





**ABSTRACT** I sistemi di posizionamento ferroviari e ferrotramviari, ad oggi impiegati, fanno un largo uso di apparati installati a terra e segnali provenienti dalla linea. La loro realizzazione ha pertanto un costo e un impatto ambientale non trascurabili.

In linea con le normative europee, la ricerca nel campo del posizionamento ferroviario si sta focalizzando verso la realizzazione di sistemi di posizionamento autonomi, i quali non debbono utilizzare alcun apparato installato a terra.

In questa Tesi si mostrano i risultati relativi alle attività di analisi condotte su di un sotto sistema di posizionamento autonomo basato sull'utilizzo di un algoritmo noto come *Sensor Fusion Algorithm*.

SFA è un algoritmo che permette di integrare le misure fornite da un insieme di sensori installati a bordo treno, al fine di attutire il rumore di misura e ottenere una stima della posizione del treno sicura e affidabile. Viene descritto il sistema a livello hardware e a livello software e, partendo dalle stesse specifiche software, viene mostrato l'ambiente di analisi realizzato al fine di condurre l'attività di analisi oggetto della Tesi.



---

## INDICE

---

1	Stato dell'Arte	9
1.1	Introduzione	9
1.1.1	Dependability	10
1.2	Robaccia	13
1.3	Il problema del posizionamento	13
1.4	Verso ETCS-3	16
2	Architettura di Sistema	19
2.1	Descrizione generale	19
2.2	Sistemi Costituenti	20
2.2.1	Sensor Set	20
2.2.2	Piattaforma di elaborazione dati	22
2.2.3	On Board Control Unit	22
2.3	Specifiche delle Interfacce	23
2.3.1	Relied Upon Interfaces	23
2.4	Interazioni	25
2.4.1	Acquisizione dei dati	25
2.4.2	Trasmissione della posizione	26
2.4.3	Interazioni con eventuali operatori umani	27
3	Architettura Software	29
3.1	Sensor Fusion Library	29
3.1.1	API	29
3.2	Listener	31
3.2.1	Database Interface	31
3.3	Interface Modules	33
4	Ambiente di Analisi	35
4.1	Principi di base	35
4.2	Software Impiegati	35
4.2.1	SensorFusionLib	35
4.2.2	SynthDataGen	36
4.2.3	Rail Track Tool	36
5	Parte Sperimentale	43
5.1	Misure di interesse	43
5.2	Esempi	44
5.2.1	Scenario 1	44
5.2.2	Scenario 2	46

4 Indice

5.2.3 Scenario 3	49
6 Conclusioni	53

---

## ELENCO DELLE TABELLE

---

Tabella 1	Specifiche Tecniche NVidia TX-Jetson	22
Tabella 2	Specifiche delle RUPi del sistema	23
Tabella 3	Specifiche delle RUMI del sistema	24
Tabella 4	Moduli di <i>interface-modules</i>	33
Tabella 5	Scenario 1, Esperimento 1.1	44
Tabella 6	Esperimento 1.1: Risultati	45
Tabella 7	Scenario 1, Esperimento 1.2	45
Tabella 8	Esperimento 1.2: Risultati	45
Tabella 9	Scenario 2, Esperimento 2.1	46
Tabella 10	Esperimento 2.1: Risultati	46
Tabella 11	Esperimento 2.1: Confronto con esperimento 1.2	46
Tabella 12	Scenario 2, Esperimento 2.2	47
Tabella 13	Esperimento 2.2: Risultati	47
Tabella 14	Esperimento 2.2: Confronto con esperimento 1.2	47
Tabella 15	Scenario 2, Esperimento 2.3	48
Tabella 16	Esperimento 2.3: Risultati	48
Tabella 17	Esperimento 2.3: Confronto con esperimento 1.2	49
Tabella 18	Scenario 3, Esperimento 3.1	49
Tabella 19	Esperimento 3.1: Risultati	49
Tabella 20	Esperimento 3.1: Confronto con esperimento 1.2	50
Tabella 21	Scenario 3, Esperimento 3.2	50
Tabella 22	Esperimento 3.2: Risultati	50
Tabella 23	Esperimento 3.2: Confronto con esperimento 1.2	51



---

## ELENCO DELLE FIGURE

---

Figura 1	Fallimento e restauro	9
Figura 2	La <i>safety</i> estende il concetto di <i>reliability</i>	11
Figura 3	Catena guasto errore fallimento	12
Figura 4	Schema di un tipico scenario ferrotramviario	14
Figura 5	Livelli ETCS	15
Figura 6	Schema SFA	19
Figura 7	<i>Inertial Measurment Unit</i>	21
Figura 8	Ricevitore GPS ublox EVK-M8T	21
Figura 9	Nvidia TX-Jetson	22
Figura 10	Modem TP-LINK M7350 LTE-4G	24
Figura 11	Architettura hardware del sottosistema di posizionamento	25
Figura 12	Sequenza di acquisizione dati	26
Figura 13	Sequenza di trasmissione della posizione	27
Figura 14	API di <i>SensorFusionLib</i>	30
Figura 15	Architettura <i>listener</i>	31
Figura 16	Class diagram DatabaseInterfaceEdges	32
Figura 17	Schema database tracce	33
Figura 18	Architettura software completa	34
Figura 19	Schema di SynthDataGen	36
Figura 20	Schema riassuntivo RTT	37
Figura 21	Interfaccia RTT	38
Figura 22	Pannello di configurazione generale <i>SynthDataGen</i>	39
Figura 23	Pannello di configurazione IMU simulato	40
Figura 24	Posizione del treno mostrata sulla mappa	40
Figura 25	Grafico valori di accelerazione assi X e Y	41
Figura 26	Grafico errore sulla stima della posizione	41
Figura 27	Traccia di analisi	43



---

## STATO DELL'ARTE

---

### 1.1 INTRODUZIONE

Negli ultimi anni, i sistemi informatici hanno assunto un ruolo sempre più centrale nelle attività umane.

Inizialmente, il computer era considerato un semplice strumento di supporto alla matematica applicata, capace di svolgere calcoli particolarmente onerosi in un tempo relativamente breve. Con lo sviluppo delle tecnologie, i sistemi informatici sono ad oggi impiegati in un vasto insieme di domini applicativi, dall'elettromedicale al trasporto aereo, fino all'*Internet of Things*.

Quanto più si diffonde l'utilizzo dei sistemi informatici, tanto più peso assume un eventuale fallimento dei medesimi. La letteratura scientifica dimostra che la valutazione della *dependability* di un sistema informatico è un problema chiave.

Per *dependability* si intende la capacità che ha un sistema di fornire un servizio in modo corretto. [1]

Un *fallimento* è una transizione compiuta da un sistema dall'erogazione di un servizio corretto verso l'erogazione di un servizio scorretto. La transizione contraria è detta *restauro*.

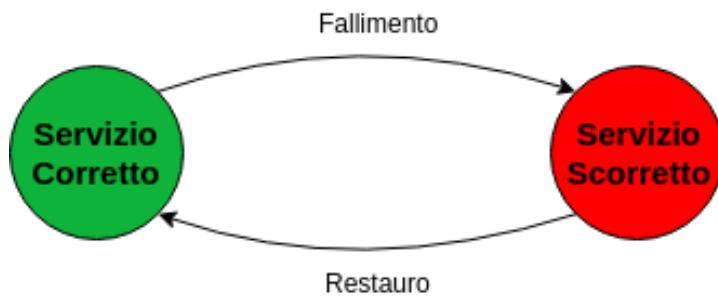


Figura 1: Fallimento e restauro

Effettuare misure sperimentalali su un sistema informatico, o su un suo prototipo, è una valida opzione per valutarne la *dependability*.

Questa Tesi descrive l'attività di osservazione del prototipo di un sistema informatico impiegato nell'ambito del posizionamento ferrotramviario.

### 1.1.1 *Dependability*

La *dependability* di un sistema è:

- Misurata rispetto a un certo insieme di proprietà note come *measures*;
- Raggiunta attraverso l'utilizzo di specifiche tecniche, i *means*;
- Minacciata dai *threats*, ossia da tutto ciò che porta il sistema ad erogare un servizio improprio.

Un sistema può fallire nel caso in cui questo non sia conforme alle specifiche, oppure perchè le specifiche non descrivono adeguatamente le sue funzioni.

La *dependability* di un sistema viene misurata rispetto alle seguenti proprietà:

- *Availability*: L'alternanza tra la fornitura di un servizio corretto e uno scorretto.

$$A(t) = \begin{cases} 1 & \text{se il servizio fornito è corretto al tempo } t \\ 0 & \text{altrimenti} \end{cases}$$

$E[A(t)]$  : probabilità che il servizio fornito sia corretto al tempo  $t$

- *Reliability*: Capacità di fornire un servizio continuamente corretto in un certo intervallo di tempo.

$R(t)$  : probabilità di fornire un servizio corretto nell'intervallo  $[0, t]$

- *Safety*: Il tempo medio a un fallimento catastrofico.

$S(t)$  : probabilità che non si verifichi alcun fallimento catastrofico nell'intervallo  $[0, t]$

La *safety* è un' estensione del concetto di *reliability*.

Si definisce uno stato sicuro in cui il sistema:

- Fornisce il servizio corretto, oppure
- Non fornisce il servizio corretto, ma il fallimento non ha conseguenze catastrofiche sull'ambiente o sulle persone

Qualunque fallimento che induca il sistema a fornire un servizio scorretto con conseguenze catastrofiche, viene modellato come una transizione verso uno stato non sicuro.

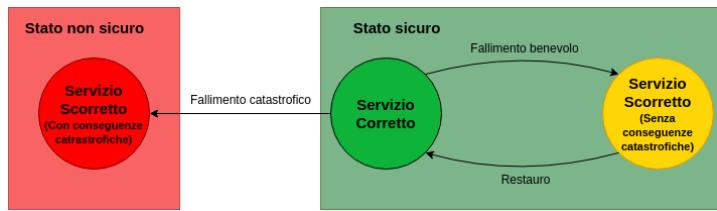


Figura 2: La *safety* estende il concetto di *reliability*

- *Time to Failure*: Il tempo che intercorre fra l'ultimo restauro e il successivo fallimento.  
Spesso è opportuno considerare il valore atteso di questa grandezza, il *Mean Time to Failure* (MTTF)
- *Maintainability*: Il tempo necessario a restaurare il sistema, dopo l'ultimo fallimento. Il valore atteso di questa misura prende il nome di *Mean Time to Repair* (MTTR)
- *Coverage*: Probabilità che il sistema sia in grado di tollerare un guasto.

I *threats* che minano la *dependability* di un sistema sono i guasti, gli errori e i fallimenti.

Un guasto è un qualunque evento interno al sistema in grado di causare un errore. Quando l'errore raggiunge l'interfaccia di servizio, ovvero altera il servizio fornito dal sistema, si parla di fallimento. In letteratura, si fa riferimento a questo rapporto di causalità come *fault error failure chain*, catena guasto errore fallimento.



Figura 3: Catena guasto errore fallimento

La *dependability* di un sistema informatico è raggiunta attraverso l'uso di quattro tecniche:

- *Fault Prevention*: Previene l'insorgenza di guasti durante il ciclo di vita del sistema
- *Fault Tolerance*: Rende il sistema in grado di fornire un servizio corretto anche in presenza di guasti
- *Fault Removal*: Riduce il numero di guasti nel sistema
- *Fault Forecasting*: Stima il numero di guasti presenti nel sistema, attualmente o in futuro

La *dependability* è la proprietà che viene misurata durante il processo di *validazione* di un sistema informatico.

La validazione di un sistema informatico è un processo atto a determinare se il sistema è conforme alle sue specifiche funzionali.

Il processo di validazione avviene durante tutte le fasi del ciclo di vita del sistema, anche prima che venga realizzato. Per questo motivo, il sistema viene opportunamente modellato al fine di condurre propriamente l'analisi.

A discrezione della fase del ciclo di vita del sistema, si utilizzano differenti tecniche e modelli di validazione: [2]

- Specifica: la validazione è basata sull'utilizzo di tecniche combinatorie che mirano a determinare le condizioni di fallimento di un sistema in funzione del fallimento dei suoi sottocomponenti, considerati indipendenti;
- *Design*: in questa fase si usano tecniche analitiche, come ad esempio le Catene di Markov, che permettono di rilassare le assunzioni di indipendenza tipiche dei modelli combinatori;
- Implementazione: con il procedere della fase di implementazione, il sistema prende forma e può essere interessante osservarne direttamente il comportamento ed effettuare misure sperimentali;

- Fase operativa: il sistema è impiegato sul campo e possono quindi essere utilizzate tutte le tecniche disponibili per l'analisi.

## 1.2 ROBACCIA

È possibile schematizzare un sistema ferroviario, o ferrotramviario, come un insieme di vetture vincolate a muoversi lungo una traccia fissata. Questa schematizzazione è, in grossolana approssimazione, valida per qualsiasi sistema ferroviario o ferrotramviario, a prescindere dal numero di veicoli transitanti o dall'estensione geografica. Ciò che invece differenzia un sistema ferroviario da un sistema ferrotramviario sono:

- Le caratteristiche fisiche del veicolo transitante, come lunghezza e massa;
- Le caratteristiche geografiche dell'ambiente operativo;
- Gli scopi del trasporto.

In generale, nel trasporto ferroviario si utilizzano veicoli pesanti atti a trasportare persone o merci su lunghe percorrenze, pertanto è comune che l'ambiente operativo di un sistema ferroviario sia prevalentemente extra urbano.

Nel trasporto ferrotramviario, di contro, si utilizzano veicoli leggeri per offrire un'alternativa al cittadino all'utilizzo di mezzi privati durante i suoi spostamenti all'interno di un'area metropolitana. Quest'ultima caratteristica implica che l'ambiente operativo di un sistema ferrotramviario sia radicalmente diverso dall'ambiente operativo di un sistema ferroviario. Le vetture si muovono lungo tracce installate su strade urbane, e di conseguenza il traffico ferrotramviario condivide l'ambiente con il traffico cittadino, come mostrato in figura 4.

## 1.3 IL PROBLEMA DEL POSIZIONAMENTO

Per posizionamento ferroviario si intende la valutazione della posizione di un treno all'interno di una traccia ferroviaria. Tale posizione viene espressa come progressiva chilometrica rispetto a una posizione nota, come ad esempio l'origine della linea. [3]



Figura 4: Schema di un tipico scenario ferrotramviario

#### *Odierne Tecniche di Posizionamento*

Gli odierni sistemi di posizionamento si basano principalmente sull'utilizzo di strumenti installati a terra, chiamati *beacon*, o *balise* in gergo ferroviario, i quali hanno lo scopo di rilevare il passaggio di un treno.[4] Esiste uno standard a livello europeo al quale gli odierni sistemi di posizionamento si devono uniformare, l'*European Train Control System* (ETCS).

Nel corso della storia, ogni paese europeo ha sviluppato autonomamente le proprie infrastrutture ferroviarie e relative regole operative. Tuttavia, ad oggi i treni possono attraversare le frontiere, pertanto è necessario sviluppare un sistema ferroviario standard che rispetti una comune normativa operazionale europea. Tale sistema prende il nome di *European Rail Traffic Management System* (ERTMS) [5], ed ETCS è il sottosistema di ERTMS dedicato al posizionamento delle vetture.

Come standard, ETCS definisce specifici livelli di *compliance* che possiede un sistema di posizionamento rispetto ad ETCS, ed essi vanno dal livello ETCS-0 al livello ETCS-3. [6]

L'obiettivo è quello di sviluppare progressivamente un sistema di posizionamento completamente autonomo (ETCS-3), partendo da un sistema interamente *non-compliant* con ETCS (ETCS-0).

Allo stato attuale, quasi tutti i sistemi di posizionamento sono ETCS-2. Nei livelli ETCS-1 e ETCS-2, le tracce vengono suddivise in blocchi, e all'entrata di ciascun blocco viene posizionato un *beacon* in grado di rilevare la presenza di un treno.

L'autorizzazione all'ingresso in un blocco viene rilasciata se nessun altro treno sta occupando il blocco al quale si vuole accedere, mentre un siste-

ma di *odometria* installato a bordo, posiziona il treno rispetto all'ultimo *beacon* incontrato.

Nel livello ETCS-3, non sono richiesti segnali provenienti dalla linea: un treno deve essere in grado di posizionarsi autonomamente. [7]  
In sintesi, i livelli ETCS possono essere descritti come segue:

- ETCS-0: Sistema non conforme a ETCS;
- ETCS-1: Utilizzo di apparati di posizionamento installati a terra, autorizzazione a procedere segnalata al macchinista attraverso indicazioni semaforiche;
- ETCS-2: Come ETCS-1, ma l'autorizzazione a procedere è gestita da un sistema automatico di scambio, denominato sistema di *interlocking*;<sup>[8]</sup>
- ETCS-3: Posizionamento autonomo, nessun utilizzo di apparati a terra.

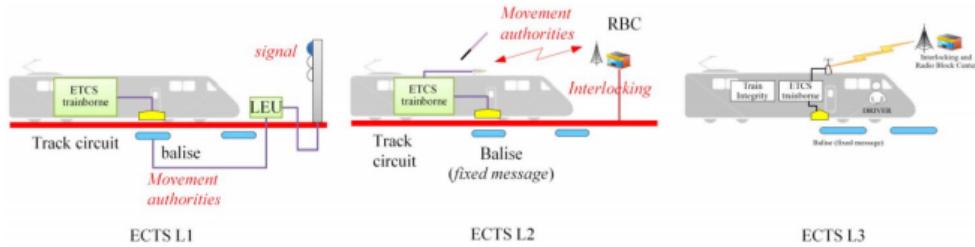


Figura 5: Livelli ETCS

Il livello ETCS-2 prevede che l'autorizzazione a procedere venga gestita dal sistema di *interlocking* e non dal solo operatore umano notificato mediante indicazioni semaforiche.

La funzionalità offerta del sistema di *interlocking* viene pertanto considerata *safety-critical*, in quanto un suo fallimento può portare a conseguenze anche catastrofiche.<sup>[9]</sup>

ETCS adotta un approccio incrementale alla realizzazione di sistemi di posizionamento autonomi. Un sistema di posizionamento ETCS-3 deve continuare a interagire con il sistema di *interlocking*, quindi deve essere considerato a sua volta un sistema *safety-critical*.

L'implementazione hardware e software di un sistema ETCS-3 viene dunque vincolata da standard generici per sistemi *safety-critical* [10] [11], e da

standard specifici del dominio ferroviario [12].

ERTMS/ETCS è pensato per sistemi ferroviari, mentre nel dominio ferrotramviario vige la regola della *marcia a vista*. Il rispetto di ETCS non è obbligatorio in detto contesto, tuttavia le tecniche di posizionamento ivi utilizzate rispettano spesso le linee guida imposte da ETCS.

#### 1.4 VERSO ETCS-3

Gli attuali sistemi di posizionamento richiedono un minimo intervento di computer installati a bordo e una grande quantità di apparati installati a terra. Gli apparati di terra sono costosi e hanno un impatto ambientale non trascurabile, pertanto è necessario iniziare a pianificare una migrazione verso sistemi ETCS-3.[13]

Il sistema oggetto della Tesi è un sottosistema di posizionamento conforme alla filosofia ETCS-3.

Nell'ottica di voler realizzare un sistema di posizionamento autonomo, il treno viene modellato come un *Cyber-Physical System* (CPS).

Un CPS è un sistema che consiste di un computer (sottosistema *cyber*) e un oggetto da esso controllato (sottosistema *physical*). Il sottosistema *cyber* è essenzialmente un elaboratore che opera in un tempo discreto, dispone di processore, memoria, e di interfacce I/O che abilitano l'interazione del CPS con eventuali operatori umani.

Il sottosistema *physical* consiste di un sistema governato dalle leggi della fisica che opera in un tempo continuo. [14][15]

Nella fattispecie, l'oggetto controllato è il treno, mentre l'elemento *cyber* è costituito da un *sistema di sistemi* composto da un'unità in grado di campionare e processare un certo insieme di misure, e da un'unità in grado di controllare il movimento del treno. Quest'ultima unità prende il nome di *On Board Control Unit* (OBCU), ed è il computer di bordo nominale che ogni treno deve possedere in accordo a ERTMS/ETCS.

Lo scopo della Tesi è quello di mostrare i risultati sperimentali delle campagne di analisi condotte sul sottosistema *cyber* del CPS descritto.

Tale sottosistema ha lo scopo di stimare la posizione di un treno attraverso l'uso combinato di un insieme di sensori installati a bordo.

I valori campionati dai sensori dovranno essere integrati al fine di ottenere una stima sicura e affidabile della posizione del treno. Tale integrazione viene realizzata grazie all'utilizzo di un algoritmo noto come *Sensor Fusion Algorithm* (SFA). [16]

Nello sviluppo di un sistema di posizionamento *safety-critical*, è fondamentale poter garantire che l'errore commesso dal sistema non superi

una determinata soglia.

Le analisi condotte si prefiggono lo scopo di poter osservare sperimentalmente una situazione in cui l'errore commesso non superi una determinata soglia definita come accettabile.



# 2

---

## ARCHITETTURA DI SISTEMA

---

In questo capitolo viene descritta l'architettura hardware del sottosistema di posizionamento, in particolare, ne vengono evidenziati i moduli costituenti e le loro interfacce di comunicazione. Vengono infine descritte le interazioni osservabili.

### 2.1 DESCRIZIONE GENERALE

Lo scopo del sistema è quello di implementare un meccanismo di posizionamento basato su SFA.

Tale algoritmo viene eseguito da una libreria software schematizzabile, ai fini di questa Tesi, come una *black-box* che rappresenta il nucleo centrale del sottosistema di posizionamento.

Ricevuti in ingresso un certo insieme di misure, essa fornisce in uscita una *stima statistica* della posizione del treno, più accurata della misura che si otterrebbe utilizzando i dati provenienti dai singoli sensori.[17]

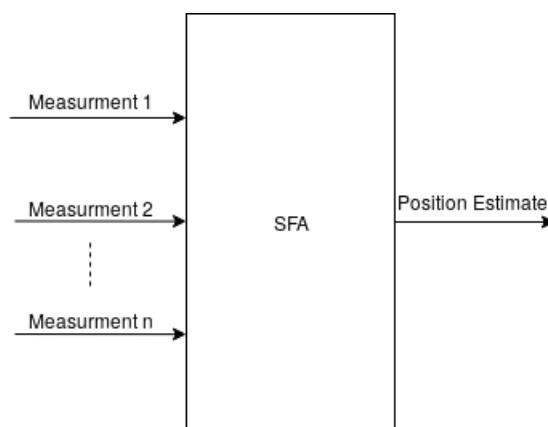


Figura 6: Schema SFA

SFA viene eseguito su di un hardware installato a bordo treno, e la sua esecuzione è volta a monitorare costantemente il moto del treno.

Il ciclo di esecuzione di SFA è essenzialmente una continua iterazione di due distinte fasi logiche:

- Acquisizione delle misure;
- Predizione della posizione del treno.

Le grandezze fisiche che dovranno essere misurate e fornite a SFA sono:

- Vettore accelerazione;
- Vettore velocità angolare;
- Coordinate geografiche;
- Velocità lineare (scalare).

In quest'applicazione, SFA utilizza tali informazioni in combinazione con un'apposita digitalizzazione della traccia tramviaria su cui si trova il treno monitorato.[18][19][20]

Queste informazioni si suppongono note a priori ed accedibili tramite un *database* caricato in memoria centrale. [21]

## 2.2 SISTEMI COSTITUENTI

Il sottosistema di posizionamento si compone dei moduli, o *Constituent Systems* (CS), descritti nel seguito di questa sezione.

### 2.2.1 Sensor Set

Il *Sensor Set* è un insieme di sensori atti a campionare le misure richieste da SFA. Esso si compone a sua volta dei seguenti moduli:

- *Inertial Measurement Unit* (IMU):  
Sensore inerziale incaricato di campionare e trasmettere a SFA i vettori accelerazione ( $\mathbf{a}$ ) e velocità angolare ( $\mathbf{v}_{\text{ang}}$ ). Le misure di IMU sono prese rispetto alla Terra e sono espresse in unità stabilite dallo standard internazionale (SI):

$$\mathbf{a} \left[ \frac{\text{m}}{\text{s}^2} \right] \quad \mathbf{v}_{\text{ang}} \left[ \frac{\text{rad}}{\text{s}} \right]$$

IMU è il sensore principale su cui si basa SFA nel predire la posizione del treno. Date le caratteristiche intrinseche del particolare SFA utilizzato, ossia un *Filtro di Kalman*, il sistema funziona anche senza i rimanenti sensori. Si osserverebbe tuttavia un calo delle performance in termini di errore commesso sulla stima della posizione del treno. [22] [23]

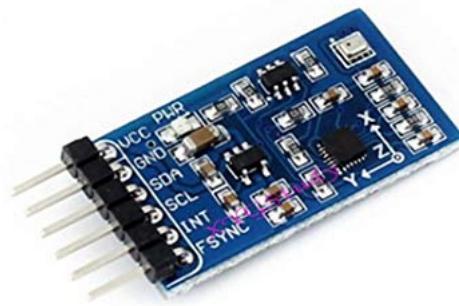


Figura 7: Inertial Measurement Unit

- Odometro:

Unità incaricata di fornire a SFA i valori di velocità lineare del treno, espressi in  $\frac{\text{m}}{\text{s}}$ .

- GPS:

Unità che fornisce a SFA le misure di posizione del treno.

Le misure di GPS sono riportate in formato standard come tripla di coordinate (*latitudine*, *longitudine*, *altitudine*), rispettivamente espresse in gradi N-S, in gradi E-O e in metri sul livello del mare.



Figura 8: Ricevitore GPS ublox EVK-M8T

### 2.2.2 Piattaforma di elaborazione dati

La piattaforma di elaborazione dati è l'hardware sul quale viene eseguito SFA. Consiste di una scheda Nvidia TX-Jetson collegata al *Sensor Set*. Da quest'ultimo essa riceve le misure da processare tramite SFA. Nvidia



Figura 9: Nvidia TX-Jetson

TX-Jetson è un'architettura specifica per sistemi *embedded*. Essa è ottimizzata per i calcoli computazionalmente onerosi tipici delle applicazioni di intelligenza artificiale.[24]

<b>GPU</b>	256-core NVIDIA Pascal
<b>CPU</b>	Dual-Core NVIDIA Denver 2 64-Bit CPU Quad-Core ARM Cortex-A57 MPCore
<b>Memoria</b>	8GB 128-bit LPDDR4 Memory
<b>Storage</b>	32GB eMMC 5.1
<b>Alimentazione</b>	7.5W / 15W

Tabella 1: Specifiche Tecniche NVidia TX-Jetson

### 2.2.3 On Board Control Unit

L'*On Board Control Unit* (OBCU) è il computer di bordo del treno. Esso non svolge alcun ruolo attivo nel sistema di posizionamento, tuttavia la

progressiva chilometrica, stimata da SFA, dovrà essere trasmessa a OBCU al fine di poter utilizzare questa informazione all'interno del sistema di *interlocking* della traccia.

## 2.3 SPECIFICA DELLE INTERFACCE

### 2.3.1 Relied Upon Interfaces

Le interfacce sono definite come punti di interazione, tra un CS e l'ambiente oppure tra un CS e un altro.

In questa sezione si evidenziano le principali interfacce del sistema, alle quali si osservano le interazioni fondamentali che avvengono al suo interno.

Tali interfacce prendono il nome di *Relied Upon Interfaces* (RUI). Le RUI si dividono in:

- *Relied Upon Physical Interfaces* (RUPI), in cui l'interazione avviene tramite osservazione diretta di una grandezza fisica;
- *Relied Upon Message Interfaces* (RUMI), dove l'interazione è rappresentata da uno scambio di messaggi a livello *cyber*.

La specifica delle RUI è di particolare importanza poiché qualunque struttura del sistema, responsabile del comportamento osservato, può essere ridotta alla specifica delle interfacce del sistema. [25]

Il CPS interagisce con l'ambiente attraverso le RUPI del *Sensor Set*, ossia gli strumenti di misura che esso integra. Queste interfacce acquisiscono, a diverse frequenze, i dati sul moto del treno che verranno elaborati da SFA. Una descrizione sintetica delle RUPI del sistema è mostrata in tabella 2.

RUPI	Grandezza Campionata	Parti interagenti
Accelerometro	Accelerazione	Ambiente - IMU
Giroscopio	Velocità angolare	Ambiente - IMU
Radar	Velocità lineare	Ambiente - Odometro
Ricevitore GPS	Coordinate geografiche	Ambiente - GPS

Tabella 2: Specifica delle RUPI del sistema

Per quanto concerne le RUMI, se ne osservano di due tipi:

- Tre bus dati, che collegano il *Sensor Set* alla scheda Nvidia TX-Jetson. Su ciascuno di essi, *Sensor Set* invia rispettivamente messaggi contenenti i dati campionati da IMU, Odometro e GPS.
- Interfaccia LTE. Essa permette di realizzare una *rete wireless ad hoc* fra la scheda e OBCU.

All'interno di tale rete vengono instradati datagrammi IP contenenti le informazioni sulla progressiva chilometrica stimata da SFA, ed eventualmente messaggi di *acknowledgment* di OBCU verso la scheda.



Figura 10: Modem TP-LINK M7350 LTE-4G

RUMI	Informazione trasmessa	Parti interagenti
Bus Dati 1	Accelerazione, Velocità angolare	Sensor Set - Nvidia TX-Jetson
Bus Dati 2	Velocità lineare	Sensor Set - Nvidia TX-Jetson
Bus Dati 3	Coordinate geografiche	Sensor Set - Nvidia TX-Jetson
LTE	Posizione del treno	Nvidia TX-Jetson - OBCU

Tabella 3: Specifica delle RUMI del sistema

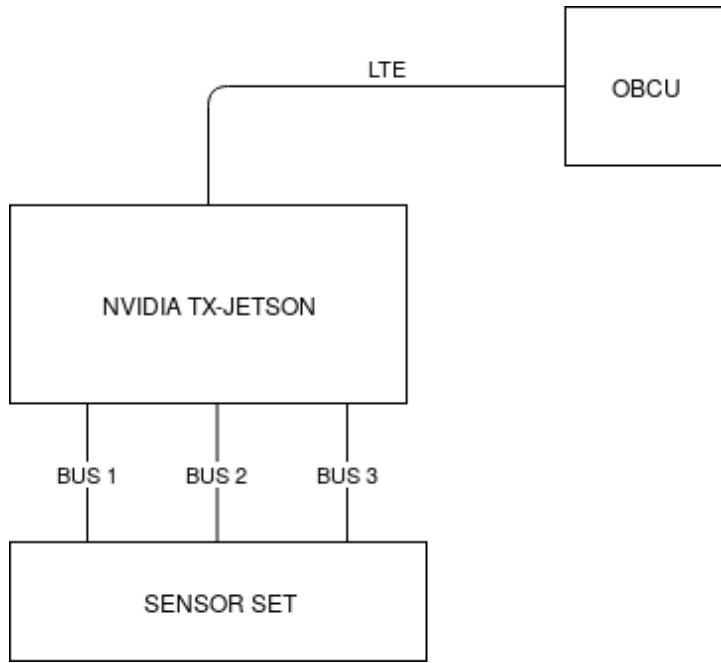


Figura 11: Architettura hardware del sottosistema di posizionamento

## 2.4 INTERAZIONI

In questa sezione si descrivono le interazioni osservabili alle interfacce sopra descritte. Queste possono essere in prima istanza categorizzate a discrezione della fase di SFA in cui esse avvengono.

Si distinguono pertanto le interazioni riguardanti l'acquisizione dei dati in ingresso a SFA, e le interazioni riguardanti l'acquisizione da parte di OBCU della posizione del treno.

### 2.4.1 Acquisizione dei dati

L'acquisizione dei dati si divide in due differenti interazioni: la prima, con l'ambiente, avviene alle RUPI del *Sensor Set*, mentre la seconda avviene alle RUMI bus dati che collegano il *Sensor Set* alla piattaforma di elaborazione dati.

I moduli che compongono il *Sensor Set* campionano ad una data frequenza le grandezze fisiche che descrivono il moto del treno. Ciascun campionamento fisico è seguito dall'invio dei valori letti alla piattaforma di elaborazione dati. I moduli del *Sensor Set* sono tra di loro indipendenti. In figura 12 viene riportato un *sequence diagram* rappresentante una possibile sequenza di campionamento e invio dei dati.

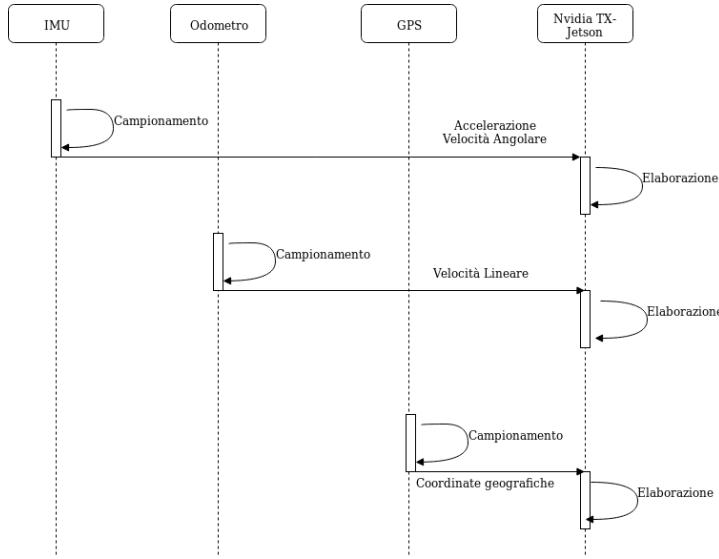


Figura 12: Sequenza di acquisizione dati

Questa tipologia di interazione è detta *time-triggered*, in quanto è determinata unicamente dallo scorrere del tempo. [28]

#### 2.4.2 Trasmissione della posizione

La piattaforma di elaborazione dati esegue SFA durante l'intero moto del treno. Le misure fornite dai sensori vengono elaborate al fine di aggiornare continuamente la stima della posizione del treno.

Ogniqualvolta venga completato un aggiornamento di SFA, si osserva un'interazione all'interfaccia LTE. Tale interazione consiste nell'invio di un messaggio contenente la posizione del treno, dalla piattaforma di elaborazione dati verso OBCU, e nella trasmissione di un messaggio di *acknowledgment* nel senso opposto.

La tipologia di scambio dei messaggi esposta è detta *event-triggered* [29] in quanto le tempistiche di interazione non sono note a priori, ma dipendono dal tempo computazionale impiegato da SFA nell'aggiornare la propria stima della posizione.

LTE è a tutti gli effetti una regolare interfaccia di rete. Il messaggio trasmesso è contenuto nel *payload* di un datagramma UDP; in accordo al modello di rete ISO-OSI. [30]

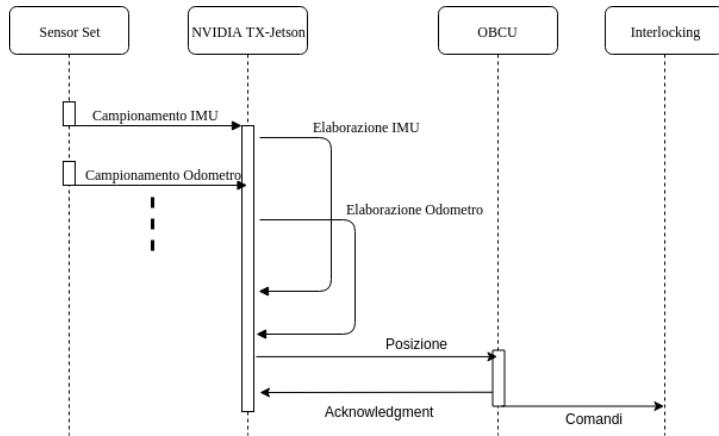


Figura 13: Sequenza di trasmissione della posizione

#### 2.4.3 Interazioni con eventuali operatori umani

La filosofia del sistema oggetto della Tesi include la minimizzazione di interventi da parte di operatori umani.

A questi viene lasciato il compito di predisporre le informazioni geografiche della traccia nel database, e di segnalare l'avvio al sistema.

Il processo che esegue SFA è in effetti un *demone* che si avvia contestualmente all'accensione della piattaforma.

Viene infine predisposta un interfaccia SSH per accedere da remoto alla piattaforma a scopi di *deployment* o di *monitoring online* attraverso la lettura dei file di *log*.



# 3

---

## ARCHITETTURA SOFTWARE

---

In questo capitolo viene completata l'esposizione dell'architettura di sistema descrivendo i moduli software in esecuzione sulla piattaforma Nvidia TX-Jetson. Su tale piattaforma è stato installato il sistema operativo Ubuntu 16.04 LTS, basato su Kernel Linux.

I sistemi software che vengono eseguiti su Nvidia TX-Jetson costituiscono l'architettura software dell'intero sottosistema di posizionamento, e pertanto dovranno essere opportunamente riprodotti nell'ambiente di analisi descritto nel prossimo capitolo.

### 3.1 SENSOR FUSION LIBRARY

Il software che esegue SFA viene fornito come libreria, denominata *SensorFusionLib*, la quale mette a disposizione dei *client* le API descritte di seguito.

#### 3.1.1 API

Le API di *SensorFusionLib* definiscono le interfacce software verso il modulo SFA che possono essere utilizzate dai *client*.

Le principali funzioni disponibili sono le seguenti:

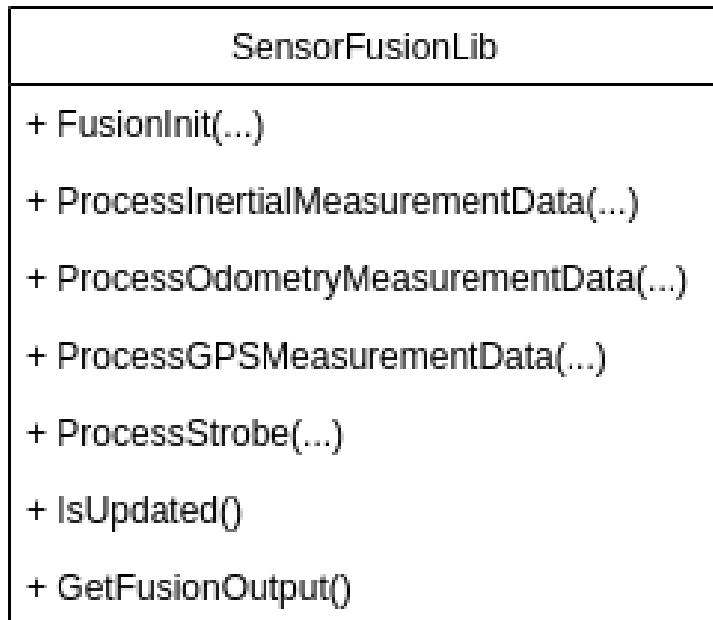
- **FusionInit(args...)**

Funzione che inizializza SFA. Tale funzione deve essere chiamata quando è necessario avviare l'algoritmo. Essa riceve come parametri i valori che caratterizzano le condizioni iniziali del moto, come l'errore iniziale rispetto alla progressiva chilometrica e alla velocità;

- **ProcessInertialMeasurementData(args...)**

Funzione che permette a SFA di ricevere ed elaborare un campionamento di IMU;

- **ProcessOdometryMeasurementData(args...)**  
Funzione che permette a SFA di ricevere ed elaborare un campionamento di Odometro;
- **ProcessGPSMeasurementData(args...)**  
Funzione che permette a SFA di ricevere ed elaborare un campionamento di GPS;
- **ProcessStrobe(args...)**  
Funzione che deve essere invocata ogni secondo, per permettere a SFA di sincronizzarsi rispetto a una *global timebase* esterna; [26]
- **IsUpdated()**  
Funzione che restituisce vero se SFA ha completato un'iterazione ed è pronto a fornire l'output prodotto;
- **GetFusionOutput()**  
Se `IsUpdated()` restituisce vero, è possibile invocare questa funzione per ricevere da SFA l'ultimo output calcolato.

Figura 14: API di *SensorFusionLib*

### 3.2 LISTENER

*SensorFusionLib* è incapsulato all'interno di un eseguibile, *listener*. Questo software possiede un'istanza di *SensorFusionLib* con la quale interagisce attraverso le API descritte in 3.1.1. Esso dispone di due *socket UDP*, una utilizzata per ricevere i dati *raw* provenienti dai sensori, l'altra per la comunicazione remota con OBCU.

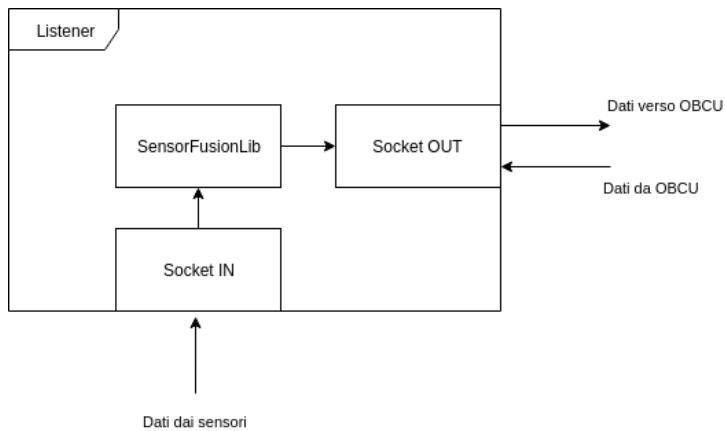


Figura 15: Architettura *listener*

#### 3.2.1 Database Interface

Come anticipato nel capitolo 2, un ulteriore ingresso a SFA è rappresentato dalle informazioni geografiche della traccia su cui si trova il treno. Prima di poter inizializzare SFA attraverso una chiamata a `FusionInit()`, occorre specificare il database da utilizzare, e la classe concreta che implementa la comunicazione con esso.

*SensorFusionLib* contiene una classe astratta `DatabaseInterfaceEdges` implementata da due classi concrete:

- `SQLITEDatabaseInterfaceEdges`, da utilizzare per connessioni verso un database *sqlite*;
- `MYSQLDatabaseInterfaceEdges`, da utilizzare per connessioni verso un database *MySql*.

Quando *SensorFusionLib* viene eseguito a bordo treno, deve essere specificato un file *sqlite* che contiene il database da caricare in memoria centrale. Il software *listener* ricerca automaticamente un file *sqlite* nella directory

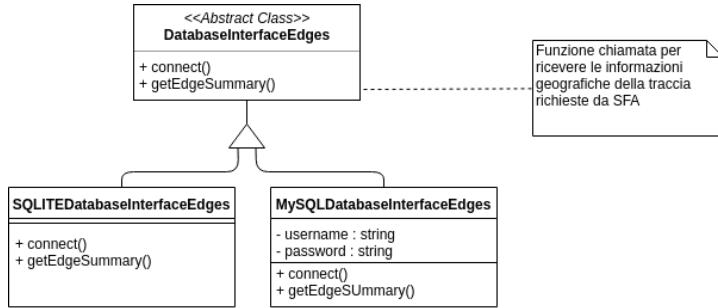


Figura 16: Class diagram `DatabaseInterfaceEdges`

in cui si trova l'eseguibile, ed istanzia una connessione verso di esso utilizzando la classe concreta `SQLITEDatabaseInterfaceEdges`.

SFA modella la determinata traccia ferrotramviaria lungo la quale si vuole eseguire l'algoritmo, come una *spline* interpolante una lista di *punti* sulla superficie della Terra.

All'interno del database, ciascuna traccia è memorizzata in una tabella *edges*.

Una tabella *edge\_vertices* memorizza le informazioni di progressiva chilometrica caratteristiche di ciascun punto geografico, associandolo alla traccia di appartenenza.

Un'ultima tabella *vertices* associa a ciascun punto geografico le sue coordinate, espresse in formato *Earth Centered Earth Fixed* (ECEF).

ECEF è uno standard per esprimere la posizione di un oggetto in un sistema di riferimento cartesiano con origine nel centro della Terra.

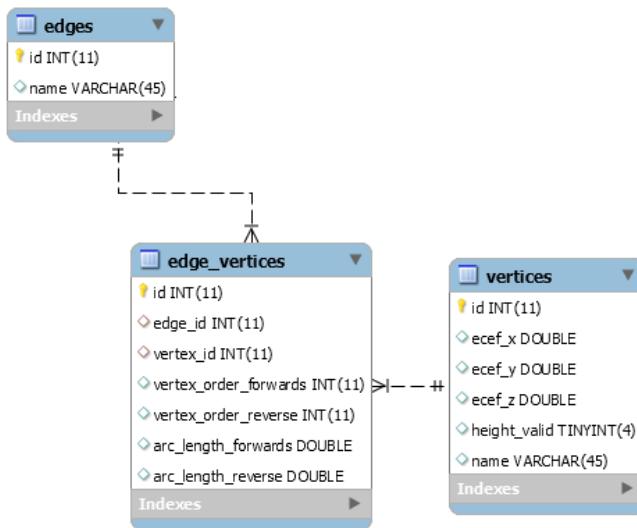


Figura 17: Schema database tracce

### 3.3 INTERFACE MODULES

La comunicazione con i sensori è gestita interamente da un set di processi denominati *interface-modules*. Ciascun sensore è collegato ad un'interfaccia seriale della piattaforma attraverso un bus dati. Per ogni interfaccia connessa, un processo resta in ascolto su di essa. Quando un sensore invia un campionamento su una specifica interfaccia, il processo in ascolto su quest'ultima si fa carico di inoltrare a *listener* i valori ricevuti.

Ciascun processo di *interface-modules* dispone di una socket *UDP* che abilita la comunicazione con *listener*.

<b>Modulo</b>	<b>Sensore</b>	<b>Dati Inviati a <i>listener</i></b>
<i>IMU Process</i>	IMU	Accelerazione, Velocità angolare
<i>ODO Process</i>	Odometro	Velocità lineare
<i>GPS Process</i>	GPS	Coordinate Geografiche
<i>Strobe Process</i>	N/A	Sincronizzazione

Tabella 4: Moduli di *interface-modules*

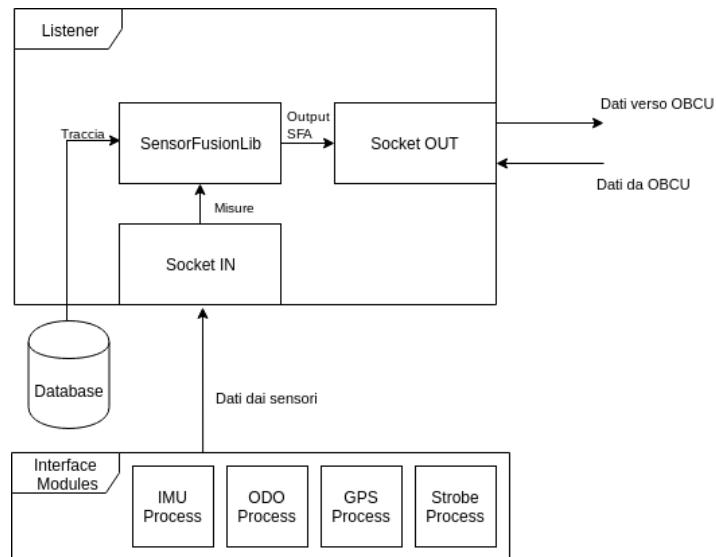


Figura 18: Architettura software completa

# 4

---

## AMBIENTE DI ANALISI

---

Al fine di osservare il comportamento di SFA, e condurre l'analisi oggetto della Tesi, è stato costruito un ambiente atto a monitorare, in modo non intrusivo, il comportamento dell'algoritmo. [27]

L'ambiente di analisi riproduce l'architettura software descritta nel capitolo 3, su una piattaforma x86 equipaggiata con il sistema operativo Windows 7.

### 4.1 PRINCIPI DI BASE

L'ambiente di analisi deve essere tale da rispettare i principi della medesima. Non c'è interesse, nella fattispecie, a osservare il sistema a *runtime*, ma lo scopo è unicamente quello di osservare il comportamento di SFA al variare dei dati in ingresso e dei parametri di configurazione.

Vengono dunque escluse dalle campagne di analisi l'acquisizione delle misure dai sensori e la trasmissione degli output di SFA verso OBCU.

L'intero processo di analisi viene condotto attraverso un software appositamente realizzato, il *Rail Track Tool* (RTT).

Il sistema di acquisizione e trasmissione delle misure viene opportunamente *mockato* [33] da una libreria usata da RTT, denominata *SynthDataGen*, che fornisce allo stesso un *framework* atto a simulare il comportamento dei sensori. RTT è dunque in grado di generare le misure che verosimilmente si otterrebbero sulla traccia in esame.

### 4.2 SOFTWARE IMPIEGATI

#### 4.2.1 *SensorFusionLib*

Libreria oggetto dell'analisi. Essa è la stessa libreria che viene utilizzata da *listener* nel sistema reale, ed è stata descritta nel capitolo 3.

#### 4.2.2 *SynthDataGen*

Nell'ambiente di analisi, la libreria *SynthDataGen* implementa la logica di *interface-modules* nel sistema reale.

*SynthDataGen* utilizza le informazioni geografiche della traccia per generare i campionamenti di IMU e Odometro, tuttavia non supporta la generazione delle informazioni GPS.

Nel sistema reale, le misure campionate da ciascun sensore sono caratterizzate da un *rumore di misura* [32], motivo per cui si palesa la necessità di utilizzare SFA.

Il rumore di misura è una caratteristica intrinseca di qualunque sensore, e viene modellato come una variabile aleatoria *gaussiana* a media nulla il cui valore assunto viene sommato alle misurazioni. La variabile aleatoria che rappresenta il rumore di misura viene dunque completamente descritta dalla sua *varianza* nel caso di generazioni univariate (Odometro), e dalla *matrice di covarianza* nel caso di generazioni multivariate (IMU).

L'ultima informazione che è necessario specificare per utilizzare *SynthDataGen* è la frequenza di campionamento di ciascun sensore simulato, espressa in *hertz*.

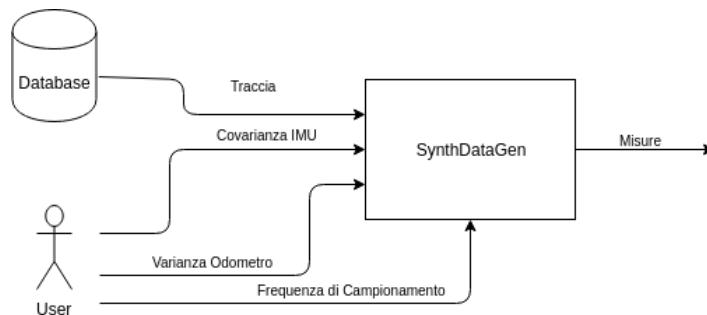


Figura 19: Schema di *SynthDataGen*

#### 4.2.3 *Rail Track Tool*

RTT è un software *front-end* realizzato al fine di valutare le performance di SFA.

Gli output di SFA non sono esclusivamente limitati alla stima della posizione del treno espressa come progressiva chilometrica; infatti date le caratteristiche dell'algoritmo, altre informazioni utili ai fini dell'analisi che esso fornisce sono:

- Coordinate ECEF della posizione stimata del treno;

- Stima della velocità proiettata lungo i 3 assi cartesiani;
- Errore commesso nella stima di posizione e velocità.

. Il termine *errore* indica la differenza, in valore assoluto, tra la stima prodotta e la misura reale.

Nel sistema reale queste informazioni vengono semplicemente *loggated* a scopi di debug o di *monitoring online*, mentre nell'ambiente simulato da RTT queste informazioni vengono mostrate per intero. Non c'è dunque la necessità di modificare il codice di *SensorFusionLib* per valutare le misure di interesse, in quanto queste vengono calcolate per definizione dell'algoritmo. Quanto esposto dimostra la non intrusività dell'attività di analisi.

RTT include tra le sue dipendenze la libreria *SensorFusionLib*, e ne sfrutta le relative API per alimentare SFA ed ottenere gli output dell'algoritmo; ed include *SynthDataGen* per simulare il comportamento dei sensori.

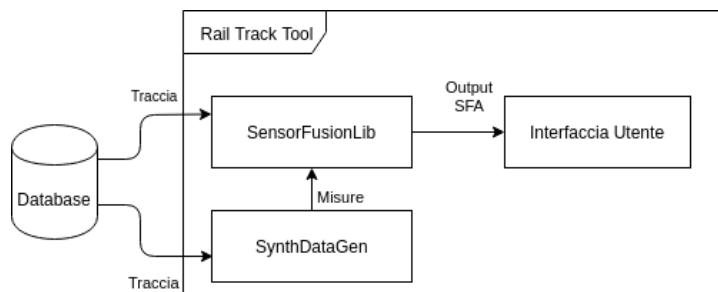


Figura 20: Schema riassuntivo RTT

### *Interfaccia Utente*

All'avvio di RTT viene mostrata una finestra contenente l'interfaccia del software verso l'utente.

Gli elementi che compongono tale interfaccia sono:

1. **Slippy map:** Mappa che visualizza le tracce memorizzate, all'interno delle quali verrà mostrata la posizione del treno durante l'esecuzione di SFA;
2. **Elevation plot:** Grafico che mostra l'altitudine della traccia in funzione della progressiva chilometrica della stessa;

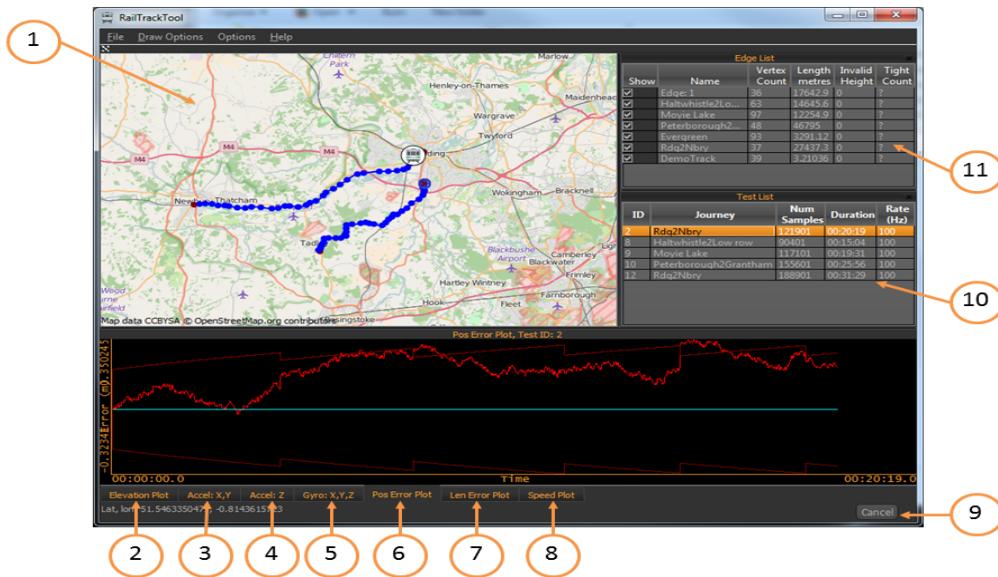


Figura 21: Interfaccia RTT

3. Accel X,Y: Grafico delle misure dell'accelerometro fornite a SFA, relative agli assi normale e tangenziale alla traccia selezionata;
4. Accel Z: Grafico delle misure dell'accelerometro fornite a SFA, relative all'asse verticale alla traccia selezionata;
5. Gyro X,Y,Z: Grafico delle misure del giroscopio fornite a SFA, relative agli assi normale, tangenziale e verticale alla traccia selezionata;
6. Pos error plot: Grafico che riporta, in funzione del tempo, la stima dell'errore commesso nel predire la posizione del treno;
7. Len error plot: Come Pos error plot, ma l'errore è espresso rispetto alla progressiva chilometrica e non rispetto al vettore posizione;
8. Speed Plot: Grafico che mostra la stima della velocità del treno come funzione del tempo;
9. Cancel button: Pulsante da premere se si ha la necessità di interrompere una simulazione;
10. Test List: Storico delle analisi effettuate;
11. Edge List: Lista delle tracce memorizzate.

### Monitoring con RTT

Tramite un'apposita interfaccia interna a RTT, l'utente ha la possibilità di definire i parametri generali con cui vengono generate le misurazioni, come ad esempio velocità massima e frequenza di campionamento (figura 22).

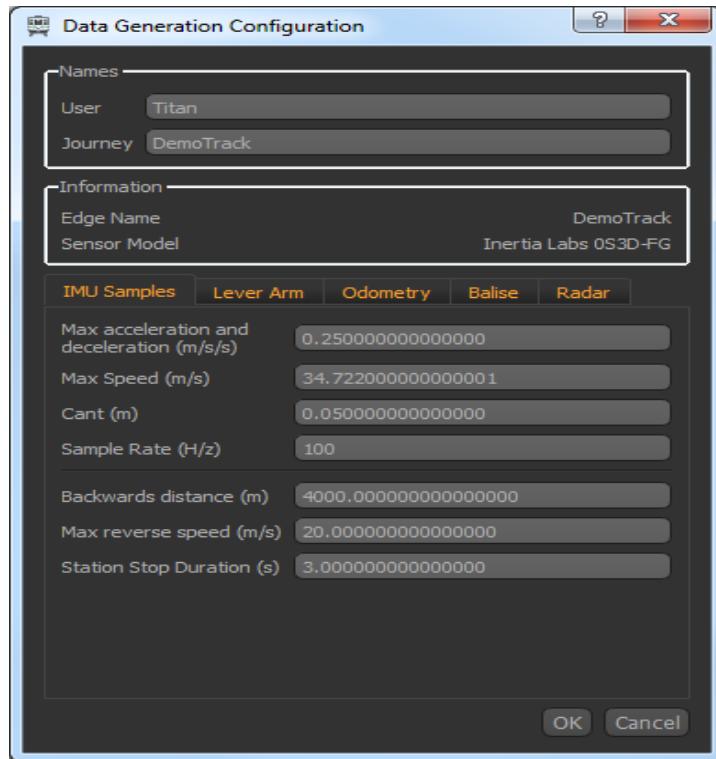


Figura 22: Pannello di configurazione generale *SynthDataGen*

Un'altra interfaccia permette infine di definire le caratteristiche dei sensori simulati, come il rumore del processo di misura (figura 23).

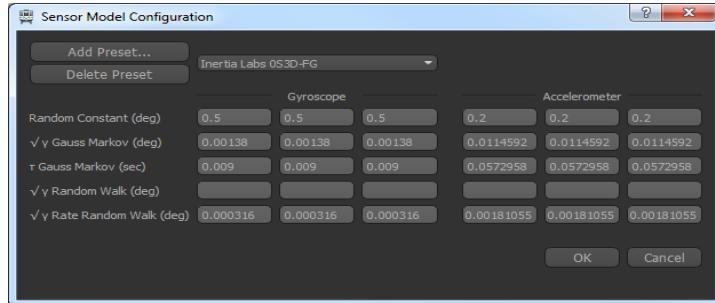


Figura 23: Pannello di configurazione IMU simulato

Definiti i parametri di configurazione è possibile selezionare una traccia, e simulare l'esecuzione di SFA.

Durante la simulazione, la posizione del treno stimata dall'algoritmo verrà visualizzata sulla mappa.

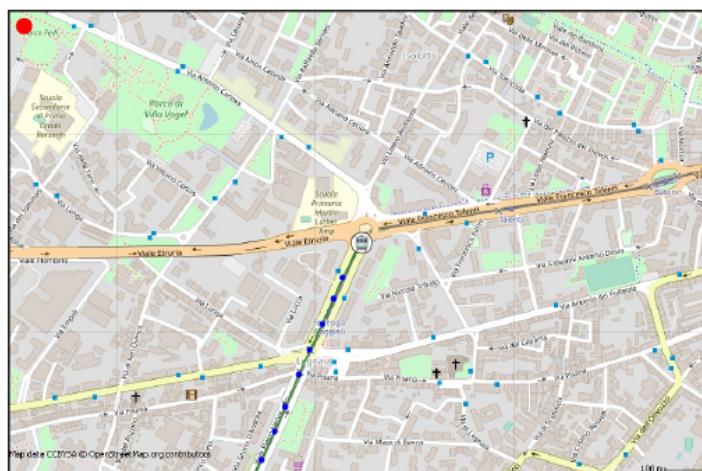


Figura 24: Posizione del treno mostrata sulla mappa

Sui grafici nel pannello inferiore è possibile inoltre visualizzare i dati forniti in ingresso all'algoritmo, e gli errori commessi dallo stesso durante il processo di stima.

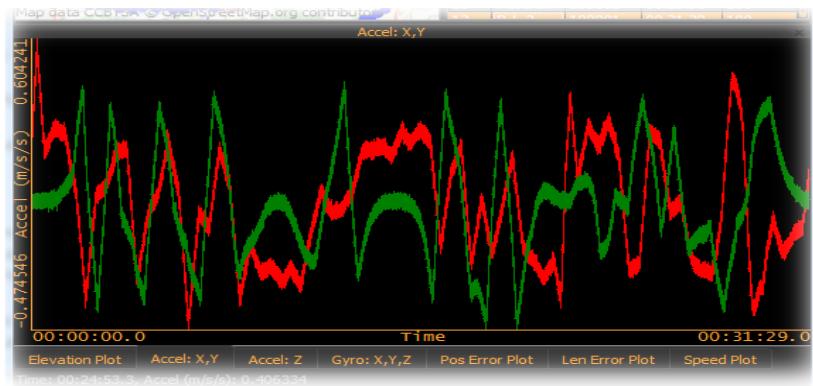


Figura 25: Grafico valori di accelerazione assi X e Y



Figura 26: Grafico errore sulla stima della posizione

Al termine di ciascuna simulazione, RTT produce un report HTML contenente la sintesi dei principali risultati relativi alla simulazione appena effettuata.

#### *Database Interface*

Come nel sistema reale, le informazioni geografiche relative alle tracce in esame sono memorizzate in un database. A differenza di quanto avviene nel sistema *embedded*, in cui si utilizza un database *sqlite*, nell'ambiente di analisi il database utilizzato è di tipo *MySql*. Permane comunque inalterato lo schema logico del database (capitolo 3).

Quanto esposto implica che all'atto di inizializzazione di SFA, RTT istanzi una connessione verso un database *MySql*, implementata dalla classe concreta *MySqlDatabaseInterfaceEdges*.



# 5

---

## PARTE Sperimentale

---

Questo capitolo rappresenta la parte sperimentale del lavoro di Tesi. L'attività di analisi effettuata rientra nella categoria del *monitoring* dei sistemi software.

Nel seguito si illustrano gli esperimenti condotti sul sistema nell'ambiente descritto nel capitolo 4, e si discutono i risultati ottenuti.

La traccia ferrotramviaria scelta per l'analisi è un sottoinsieme della linea T1 della Tramvia di Firenze, che si estende per 4.25044 chilometri dal terminal di *Villa Costanza*, nel comune di Scandicci.



Figura 27: Traccia di analisi

### 5.1 MISURE DI INTERESSE

Le performance del sistema saranno valutate in termini degli errori che SFA commette nella stima delle seguenti grandezze:

- Coordinate ECEF della posizione del treno;

- Velocità proiettata sugli assi cartesiani;

Al variare dei seguenti parametri:

- Numero di sensori integrati (Scenario 1);
- Frequenza di campionamento IMU (Scenario 2);
- Frequenza di campionamento odometro (Scenario 3);
- Varianza rumore di misura odometro (Scenario 4);
- Covarianza rumore di misura IMU (Scenario 5).

Gli scenari da 1 a 3 hanno lo scopo di valutare l'impatto della frequenza di campionamento di ciascun sensore su SFA, al fine di determinare una configurazione ottimale. Si assume di utilizzare sensori ideali caratterizzati da un rumore di misura nullo.

Il focus dell'analisi si sposta in seguito alla valutazione dell'impatto sulle performance di SFA del rumore di misura, fissate le frequenze di campionamento.

Per ciascuna grandezza osservata viene riportato il valore medio, il valore massimo e la deviazione standard (dev. std.) dell'errore commesso da SFA nel relativo processo di stima.

## 5.2 ESPERIMENTI

### 5.2.1 Scenario 1

#### *Esperimento 1.1*

Sensori integrati	Frequenza IMU	Frequenza odometro	Errore iniziale
IMU	100 Hz	-	20 m

Tabella 5: Scenario 1, Esperimento 1.1

Alimentare SFA utilizzando esclusivamente le misurazioni di IMU conducono a una netta divergenza dell'errore, sia in termini di posizione che in termini di velocità.

Si osserva che l'errore sulla posizione non varia significativamente lungo l'asse z, poiché la linea scelta per gli esperimenti si sviluppa in un'area pianeggiante, non caratterizzata da importanti cambi di altitudine.

Questi risultati non sono accettabili.

Misura	Errore medio	Errore massimo	Dev. std. errore
ECEF X	861.883 m	2431.1 m	678.953 m
ECEF Y	348.814 m	1518.65 m	499.222 m
ECEF Z	0.123305 m	0.155086 m	0.567656 m
Velocità X	8.20331 m/s	30.782 m/s	7.32822 m/s
Velocità Y	23.4213 m/s	75.1929 m/s	20.0333 m/s
Velocità Z	87.7399 m/s	87.1907 m/s	245.723 m/s

Tabella 6: Esperimento 1.1: Risultati

*Esperimento 1.2*

Sensori integrati	Frequenza IMU	Frequenza odometro	Errore iniziale
IMU, Odometro	100 Hz	10 Hz	20 m

Tabella 7: Scenario 1, Esperimento 1.2

Misura	Errore medio	Errore massimo	Dev. std. errore
ECEF X	3.5826 m	20.1141 m	5.60308 m
ECEF Y	0.0243133 m	0.362813 m	0.0452763 m
ECEF Z	$3.56432 \cdot 10^{-6}$ m	$3.19201 \cdot 10^{-5}$ m	$8.81543 \cdot 10^{-6}$ m
Velocità X	0.0169528 m/s	0.124472 m/s	0.0199173 m/s
Velocità Y	0.0394826 m/s	0.847261 m/s	0.0828195 m/s
Velocità Z	0.00382241 m/s	0.0192343 m/s	0.00314704 m/s

Tabella 8: Esperimento 1.2: Risultati

Utilizzando anche l'odometro, si ottiene una netta riduzione degli errori commessi. L'errore massimo commesso sulla stima della posizione permane in un intorno del valore iniziale di 20 metri.

Questo esperimento viene considerato *golden run*, in quanto corrisponde alla configurazione standard del sistema. I risultati osservati durante questo esperimento saranno confrontati. Tale confronto viene riportato in termini di differenza percentuale.

### 5.2.2 Scenario 2

#### Esperimento 2.1

Sensori integrati	Frequenza IMU	Frequenza odometro	Errore iniziale
IMU, Odometro	50 Hz	10 Hz	20 m

Tabella 9: Scenario 2, Esperimento 2.1

Misura	Errore medio	Errore massimo	Dev. std. errore
ECEF X	3.57373 m	20.1609 m	5.60304 m
ECEF Y	0.0234386 m	0.366496 m	0.0445943 m
ECEF Z	3.55578e-06 m	3.19863e-05 m	8.7636e-06 m
Velocità X	0.0184494 m/s	0.129497 m/s	0.0222426 m/s
Velocità Y	0.0396467 m/s	0.863711 m/s	0.084737 m/s
Velocità Z	0.00355928 m/s	0.0189619 m/s	0.00306268 m/s

Tabella 10: Esperimento 2.1: Risultati

Misura	Errore medio	Errore massimo	Dev. std. errore
ECEF X	-0.247586 %	+0.232673 %	-0.000714 %
ECEF Y	-3.59762 %	+1.01512 %	-1.50631 %
ECEF Z	-0.239597 %	+0.207393 %	-0.587946 %
Velocità X	+8.82804 %	+4.03705 %	+11.6748 %
Velocità Y	+0.415626 %	+1.94155 %	+2.31528 %
Velocità Z	-6.88388 %	-1.41622 %	-2.68061 %

Tabella 11: Esperimento 2.1: Confronto con esperimento 1.2

Il dimezzamento della frequenza di campionamento di IMU ha causato un lieve degrado delle performance di SFA nell'errore massimo commesso nella stima della posizione e della velocità.

È ragionevole supporre che 50 hertz sia un valore di frequenza IMU comunque sufficiente a garantire risultati accettabili, considerato che l'errore rimane comunque nell'ordine del valore iniziale.

### *Esperimento 2.2*

Sensori integrati	Frequenza IMU	Frequenza odometro	Errore iniziale
IMU, Odometro	20 Hz	10 Hz	20 m

Tabella 12: Scenario 2, Esperimento 2.2

Misura	Errore medio	Errore massimo	Dev. std. errore
ECEF X	4.3248 m	20.5617 m	5.41018 m
ECEF Y	0.0226056 m	0.39742 m	0.04816 m
ECEF Z	3.81041e-06 m	3.30294e-05 m	9.04865e-06 m
Velocità X	0.0248871 m/s	0.150687 m/s	0.0260141 m/s
Velocità Y	0.0475246 m/s	0.908185 m/s	0.0899591 m/s
Velocità Z	0.00406042 m/s	0.0228906 m/s	0.00340827 m/s

Tabella 13: Esperimento 2.2: Risultati

Misura	Errore medio	Errore massimo	Dev. std. errore
ECEF X	+20.7168 %	+2.22531 %	-3.44275 %
ECEF Y	-7.02373 %	+9.53852 %	+6.36912 %
ECEF Z	+6.90426 %	+3.47524 %	+2.64559 %
Velocità X	+46.8023 %	+21.061 %	+30.6106 %
Velocità Y	+20.3685 %	+7.1907 %	+8.62068 %
Velocità Z	+6.2267 %	+19.0093 %	+8.30082 %

Tabella 14: Esperimento 2.2: Confronto con esperimento 1.2

L'ulteriore diminuzione della frequenza di campionamento IMU ha questa volta prodotto un degrado delle performance di SFA non trascurabile. In particolare, la velocità risulta la grandezza più sensibile al variare dell'frequenza di IMU.

### *Esperimento 2.3*

Il calo della frequenza IMU fino al valore di 10 hertz ha impattato negativamente in maniera significativa e piuttosto diffusa sulle performance di

Sensori integrati	Frequenza IMU	Frequenza odometro	Errore iniziale
IMU, Odometro	10 Hz	10 Hz	20 m

Tabella 15: Scenario 2, Esperimento 2.3

Misura	Errore medio	Errore massimo	Dev. std. errore
ECEF X	5.2473 m	21.1805 m	5.35502 m
ECEF Y	0.0267588 m	0.512581 m	0.0640117 m
ECEF Z	4.47911e-06 m	3.49469e-05 m	9.3992e-06 m
Velocità X	0.0379368 m/s	0.223375 m/s	0.0360634 m/s
Velocità Y	0.0634147 m/s	1.05148 m/s	0.108924 m/s
Velocità Z	0.00396413 m/s	0.0226519 m/s	0.00362388 m/s

Tabella 16: Esperimento 2.3: Risultati

SFA. Risultano particolarmente degradate sia le stime di velocità che le stime di posizione.

Misura	Errore medio	Errore massimo	Dev. std. errore
ECEF X	+46.4663 %	+5.30175 %	-4.42721 %
ECEF Y	+10.0583 %	+41.2797 %	+41.3801 %
ECEF Z	+25.6652 %	+9.48243 %	+6.62214 %
Velocità X	+123.779 %	+79.458 %	+81.0657 %
Velocità Y	+60.6143 %	+24.1034 %	+31.5198 %
Velocità Z	+3.70761 %	+17.7683 %	+15.152 %

Tabella 17: Esperimento 2.3: Confronto con esperimento 1.2

### 5.2.3 Scenario 3

#### Esperimento 3.1

Sensori integrati	Frequenza IMU	Frequenza odometro	Errore iniziale
IMU, Odometro	100 Hz	20 Hz	20 m

Tabella 18: Scenario 3, Esperimento 3.1

Misura	Errore medio	Errore massimo	Dev. std. errore
ECEF X	3.24767 m	20.1043 m	5.63575 m
ECEF Y	0.022779 m	0.314233 m	0.0412647 m
ECEF Z	3.43522e-06 m	3.19257e-05 m	8.73175e-06 m
Velocità X	0.0132552 m/s	0.0951454 m/s	0.0153522 m/s
Velocità Y	0.0340335 m/s	0.785133 m/s	0.0745504 m/s
Velocità Z	0.00326259 m/s	0.0182945 m/s	0.00296276 m/s

Tabella 19: Esperimento 3.1: Risultati

In linea con le aspettative, il raddoppio della frequenza dell'odometro ha portato a un miglioramento diffuso delle performance di SFA.

Misura	Errore medio	Errore massimo	Dev. std. errore
ECEF X	-9.3488 %	-0.048722 %	+0.583072 %
ECEF Y	-6.31054 %	-13.3898 %	-8.86027 %
ECEF Z	-3.62201 %	+0.017544 %	-0.949245 %
Velocità X	-21.8111 %	-23.5608 %	-22.9203 %
Velocità Y	-13.8013 %	-7.3328 %	-9.98448 %
Velocità Z	-14.6457 %	-4.88606 %	-5.85566 %

Tabella 20: Esperimento 3.1: Confronto con esperimento 1.2

*Esperimento 3.2*

Sensori integrati	Frequenza IMU	Frequenza odometro	Errore iniziale
IMU, Odometro	100 Hz	5 Hz	20 m

Tabella 21: Scenario 3, Esperimento 3.2

Misura	Errore medio	Errore massimo	Dev. std. errore
ECEF X	7.79598 m	20.0671 m	6.6539 m
ECEF Y	0.0696885 m	1.68216 m	0.186058 m
ECEF Z	8.29376e-06 m	3.18196e-05 m	1.00588e-05 m
Velocità X	0.0387411 m/s	0.749128 m/s	0.106435 m/s
Velocità Y	0.219215 m/s	5.72096 m/s	0.707916 m/s
Velocità Z	0.00502424 m/s	0.0809832 m/s	0.00758546 m/s

Tabella 22: Esperimento 3.2: Risultati

In questo caso, il dimezzamento della frequenza dell'odometro ha impattato negativamente sulle performance dell'algoritmo in maniera più significativa di quanto abbia impattato positivamente il raddoppio. È dunque ragionevole pensare che 10 hertz sia un valore accettabile come frequenza di campionamento dell'odometro.

Misura	Errore medio	Errore massimo	Dev. std. errore
ECEF X	+117.607 %	-0.233667 %	+18.7543 %
ECEF Y	+186.627 %	+363.644 %	+310.939 %
ECEF Z	+132.688 %	-0.314849 %	+14.1045 %
Velocità X	+128.523 %	+501.845 %	+434.385 %
Velocità Y	+455.219 %	+575.23 %	+754.77 %
Velocità Z	+31.4417 %	+321.035 %	+141.035 %

Tabella 23: Esperimento 3.2: Confronto con esperimento 1.2



# 6

---

## CONCLUSIONI

---

Lo scopo della Tesi era mostrare i risultati sperimentali ottenuti durante le attività di analisi condotte su un sistema di posizionamento ferrotramviario innovativo.

Al giorno d'oggi, i costruttori dei sistemi ferrotramviari tendono a rispettare le normative operazionali imposte dallo standard europeo *ERTMS*. *ERTMS* nasce per uniformare i regolamenti dei paesi dell'Unione Europea in materia ferroviaria. È necessario uniformare detti regolamenti poichè le linee ferroviarie non sono più esclusivamente limitate a territori nazionali.

*ETCS* è la parte di *ERTMS* che regolamenta il posizionamento dei rotabili. La filosofia di *ETCS* mira a realizzare sistemi di posizionamento completamente autonomi, conformi al livello *ETCS-3*, i quali non fanno alcun uso di apparati installati a terra.

Nell'ottica di evoluzione verso una completa autonomia, è stato mostrato ed analizzato un possibile sistema di posizionamento autonomo operante in un contesto ferrotramviario. Tale sistema fa uso di un insieme di sensori installati a bordo treno, capaci di campionare le grandezze fisiche che caratterizzano il moto del medesimo: accelerazione, velocità angolare, velocità lineare e coordinate geografiche.

Il rumore che caratterizza ciascun processo di misura viene attenuato dall'utilizzo di un algoritmo SFA, che integra le misure campionate dai sensori al fine di stimare in modo affidabile la posizione del treno lungo la traccia entro cui si sta muovendo.

Modellando il treno come un *Cyber-Physical System*, ossia come un sistema composto da una parte *cyber* che controlla un oggetto fisico, è stato individuato il sottosistema di posizionamento oggetto della Tesi, quale parte *cyber* del sistema treno.

Tale sottosistema è stato descritto a livello hardware e a livello software, ne sono stati individuati i *sistemi costituenti*, le loro interfacce e modalità

di interazione.

È stato descritto l'apparato software che implementa il funzionamento logico del sistema di posizionamento, tale apparato è stato riprodotto in un ambiente simulato per eseguire le attività di analisi in modo non intrusivo. Tali attività di analisi rientrano nella categoria del *monitoring* di sistemi software.

Il lavoro di Tesi si è poi concluso con l'esposizione degli esperimenti effettuati e la discussione dei risultati ottenuti.

---

## BIBLIOGRAFIA

---

- [1] J.C. Laprie, *Dependability - its attributes impairments and means, Predictability Dependable Computing Systems*, Springer (1995) (Cited on page 9.)
- [2] A. Bondavalli, *L'analisi quantitativa dei Sistemi Critici, Fondamenti e Tecniche per la Valutazione - Analitica e Sperimentale - di Infrastrutture Critiche e Sistemi Affidabili* (2011) (Cited on page 12.)
- [3] J. Otegui, *A Survey of Train Positioning Solutions*, IEEE Sensors Journal, Vol. 17, No. 20 (2017) (Cited on page 13.)
- [4] T. Albrecht et al, *A precise and reliable train positioning system and its use for automation of train operation*, Proc. IEEE Int. Conf. Intell. Rail Transp. (2013) (Cited on page 14.)
- [5] European Commission, *Delivering an effective and interoperable European Rail Traffic Management System (ERTMS) - the way ahead* (2017) (Cited on page 14.)
- [6] J. Marais, J. Beugin, M. Berbineau, *A Survey of GNSS-Based Research and Developments for the European Railway Signaling*, IEEE transactions on intelligent transportation system (2017) (Cited on page 14.)
- [7] A. Neri, F. Rispoli, P. Salvatori, *An analytical assessment of a GNSS-based train integrity solution in typical ERTMS level 3 scenarios*, in Proc. Eur. Navigat. Conf. (ENC), Bordeaux, France, (2015) (Cited on page 15.)
- [8] P. Josserand, F.H Willard, *Rights of Trains* (5th ed.), Simmons-Boardman Publishing Corporation, New York (1957) (Cited on page 15.)
- [9] N.A. Zafar et al, *Towards the safety properties of moving block railway interlocking system*, International Journal of Innovative Computing Information and Control (2012) (Cited on page 15.)
- [10] MISRA, *MISRA-C:2004, Guidelines for the use of the C language in critical systems* (2004) (Cited on page 15.)

- [11] International Electrotechnical Commission, *61508-1: Functional safety of electrical/electronic/programmable electronic safety-related systems, edition 2.0* (2010) (Cited on page 15.)
- [12] CENELEC European Committee for Electrotechnical Standardization. *EN 50128:2011 - Railway Applications - Communications, signalling and processing systems - Software for railway control and protection systems* (2011) (Cited on page 16.)
- [13] R. S. Hosse, H. Manz, K. Burmeister, E. Schnieder, *Market analysis for satellite train localisation for train control systems, Proc. 5th Conf. Transp. Solutions Res. Deployment Transp.* (2014)
- [14] D. Basile et al, *A Refinement Approach to Analyse Critical Cyber-Physical Systems, Software Engineering and Formal Methods* (2017) (Cited on page 16.)
- [15] A. Ceccarelli et al, *Basic Concepts on Systems of Systems, Cyber-Physical Systems of Systems*, Springer (2017) (Cited on page 16.)  
(Cited on page 16.)
- [16] A. Mirabadi et al, *Application of sensor fusion to railway systems*, IEEE (1996) (Cited on page 16.)
- [17] F. Bohringer, A. Geistler, *Adaptation of the kinematic train model using the interacting multiple model estimator, Advances in Transport*, vol. 74, no. 7. Southampton (2004) (Cited on page 19.)
- [18] B. Cai, X. Wang, *Train positioning via integration and fusion of GPS and inertial sensors, WIT Transactions on the Built Environment*, Southampton (2000) (Cited on page 20.)
- [19] M. Malvezzi et al, *A localization algorithm for railway vehicles based on sensor fusion between tachometers and inertial measurement units, Proc. Inst. Mech.* (Cited on page 20.)
- [20] C. Reimer et al, *INS/GNSS/odometer data fusion in railway applications, Proc. DGON Intertial Sensors Syst. (ISS)*, vol. 2. Karlsruhe, Germany (2016) (Cited on page 20.)
- [21] L. Junyan et al, *Application Research of Embedded Database SQLite*, IEEE (2009) (Cited on page 20.)

- [22] X. Liu, A. Goldsmith, *Kalman Filtering with Partial Observation Losses*, Department of Electrical Engineering, Stanford University, Stanford, USA (Cited on page 21.)
- [23] R. Mazl, L. Preucil, *Sensor Data Fusion for Inertial Navigation of Trains in GPS Dark Areas (Mathematics in Science and Engineering)*, San Diego, CA, USA (2003) (Cited on page 21.)
- [24] S. Mittal, *A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform*, Journal of Systems Architecture Volume 97 (2019) (Cited on page 22.)
- [25] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications 2nd edn*, Springer, New York (2011) (Cited on page 23.)
- [26] H. Kopetz, W. Ochsenreiter, *Clock synchronization in distributed real-time systems*, IEEE (1987) (Cited on page 30.)
- [27] K. Wolter et al, *Resilience Assessment and Evaluation of Computing Systems*, Springer (2012) (Cited on page 35.)
- [28] M.J. Pont, *Patterns for Time-Triggered Embedded Systems*, Addison-Wesley (2001) (Cited on page 26.)
- [29] H. Kopetz, *Event-Triggered versus Time-Triggered Real-Time Systems*, Springer (1991) (Cited on page 26.)
- [30] ISO/IEC 7498-1:1994, *Information technology, Open Systems Interconnection - Basic Reference Model: The Basic Model* (1994) (Cited on page 26.)
- [31] D.E. Katlin, *Estimation, Control, and the Discrete Kalman Filter*, Applied Mathematical Science, Springer (1987)
- [32] S. Dilhaire, D. Maillet, *Dealing with the measurement noise of a sensor* (2015) (Cited on page 36.)
- [33] M. Karlesky et al, *Mocking the Embedded World, Test-Driven Development, Continuous Integration, and Design Patterns Embedded Systems Conference, Silicon Valley (San Jose, California)* (2007) (Cited on page 35.)