UNIVERSITÀ DI PARMA

Dipartimento di Ingegneria e Architettura

Corso di Laurea Triennale in Ingegneria Informatica, Elettronica

e delle Telecomunicazioni

Software di Comunicazione con Distanziometro Laser per Calibrazione Sensori ADAS

Software for Communication with Laser Device for ADAS Sensor Calibration

Relatore:

Prof. Andrea Prati

Tesi di Laurea di:

Laura Rivi

Correlatore:

Ing. Mauro Disaro

Anno Accademico 2019-2020

Grazie alla mia famiglia e

a tutte le persone che fanno o hanno fatto parte della mia vita in questi anni

per l’affetto, il sostegno e l’incoraggiamento.

Introduzione **1**

Cos'è la calibrazione di sensori ADAS? ... 3

Cosa sono i Cruise Control Adattivi? ... 4

Come viene effettuata la calibrazione? ... 5

Strumenti di sviluppo **7**

1.0 Hardware ... 8

1.0.1 Distanziometro Laser Bosch Glm50C Professional ... 8

1.1 Software ... 9

1.1.1 Microsoft Visual Studio 2019 ... 9

1.1.2 Mozilla Firefox ... 10

1.2 Architettura ... 11

1.2.1 Master-slave ... 11

1.3 Protocolli ... 13

1.3.1 Bluetooth 4.0 ... 13

1.3.2 RFCOMM ... 14

1.4 Linguaggi ... 15

1.4.1 C++ ... 15

1.4.2 HTML5 ... 17

1.4.3 CSS ... 18

1.4.4 JavaScript ... 18

1.5 Utility ... 19

1.5.1 Normalize ... 19

1.5.2 Json 19
1.5.3 Web socket 20
1.5.4 SHA-1 21
Descrizione del Sistema Sviluppato 22
2.0 Struttura del software 23
2.1 Modalità di misurazione 23
2.1.1 Modalità Live Measure (Master) 24
2.1.2 Modalità Wheel Measure (Slave) 25
2.2 Modello client - server 26
2.3 Server 27
2.3.1 Design Pattern di programmazione 27
2.3.1.1 Observer 28
2.3.1.2 Abstract factory 29
2.3.2 Diagramma UML delle Classi 30
2.3.3 Elenco delle classi 31
2.3.3.1 Classe Device 31
2.3.3.2 Classe DeviceFactory 32
2.3.3.3 Classe Test 32
2.3.3.4 Classe Glm 33
2.3.3.5 Classe WheelMeasure 33
2.3.3.6 Classe Observer 34
2.3.3.7 Classe MeasureCollector 35
2.3.3.8 Classe WebSocket 36
2.4 Client 38
2.4.1 Struttura Pagine Web 39

2.4.1.1 Head 39
2.4.1.2 Body 39
2.4.2 Elenco Pagine Web 41
2.4.2.1 index.html 41
2.4.2.2 instruction.html 42
2.4.2.3 settings.html 42
2.4.2.4 start.html 43
2.4.2.5 wheelMeasureAuto.html 44
2.4.2.5.1 Avvio del server 45
2.4.2.5.2 Save 46
2.4.2.6 liveMeasure.html 46
2.4.2.7 style.css 48
Conclusioni **49**
Bibliografia **51**

Introduzione

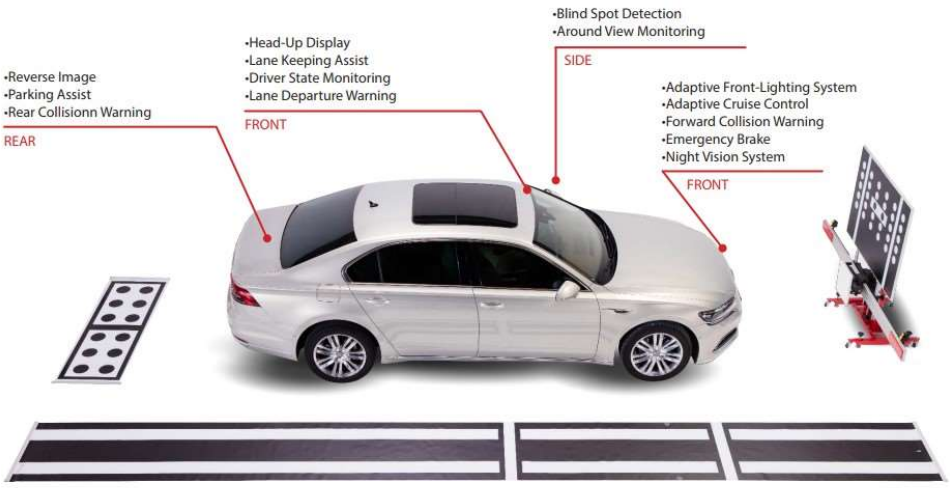
Lo scopo di questo progetto di tesi consiste nella progettazione e relativo sviluppo di un software finalizzato ad automatizzare alcune procedure aziendali.

Questo progetto è avvenuto in collaborazione con Bosch Automotive Service Solutions S.r.l. affiliata dell'azienda multinazionale tedesca ROBERT BOSCH S.p.A.

L'azienda, avente sede a Corcagnano in provincia di Parma, si occupa di produzione e commercializzazione apparecchi e servizi per la manutenzione di autoveicoli, motocicli e veicoli pesanti.

Il software sviluppato nasce dal desiderio di digitalizzare la rilevazione di alcune distanze utili in fase di preparazione alla calibrazione dei sensori ADAS.

Queste distanze, ad oggi, vengono misurate con metro a nastro o con l'utilizzo di distanziometro laser e, in seguito, registrate manualmente.

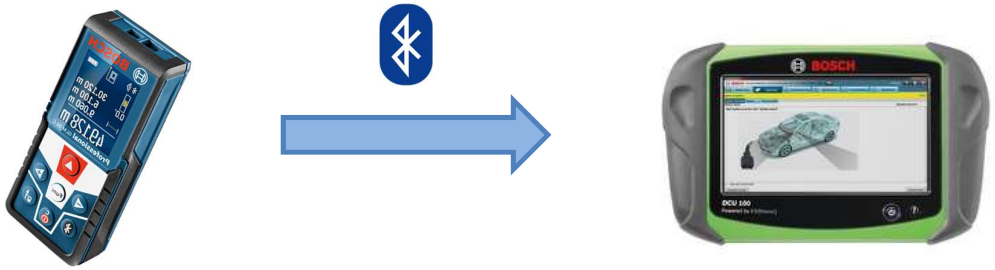


La rilevazione con distanziometro laser permette di stabilire le distanze con un margine di precisione e accuratezza elevato rispetto alla tradizionale, ma ormai in disuso, misurazione con metro a nastro.

Per velocizzare la sopracitata fase di preparazione del veicolo, è stato quindi pensato di sfruttare la funzione di connessione, tramite bluetooth, offerta dal distanziometro laser per ricevere direttamente in formato digitale il risultato delle misurazioni.

Questa ricezione diretta di dati consente di disporre dei risultati in tempo reale e di evitare tutti quei possibili errori legati alla lettura della distanza o alla inesatta trascrizione della stessa, elimina inoltre il rischio che un dato, seppur corretto, venga inserito o registrato in campo non coerente.

Nella progettazione del software è stata inoltre predisposta la funzione di salvataggio delle misurazioni, che offre come vantaggi la storicizzazione ed il futuro recupero delle distanze registrate o di una loro verifica in caso di necessità.

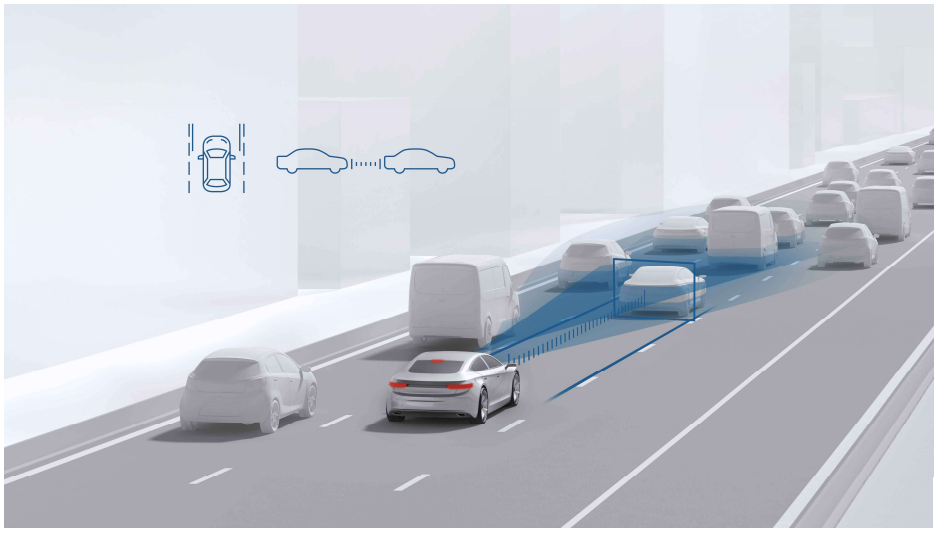


Cos'è la calibrazione di sensori ADAS?

Gli Advanced Driver Assistance Systems sono, in breve, ausili tecnologici installati sull'autovettura che possono intervenire in maniera autonoma qualora se ne ravvisasse la necessità.

Sono dunque dotati di un sistema di sensori (videocamere, radar o lidar) in grado di riconoscere oggetti e calcolarne posizione, dimensione e distanza e di regolare esposizione e messa a fuoco su di essi; per poterlo fare analizzano con sofisticati algoritmi i dati delle loro foto/videocamere e si avvalgono di informazioni radar.

Quest'ultimo sensore è inoltre indispensabile per il funzionamento dei Cruise Control Adattivi.



Cosa sono i Cruise Control Adattivi?

Il Cruise Control Adattivo (ACC – Adaptive cruise control) è un'evoluzione del cruise control classico. Un dispositivo che, se inserito, mantiene la velocità impostata, indipendentemente dalle condizioni esterne e rende quindi indispensabile l'intervento del conducente per rallentare o accelerare.

L'ACC monitora anche la distanza rispetto al veicolo che viaggia davanti, riducendo o aumentando l'andatura in base alle circostanze stradali, per farlo interviene su freni e motore per mantenere costante la distanza impostata e, se possibile, raggiungere la velocità desiderata quando la corsia torna ad essere più libera.

Esiste inoltre l'adaptive cruise control intelligence (i-ACC) che, oltre a prevedere ere e reagire prontamente all'eventuale "invasione" di corsia da parte di altri veicoli, rileva i limiti di velocità permettendo al conducente di reimpostare la velocità della vettura adeguandola al nuovo limite semplicemente utilizzando i comandi sul volante.

In qualsiasi momento è sufficiente una breve pressione dell'acceleratore o del freno per disattivare il sistema e assumere il pieno controllo del veicolo.

L'ACC offre numerosi vantaggi in termini di sicurezza rilevando e intervenendo in pochi millesimi di secondo in caso di pericolo, tempo decisamente inferiore rispetto all'essere umano che in media reagisce in un secondo.

Un sensore può non essere più calibrato a causa di un suo spostamento dovuto ad un urto o in seguito a sostituzione del suo supporto o di parti dell'auto che si frappongono tra il sensore e l´ambiente esterno (ad esempio il parabrezza per le telecamere) e necessita di essere nuovamente calibrato.

Come viene effettuata la calibrazione?

Per calibrare il radar si usa specifica superficie metallica in grado di riflettere il segnale radar da posizionare a terra; l'auto va poi sistemata in una specifica posizione rispetto allo specchio stesso tramite un puntatore laser.

La calibrazione viene attivata utilizzando un protocollo specifico del veicolo.

Per questa operazione viene utilizzato un sistema di diagnosi collegato al veicolo tramite l'interfaccia standard OBD.

Per la videocamera anteriore si posiziona invece davanti alla vettura un target dotato di specifico pattern che varia a seconda della marca di autovettura da ricalibrare e si procede poi con lo strumento di diagnosi.

Le distanze e gli angoli di inclinazione dell'auto rispetto al target e agli altri pannelli vengono calcolate direttamente da telecamere.

Nell'ambito di questa fase può essere richiesto lo spostamento del target a diverse distanze dal veicolo, assicurandosi di mantenere costante l'altezza del target rispetto a terra, rilevata a partire dal suo baricentro.

La procedura di calibrazione di alcuni veicoli richiede come input l'altezza degli apici dei quattro passaruota dalla superficie su cui poggiano le ruote.

Questa informazione viene utilizzata dal veicolo per compensare eventuali angoli di beccheggio e rollio della scocca del veicolo (ad esempio a causa di carichi presenti sul veicolo, pressione delle gomme, posizione delle sospensioni...)

Capitolo 1

Strumenti di sviluppo

In questo capitolo, si riporteranno i dettagli sulle risorse hardware e software che sono state utilizzate durante lo sviluppo del progetto. Verranno descritti il Distanziometro Laser, gli ambienti di sviluppo, i protocolli di comunicazione ed i linguaggi di programmazione utilizzati.

1.0 Hardware

In questo sotto-capitolo verrà riportata una descrizione dettagliata del componente hardware utilizzato specificandone caratteristiche tecniche e principio di funzionamento

1.0.1 Distanziometro Laser Bosch Glm50C Professional

Il Distanziometro Laser è uno strumento di alta precisione per la misura di distanze fra due punti, il cui funzionamento si basa sull'emissione di un raggio laser a bassa energia.

Il principio di funzionamento dei Distanziometri Laser portatili si basa sulla misura della frequenza di auto-oscillazione, il cui periodo è determinato dal tempo di andata-ritorno del raggio emesso.

Per lo sviluppo e la successiva fase di test del software realizzato è stato utilizzato il Distanziometro laser Bosch GLM 50C Professional, che consente un trasferimento rapido di dati tramite il collegamento Bluetooth 4.0.

1.1 Software

In questo sotto-capitolo verrà riportata una descrizione dettagliata delle componenti software utilizzate, ovvero l'ambiente di sviluppo ed il browser specificandone caratteristiche e motivazione della scelta.

1.1.1 Microsoft Visual Studio 2019

Microsoft Visual Studio 2019 è un IDE sviluppato da Microsoft che supporta numerosi linguaggi di programmazione come C, C++, C#, Visual Basic ed F#, ed è dotato di un editor di codice avanzato, un debugger e vari tool di analisi.

Supporta, inoltre, un gran numero di plug-in, che facilitano lo sviluppo e l'integrazione con altri sistemi.

Visual Studio è l'ambiente di sviluppo scelto per la realizzazione del software sviluppato, utilizzando il compilatore MSVC++ 14.2.

1.1.2 Mozilla Firefox

È un browser web open source prodotto da Mozilla Foundation, l'organizzazione non-profit fondata per supportare, organizzare e rilasciare le versioni dei software Mozilla.

La sua struttura, realizzata seguendo il rispetto degli standard web, garantisce stabilità e sicurezza e risulta sufficientemente protetta dall'attacco di fastidiosi spyware/adware.

L'interfaccia grafica del software sviluppato è stata progettata e testata utilizzando Mozilla Firefox come browser, ma resta compatibile anche con gli altri browser maggiormente utilizzati.

1.2 Architettura

In questo sotto-capitolo verrà indicata l'architettura del software realizzato.

1.2.1 Master-slave

Per la comunicazione e lo scambio di dati tra distanziometro laser ed il software sviluppato si ricalca un modello architetturale di tipo master-slave.

Un'architetture Master-Slave permette di creare un rapporto tra componenti hardware o software, in cui uno ha il pieno controllo dell'altro.

Nello specifico, un dispositivo alla volta impegna il canale di comunicazione, prendendo il ruolo di Master, trasmette un messaggio che viene ricevuto da uno o più slave che rispondono secondo una gerarchia prestabilita.

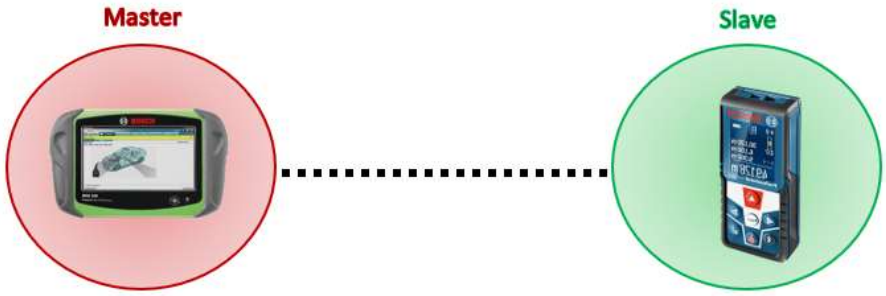
La Comunicazione si conclude quando l'ultimo messaggio di risposta di uno slave giunge al Master o scade un time-out.

Va sottolineato che in questo tipo di architettura il ruolo di Master o Slave è dinamico, in quanto in ogni "sessione" il Master è il dispositivo che inizia la comunicazione.
Tipicamente l'architettura Master/Slave si usa quando si devono stabilire comunicazioni solide, in cui bisogna assicurarsi che le richieste vengano sempre evase dal dispositivo destinatario.

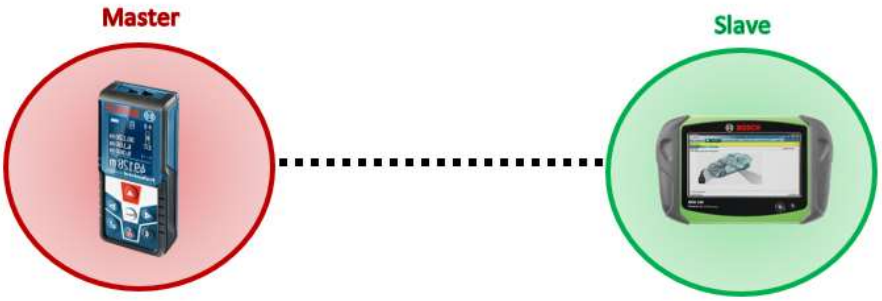
Tutte le comunicazioni seriali a basso livello sono di tipo Master/Slave.

Il software sviluppato prevede due modalità di invio/ricezione dati:

- Master: il software chiede al distanziometro laser di effettuare una misurazione e preleva i dati rilevati



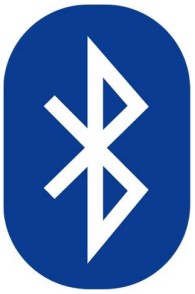
- Slave: il software resta in ascolto del distanziometro, il quale effettua la misura e invia i dati rilevati



1.3 Protocolli

In questo sotto-capitolo verranno riportati i protocolli di comunicazione utilizzati per lo scambio di dati tra distanziometro laser e software sviluppato, quali bluetooth e RFCOMM.

1.3.1 Bluetooth 4.0



Bluetooth è uno standard tecnico-industriale di trasmissione dati per reti personali senza fili (WPAN: *Wireless Personal Area Network*).

Fornisce un metodo standard, economico e sicuro per scambiare informazioni tra dispositivi diversi attraverso una frequenza radio sicura a corto raggio, in grado di ricercare i dispositivi coperti dal segnale radio entro un determinato raggio, mettendoli in comunicazione tra loro.

Il protocollo Bluetooth 4.0 prevede una nuova modalità di funzionamento a basso consumo (“Low Energy”) nella banda di frequenza a 2,4 GHz, utilizzata dalle precedenti versioni di questo standard.

Il software sviluppato utilizza questo protocollo di comunicazione per instaurare una connessione con il distanziometro laser e scambiare dati con esso.

Si è scelto di utilizzare il protocollo Bluetooth in quanto e l'unico metodo di scambio dati a distanza tra distanziometro laser e software installato su PC o tablet.

1.3.2 RFCOMM

È un protocollo che emula una connessione seriale RS-232 tra due dispositivi.

Consente di ricalcare i segnali RS-232 di dati e controllo, nella Baseband di Bluetooth; inoltre fornisce abilità di trasmissione per servizi di livello superiore che, in caso contrario, utilizzerebbero una connessione seriale come meccanismo di trasmissione.

La comunicazione RFCOMM avviene attraverso una socket Bluetooth; le operazioni da effettuare per instaurare la connessione sono:

- apertura del socket
- scelta della modalità di comunicazione, connessione in modalità master o listening in modalità slave
- invio o ricezione di dati (rispettivamente Wright o Read dal Socket)
- chiusura del socket

Si è scelto di utilizzare questo protocollo di comunicazione per lo sviluppo del software in quanto indicato per trasferire stream di dati esattamente come si farebbe con una porta seriale.

Questo protocollo ha come svantaggio l'ordinamento dei byte soprattutto interfacciandosi con sistemi diversi, usa infatti un ordinamento Little-Endian.

1.4 Linguaggi

In questo sotto-capitolo verranno descritti i principali linguaggi di programmazione utilizzati.

1.4.1 C++

C++ è un linguaggio di programmazione standardizzato secondo l'ISO (International Organization for Standardization) ed è considerato vicino al linguaggio macchina ed efficiente, nonché altamente astratto e complesso.

C++ si basa su uno dei linguaggi di programmazione più vecchi: C.

I maggiori punti di forza di C++ sono la varietà di combinazione e l'efficace programmazione vicina al linguaggio macchina. Anche i processi altamente complessi si possono raggruppare in funzioni di base.

Il software sviluppato è stato scritto utilizzando il linguaggio nella sua versione C++11 e sono state sfruttate alcune delle utility con esso introdotte.

La libreria per il threading è una delle caratteristiche più importanti introdotte con C++11. Si tratta di una libreria piuttosto vasta i cui capisaldi sono:

- sdt:thread, una classe che rappresenta un thread in esecuzione
- costrutti per la sincronizzazione
- la funzione template async per l'avvio di task simultanei
- il tipo di archiviazione thread_local per la dichiarazione di dati unici per thread

Possiamo considerare un thread una parte del programma in esecuzione, esso può condividere uno spazio di indirizzamento con altri thread, a differenza di un processo che, generalmente, non condivide i dati in maniera diretta con altri processi.

Thread-Local Storage

Gli oggetti thread sono simili agli oggetti globali. Mentre gli oggetti globali hanno un campo di esistenza che copre tutta l'esecuzione del programma, gli oggetti thread hanno un campo di esistenza limitato ad un singolo thread, al cui termine la variabile non può essere più acceduta. Un oggetto thread può essere inizializzato come una qualsiasi variabile di durata statica, eventualmente impiegando un costruttore; se possiede un distruttore, questo verrà chiamato al termine del thread.

Puntatore nullo

Dal 2011 è stata inclusa una nuova parola chiave (nullptr) riservata esclusivamente per indicare il puntatore nullo.

Il nullptr non può essere assegnato ad un tipo intero, mentre può essere assegnato a qualsiasi puntatore e confrontato con esso.

Enumerazioni

A parte i tipi fondamentali e quelli composti, il linguaggio C++ consente di definire le enumerazioni.

Ogni enumerazione è un tipo. I valori che possono essere assunti da una variabile enumerazione sono ristretti ad un insieme di valori interi costanti, ad ognuno dei quali viene associato un nome. La coppia nome-costante è detta enumeratore.

Il compilatore assegna automaticamente i valori costanti associati ai nomi degli enumeratori ed il tipo intero di riferimento.

1.4.2 HTML5

HTML è l'acronimo di Hyper Text Markup Language, non è un linguaggio di programmazione, ma un linguaggio di markup che permette di indicare come disporre gli elementi all'interno di una pagina.

La disposizione dei vari componenti non richiede particolari vincoli, tuttavia si è scelto di utilizzare una programmazione modulare per avere maggiore leggibilità del codice in fase di scrittura, di correzione e soprattutto di eventuali modifiche future.

Questo permette al codice di poter essere facilmente letto ed eventualmente riutilizzato in futuro per realizzare interfacce collegate o per l'implementazione e ampliamento della stessa.

1.4.3 CSS

Si tratta di una serie di regole che permettono di definire l'aspetto e lo stile che devono assumere gli elementi sulla pagina. Dimensioni, colori, animazioni, ogni caratteristica visuale può essere manipolata.

Nello specifico la scelta è caduta su uno stile semplice ma elegante, che permetta un semplice utilizzo delle varie funzionalità ed una rapida individuazione della sezione relativa all'operazione che si desidera effettuare.

Per rendere questa interfaccia intuitiva sono stati inseriti pochi ed esaustivi comandi attivabili tramite il click di appositi pulsanti e corredati di immagini illustrative.

1.4.4 JavaScript

È un linguaggio di programmazione che consente di manipolare oggetti della pagina HTML: lo stile, i contenuti della pagina, ma soprattutto l'interazione con l'utente. Ci permette di creare la logica dell'interfaccia utente e di sfruttare le API messe a disposizione dal browser.

Anche nella scrittura di questo codice uno degli obiettivi di progettazione è stata oltre alla funzionalità del codice, la sua facile lettura a posteriori, dotandolo di commenti e mantenendo ordine e linearità.

1.5 Utility

In questo sotto-capitolo verranno descritti in dettaglio Normalize, Json, WebSocket e l'algoritmo SHA-1.

1.5.1 Normalize

È un file CSS personalizzabile che permette ai browser di visualizzare gli elementi in modo più coerente e in linea con gli standard moderni (HTML5 compreso).

Gli sviluppatori hanno studiato le differenze tra gli stili di default del browser, al fine di personalizzare solo gli stili che hanno bisogno di normalizzazione.

1.5.2 Json

JSON (JavaScript Object Notation) è un formato leggero per lo scambio di dati, facile da leggere e scrivere per gli esseri umani e facile da generare ed analizzare da parte delle macchine. Si basa su di un sottoinsieme del linguaggio di programmazione JavaScript, definito nelle specifiche ECMA-262, 3a edizione, del dicembre 1999. JSON è un formato di testo completamente indipendente dal linguaggio ma che usa convenzioni già familiari ai programmatori dei linguaggi derivati dal C, tra cui C, C++, C#, Java, JavaScript, Perl, Python e molti altri. Queste caratteristiche rendono JSON un linguaggio ideale per lo scambio di dati.

1.5.3 Web socket

I WebSockets forniscono una comunicazione bidirezionale in tempo reale tra client e server.

Un WebSocket è una connessione TCP persistente, bi-direzionale, full-duplex, garantita da un sistema di handshaking client-key ed un modello di sicurezza origin-based.

Il sistema inoltre maschera le trasmissioni dati per evitare lo sniffing di pacchetti di testo in chiaro.

La WebSocket API con la quale avremo a che fare in JavaScript risulta semplice ed elegante e definisce un oggetto che contiene:

- Informazioni sullo stato della connessione (connecting, open, closing e closed);
- metodi per interagire con la connessione WebSocket (chiudere una connessione ed inviare dati);
- eventi che vengono sollevati all'occorrenza di un evento WebSocket (quando un socket viene aperto, chiuso o riceve in risposta un messaggio di errore).

1.5.4 SHA-1

Con il termine SHA (Secure Hash Algorithm) si indica una famiglia di funzioni crittografiche di hash, sviluppate dalla National Security Agency (NSA) e pubblicate dal NIST come standard federale dal governo degli USA.

Come ogni algoritmo di hash, l'SHA produce un message digest di lunghezza fissa partendo da un messaggio di lunghezza variabile.

La sicurezza di un algoritmo di hash risiede nel fatto che la funzione non sia reversibile e che non deve essere mai possibile creare intenzionalmente due messaggi diversi con lo stesso digest.

L'SHA-1 produce un digest di 160 bit da un messaggio con una lunghezza massima di 264-1 bit ed è basato su principi simili a quelli usati da Ronald L. Rivest del MIT nel design degli algoritmi MD4 e MD5.

L'algoritmo sopra descritto viene utilizzato all'interno della websocket di comunicazione tra client e server nel seguente modo:

- Il client invia al server una chiave segreta di 16 byte con codifica in base 64
- Il server utilizza le funzioni di codifica e decodifica del testo, aggiunge una stringa, elabora la stringa così ottenuta con algoritmo SHA1 e rimanda il risultato al client.
- Quest'ultimo può essere certo che il server cui aveva inviato la sua chiave è lo stesso che apre la connessione.

Capitolo 2

Descrizione del Sistema Sviluppato

In questo capitolo, si riporterà la realizzazione effettiva del software di comunicazione. Verranno illustrate in dettaglio le modalità tramite le quali è possibile effettuare una misurazione, le scelte progettuali e le implementazioni di client e server.

2.0 Struttura del software

Il software sviluppato per la comunicazione con Distanziometro Laser si compone principalmente di due componenti, una componente back-end ed una front end.

La componente back-end, definita server, si occupa della connessione con il distanziometro laser tramite protocollo bluetooth e dello scambio di informazioni e dati con esso.

La componente front-end, definita client, permette all'utente di selezionare la modalità di misurazione (ovvero la modalità in cui il Distanziometro Laser effettuerà le misure, slave o master), di avviare e fermare le misurazioni e di salvare una copia in formato JSON.

2.1 Modalità di misurazione

Come già anticipato il software sviluppato è basato su un modello master slave ed è possibile selezionare la modalità con la quale effettuare la misurazione.

2.1.1 Modalità Live Measure (Master)

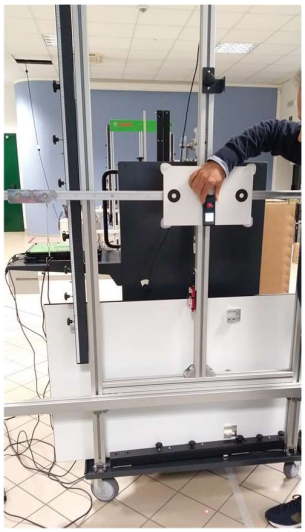
Scegliendo la misurazione Live Measure, l'applicazione assumerà il ruolo di master ed il distanziometro ricoprirà, di conseguenza, quello di slave.

Il Distanziometro, selezionando questa modalità di lettura, è fissato al centro del supporto del target e rileverà la distanza dello stesso misurata rispetto a terra.

Il target è, a sua volta, disposto frontalmente rispetto all'autovettura e subirà spostamenti nelle quattro direzioni, a seconda della richiesta da parte del software di calibrazione.

Una volta avviata la misurazione dalla sezione Live Measure, l'applicazione chiederà al distanziometro la lettura di distanze ad un intervallo di secondi costante l'una dall'altra.

L'intervallo di misurazione è impostato, di default, a 2 secondi, ma è possibile modificare il valore a piacere.



2.1.2 Modalità Wheel Measure (Slave)

Scegliendo la misurazione Wheel Measure, l'applicazione assumerà il ruolo di slave ed il distanziometro ricoprirà, di conseguenza, quello di master.

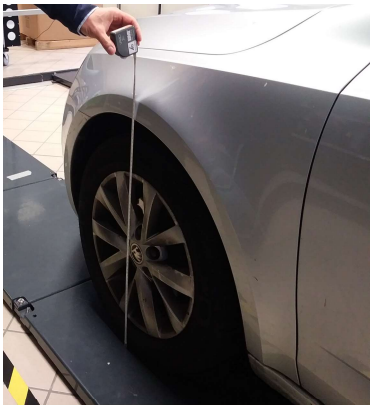
Il distanziometro, selezionando questa modalità di lettura, non avrà una posizione assoluta e verrà manualmente posizionato in prossimità delle quattro ruote dell'autovettura secondo un ordine prestabilito.

Una volta avviata la misurazione dalla sezione Wheel Measure, l'applicazione resterà in ascolto del distanziometro, che invierà una lettura sequenziale dell'altezza da terra, misurata rispetto al passaruota.

L'addetto alla calibrazione posizionerà il distanziometro in prossimità della ruota frontale destra ed effettuerà la misurazione, la quale verrà inviata all'applicazione, che mostrerà a video la distanza letta e resterà in attesa del prossimo valore.

L'addetto si posizionerà ora in corrispondenza della ruota posteriore destra, effettuerà la misurazione, ripeterà spostandosi sulla ruota posteriore sinistra e concluderà il ciclo con la ruota anteriore sinistra.

Una volta effettuate queste quattro misurazioni sequenziali potrà decidere se ripetere il ciclo sovrascrivendo le distanze lette o salvare i dati.



2.2 Modello client - server

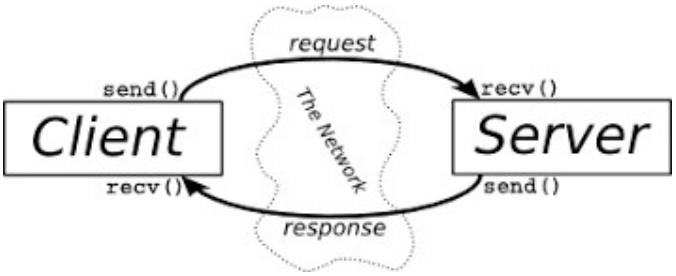
Il modello client - server è costituito da due o più processi in esecuzione, eventualmente su diversi Host, i processi che gestiscono una o più risorse sono detti server, mentre quelli che richiedono l'accesso ad alcune di queste risorse distribuite sono detti client.

Schema di funzionamento di un modello client - server:

1. Il client manda una richiesta al server;
2. Il server (in stato di ascolto, **listening**) riceve la richiesta;
3. Esegue il servizio richiesto (mediante un thread concorrente);
4. Manda una risposta ed eventualmente dei dati al client;
5. Il client riceve la risposta ed eventualmente i dati.

Queste attività avvengono in maniera trasparente sia per i processi server, che per i processi client. Il meccanismo di comunicazione è utilizzato sia se client e server si trovano in esecuzione sullo stesso calcolatore, sia su calcolatori diversi.

Nello specifico il server comunica ed effettua scambio di dati verso il client utilizzando una Web Socket.



2.3 Server

In questo sotto-capitolo si riporterà una descrizione dettagliata della parte relativa al back-end, ovvero al server.

Verranno indicati i pattern di programmazione scelti ed utilizzati, le classi realizzate ed una descrizione del loro scopo, con dettaglio dei metodi in esse contenuti.

2.3.1 Design Pattern di programmazione

"Un pattern descrive un problema che ricorre nell'ambiente e l'essenza della soluzione del problema, in modo tale che si possa riusare questa soluzione milioni di volte senza mai ripeterla in maniera identica due volte"

(Christopher Alexander, 1977, riferendosi a edifici e città)

I Design Pattern sono degli schemi utilizzabili nel progetto di un sistema che permettono di non inventare da capo soluzioni a problemi già risolti, ma di utilizzare istruzioni di provata efficacia in situazioni simili.

2.3.1.1 Observer

È un design pattern intuitivamente utilizzato come base architetturale di molti sistemi di gestione di eventi. Sostanzialmente il pattern si basa su uno o più oggetti, chiamati osservatori o observer, che vengono registrati per gestire un evento che potrebbe essere generato dall'oggetto "osservato", che può essere chiamato soggetto. Oltre all'observer esiste il concrete Observer, che si differenzia dal primo in quanto implementa direttamente le azioni da compiere in risposta ad un messaggio; riepilogando, il primo è una classe astratta, il secondo no. Uno degli aspetti fondamentali è che tutto il funzionamento dell'observer si basa su meccanismi di callback, implementabili in diversi modi, tramite funzioni virtuali o tramite puntatori a funzioni passati quali argomenti nel momento della registrazione dell'observer e spesso a questa funzione vengono passati dei parametri in fase di generazione dell'evento.

Vantaggi:

- Accoppiamento lasco fra classi
- Aumento della probabilità che una classe possa essere usata in modo indipendente e della comprensibilità, modificabilità ed estensibilità dell'intero sistema
- Composizione di oggetti come alternativa alla creazione di sottoclassi
- Evitare l'esplosione del numero delle classi

Come svantaggio, però, la funzione registrata dell´observer viene eseguita nel thread dell´osservable.

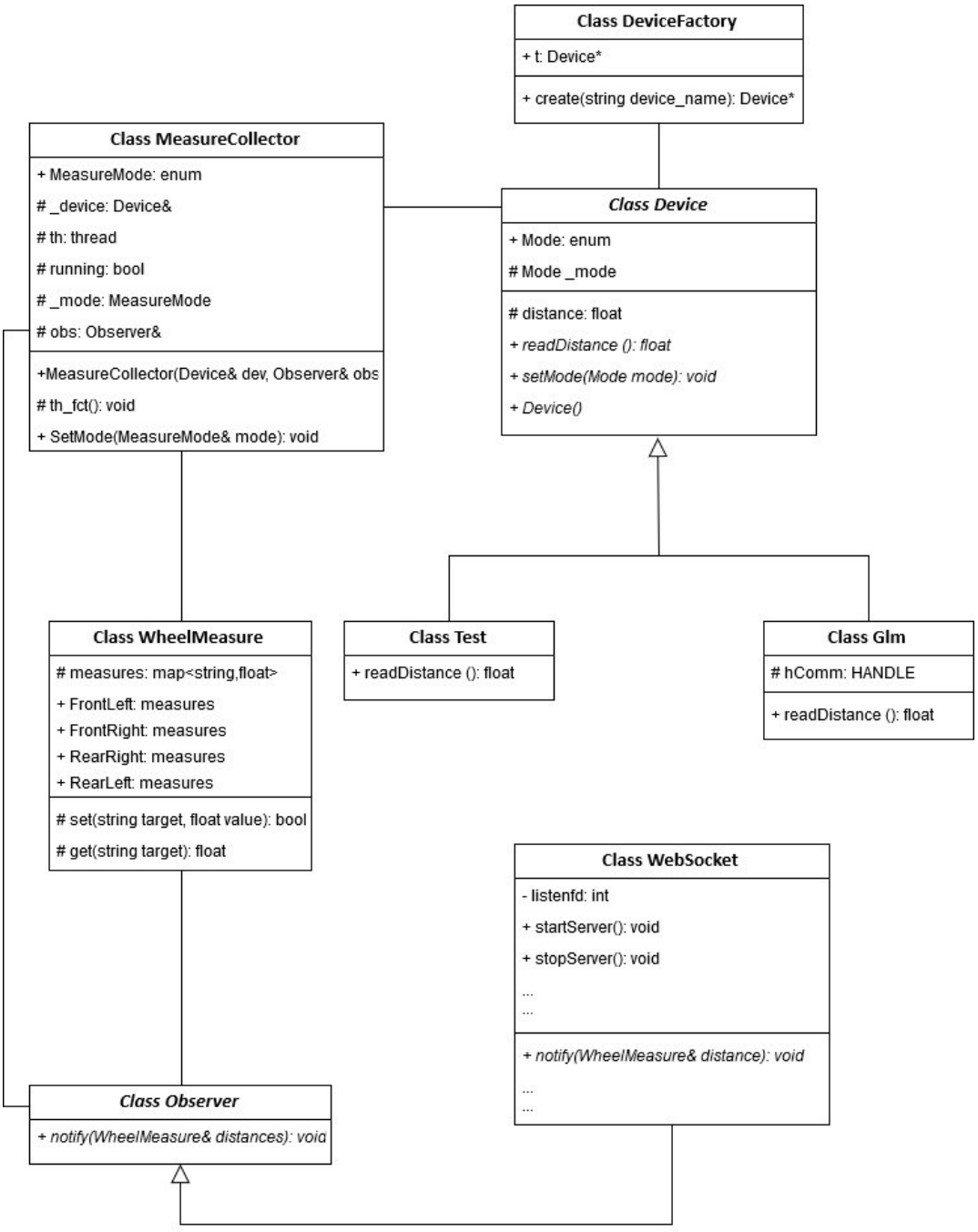
2.3.1.2 Abstract factory

Abstract factory è un design pattern creazionale, fornisce un'interfaccia per creare famiglie di oggetti connessi o dipendenti tra loro, in modo che non ci sia necessità da parte dei client di specificare i nomi delle classi concrete all'interno del proprio codice.

Vantaggi:

- Creare oggetti indirettamente, specificando non il nome della classe ma quello di un'interfaccia
- Si evita di legarsi ad una particolare implementazione, semplificando così futuri cambiamenti
- Limitare le dipendenze (interfaccia verso sistema operativo e API) dalla piattaforma hw e sw
- Aumento della portabilità del sw
- Nascondere agli oggetti client i dettagli circa rappresentazione e implementazione degli oggetti server
- Si evita che i cambiamenti dei server richiedano in cascata cambiamenti nei client

2.3.2 Diagramma UML delle Classi



2.3.3 Elenco delle classi

Come anticipato in questa sezione si trova l'elenco delle classi c++ concrete o astratte utilizzate nel software con il dettaglio delle variabili in esse dichiarate e dei metodi pubblici e privati che contengono.

2.3.3.1 Classe Device

È una classe astratta che contiene al suo interno i metodi astratti ReadDistance e setMode; entrambi i metodi dovranno essere definiti in una o più classi derivanti dalla classe astratta Device.

Il metodo ReadDistance verrà utilizzato per la lettura di una misurazione dal distanziometro laser, mentre il metodo setMode servirà ad importare la modalità di lettura della misurazione, ovvero misuratore in modalità master oppure slave.

```
class Device
public:
      enum Mode;
      virtual ~Device()
            virtual void setMode(Mode mode)
            virtual float readDistance() = 0;
protected:
      Mode _mode;
      float distance;
```

2.3.3.2 Classe DeviceFactory

È una classe che contiene al suo interno un oggetto di tipo puntatore alla classe device, che invoca una funzione che crea un oggetto di classe Test o Glm a seconda di cosa ha letto nel file di configurazione.

È possibile modificare la scelta del modello di Distanziometro Laser Bosch utilizzato, ovvero il Glm 50C, applicando una eventuale integrazione di codice.

```
#include "glm.h"
#include "device.h"

class DeviceFactory
public:
      Device* create(string device_name);
      Device* t;
```

2.3.3.3 Classe Test

Classe derivante dalla classe astratta Device, eredita e definisce il metodo readDistance simulando la lettura di dati dal misuratore laser.

Per la simulazione viene chiamato il metodo random della libreria time.h per la generazione di numeri casuali che verranno successivamente convertiti da interi a float.

```
class Test : public Device
public:
      float readDistance();
```

2.3.3.4 Classe Glm

Classe derivante dalla classe astratta Device che rappresenta il distanziometro Laser Bosch Glm50C. Eredita e definisce il metodo readDistance occupandosi della creazione di un file che verrà utilizzato per lo scambio di comandi e dati tra il device fisico ed il software.

Si appoggia al protocollo di sostituzione cavo RFCOMM ed una volta letti i dati ricevuti dal device li decodifica con operazioni Shift sinistro e Shift destro per renderli disponibili alla lettura.

```
class Glm : public Device
public:
      float readDistance();
protected:
      HANDLE hComm;
```

2.3.3.5 Classe WheelMeasure

Classe che definisce i campi delle varie misurazioni: distance, front left, front right, rear left, rear right.

```
class WheelMeasure
public:
      measures["FrontLeft"] = 0;
      measures["FrontRight"] = 0;
      measures["RearLeft"] = 0;
      measures["RearRight"] = 0;
      measures["Distance"] = 0;

      bool set(string target, float value);
      float get(string target);

protected:
      map<std::string, float> measures;
```

2.3.3.6 Classe Observer

E' una classe astratta che contiene al suo interno il metodo astratto notify, il quale prende come parametro di ingresso l'indirizzo di memoria di un oggetto WheelMeasure.

```
#include "WheelMeasure.h"

class Observer
public:
	virtual void notify(WheelMeasure& distances) = 0;
```

2.3.3.7 Classe MeasureCollector

Classe che definisce la modalità di lettura dei dati ed invoca un thread.

Al suo interno viene creato un oggetto wheelMeasure ed un collegamento ad un oggetto di tipo Device.

Una volta partito il thread tramite uno switch si accede alla modalità di lettura master o slave, vengono letti i dati di Device chiamando il metodo ReadDistance popolando l'oggetto wheelMeasure, che successivamente verrà trasmesso tramite la funzione notify della classe observer.

```
#include <thread>
#include "WheelMeasure.h"
#include "Observer.h"
#include "Device.h"

class MeasureCollector
{
public:
	MeasureCollector(Device& dev, Observer& obs);
	enum MeasureMode;
	void SetMode(MeasureMode& mode);

protected:
	MeasureMode _mode = SEQ_MEASURE;
	Device& _device;
	thread th;
	bool running;
	Observer& _obs;
	void th_fct();
```

2.3.3.8 Classe WebSocket

Classe derivante dalla classe astratta Obserever, si occupa di instaurare la comunicazione con il client web.

All'interno di questa classe viene creato un oggetto WheelMeasure, che sarà stato precedentemente popolato con il metodo notify, ereditato dalla classe astratta Observer.

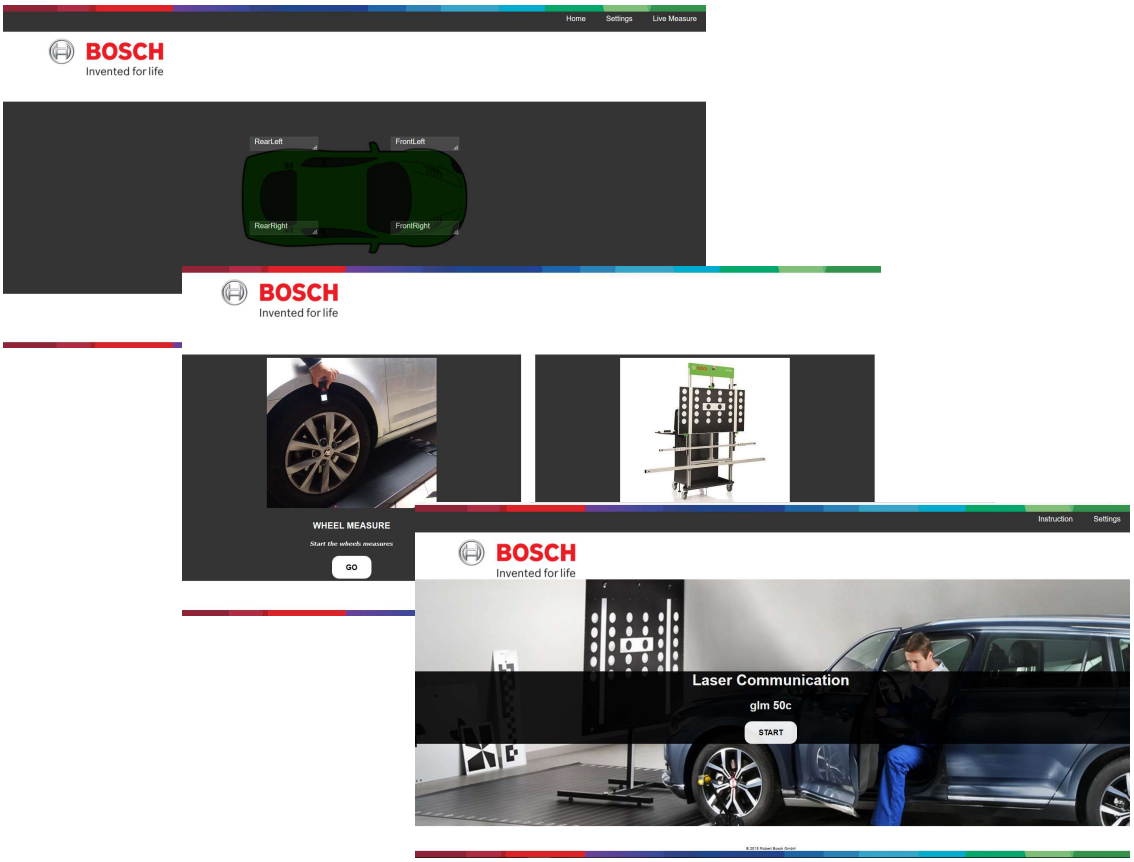
Utilizzando il metodo StartServer preleva e spedisce al client il contenuto di WheelMeasure al client.

```
#include "Observer.h"

class webSocket : public Observer
public:
	void setOpenHandler(defaultCallback callback);
	void setCloseHandler(defaultCallback callback);
	void setMessageHandler(messageCallback callback);
	void setPeriodicHandler(nullCallback callback);
	void startServer();
	void stopServer();
	bool wsSend(int clientID, string message, bool binary = false);
	void wsClose(int clientID);
	int transmitJSON(int clientID, std::string& buffer);
	vector<int> getClientIDs();
	string getClientIP(int clientID);
	string base64_encode(unsigned char const*, unsigned int len);
	string base64_decode(std::string const& s);

	mutex _mutexOutput;
	condition_variable _condOutput;

	void notify(WheelMeasure& distance);
	void startThread(int port);

private:
	vector<wsClient*> wsClients;
	map<int, int> socketIDmap;
	fd_set fds;
	int fdmax;
	int listenfd;

	void wsCheckIdleClients();
	bool wsSendClientMessage(int clientID, unsigned char opcode, string
	message);
	void wsSendClientClose(int clientID, unsigned short status = -1);
	bool wsCheckSizeClientFrame(int clientID);
	void wsRemoveClient(int clientID);
	bool wsProcessClientMessage(int clientID, unsigned char opcode, string
	data, int dataLength);
	bool wsProcessClientFrame(int clientID);
	bool wsBuildClientFrame(int clientID, char* buffer, int bufferLength);
	bool wsProcessClientHandshake(int clientID, char* buffer);
	bool wsProcessClient(int clientID, char* buffer, int bufferLength);
	int wsGetNextClientID();
	void wsAddClient(int socket, in_addr ip);

	defaultCallback callOnOpen;
	defaultCallback callOnClose;
	messageCallback callOnMessage;
	nullCallback callPeriodic;

	std::map<std::string, std::string> cameras;
	std::map<std::string, std::string> targets;
	std::thread webthread;
```

2.4 Client

In questo sotto-capitolo si riporterà descrizione dettagliata della parte relativa al front-end, ovvero al client.

Verrà indicata la struttura generale delle diverse pagine web che la compongono ed il dettaglio delle singole.



2.4.1 Struttura Pagine Web

Le interfacce web realizzate si compongono della stessa struttura e si differenziano per le specifiche caratteristiche che ciascuna pagina contiene, per soddisfare determinate esigenze.

2.4.1.1 Head

Composto dal tag <title> relativo alla pagina html e da due tag <link>, il primo relativo al file normalize.css che determina gli adattamenti da effettuare in base al browser utilizzato, il secondo al file style.css che stabilisce lo stile e la grafica.

```
<!DOCTYPE html>
<head>
	<meta charset="UTF-8">
	<title>Home</title>
<!-- normalize-->
	<link rel="stylesheet" href=" normalize.css">
	<link rel="stylesheet" href="style.css">
</head>
```

2.4.1.2 Body

Composto dal tag <header> che rappresenta la parte superiore della pagina web e che è suddiviso in classi e sottoclassi per la definizione del menù.

```
<!-- header-->

<header class="header clearfix">

	<ul class="header__menu animate">
		<li class="header__menu__item"><a href="Instruction.html">Instruction</a></li>
		<li class="header__menu__item"><a href="settings.html">Settings</a></li>

	</ul>
</header>
```

Successivamente vengono definiti due tag <section> posti in successione al di sotto del menù che contengono, rispettivamente, le classi e sottoclassi per la definizione del logo, le classi e sottoclassi per la definizione della cover e di eventuali filtri e sottosezioni della cover.

```
<section class="box_logo">
	<div class="logo">
		<img src="bosch_logo_w250.png" alt="">
	</div>
</section>
```

```
<section class="cover">

<div class="cover__caption">

	<div class="cover__caption__copy">
		<div class="cover__caption__copy__filter">
			<h1>Laser Communication </h1>
			<h2>glm 50c</h2>
			<a href="start.html" class="button"> <strong>START</strong> </a>
		</div>
	</div>
</div>
</section>
```

Infine, alcuni tag definiscono le bande grafiche superiore ed inferiore ed il copyright.

```
<div class="page__header">
	<div class="supergraphic bar"></div>
</div>

<div class="page__footer">
	<span class="copyright">
	&copy; 2018 Robert Bosch GmbH
	</span>

<div class="supergraphic bar"></div>
</div>
```

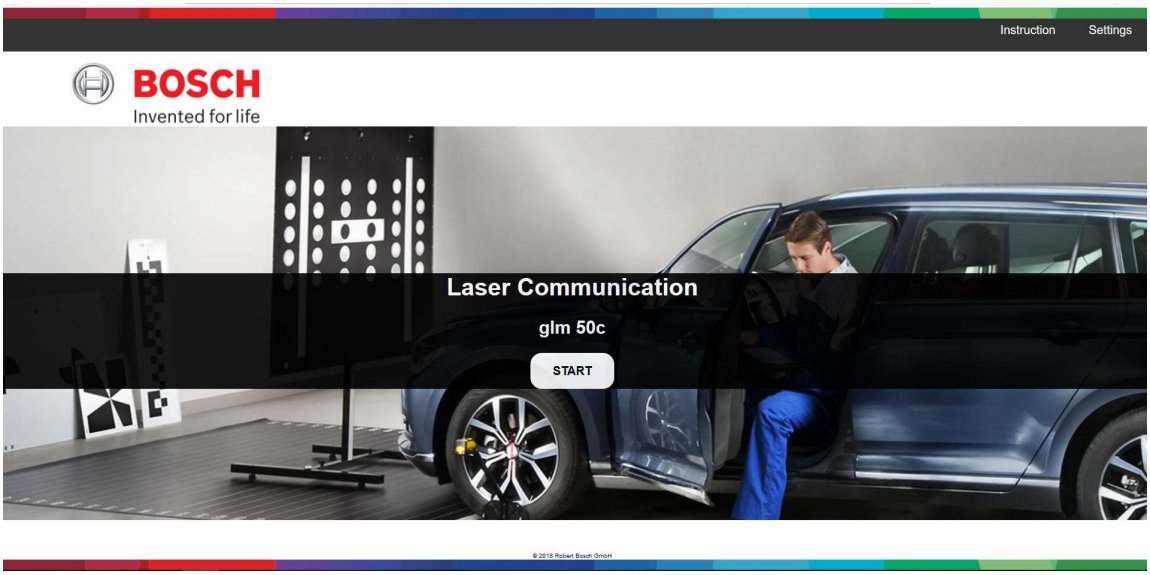
2.4.2 Elenco Pagine Web

In questa sezione verranno elencate le pagine web realizzate, il loro scopo, una breve descrizione del funzionamento e cosa accade dopo la connessione con il server.

2.4.2.1 index.html

Rappresenta la home, pagina principale nonché la prima schermata che visualizza l'addetto alla calibrazione effettuando l'accesso al software.

Questa pagina indirizza alla vera e propria sezione di scambio dati tra client e server, premette di consultare le istruzioni per capire, al primo accesso o per altri motivi, come utilizzare le funzionalità e consente l'indirizzamento verso la sezione di modifica delle impostazioni.

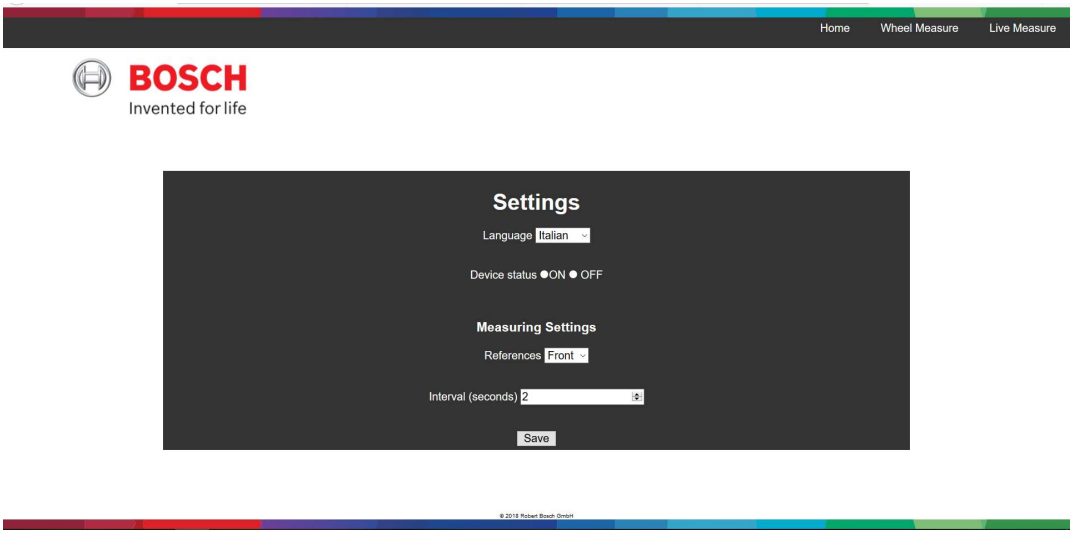


2.4.2.2 instruction.html

È una pagina accessoria, di sola consultazione, che fornisce le informazioni necessarie all'utilizzo del software e della sua interfaccia per procedere alla misurazione delle distanze.

2.4.2.3 settings.html

Questa pagina consente di modificare le impostazioni fornite di default al primo accesso, quali la lingua di visualizzazione, il punto di riferimento per la misurazione, l'intervallo di misurazione nella modalità live e permette di effettuare la connessione al dispositivo.

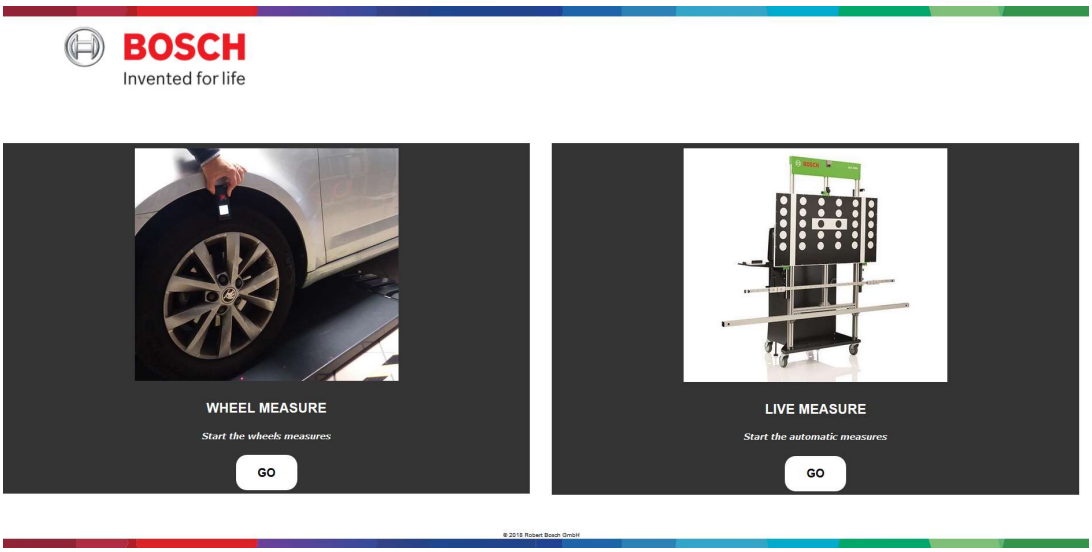


2.4.2.4 start.html

È la pagina intermedia tra la schermata di home e la misurazione vera e propria.

Presenta la possibilità di scegliere come effettuare la misurazione ovvero la modalità con la quale il software comunicherà con il distanziometro laser.

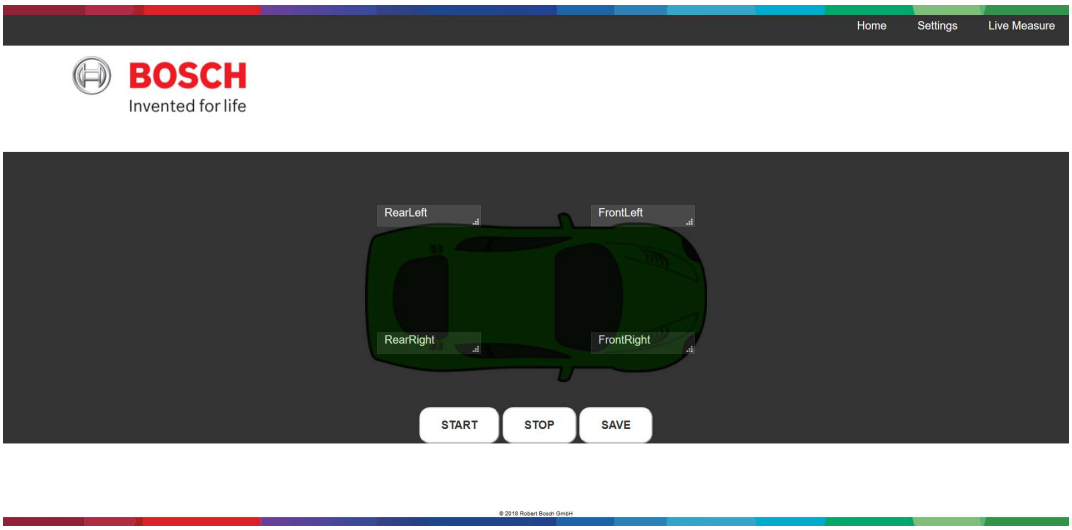
Consente inoltre di tornare alla pagina home e di modificare le impostazioni.



2.4.2.5 wheelMeasureAuto.html

È la pagina che permette di avviare la misurazione in modalità slave, dopo aver cliccato il pulsante “start” l’applicazione, tramite funzione web socket javascript, instaura la connessione con il server, attende l’arrivo sequenziale delle quattro distanze e le visualizza a video.

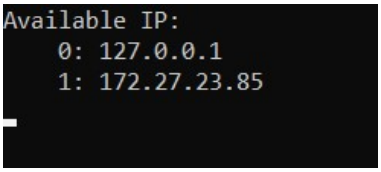
Cliccando su “stop” si termina la connessione con il server e utilizzando il pulsante “salva” è possibile salvare le distanze visualizzate su file in formato standard json.



2.4.2.5.1 Avvio del server

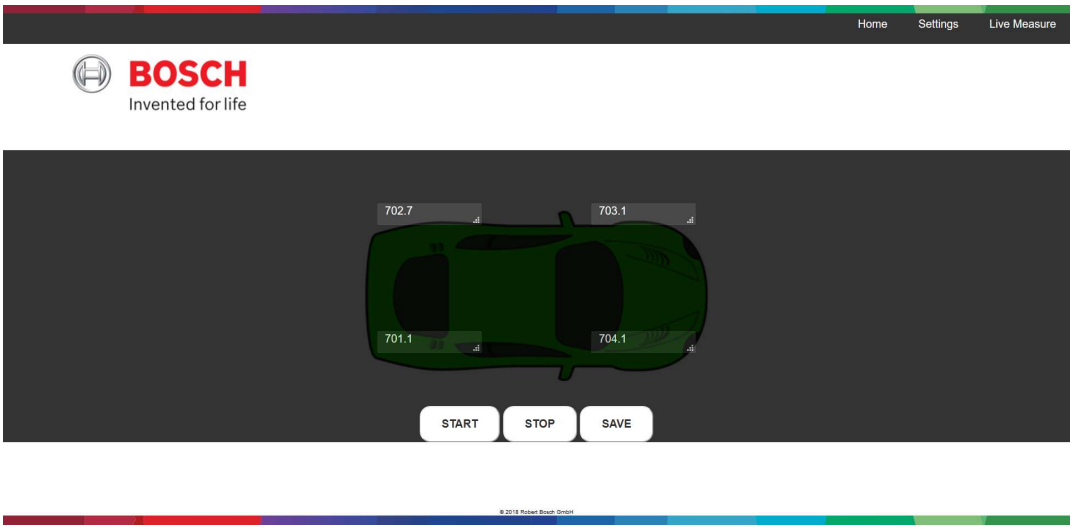
Avviando il server verrà instaurata la connessione con il client e si visualizzeranno a video gli indirizzi IP relativi a client e server.

A questo punto la connessione è avvenuta con successo ed entrambi restano in ascolto per la successiva fase di scambio dati.



Lo scambio di dati avviene nel momento in cui l'utente clicca sul pulsante di "start" avviando così la richiesta di dati al server.

Di seguito dettaglio della pagina web dopo la ricezione da parte del server delle quattro altezze dei passaruota.



2.4.2.5.2 Save

Cliccando sul pulsante “save” si procede al salvataggio dei dati ricevuti, e visualizzati a video, in formato standard JSON.

2.4.2.6 liveMeasure.html

È la pagina che permette di avviare la misurazione in modalità master.

Dopo aver cliccato il pulsante “start” l’applicazione, tramite funzione web socket javascript, instaura la connessione con il server e stampa a video “connessione effettuata” o “errore di connessione”.

Se la connessione è avvenuta correttamente richiede la lettura di distanze ad un intervallo costante prestabilito e le visualizza a video stampando l’ultima lettura effettuata sempre in testa alla lista.

Cliccando su “stop” si termina la connessione con il server.

Infine, utilizzando il pulsante “save” sarà possibile salvare le distanze visualizzate a video su file in formato standard JSON.

Ora si potrà richiedere una nuova connessione, cambiare modalità di lettura o chiudere l’applicazione.

2.4.2.7 style.css

File unico dove vengono definiti i font, il colore, la dimensione etc dei caratteri all'interno delle varie pagine.

Per agevolarne la lettura e la modifica a posteriori è suddiviso in sezioni, ovvero:

general per le parti generali e che sono comuni a tutte le pagine;

header per lo stile della omonima sezione;

logo per ciò che caratterizza dimensione e altre caratteristiche del logo;

Cover Home per la struttura della cover nella pagina index ovvero la home;

Cover Wheel Measure si differenzia per alcune caratteristiche dalla cover precedente quindi richiedeva una implementazione a parte;

Cards sono le varie sottosezioni presenti solo in alcune pagine e fisicamente situate sotto la cover;

Clearfix sono alcune istruzioni per adattare il contenuto alla sezione in cui è incluso.

Conclusioni

Il software sviluppato ha raggiunto i risultati prefissati.

La connessione mediante bluetooth con il dispositivo è stata instaurata con successo, lo scambio di informazioni e dati tra applicazione e distanziometro Laser Bosch Glm50C, ha dimostrato una buona velocità di trasferimento ed un alto livello di efficienza e affidabilità.

Il software è stato sviluppato in un linguaggio di programmazione sia lato server che lato client, facilmente integrabili con l'attuale strumento di calibrazione dei sensori ADAS.

Sviluppo futuro potrebbe certamente essere l'effettiva integrazione con il sopracitato software di calibrazione.

Ulteriori sviluppi potrebbero focalizzarsi sulla modalità di lettura sequenziale, la Wheel Measure che, ad oggi, è stata realizzata e testata esclusivamente in modalità simulata, provvedendo sia all'integrazione del codice, che all'effettivo test sul campo.

Ritengo, in conclusione, che il software realizzato possa essere un concreto supporto al personale addetto alla calibrazione ed un investimento favorevole per l'azienda da un punto di vista economico.

Il personale addetto potrà così beneficiare di un miglioramento sia in termini di tempo che di marginalità d'errore.

In termini di tempo perché permette di ottimizzare la fase preliminare di preparazione, di marginalità d'errore perché azzera l'errore umano in cui si potrebbe facilmente incorrere.

Per quanto concerne l'azienda, a seguito di un investimento di capitale iniziale, si prevede un ritorno economico su lungo termine, scaturito dalla riduzione dei tempi di lavoro sia in fase preliminare, che per eventuale correzione del suddetto errore umano.

Bibliografia

https://www.bosch-professional.com/it/it/products/glm-50-c-0601072C00

https://www.html.it/articoli/introduzione-ai-websocket/

https://www.sicurauto.it/

https://it.wikipedia.org/wiki/Telemetro_laser

https://docs.microsoft.com/it-it/visualstudio

https://www.webit.it/magazine/tutte-le-caratteristiche-del-browser-web-mozilla-firefox/

https://it.wikipedia.org/wiki/Architettura_master-slave

https://it.wikipedia.org/wiki/Bluetooth

http://elettronica-plus.it/bluetooth-4-2-lo-standard-ideale-per-applicazioni-iot_90456/

https://appuntisuperiori.altervista.org/blog/il-modello-client-server

https://it.wikipedia.org/wiki/Trasmissione_seriale

https://diazilla.com/doc/388083/oo-design-pattern-design-pattern

https://www.ionos.it/digitalguide

https://www.html.it/pag/61287/enumerazioni-2/

https://it.wikipedia.org/wiki/

https://www.html.it/

http://www.json.org/json-it.html