediante sottomissione di query.

Una volta linkata la libreria è necessario includere le librerie di interesse, che nello specifico sono:

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;
.
.
```

mediante tali inclusioni è possibile richiamare le funzioni della libreria per settare la connessione richiamando l'opportuno driver, database e struttura dati per il retrieve dei dati delle query una volta definite le istanze delle classi necessarie,

```
private Connection connectSensing = null, connectClassified = null;
private Statement statement = null;
private PreparedStatement preparedStatement = null;
private ResultSet resultSet = null;
```

rispettivamente mediante le seguenti funzioni:

```
Class.forName("com.mysql.jdbc.Driver");

connectSensing = DriverManager.getConnection("jdbc:mysql://localhost/prova?"
                    + "user=root&password=root");


statement = connectSensing.createStatement();


resultSet= statement.executeQuery(QUERY);
```

a partire poi dalla variabile resultSet sarebbe possibile estrarre sequenzialmente tutti i dati medianti un ciclo

```
while (resultSet.hasNext())
{
      .
      .
}
```

LIBRERIA BLUETOOTH

La libreria bluetooth utilizzata è bluez, scaricabile dal sito http://www.bluez.org/download/, mediante tale libreria è possibile effettuare un discovery dei dispositivi presenti discriminando quelli già associati, e una volta stabilita una connessione permettere lo scambio dati sia sotto forma di file che stream di dati.

LIBRERIA WEIKA

Il modello utilizzato per effettuare la classificazione si basa sul framework di Weka[17] , mediante il quale è stato possibile definire un albero decisionale di classificazione.

Mediante tale framework è stato possibile anche valutare il modello definito, i risultati più avanti motivano ulteriormente la scelta del modello definito.

Estraendo dalla cross-classificazione degli stessi dati di training il livello di affidabilità e corretta

classificazione compiuta.

Il seguente codice definisce tali funzionalità:

```
.

.

caricaModello();

try
{
      data_classify=new Dataset(CLASSIFICATION_FILE);
      classificaDataset(data_classify);
      resetClassificationFile();
}
catch(IOException e)
{

                  e.printStackTrace();
}
.

.

public void caricaModello() stile

{

            try
            {
                  data_learn = new Dataset(TRAINING_FILE);
                  valutaModello(data_learn.instances);

            }
            catch (IOException e)
            {
                  e.printStackTrace();
            }

}

.
.
public void valutaModello(Instances data)
{

      try
      {
                   tree = new RandomForest();    // new instance of tree

                   tree.buildClassifier(data);   // build classifier
                   Evaluation eval = new Evaluation(data);
                   eval.crossValidateModel(tree,data,10, new Random(1));
                   System.out.println(eval.toSummaryString(" ", true));
      }catch( Exception e){System.out.println("errore validazione");}
}
```

Relativamente alla classificazione di nuovi dati invece viene effettuata una vera e propria classificazione mediante il seguente codice che caricato il modello di training valuta le istanze acquisite:

```
public void classificaDataset(Dataset tolabel)
{
      // Create a copy for manipulating the dataset.
       Dataset unlabeled = new Dataset(tolabel);

       // Label the instances and get the distribution.
       double clsVal = 0;
       double[] clsDistro = null;
       String s=Integer.toString(unlabeled.numInstances());

       try{tree.buildClassifier(data_learn.instances);}
       catch (Exception e1) {e1.printStackTrace();}
       for (int i=0; i<unlabeled.numInstances(); i++)
       {
            try
            {
                  clsVal = tree.classifyInstance(unlabeled.getInstance(i));
            }
            catch(Exception e){}
            try
            {
             clsDistro =tree.distributionForInstance(unlabeled.getInstance(i));
            }
            catch(Exception e){}
            tolabel.setDistroForInstanceAt(clsDistro, i);
            tolabel.setClassValueAt(clsVal, i);
        }
        String distr=tolabel.distroToString();
    }
.
.
```

5.2 SOFTWARE ON-PHONE

Il software on_Phone rappresenta un sistema quasi autonomo, in quanto si basa sulla necessità di ricevere le info già classificate al sistema On_Board, ma potrebbe essere un sistema autonomo supponendo la mancanza di un sistema all'interno dell'auto , mediante la presenza di un adattatore direttamente connesso all'OBD mediante il quale ricevendo direttamente i dati spuri sia possibile classificarli.

Si è scelto di sviluppare il prodotto sulla piattaforma Android, piuttosto che IOS o Windows phone

per garantire la compatibilità massima con tablet e smartphone in commercio, e per la garanzia di lavorare su una piattaforma completamente open source.

Sviluppare un prodotto che risulti largamente compatibile con i comuni dispositivi in commercio permetterebbe di rendere il prodotto in futuro realizzabile vincente e flessibile rispetto a soluzioni brandizzate.

5.2.1 CARATTERISTICHE

Le caratteristiche principali del software On_Phone sono la possibilità oltre che di presentare le info prelevate dal sistema CAN-OBD, anche di permettere un livello di astrazione rispetto alle info ricevute, e di mantenere uno storico grafico delle info relative alla velocità e gli rpm facilmente gestiti su due interfacce grafiche accessibili mediante swipe.

5.2.2 STRUTTURA PROGETTUALE

Il software On_Phone diversamente dal sistema On_Board presenta una struttura più modulare in quanto ogni singola activity rappresenta una scomposizione del sistema così definito:

- MAIN ACTIVITY & SERVICE
    - AwesomePagerActivity
    - MonitorService
- SERVICE ACTIVITY
    - WebWeather
- BLUETOOTH ACTIVITY & SERVICE
    - DeviceListActivity
    - BluetoothService
- UTILS

    - About
    - Utils
- GUI CLASS
    - CustomImageView
    - FuelIndicator
    - DepthPageTrasformer

ACTIVITY, CLASS & SERVICE

I MAIN ACTIVITY & SERVICE Rappresentano l'activity e service principali, l'activity AwesomePageActivity rappresenta l'entry point del sistema e sul quale viene definita la struttura dell'intero sistema, costruita l'opportuna interfaccia grafica a swipe per differenziare dati di immediata utilità da quelli riepilogativi.

Il MonitorService rappresent il service in attesa di messaggi dal bluetooth e pronto a ritrasferire le info acquisite mediante BroadcastIntent all'AwesomePageActivity.

Il service BluetoothService rappresenta il modulo capace di definire ed instaurare le comunicazioni bluetooth con il terminale On_Board.

Tale modulo è associato all'activity DeviceListActivity, mediante la quale è possibile effettuare un discovery dei dispositivi disponibili e di sceglierne uno per instaurare la connessione.

Il service WebWeather, instaura invece una connessione con il web server OpenWeather[18] dal quale effettua un retrieve delle condizioni meteo mediante coordinate geografiche del dispositivo mobile, precedentemente ottenute.

I moduli CustomImageView e FuelIndicator invece rappresentano le opportune classi che estendono la classe View di android per definire rispettivamente il widget grafico del tachimetro, del contagiri, e l'indicatore del carburante.

Infine la classe About e Utils rappresentano un descrittore dell'applicativo e un container di tutte le costanti utilizzate nei vari moduli.

5.2.3 LIBRERIE

Diversamente dal sistema On_Board, le librerie utilizzate dal sistema mobile si riducono alla sola libreria grafica GraphView[19] e alla libreria bluetooth Bluez, piuttosto che quella integrata in javax.

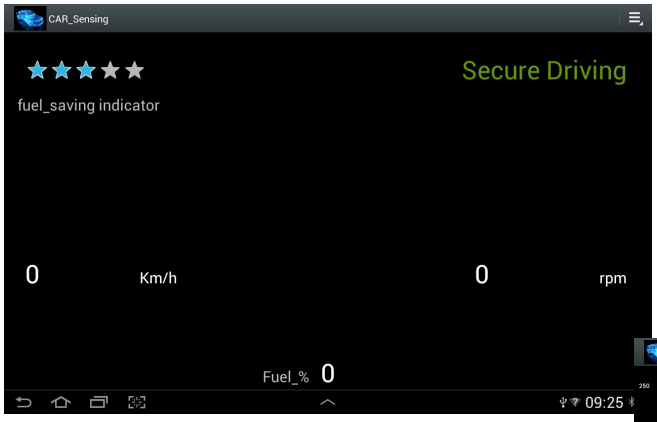
Mediante la libreria grafica GraphView, vengono prodotti i grafici di velocità e giri del motore.

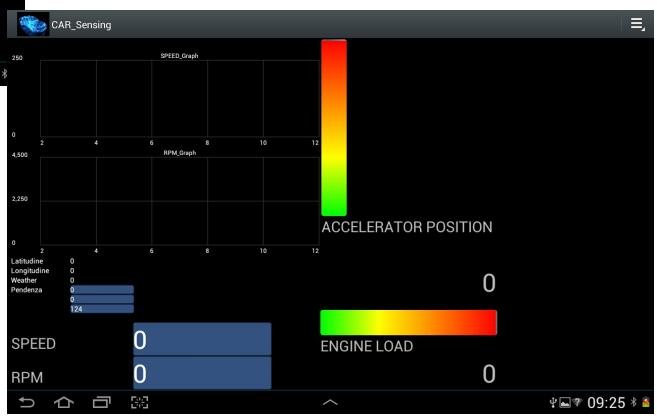
GraphView permette di definire un set di dati a partire dalle informazioni raccolte e di plottarli a video.

Caratteristica importante di tale libreria è la possibilità di zoommare e visionare temporalmente le informazioni plottate, così da avere una visione di insieme più completa.

5.2.4 LOGICA DI FUNZIONAMENTO

All'avvio l'applicazione On_Phone si presenta in tal modo:

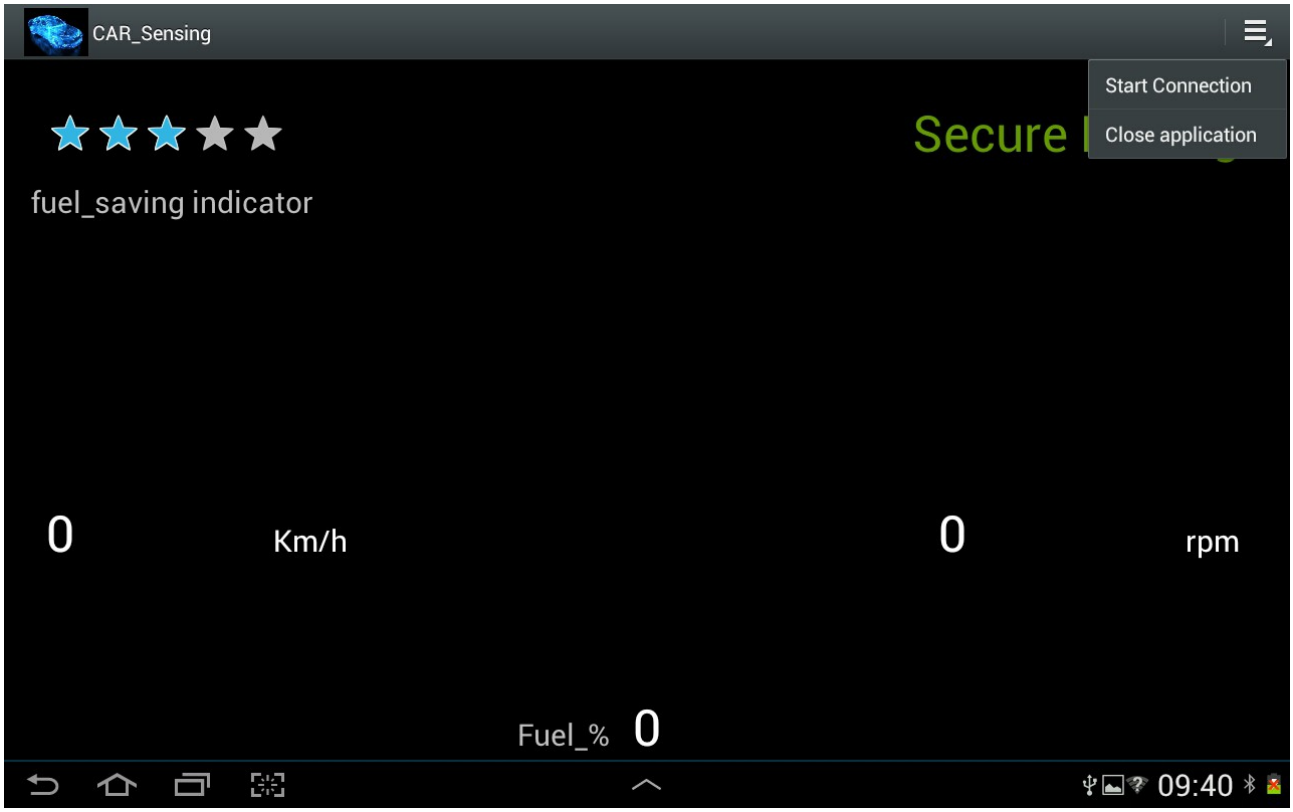




il layout dello screen principale presenta nella parte superiore, rispettivamente un indicatore dello stile di guida relativo al fuel consuption sulla sinistra e un indicatore di sicurezza sulla destra.

Nella parte inferiore invece sono presenti gli indicatori grafici di velocità livello del carburante e giri del motore.

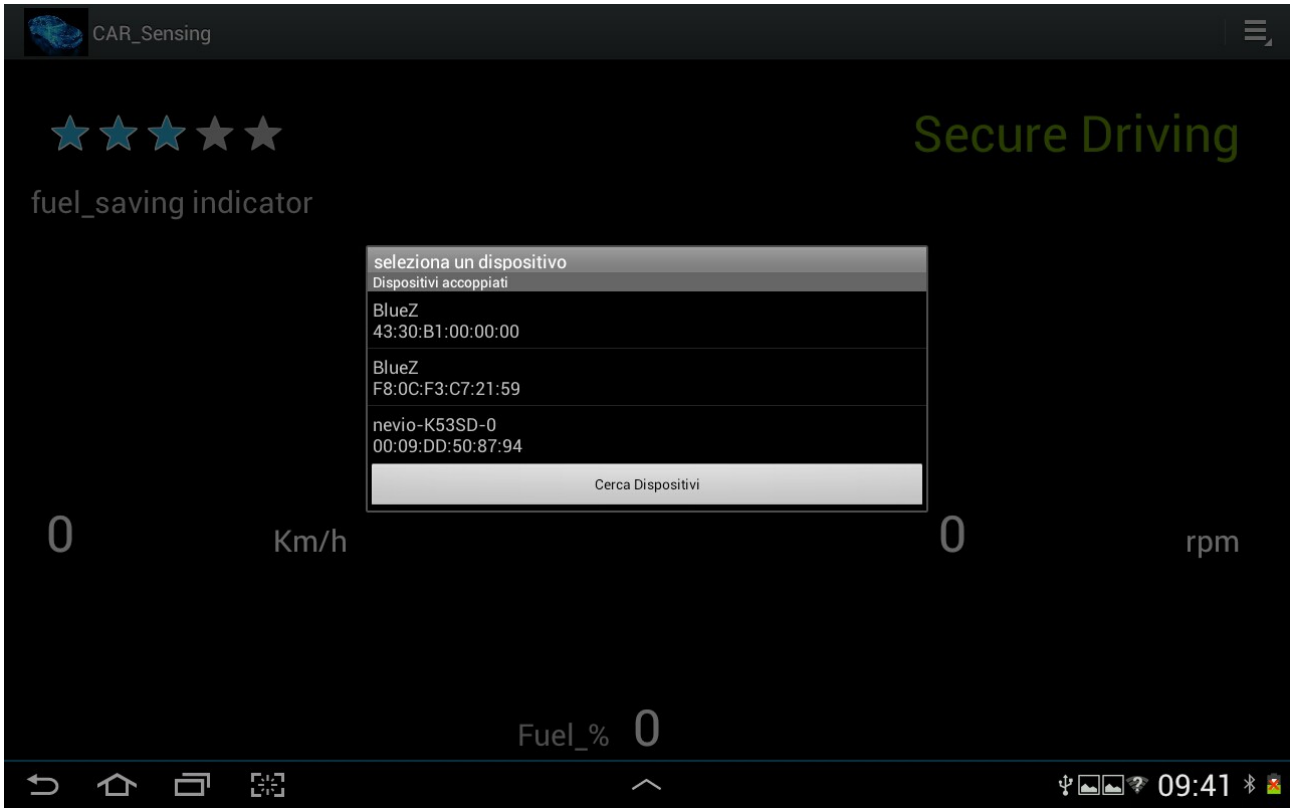
Mediante l'apposito menu delle opzioni è possibile avviare la connessione, selezionando la voce “Start Connection” oppure chiudere l'applicazione rilasciando le risorse utilizzate mediante la voce “Close application”.



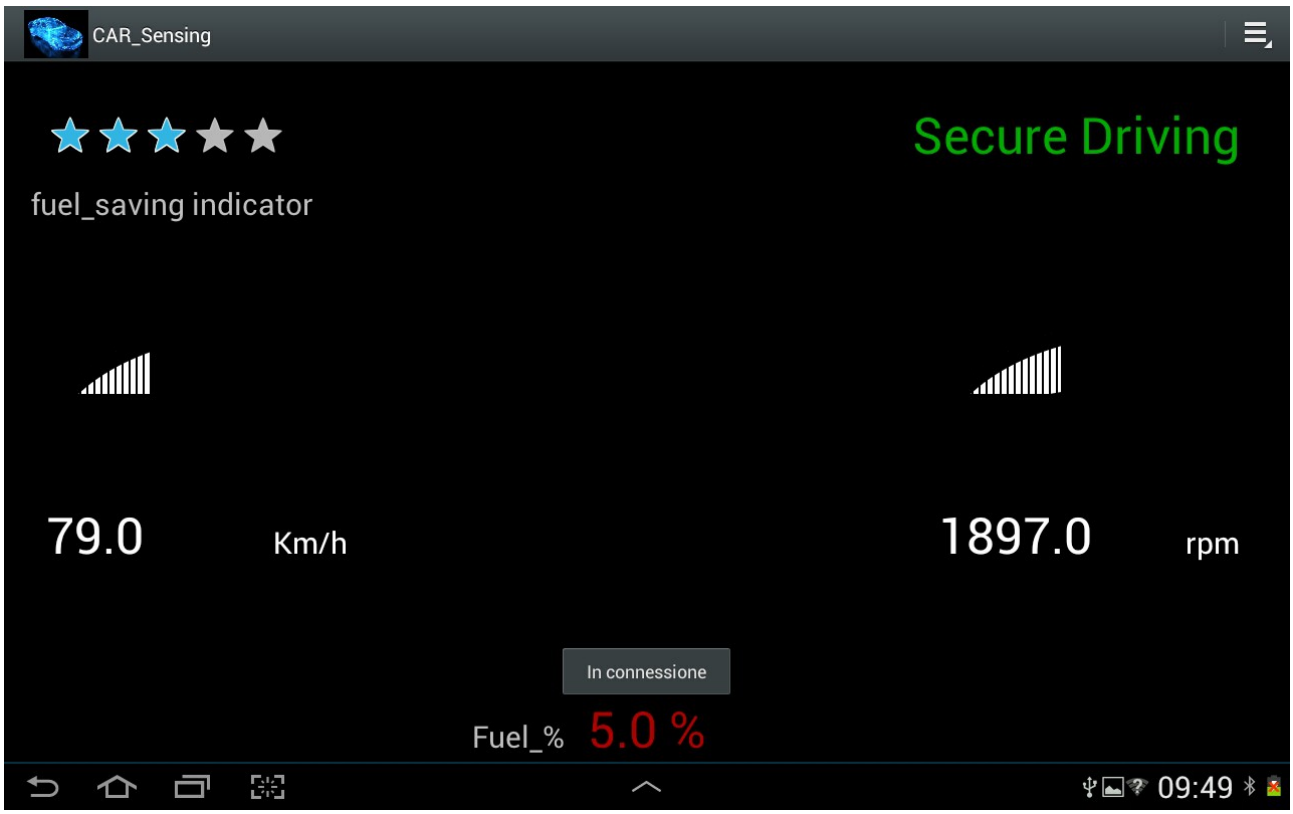
Selezionando il comando di avvio connessione il dispositivo ricercherà tutti i terminali dotati di

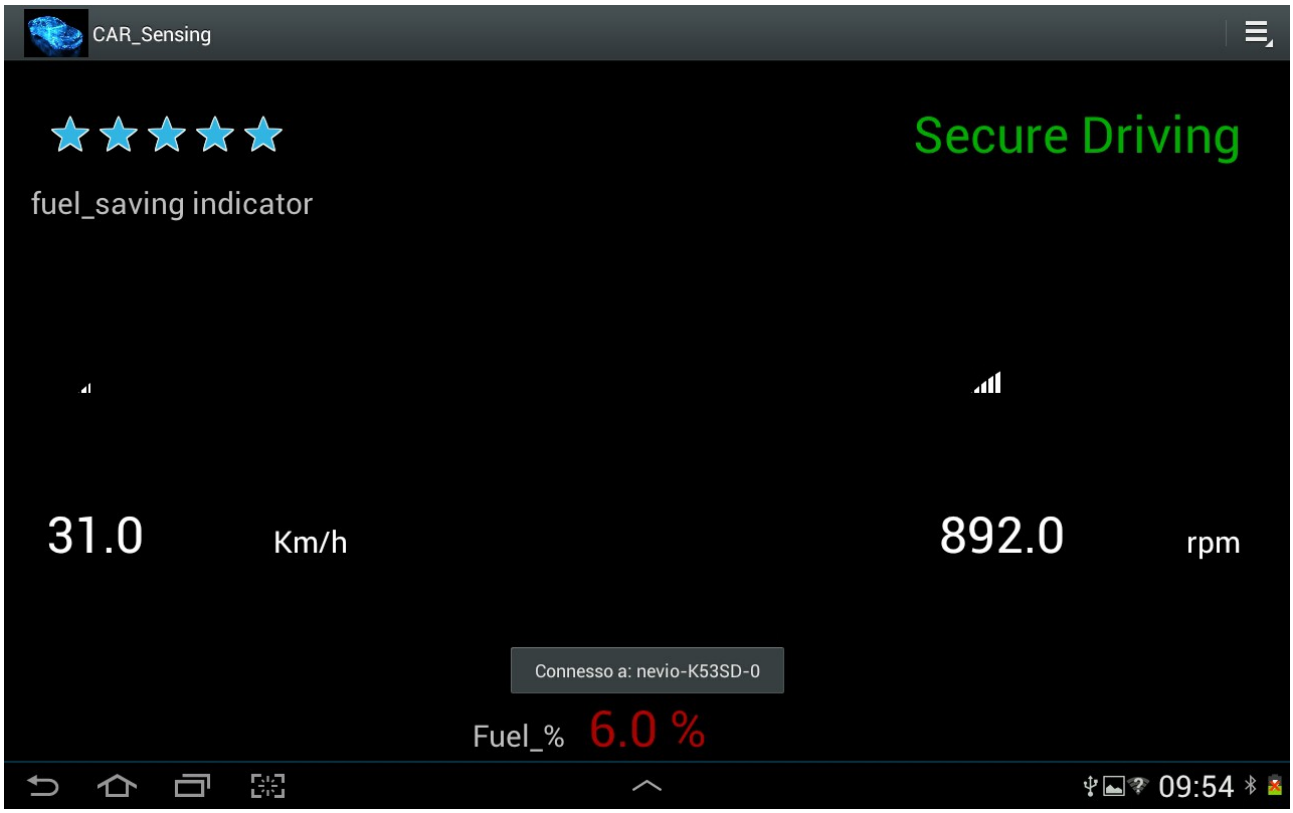
bluetooth disponibili, discriminando quelli già noti o “paired”.

Da tale menù è possibile quindi scegliere un dispositivo noto oppure avviare il discovery alla ricerca di nuovi terminali.

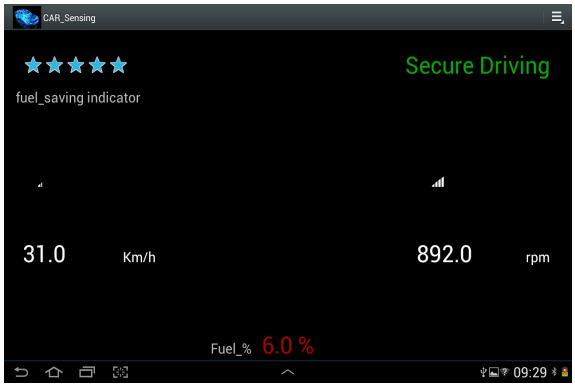


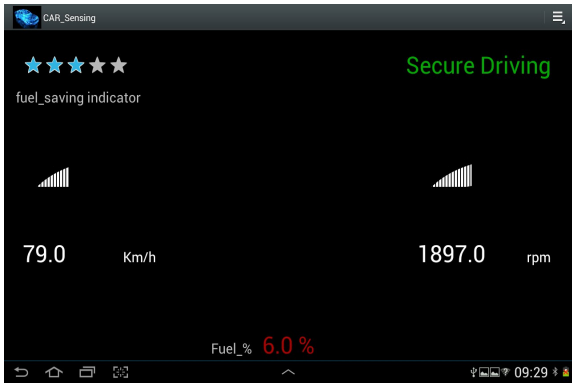
Una volta effettuata la scelta il sistema tenta di connettersi al dispositivo scelto ed in caso positivo avvia un socket di comunicazione tra i dispositivi, oppure mostra un messaggio sul problema riscontrato.

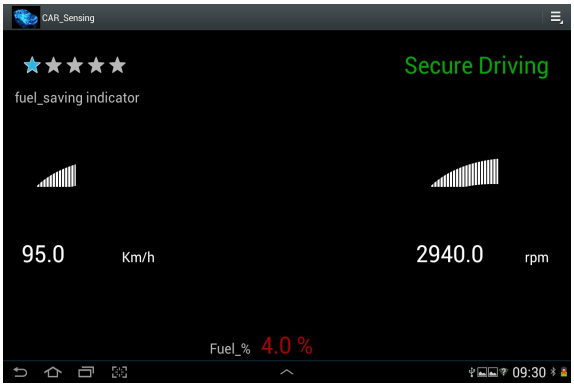


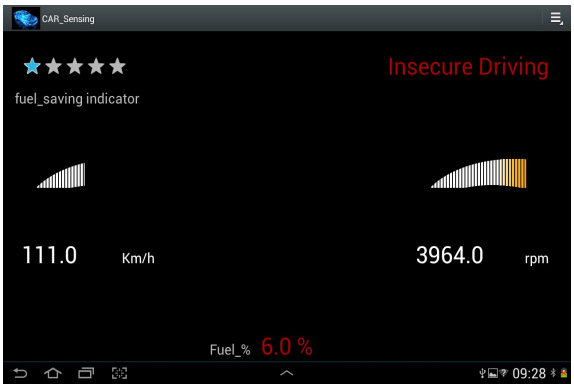


Una volta instaurata la connessione il dispositivo mobile si interessa di ricevere le informazioni elaborate dal sistema On_Board mostrandole a video sotto forma testuale e in grafici riepilogativi:

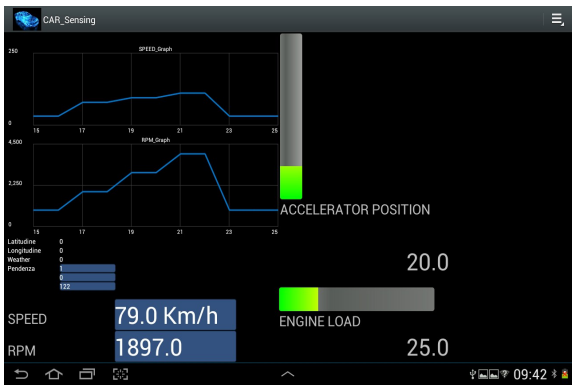


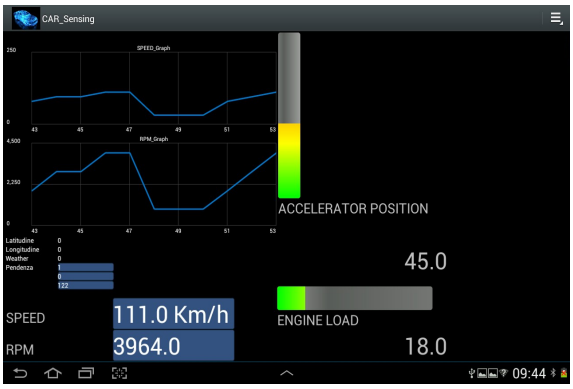




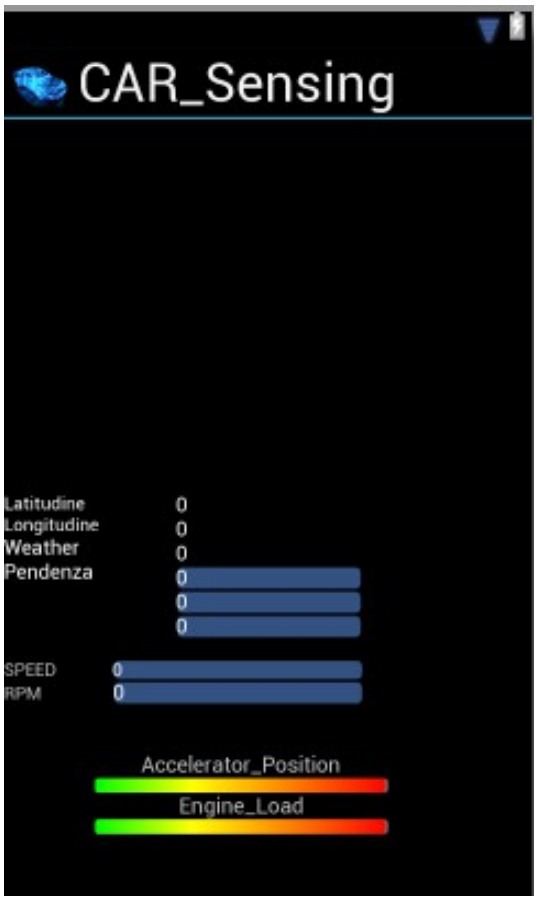


Nello screen secondario vengono mostrate informazioni riepilogative relative ai grafici dell' rpm e della velocità, indicatore grafico del carico del motore e della posizione dell'acceleratore , pendenza veicolo, posizione geografica e condizione meteo se presente una connessione internet.





Per compatibilità con i dispositivi di dimensione inferiore a quelle di un comune tablet, sono stati predisposti altri due layout per schermi di 4 pollici come di seguito:



mediante tale layout è possibile usufruire del sistema di consultazione anche su dispositivi comuni come gli smartphone, spesso in possesso di un utente rispetto ad un tablet dalle dimensioni maggiori.

5. RISULTATI

La scelta di utilizzare un modello Random Forest deriva da un'analisi compiuta mediante l'avvio del sistema attraverso il modulo Tester, e dal quale raccolti i dati qualitativi ottenuti tra il Random Forest J48 e Naïve Bayes si è deciso di utilizzare quello che meglio descrivesse la realtà.

Partendo dal presupposto che i modelli utilizzati non si discostano particolarmente nei risultati ottenuti, la scelta è stata legata alla ricerca di un trade-off tra efficienza e occupazione di spazio.

Per quanto riguarda il livello di bontà di ogni modello è stato valutato come il modello stesso riuscisse a classificare tuple la cui appartenenza ad un definito attributo classe fosse già noto, e mediante un metodo di analisi misto tra valutazione del Training set e cross correlazione con divisione dei dati in più “folds” ha prodotto i seguenti risultati, a partire da un set di 231 elementi derivanti dalla creazione del modello.

Analizzando la percentuale di entry correttamente classificate o meno, si riscontra che la percentuale ottenuta dal RandomForest come si può vedere dall 'illustrazione 38 è maggiore rispetto agli altri algoritmi.

La scelta deriva da un'attennta analisi comparativa, tra i modelli basati su albero decisionale , la cui predisposione per tali analisi è ben nota, e il modello a reti Bayesiane il cui carico computazionale e occupazione di spazio hanno reso interessante l'analisi comaprativa,

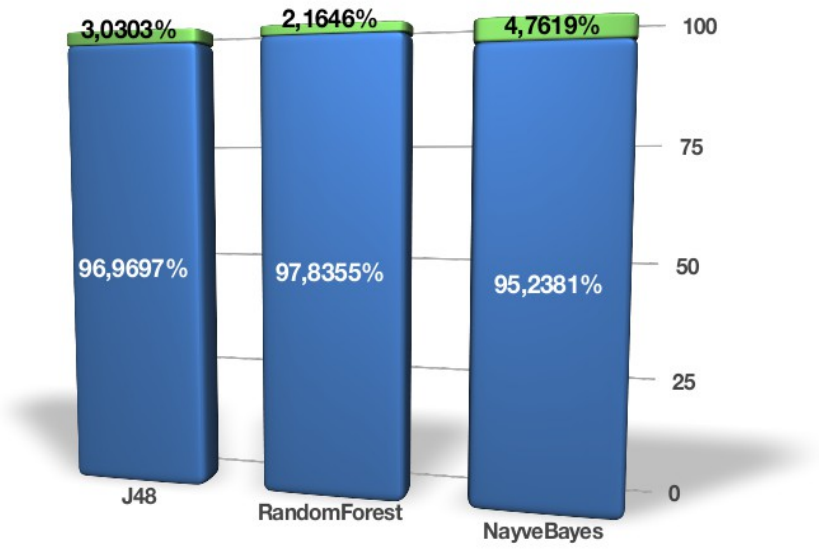


Illustrazione 37: percentuale di corretta Classificazione J48-RandomForest- NaïveBayes

Analogamente mediante l'analisi del Kappa Statistic descritto nel terzo Capitolo si evince che il grado di affidabilità della classificazione svolta con il modello Random Forest risulta la più coerente con il contesto analizzato, come si evince dalla figuraY.

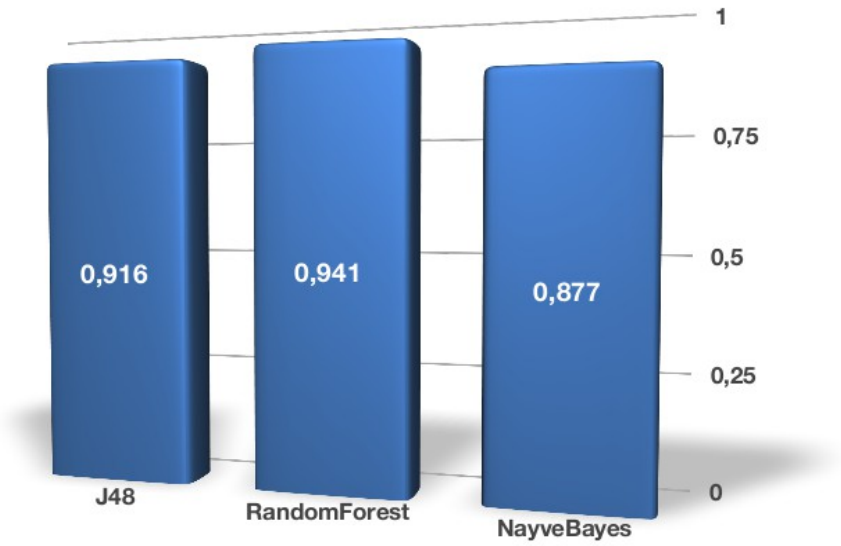


Illustrazione 38: Kappa Statistic classificazione J48-RandomForest- NaïveBayes

Come si evince dai risultati proposti la scelta di valutare l'intero dataset con l'algoritmo random

forest è legato alla sua percentuale di corretta classificazione in ambito di cross classificazione, relativo alla valutazione della classificazione del dataset di apprendimento già noto.

Analizzando i parametri riportati in precedenza si evince che tutti e tre i modelli riescono ben a rappresentare un modello della realtà analizzata avendo indici di corretta classificazione e kappa statistic molto alti.

Come riportato nei primi capitoli un indice di corretta classificazione del dataset di training del 0,95:0,97 è un kappa statistic quasi prossimo ad 1(valore massimo raggiungibile), ricordando che un 0,8<kappa<1, indica un'ottima concordanza dell'indice nel valutare accuratezza e affidabilità).

La scelta di utilizzare un classificatore Random Forest è da attribuire anche a logiche di occupazione di spazio, sfruttando infatti l'indice di Konenko & Bratko Information score espresso in bits si ottengono i seguenti risultati riportati in Illustrazione 40:

	K&B Information Score		Class Complexity I Order 0		Class Complexity I Scheme		Total Number of Istances
J48	197.1957 bits	0.8537 bits/instance	223.89 bits	0.9692 bits/instance	4313.4222 bits	18.6728 bits/instance	231
RandomForest	195.6915 bits	0.8471 bits/instance	223.89 bits	0.9692 bits/instance	1094.0853 bits	4.7363 bits/instance	231
NaïveBayes	187.337 bits	0.811 bits/instance	223.89 bits	0.9692 bits/instance	849.4723 bits	3.6774 bits/instance	231

Illustrazione 39: Occupazione in bits Classificazione J48-RandomForest- NaïveBayes

si nota quindi che per un classificatore con 231 to-learning tuples ed un file di 5 to-classify tuples il

risparmio in quantità di bits utilizzati per classificare è migliore utilizzando un modello Naïve Bayes, ed in ordine random forest e j48, ed analogamente considerando la complessità in ordine di bits utilizzati per costruire il modello.Da questi risultati si evince che dovendo scegliere il modello sulla base della quantità di bit utilizzati sia per descrivere il modello che i dati da classificare,si dovrebbe optare per un Naïve Bayes, mentre per la accuratezza un Random forest. Analizzando comunque il rapporto di qualità di classificazione e spazio occupato si evince che il vantaggio di una classificazione più accurata soprattutto considerando che tale accuratezza risulta importante all'aumentare dei dati del training test. Considerando Naïve Bayes e random forest la differenza è circa 30 byte in più del random forest per una accuratezza pari a circa il 97% rispetto al 95% circa raggiunto dal modello Naïve.

Da tali risultati quindi si dimostra l'aderenza a quanto detto in relazione allo studio “artemisia “ secondo il quale il modello più semplice risulta essere il Naïve Bayes ma relativamente alle performance risulta carente, ed analogamente il J48 pur ottenendo valori di accuratezza molto vicini, ma mai superiori, presenta un carico computazionale e di spazio occupato maggiore rispetto al random forest che quindi viene scelto come algoritmo e modello di classificazione.

Relativamente alla raccolta dati ,esclusa la fase preliminare per costruire il modello, tutti i dati sono stati riprodotti artificialmente con il modulo tester. Valutando il lavoro svolto su strada si è comprovata una altissima coerenza e usabilità del sistema , in quanto le logiche definite risultano perfettamente adatte a classificare le dinamiche che avvengono durante una prova di guida rispondendo positivamente alle variazioni dello stile di guida sia per quanto riguarda la componente di Fuel Saving che Safety Driving. Di seguito i risultati ottenuti provando il prodotto su strada mediante l'ausilio di un pc corredato del modulo On_Board e uno smartphone. Provando infatti lo stesso circuito utilizzato per la classificazione del modello e valutando una guida libera da qualsiasi imposizione si è riscontrato che il sistema riesce a catturare le dinamiche della guida svolta classificandone opportunamente ogni componente.

6.CONCLUSIONI

Relativamente a quanto prefissato in fase di progettazione, si è definita l'intera architettura di classificazione sul sistema centrale, realizzato un framework di gestione della connessione al sistema OBD e al dispositivo mobile che funge da terminale aggiuntivo di presentazione delle elaborazioni e classificazioni effettuate.

Rispetto alle soluzioni alternative valutate il progetto sviluppato presenta il vantaggio di una immediata spendibilità del risultato di classificazione, oltre alla possibilità di estendere le variabili di analisi con le componenti dello smartphone quali posizione gps o pendenza del veicolo non analizzabili in sistemi privi di gps e accelerometro.

Relativamente alla possibilità di estendere le funzionalità si demanda al capitolo seguente

7.SVILUPPI FUTURI

A partire dal prodotto sviluppato, diversi possono essere gli aspetti evolutivi implementabili, soprattutto legati all'aspetto di rendere il modulo smartphone più indipendente integrando in esso la classificazione, relativamente al miglioramento del modello di classificazione e clouding delle info per ricostruzioni e analisi successive dei dati memorizzati.

L'aspetto di indipendenza del modulo presente sullo smartphone rientra nell'ottica di progettare un sistema in cui ogni componente risulti indipendente. L'unico elemento che rende lo smartphone legato alla presenza di un dispositivo on-board che ritrasmetta ad esso i dati elaborati , nasce solo da motivazioni logistiche , che nel contesto del lavoro sviluppato, hanno reso necessario l'uso di tale logica.

Mediante un comune adattatore OBD-Bluetooth sarebbe comunque possibile trasformare lo smartphone da visualizzatore ad elaboratore dei dati integrando in esso la libreria weka ed il codice

sviluppato sul sistema On_Board e relativo alla lettura dei dati e classificazione al momento non presenti. Per quanto riguarda invece il miglioramento della classificazione definita, questa rientra nella casistica di espandere il set di attributi utilizzati mediante miglioramento dell'analisi a seguito di prove su strada del lavoro stesso. Inoltre sarebbe anche possibile integrare gli attributi rilevabili solo dallo smartphone in un contesto in cui esso sia sicuramente presente in tutte le fasi di analisi.

Tale particolare, ovvero la possibile assenza di un dispositivo mobile comporterebbe una impossibilità di avviare l'algoritmo di classificazione qualora quei dati siano assenti data la rigida strutturazione imposta dal framework weka.

L'ultimo aspetto espansivo considerabile è relativo all'aggiunta di una componente cloud, mediante la quale memorizzare in una base dati centralizzata tutte le informazioni ricevute dallo smartphone mediante la connessione dati presente su esso.

Tale aspetto permetterebbe di ricostruire periodi di analisi molto più grandi rispetto alla quantità di memoria a disposizione degli smartphone con capacità di calcolo sicuramente a livello di mainframe piuttosto che quelli di uno smartphone.

Tra le logiche implementabili sicuramente si potrebbe acquisendo cognizione del tipo di strada e del tipo di traffico(per esempio mediante le google maps) rilevando determinate condizioni in cui la classificazione risulterebbe fortemente influenzata, permettendo di condividere un po' come nei servizi caratteristici dei social network informazione personali prelevando informazioni per le quali si è manifestato interesse come per il servizio di google now.

La centralizzazione di tali informazioni potrebbero essere monitorate anche da enti terze come assicurazioni o società di controllo rivoluzionando come già si può constatare il modo di viaggiare e condividere dati utili quando necessari.

La memorizzazione potrebbe risultare un aspetto utile nella ricostruzione di incidenti o nella produzione di allarmi che manifestino agli utenti in prossimità di luoghi che manifestano condizioni particolari di acquisire cognizione di cosa stia succedendo.

8. BIBLIOGRAFIA

[1]Kuhler, M., Kartens, D., “Improved driving cycle for testing automotive exhaust emissions”. SAE Technical Paper Series 780650. 1978

[2]André, M., 1996. Driving cycles develop-ment: Characterization of the methods. SAE Technical Papers Series 961112

[3]Fomunung, I., Washington, S., Guensler, R. “A statistical model for estimating oxides emissions from light duty motor vehicles”. Transporta-tion Research Part D, pp.333-352. 1999

[4]Probabilistic driving style determination by means of a situation based analysis of the vehicle data Bär, Tobias ; Nienhüser, Dennis ; Kohlhaas, Ralf ; Zöllner, J. Marius Publication Year: 2011 , Page(s): 1698 - 1703

[5]ELman j.l.(1990) FINDING STRUCTURE IN TIME, COGNITIVE sCIENCE, 14,179-211

[6]Rutkowski, L. (2005), Method and Techniques of artificial intelligence,polish scientistpublisher pwn. warsaw.

[7]Augustynowicz (2004) Estimation of driver's intention based on accelaration pedal signal, Int. Automotive Cong.Konmot-Autoprogres, Zakopane,Poland, 59-66

[8]Augustynowicz and Bartecki, (2006) Estimation of driving characteristics by the application of

Elman's recurrent neural network. The Archives of Trasport,Warsaw 18,4,5-13

[9]Constantinescu Z., Marinoiu C., Vladoiu M., Driving Style Analysis Using Data Mining Techniques,*International Journal of Computers Communications & Control*, ISSN 1841-9836, 5(5): 654-663, 2010.

[10]Ward'method, http://en.wikipedia.org/wiki/Ward%27s_method

[11]Driver Modeling Based on Driving Behavior and Its Evaluation in Driver Identification V.Corcoba Magaña M. Muñoz-Organero vcorcoba@it.uc3m.es mmunoz@it.uc3m.es

[12]Random Forest, http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

[13]Artemisa: An eco-driving assistant for Android Os ,Corcoba Magana, V. ; Munoz-Organero, M. Publication Year: 2011 , Page(s): 211 - 215

[14]Artemisa: Using an Android device as an Eco-Driving assistant V. Corcoba Magaña and M. Muñoz Organero

[15]Database pubblico, http://archive.ics.uci.edu/ml/

[16] CAN_BUS, https://it.wikipedia.org/wiki/Controller_Area_Network

[17] WEKA FRAMEWORK http://www.cs.waikato.ac.nz/ml/weka/

[18] OpenWeather webServer http://openweathermap.org/

[19] graphView library, http://www.jjoe64.com/p/graphview-library.html

Sitografia

[1] http://dev.mysql.com/downloads/connector/

[2] http://dev.mysql.com/downloads/connector/j/#downloads

Ringraziamenti

Ringrazio Dio per qualsiasi persona mi abbia fatto incontrare ogni giorno, nel bene o nel male mi hanno reso ciò che sono oggi.

Nevio Longo